# Métrica de Coesão de Responsabilidade - A Utilidade de Métrica de Coesão na Identificação de Classes com Problemas Estruturais

**Kecia A. M. Ferreira[1], Mariza A. S. Bigonha[2], Roberto S. Bigonha[2], Heitor C. Almeida[2], Roberta Coeli das Neves[1]**

[1]Departamento de Computação – CEFET-MG
Av. Amazonas, 7675 – Nova Gameleira – Belo Horizonte-MG

[2]Departamento de Ciência da Computação – UFMG
Av. Antônio Carlos, 6627 – Pampulha – CEP: 31270-010 – Belo Horizonte-MG

{kecia,mariza,bigonha}@dcc.ufmg.br

***Resumo.*** *Muitas métricas de coesão de classe têm sido propostas na literatura. Entretanto, ainda não há um consenso sobre a melhor abordadem para medir coesão. Uma questão importante nesse tópico é que o grau de coesão interna de uma classe é dificilmente capturado por meio automático, pois esse tipo de avaliação é estreitamente dependente do conhecimento do domínio de problema da aplicação. Os resultados relatados neste artigo identificam evidências de que, embora métricas de coesão possam não ser indicadores precisos, elas são úteis para a avaliação da qualidade estrutural de uma classe. São avaliadas quatro métricas: LCOM, LCOM4, TCC e COR. Coesão de Responsabilidade (COR) é definida neste trabalho como um indicador do número de responsabilidades implementadas por uma classe. A métrica COR foi definida de maneira a tornar mais simples a interpretação dos resultados da avaliação da coesão interna de uma classe. Os resultados deste trabalho mostram que a aplicação de métricas como COR podem auxiliar a identificar classes com problemas estruturais.*

***Abstract.*** *Several class cohesion metrics have been proposed in the literature. However, there is no consensual approach to measure class cohesion yet. An important issue in this topic is that it is difficult to assess cohesion quantitatively because this is a semantical concept. In the present work, we carried out an experimental study in order to find out evidences that cohesion metrics are significant indicators of the structural quality of classes. We evaluated four cohesion metrics: LCOM, LCOM4, TCC and COR. Cohesion of Responsibility (COR) is defined in this work as an indicator of the number of responsibilities implemented by a class. The definition of COR aims to generate values which are simple and easy to interpret. Findings of this work show that metrics like COR are usefull in identifying classes with design deviances.*

## 1. Introdução

Modularidade, a característica de um software construído a partir de unidades denominadas módulos, contribui para a redução da complexidade estrutural de software e facilita

a realização de modificações [Pressman 2006]. Há dois aspectos essenciais para se obter modularidade: minimizar o relacionamento entre módulos, o que se denomina acoplamento (*coupling*), e maximizar o relacionamento entre elementos no mesmo módulo, o que se denomina coesão (*strength* ou *cohesion*) [Myers 1975]. Um módulo é coeso quando implementa idealmente um único conceito. Pressman [Pressman 2006] define a coesão como "uma indicação qualitativa do grau em que um módulo focaliza em apenas uma coisa", avaliando que é preciso buscar alta coesão de módulos, mas que não é essencial determinar precisamente o nível de coesão. Para Pressman [Pressman 2006] o que é de fato relevante é poder identificar a ocorrência de baixa coesão em um projeto com o objetivo de poder modificá-lo para obter maior independência funcional. A avaliação do grau de coesão de um módulo é uma tarefa estreitamente dependente de conhecimento do domínio do problema, demandando avaliação humana. Contudo, a avaliação manual de classes, sobretudo em softwares de grande porte, para identificar classes com problemas de coesão é uma tarefa desafiadora.

Muitas métricas de coesão têm sido propostas [Chidamber and Kemerer 1994, Briand et al. 1998a, Marcus and Poshyvanyk 2005, Counsell et al. 2006, Mäkelä and Leppänen 2007, Al-Dallal 2009]. Esses trabalhos visam a definição de uma métrica que capture com fidelidade o grau de coesão interna de classes. Todavia, obter esse indicador automático parece ser uma meta inatingível, visto que até mesmo a avaliação manual realizada por especialista, que possua conhecimento semântico do software, pode não ser tão precisa. Boa parte dos trabalhos que se propõem a avaliar métricas de software, incluindo métricas de coesão, visam verificar a aplicabilidade delas na predição de falhas [Basili et al. 1996, Briand et al. 1998b, Singh et al. 2010]. Entretanto, o fato de uma métrica não servir para esse tipo de predição não implica que ela seja desqualificada para outros propósitos. Particularmente no caso de coesão, é importante avaliar se a métrica auxilia a identificar classes com necessidades de refatoração, o que é um dos aspectos mais importantes na avaliação de coesão, como pondera Pressman [Pressman 2006].

O presente trabalho tem por objetivo avaliar a utilidade de quatro métricas de coesão na identificação de classes com problemas estruturais: LCOM (*lack of cohesion in methods*) [Chidamber and Kemerer 1994], LCOM4 [Hitz and Montazeri 1995], TCC (*tight class cohesion*) [Bieman and Kang 1995] e COR. LCOM foi uma das primeiras métricas de coesão de classes propostas na literatura. Ela tem sido amplamente criticada por gerar valores que não representam satisfatoriamente o nível de coesão de classes. Apesar disso, muitas das ferramentas de coleta de métricas coletam essa métrica [Lincke et al. 2008]. Essa é uma das razões pela qual esta métrica é avaliada neste estudo. Uma outra razão é observar se esta métrica, que tem sido considerada falha, é de fato insignificante na identificação de classes com problemas estruturais. A métrica LCOM4 é definida com base nos mesmos conceitos de LCOM, porém de forma a contornar alguns dos problemas de LCOM. Tanto LCOM como LCOM4 resultam em valores a partir de 0 e correspondem a uma medida inversa de coesão, já que avaliam ausência de coesão em classes. A métrica TCC é normalizada no intervalo [0,1], sendo que o valor máximo indica alta coesão. Essas métricas são descritas em maiores detalhes na Seção 2. COR (Coesão de Responsabilidades) é uma métrica definida neste trabalho para avaliar coesão interna de classes. Ela é baseada nos conceitos utilizados em LCOM4, porém é uma medida direta da coesão e gera valores normalizados no intervalo [0,1]. Embora TCC tenha

também essas características, os valores reportados por COR permitem uma interpretação mais simples da estrutura da classe.

Para avaliar as métricas, os valores delas foram comparados a avaliações qualitativas realizadas em classes de softwares abertos desenvolvidos em Java. Os resultados dessa avaliação mostram que as métricas consideradas podem ser utilizadas para identificar classes com potenciais necessidades de refatoração. Um segundo estudo de caso foi realizado com o objetivo de verificar o comportamento de tais métricas na avaliação de um software cuja estrutura tem sido avaliada qualitativamente como boa em outros estudos. Os resultados desse estudo de caso mostram que as métricas que mais se aproximam dessa avaliação qualitativa são COR e LCOM4.

Este trabalho realiza uma avaliação prática das referidas métricas de software, identificando evidências empíricas de que métricas de coesão são úteis em cenários de Engenharia de Software, especialmente no que compete à avaliação e à melhoria da qualidade estrutural de produtos de software. Os achados de trabalhos dessa natureza podem encorajar o uso efetivo de métricas de coesão como instrumentos na identificação de necessidades de refatoração em software.

O restante deste artigo está organizado da seguinte forma: na Seção 2 são discutidos trabalhos relacionados ao tema do presente trabalho; a metodologia aplicada na condução deste trabalho é apresentada na Seção 3; a métrica COR é definida na Seção 4; a Seção 5 identifica valores referência para as métricas LCOM4 e COR; a Seção 6 descreve a avaliação das métricas; as ameaças à validade do estudo são discutidas na Seção 7; a Seção 8 apresenta as conclusões do trabalho e a indicação de trabalhos futuros.

## 2. Trabalhos Relacionados

Um estudo comparativo entre métricas de coesão interna de classes realizado por Briand et al. [Briand et al. 1998a] cita sete métricas com esse propósito. Outras métricas de coesão foram propostas depois desse levantamento. Nesta seção, são descritas seis dessas métricas que utilizam diferentes abordagens de avaliação de coesão.

A métrica LCOM [Chidamber and Kemerer 1994] mede a ausência de coesão entre os métodos de uma classe. A definição desta métrica considera que a coesão entre os métodos de uma classe é dada pela similaridade entre eles. Um método é considerado similar a outro se ambos usam pelo menos uma variável de instância da classe em comum. 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. O cálculo de LCOM é dado por: $LCOM = \mid P \mid - \mid Q \mid$, se $\mid P \mid > \mid Q \mid$ e $LCOM = 0$, caso contrário. LCOM resulta em valores a partir de zero, sendo que baixos valores indicam bom nível de similaridade, portanto de coesão, entre os métodos da classe avaliada. Entretanto, o valor nulo não necessariamente indica boa coesão da classe. Essa é uma das razões para LCOM ser considerada inválida [Kitchenham 2009]. Apesar disso, ela é implementada em muitas ferramentas de coleta de métricas [Lincke et al. 2008].

Hitz e Montazeri [Hitz and Montazeri 1995] definem um métrica que é conhecida na literatura como LCOM4. Eles modelam o relacionamento entre métodos de uma classe como um grafo, no qual os métodos são os vértices. Neste grafo, há uma aresta entre

dois métodos se eles usam um atributo da classe em comum, ou se um método chama o outro. A métrica LCOM4 é definida como o número de componentes conectados no grafo. Assim como LCOM, os valores dessa métrica não são normalizados no intervalo [0,1]. De acordo com Abreu e Carapuça [Abreu and Carapuça 1994] , este tipo de normalização é apropriado porque permite uma avaliação independente do tamanho do sistema, o que torna possível comparar sistemas de tamanhos diferentes. Além disso, tanto LCOM como LCOM4 são medidas inversas de coesão, visto que avaliam *ausência* de coesão em classes: valores baixos correspondem a alta coesão, enquanto valores altos correspondem a baixa coesão.

Bieman e Kang [Bieman and Kang 1995] definem a métrica TCC. Na definição de TCC, considera-se uma classe *C*, na qual *NP(C)* é o número máximo de pares de métodos da classe. Em uma classe com *n* métodos, $NP(C) = n(n-1)/2$. O número de pares de métodos que possuem conexão direta na classe é representado por NDC(C). Um método está diretamente conectado a outro se ambos usam pelo menos uma variável de instância em comum da classe. TCC é dada por $NDC(C)/NP(C)$, sendo um percentual de pares de métodos com conexão direta na classe. Os resultados de TCC são normalizados no intervalo [0,1], sendo que o valor 1 corresponde a alta coesão.

Bansiya et al.[1](1999 apud Counsell et al. (2006)) propõem a métrica CAMC (*cohesion among methods in a class*), que considera uma classe coesa quando seus métodos usam o mesmo conjunto de tipos de parâmetros. Marcus e Poshyvanyk (2005) propõem medir a coesão de classes com base na análise de informação semântica no código fonte, tais como comentários e identificadores, e, para isso, propõem a métrica C3, denominada *coesão conceitual*. O cálculo da métrica baseia-se em técnicas de recuperação de informação. Mäkelä e Leppänen (2007) propõem a medição de coesão externa de classe, com base na premissa de que para o cliente de uma classe, a sua representação interna não é tão relevante quanto a sua interface. A métrica, denominada ELCOM (*external lack of cohesion in methods*), é dada por $ELCOM(c) = 1 - (|C(c)|/(|M(c)|.|V(c)|))$, onde $M(c)$ é o conjunto de classes clientes de *c*, $V(c)$ é o conjunto de variáveis de instância de *c*, e $C(c)$ é o conjunto de pares ordenados $(m, v)$ pertencentes ao produto cartesiano entre $M(c)$ e $V(c)$, tal que $m$ possua algum método que use $v$.

As métricas de coesão propostas na literatura sofrem de três problemas principais: (i) algumas têm definição complexa, de difícil compreensão, como é o caso de ELCOM, C3 e CAMC; (ii) os valores resultantes de boa parte das métricas não se traduzem em indicadores de fácil compreensão, que possam orientar o especialista na decisão de reestruturação da classe; (iii) muitas dessas métricas avaliam *ausência de coesão* e não a coesão interna da classe, o que também prejudica a interpretação de seus resultados. A métrica proposta no presente trabalho, COR, foi definida com o objetivo de contornar esses problemas. Com base no conceito de que uma classe coesa deve implementar uma única responsabilidade, a métrica foi definida de forma a resultar em valores no intervalo [0,1] e que indiquem a quantidade de responsabilidades que uma classe implementa. Desta forma, seus resultados podem apoiar melhor a tarefa de avaliação e reestruturação de classes, que é o objetivo principal de uma métrica de coesão de classe.

Muitos trabalhos concentram-se em verificar se métricas de coesão podem

[1]Bansyia, J., Etzkorn, L. *A class cohesion metric for object-oriented designs*. Journal Object-Oriented Program. Volume 11, pp. 47-52, 1999.

ser utilizadas como instrumentos de predição de falhas em software. Basili et al. [Basili et al. 1996] concluíram que LCOM não é significante para a predição de falhas em software, enquanto os resultados de outros trabalhos [Gyimothy et al. 2005, Zhou and Leung 2006, Olague et al. 2007, Singh et al. 2010] concluíram o oposto. Marinescu [Marinescu 1998] realizou um estudo de caso com três softwares desenvolvidos em Visual C++, concluindo que TCC pode ser utilizada para identificar classes com falhas de coesão. Olbrich et al. [Olbrich et al. 2009] definem estratégias para identificar *code smells* [Fowler 1999] utilizando-se métricas de software. Um dos *code smells* estudados por eles é o *God Class*, que são classes que realizam muitas funcionalidades. Eles recomendam utilizar uma métrica de coesão, dentre outras, para identificar *God Classes*. A métrica recomendada no estudo deles é TCC. Entretanto, eles não avaliam se TCC identifica adequadamente esse tipo de *code smell*. Há uma carência de estudos empíricos para avaliar se métricas de coesão auxiliam a identificar classes com problemas de coesão adequadamente. Um estudo de Kitchenham (2009) sobre a produção científica na área de métricas de software mostra que os principais aspectos tratados nos estudos de avaliação de métricas de software orientado por objetos são estimativa de falhas, estimativa de esforço e tamanho.

No presente trabalho é realizado um estudo empírico para avaliar se as métricas LCOM, LCOM4, TCC e COR podem auxiliar a identificação de classes com problemas estruturais. O comportamento das métricas é comparado à avaliação qualitativa de um conjunto de classes. Como essas classes são de software aberto, os resultados da análise qualitativa é verificável e replicável. Esse tipo de avaliação não é comum em trabalhos anteriores. Além disso, as métricas são utilizadas para avaliar um software aberto cuja estrutura tem sido avaliada qualitativamente como boa. A principal contribuição deste trabalho são os resultados de um estudo empírico com uma quantidade razoável de classes que evidencia a utilidade do uso de métricas de coesão no auxílio à avaliação e à reestruturação de software.

## 3. Metodologia

Em um estudo anterior [Ferreira et al. 2009], foram identificados valores referência para métricas de software orientado por objetos, dentre elas LCOM. No estudo, foram analisados dados de mais de 26000 classes de software aberto Java. O objetivo do estudo foi identificar faixas de valores referência para as métricas estudadas da seguinte forma: a faixa *bom* corresponde aos valores da métrica que ocorrem com maior frequência, a faixa *regular* corresponde a valores que ocorrem com uma frequência baixa, mas não irrelevante, e a faixa *ruim*, corresponde a valores que ocorrem com uma frequência desprezível. Essas faixas de valores não necessariamente representam as melhores práticas em Engenharia de Software, mas fornecem um parâmetro para avaliar software em comparação à prática comum. Para a métrica LCOM, identificaram-se os seguintes valores referência: *bom*: 0 – *regular*: 1 a 20 – *ruim*: superior a 20. Esses valores referência, bem como aqueles para as demais métricas utilizadas no estudo, foram avaliados por meio de estudos de caso [Ferreira 2011]. O resultado dessa avaliação mostrou que os valores referência permitem identificar classes com problemas estruturais.

No presente trabalho, foi utilizado o mesmo conjunto de dados utilizado na derivação desses valores referência. Como a tarefa de examinar manualmente essa grande quantidade de classe é proibitiva, foram selecionadas classes avaliadas como *ruim* nos

valores referência de LCOM. O critério para aplicação desse filtro é que para as demais métricas ainda não havia definição de valores referência. Esse filtro resultou em 22 classes. A métrica TCC foi, então, coletada nesse conjunto de classes pela ferramenta VizzAnalyzer[2]. As métricas LCOM4 e COR foram coletadas pela ferramenta Connecta [Ferreira 2006] no conjunto completo de classes. O motivo pelo qual foi realizada coleta de TCC somente nas 22 classes é que sua coleta foi realizada por uma ferramenta diferente daquela utilizada para coletar LCOM, LCOM4 e COR, e a tarefa de coletar esses dados consome uma grande quantidade de tempo.

Utilizando-se a mesma abordagem aplicada para a identificação dos valores referência de LCOM, foram identificados valores referência para LCOM4 e COR. Em linhas gerais, o método para identificar esses valores consiste em coletar os valores das métricas, observar as propriedades das distribuições desses valores e identificar três faixas de valores: uma que ocorre com uma frequência alta, uma que ocorre com uma frequência intermediária, e outra que ocorre com uma frequência baixa. Essas três faixas são denominadas, respectivamente, como *bom*, *regular* e *ruim*.

As classes foram inspecionadas manualmente e avaliadas qualitativamente. Os resultados dessa avaliação foram comparados aos valores reportados pelas métricas. O critério para essa comparação foi identificar se as classes com problemas estruturais são classificadas nas faixas *ruim* das métricas. No caso da métrica TCC, para a qual não foram derivados valores referência, observou-se quão próximos os resultados são de zero, que corresponde a baixa coesão na avaliação desta métrica.

## 4. Definição da Métrica de Coesão de Responsabilidade

Idealmente, uma classe deve ter um único propósito bem delimitado no software, implementando um único conceito. A abordagem empregada em métricas como LCOM, LCOM4 e TCC é avaliar a coesão interna de classe por meio do uso de variáveis de instância da classe por seus métodos. A característica comum entre essas métricas é que elas consideram que um método é *similar* ou está *conectado* ao outro se ambos utilizam pelo menos uma variável de instância em comum. A métrica LCOM gera valores a partir de zero que representam a diferença entre o número de pares de métodos sem similaridade e o número de pares de métodos com similaridade. A interpretação desses valores não é trivial. Por exemplo, se uma classe tem LCOM igual a 20 significa que há 20 pares de métodos sem similaridade a mais do que pares de métodos com similaridade. É possível que esse valor de fato indique falha estrutural da classe. Contudo, extrair a partir desse número um direcionamento prático sobre o que poderia ser feito para melhorar a classe é difícil. A métrica LCOM4 vem contornar essa dificuldade. Ela fornece o número de *componentes conectados* dentro da classe. Cada componente conectado é constituído por métodos que estão relacionados entre si. Por exemplo, se LCOM4 resulta em dois, significa que há dois grupos independentes de métodos na classe. Isso é um indicador de que a classe poderia ser dividida em duas. A métrica TCC resulta no percentual de pares de métodos públicos que usam pelo menos uma variável de instância da classe em comum. Por exemplo, se TCC resulta em 0,5, significa que 50% dos pares de métodos usam variáveis de instância em comum. Tal como ocorre com LCOM, é difícil interpretar esse tipo de resultado em termos de recomendações práticas a cerca das melhorias a

[2]http://www.arisa.se/vizz_analyzer.php

serem realizadas na classe. Das três métricas, a que parece permitir uma interpretação mais simples neste contexto é LCOM4. Porém, LCOM4 não gera valores no intervalo [0,1].

Esta seção apresenta uma métrica de coesão que aplica conceitos semelhantes aos de LCOM4, mas que gera valores no intervalo [0,1]. A métrica é denominada Coesão de Responsabilidade (COR) por ser um indicador do número de *responsabilidades* que uma classe implementa. A definição de COR baseia-se nos seguintes conceitos:

1. *Responsabilidade:* um conjunto de métodos dentre os quais há *relacionamento* representa uma responsabilidade implementada pela classe.

2. *Relacionamento:* dois métodos de uma classe *C* estão relacionados se utilizam pelo menos um atributo da classe *C* em comum, ou se um utiliza o outro, ou se usam direta ou indiretamente métodos da classe *C* em comum. Um método $a$ utiliza diretamente um método $b$ se há uma chamada a $b$ no corpo de $a$. Um método $a$ utiliza indiretamente um método $b$ se em $a$ há uma chamada a algum método que pode levar a chamada de $b$.

3. *Propriedade transitiva de relacionamento:* se um método *a* está relacionado com um método *b* e *b* está relacionado com um método *c*, então *a* está relacionado com $c$.

O cálculo da métrica COR é realizado da seguinte forma:

- $S$: é o conjunto de conjuntos disjuntos formados por métodos com relacionamento entre si na classe.

- $r$: corresponde ao número de responsabilidades da classe. É dado pela quantidade de conjuntos disjuntos formados por métodos com relacionamento entre si na classe, $r = |S|$.

Assim,

$COR = 1/r$, se $r > 0$
$COR = 0$, caso contrário.

A métrica COR pode assumir valores no intervalo $[0, 1]$, sendo que quanto mais próximo de 1, melhor a coesão da classe. Opcionalmente, pode-se utilizar o valor de *r*, que indica diretamente o número de responsabilidades da classe. COR assume valor igual a zero quando não há métodos na classe.

### 4.1. Restrições

De acordo com Briand et al. [Briand et al. 1998a], a definição de uma métrica de coesão deve levar em consideração três aspectos principais: herança, métodos de acesso a dados da classe (métodos *get* e *set*) e métodos construtores. Em relação à herança, é possível considerar ou não os métodos e atributos herdados. A definição de COR não restringe qualquer uma dessas possibilidades. A exclusão de elementos herdados na análise da coesão da classe indica que se pretende avaliar o grau de coesão da extensão da classe. A

inclusão desses elementos indica que se pretende avaliar a coesão da classe como um todo. Entretanto, posto que um software é constituído por módulos que se relacionam entre si, que uma classe é um módulo, que herança é um tipo de relacionamento entre classes e que se busca avaliar coesão interna de módulos (classes), a avaliação dos elementos definidos na classe, ou seja, excluindo os elementos herdados, pode favorecer uma avaliação mais fiel da coesão interna de classe.

A inclusão de métodos de acesso no cálculo da métrica também não é restringida na definição da métrica. No caso de algumas métricas propostas na literatura, como é o caso de LCOM, a inclusão de métodos de acesso diminuem artificialmente a coesão da classe. Na métrica COR, entretanto, isso não ocorre, pois tais métodos estarão inseridos no conjunto de outros métodos que usam os respectivos atributos. Classes que possuem apenas campos e métodos de acesso são definidas por Fowler (1999) como o *bad code smell* denominado *Data Class* porque elas podem não representar um comportamento relevante no sistema. Nesse tipo de classe, a métrica COR indicará a quantidade de atributos da classe, refletindo falta de coesão da classe. Esse comportamento é desejável, já que classes com essa característica são consideradas um *bad smell*.

Semelhante ao que ocorre com outras métricas de coesão, a inclusão de método construtor pode aumentar artificialmente o valor da métrica COR, uma vez que é comum que esse tipo de método faça referência a uma grande quantidade de atributos da classe. Desta forma, o ideal é excluir esses métodos do cálculo de COR.

### 4.2. Comparação de COR com Outras Abordagens

Os conceitos empregados em COR não diferem substancialmente dos conceitos empregados nas métricas mais conhecidas para coesão. A ideia de avaliar coesão por meio da análise de relacionamento entre métodos é empregada em métricas como LCOM. Em particular, a métrica LCOM4 [Hitz and Montazeri 1995] modela o relacionamento entre métodos de uma classe como um grafo, no qual os métodos da classe são os vértices e as arestas representam relacionamentos entre dois métodos. Na abordagem da métrica COR, um componente conectado na métrica LCOM4 corresponde a uma responsabilidade.

A principal vantagem de COR em relação às demais métricas já propostas na literatura é a facilidade de interpretação de seus resultados e o fato de gerar valores independentes do tamanho da classe, no intervalo $[0, 1]$. Além disso, em métricas como LCOM, valores baixos indicam alta coesão e valores altos indicam baixa coesão. Os valores de COR, ao contrário, indicam o nível de coesão da classe: valores baixos indicam baixa coesão, e valores altos indicam alta coesão.

## 5. Valores Referência das Métricas COR e LCOM4

Esta seção identifica valores referência para a métrica COR e LCOM4. Para derivar esses valores, foram utilizadas as classes de 40 programas abertos desenvolvidos em Java. Esses software são os mesmos utilizados em um estudo anterior que identificou valores referência de outras métricas de software [Ferreira et al. 2009]. O método para derivar esses valores referência consiste em observar as propriedades das distribuições dos valores a fim de identificar as faixas de valores mais frequentes, uma faixa de frequência intermediária e outra de frequência baixa.

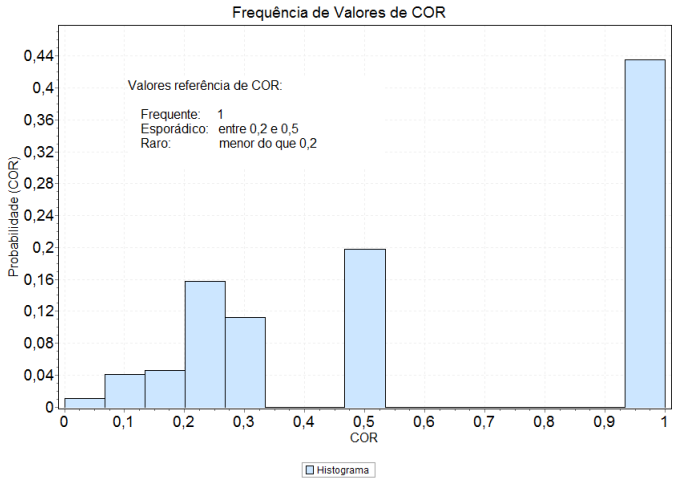
A frequência dos valores de COR é mostrada na Figura 1a. Nesta distribuição de valores, observa-se que o valor 1 é o mais frequente, com 44% das ocorrências. Pela própria definição da métrica, valores entre 0,5 e 1 não ocorrem. Valores maiores ou iguais a 0,2 e menores ou iguais a 0,5 ocorrem com uma frequência intermediária. Se tomados em conjunto, valores maiores ou iguais a 0,2 e menores ou iguais a 0,5 ocorrem com uma frequência próxima à do valor 1. Entretanto, se tomados isoladamente, valores maiores ou iguais a 0,2 e menores ou iguais a 0,5 ocorrem com uma frequência intermediária. Por exemplo, o valor 0,5 ocorre em 20% dos casos. Por esta razão, classificamos esta faixa de valores, de 0,2 a 0,5, como esporádicos. Valores menores do que 0,2 são raros. Desta forma, identificam-se os seguintes valores referência para a métrica COR: *bom*: 1 – *regular*: de 0,2 a 0,5 – *ruim*: inferior a 0,2. A identificação desses valores referência são úteis no processo de avaliação de software na medida em que fornecem dados da prática usual. No caso de COR, usualmente as classe tem valor 1, o que indica que a maior parte das classes possuem apenas uma responsabilidade. Isso é uma evidência de que, em geral, a diretriz de que uma classe deve ter um único propósito é seguida pelos desenvolvedores. Uma parcela baixa, mas não desprezível, das classes tem COR entre 0,5 e 0,2, o que indica que, na prática, é geralmente tolerável classes com até cinco responsabilidades. Esse resultado é um parâmetro para avaliação de software para o desenvolvedor que deseja alcançar o mesmo patamar de qualidade de software geralmente praticado.

A frequência dos valores de LCOM4 é mostrada na Figura 1b. Nesta distribuição de valores, observa-se que o valor 1 é o mais frequente, com cerca de 44% das ocorrências. Valores de 2 a 5 ocorrem com uma frequência intermediária. Valores superiores a 5 ocorrem com uma frequência muito baixa. Desta forma, identificam-se os seguintes valores referência para a métrica LCOM4: *bom*: 1 – *regular*: de 2 a 5 – *ruim*: superior a 5. Esse resultado mostra que usualmente as classes têm LCOM4 igual a 1, o que indica que, no conceito aplicado na definição da métrica, as classes têm boa coesão em geral.

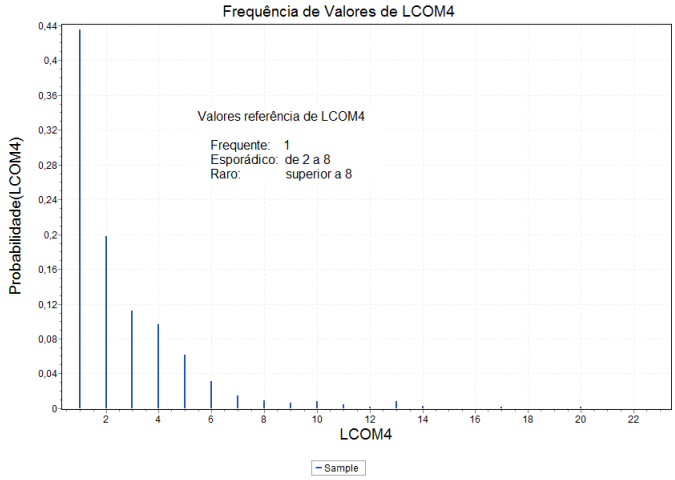
## 6. Avaliação das Métricas de Coesão como Indicadores de Necessidades de Reestruturação

Para avaliar a utilidade das métricas LCOM, LCOM4, TCC e COR na identificação de necessidades de reestruturação de classes, foram inspecionadas 22 classes. Os dados dessas classes são relatados na Tabela 1. A descrição detalhada dessa avaliação pode ser obtida na tese de doutorado de Ferreira [Ferreira 2011]. As classes avaliadas apresentam os seguintes problemas: sete possuem atributos públicos não constantes, o que fere o princípio de ocultação de informação; três são meramente classes utilitárias no sistema, possuindo métodos estáticos de propósitos distintos; quatro possuem somente métodos *get* e *set*, o que caracteriza o *code smell* denominado *Data Class*; treze delas possuem baixa coesão, pois não parecem desempenhar um propósito unico no sistema.

A métrica LCOM foi utilizada para filtrar essas classes. Desta forma, todas as classes desse conjunto avaliado possuem valor de LCOM superior a 20, que é o limite inferior da faixa *ruim* dessa métrica. Isso indica que, apesar das críticas realizadas a LCOM a respeito de sua validade, a métrica pode ser útil para auxiliar na identificação de classes com deficiência de coesão. Quando essa métrica assume valores muito altos, significa que a classe possui uma grande quantidade de métodos. Como avalia Fowler [Fowler 1999], o fato de uma classe possuir muitos métodos é um sintoma de que ela tem muitas responsabilidades, o que pode resultar na dificuldade de entendimento e manutenção da classe.




(a)




(b)

**Figura 1. Frequência de valores – (a) COR e (b) LCOM4**

De acordo com a avaliação dada por LCOM4, nenhuma das classes avaliadas tem alto grau de coesão, pois todas elas têm LCOM4 menor do que 1, sendo que nove delas foram classificadas na faixa *ruim* de LCOM4. Esse resultado é consistente com a avaliação qualitativa das classes. Embora não tenham sido derivados valores refência para TCC, observa-se que as classes avaliadas apresentam valores muito baixos para essa métrica. O maior valor de TCC no conjunto avaliado é 0,39. As demais classes apresentam valor inferior a 0,20, sendo que 11 delas apresentam valores muito próximos de zero, que corresponde ao menor nível de coesão na avaliação dessa métrica. Segundo a avaliação de COR, nenhuma das classes apresentam alto grau de coesão, sendo que 15 delas são classificadas na faixa *ruim* de COR. Esse resultado também é consistente com a avaliação qualitativa das classes.

LCOM e LCOM4 geram valores a partir de zero, não normalizados. As interpretações dos valores gerados por essas métricas são distintas. O mesmo ocorre com COR e TCC, que geram valores no intervalo [0,1], mas com significados muito diferentes.

Os resultados dessa avaliação mostram que, apesar dessas diferenças de interpretações, as métricas utilizadas podem auxiliar a identificar classes com problemas de coesão. Isso não implica que essa métricas avaliem fielmente e com acurácia o nível de coesão das classes. Todavia, elas se mostram capazes de fornecer um indício de que pode haver algo de errado com a estrutura da classe. Desta forma, essas métricas são potencialmente úteis para auxiliar a identificar necessidades de refatoração, principalmente em softwares de grande porte, nos quais a tarefa de realizar avaliação qualitativa pode consumir muito tempo.

**Tabela 1. Classes avaliadas e valores de LCOM, LCOM4, TCC e COR**

| Software | Classe | Baixa Coesão | LCOM | LCOM4 | TCC | COR |
|---|---|---|---|---|---|---|
| *JasperReports* | `net.sf.jasperreports.engine.xml.JRXmlConstants` | Sim | 957 | 7 | 0,001 | 0,14 |
| *KolMafia* | `net.sourceforge.kolmafia.textui.DataTypes` | Sim | 261 | 6 | 0,004 | 0,167 |
| *JavaX11Library* | `gnu.x11.Display` | Sim | 67 | 4 | 0,003 | 0,25 |
| *KolMafia* | `net.sourceforge.kolmafia.request.UseItemRequest` | Sim | 67 | 3 | 0,103 | 0,333 |
| *KolMafia* | `net.sourceforge.kolmafia.KoLmafia` | Sim | 2826 | 32 | 0,007 | 0,031 |
| *Hibernate* | `org.hibernate.Hibernate` | Sim | 91 | 13 | 0 | 0,08 |
| *JavaX11Library* | `gnu.x11.Window` | Sim | 1345 | 20 | 0,013 | 0,05 |
| *KolMafia* | `net.sourceforge.kolmafia.KoLAdventure` | Sim | 147 | 3 | 0,12 | 0,333 |
| *JavaX11Library* | `gnu.x11.extension.glx.VisualConfig` | Sim | 878 | 2 | 0,12 | 0,5 |
| *KolMafia* | `net.sourceforge.kolmafia.request.GenericRequest` | Sim | 872 | 14 | 0,93 | 0,071 |
| *Jpilot* | `org.jpilotexam.ui.util.UIUtilities` | Sim | 21 | 7 | 0 | 0,14 |
| *CodeGeneration* | `net.sf.cglib.core.CodeEmitter` | Sim | 2175 | 10 | 0,011 | 0,1 |
| *KolMafia* | `net.sourceforge.kolmafia.persistence.ConcoctionDatabase` | Sim | 328 | 12 | 0,032 | 0,083 |
| *Bcel* | `org.apache.bcel.generic.Type` | Sim | 24 | 2 | 0,18 | 0,5 |
| *KolMafia* | `net.sourceforge.kolmafia.AdventureResult` | Sim | 46 | 4 | 0,39 | 0,25 |
| *KolMafia* | `net.sourceforge.kolmafia.request.UseSkillRequest` | Sim | 285 | 9 | 0,16 | 0,111 |
| *KolMafia* | `net.sourceforge.kolmafia.KoLCharacter` | Sim | 15507 | 15 | 0,01 | 0,067 |
| *JasperReports* | `net.sf.jasperreports.engine.util.JRProperties` | Sim | 223 | 17 | 0,14 | 0,06 |
| *JasperReports* | `net.sf.jasperreports.engine.design.JRDesignChart` | Sim | 4449 | 13 | 0,004 | 0,08 |
| *KolMafia* | `net.sourceforge.kolmafia.request.EquipmentRequest` | Sim | 62 | 3 | 0,09 | 0,333 |
| *KolMafia* | `net.sourceforge.kolmafia.persistence.SkillDatabase` | Sim | 206 | 7 | 0,35 | 0,143 |
| *KolMafia* | `net.sourceforge.kolmafia.swingui.panel.ItemManagePanel` | Sim | 53 | 5 | 0,073 | 0,2 |

Um segundo estudo de caso foi realizado com o objetivo de verificar se as métricas avaliadas são também úteis para identificar classes de boa coesão. O software utilizado neste estudo de caso foi JHotDraw[3], que tem sido utilizado em outros trabalhos relacionados à qualidade de software e cuja estrutura tem sido considerada boa [Riehle 2000, Czibula and Czibula 2008, Jancke 2010, Kessentini et al. 2010]. Esse software foi desenvolvido em Java e possui 1095 classes.

[3]http://www.jhotdraw.org/

No JHotDraw, 215 classes possuem LCOM superior a 20, que é o limite inferior da faixa *ruim* da métrica, 335 classes possuem LCOM de 1 a 20, e 545 classes possuem LCOM igual a 0. Pela avaliação de LCOM, então, 80% das classes têm coesão boa ou pelo menos regular. O número de classes com TCC igual a zero em JHotDraw, que corresponde ao menor nível de coesão de acordo com a métrica, é 452, enquanto apenas 48 classes tem TCC igual a um. Na avaliação de TCC, então, mais de 40% das classes de JHotDraw tem baixa coesão. No software, 532 classes tem COR igual a 1, o que corresponde à faixa *bom* dessa métrica e ao maior nível de coesão segundo a avaliação de COR, 490 classes são avaliadas como *regular* e 73, como *ruim*. O mesmo resultado é observado para LCOM4. Esse resultado mostra que na avaliação de LCOM4 e COR, 93% das classes de JHotDraw têm boa coesão.

Tendo como premissa que JHotDraw possui boa qualidade estrutural, conforme trabalhos anteriores têm considerado, os resultados desse segundo estudo de caso evidenciam que a métrica TCC é propensa a indicar que uma classe tem baixa coesão quando na realidade ela não tem. Sendo assim, a aplicação dessa métrica pode prejudicar o processo de identificação de necessidades de reestruturação, já que pode sinalizar ao desenvolvedor uma demanda de reestruturação inexistente. Já as métricas LCOM, LCOM4 e COR mostraram-se apropriadas para avaliar adequadamente tanto classes com deficiências de coesão quanto classes de boa coesão neste estudo. A métrica LCOM, porém, possui a anomalia de o valor 0 não necessariamente indicar boa coesão. Esse problema reduz a aplicabilidade da métrica pois pode levar a considerar uma classe como coesa quando ela possui problemas estruturais. Considerando-se as implicações dessa anomalia de LCOM, os resultados desta avaliação indicam que as métricas LCOM4 e COR são mais adequadas para a avaliação da coesão interna de classes.

Utilizamos COR em um modelo de predição de propagação de modificações em software orientado por objetos, denominado K3B [Ferreira 2011]. Uma dos parâmetros do modelo é o grau médio de coesão interna das classes do software avaliado. A métrica COR foi utilizada no modelo como esse parâmetro. Os resultados gerados pelo modelo foram comparados aos de simulações de modificações em mais de 35 programas Java. Essa avaliação mostrou que K3B reporta valores que estimam os valores obtidos nas simulações. No que se refere à métrica COR, esse resultado evidencia que a métrica é capaz de avaliar apropriadamente a coesão interna de classe, de forma que seus valores podem ser aplicados em modelos de predição como K3B.

## 7. Ameaças para a Validade do Estudo

A principal ameaça para a validade deste estudo é o fato da análise ter sido baseada em avaliação qualitativa das classes. Esse tipo de avaliação é propensa a falhas, especialmente porque demanda conhecimento do domínio do problema dos softwares utilizados no estudo. Além disso, parte da análise baseou-se na premissa de que as classes do software JHotDraw possuem boa qualidade estrutural, com base em relatos prévios na literatura. A validade dessa premissa é dependente da validade desses estudos.

No caso da identificação dos valores referência das métricas LCOM4 e COR, a metodologia aplicada e o conjunto de softwares utilizados foram os mesmos utilizados em um estudo anterior, no qual foram identificados e avaliados valores referência para um conjunto de métricas de software orientado por objetos. Embora tenha sido utilizada uma

grande quantidade de softwares, não se pode garantir que eles sejam os melhores representantes da prática comum, tampouco que possuem alta qualidade estrutural. Todavia, considerando-se a avaliação de coesão definidas pelas métricas, os resultados indicam que a maior parte das classes dos softwares utilizados possuem boa qualidade estrutural.

Como a análise das métricas de coesão foi baseada em inspeção manual de classes, que é uma tarefa dispendiosa, o número de classes avaliadas neste estudo não é grande. Por esta razão, a generalização dos resultados obtidos neste trabalho é limitada.

## 8. Conclusão

Coesão interna de módulo, definida como o grau de interrelacionamento entre os elementos de um módulo, é um importante fator da modularidade e da qualidade de um software. Um grande número de métricas de software, de diferentes abordagens, têm sido propostas para avaliação de coesão interna de classes. Apesar das importantes contribuições desses trabalhos, ainda não há um consenso sobre uma abordagem ideal de medição de coesão.

A métrica LCOM (ausência de coesão de métodos) é uma das mais referenciadas na literatura. Essa métrica tem sofrido muitas críticas, mas também tem sido inspiração para muitas outras propostas, tais como LCOM4 e TCC. O presente trabalho definiu uma nova métrica de coesão interna de classes denominada Coesão de Responsabilidade (COR). COR é um indicador do número de responsabilidades que uma classe implementa. No cálculo dessa métrica, uma responsabilidade corresponde a um conjunto de métodos dentre os quais verifica-se relacionamento. Foram identificados os valores referência para as métricas LCOM4 e COR. A identificação de valores referência é importante para permitir a interpretação apropriada dos valores gerados pelas métricas.

Um estudo empírico foi realizado para avaliar a utilidade das métricas LCOM, LCOM4, TCC e COR na identificação de classes com deficiências de coesão e necessidades de reestruturação. Foram inspecionadas 22 classes manualmente, e um software de 1095 também foi utilizado neste estudo. Os resultados do estudo indicaram que as quatro métricas avaliadas mostram-se úteis para esse propósito. Entretanto, somente as métricas LCOM, LCOM4 e COR mostraram-se adequadas para identificar classes de boa coesão. Porém, quando LCOM resulta em valor 0, não necessariamente significa que a classe possui boa coesão, o que pode levar a interpretações enganosas de seu resultado neste caso. Considerando-se essa anomalia de LCOM, as métricas LCOM4 e COR mostraram-se mais adequadas na tarefa de identificar classe de boa coesão.

Os resultados evidenciam que métricas de coesão têm papel importante em cenários de avaliação e melhoria da qualidade estrutural de classes. Isso pode encorajar o uso efetivo de tais métricas no processo de desenvolvimento e reestruturação de software. Como trabalhos futuros, identifica-se a necessidade de ferramentas de apoio ao desenvolvimento que possuam recursos para identificar os grupos de métodos relacionados entre si. Esse tipo de ferramenta poderia orientar ainda mais a reestruturação de classes. Além disso, para consolidar os achados deste trabalho, é importante a realização de estudos de caso com softwares proprietários, já que o presente trabalho foi conduzido exclusivamente com software aberto.

## Referências


Abreu, F. B. and Carapuça, R. (1994). Object-oriented software engineering: Measuring and controlling the development process. In *Proceedings of 4th Int. Conf. of Software Quality*, McLean, Estados Unidos.

Al-Dallal, J. (2009). Software similarity-based functional cohesion metric. *IET Software*, 3(1):46–57.

Basili, V. R., Briand, L. C., and Melo, W. L. (1996). A validation of object-oriented design metrics as quality indicators. *IEEE Trans. Softw. Eng.*, 22:751–761.

Bieman, J. M. and Kang, B.-K. (1995). Cohesion and reuse in an object-oriented system. *SIGSOFT Softw. Eng. Notes*, 20:259–262.

Briand, L. C., Daly, J. W., and Wüst, J. (1998a). A unified framework for cohesion measurement in object-oriented systems. *Empirical Softw. Eng.*, 3(1):65–117.

Briand, L. C., Wüst, J., Daly, J., and Porter, V. (1998b). A comprehensive empirical validation of design measures for object-oriented systems. In *Proceedings of the 5th International Symposium on Software Metrics*, METRICS '98, pages 246–, Washington, DC, USA. IEEE Computer Society.

Chidamber, S. R. and Kemerer, C. F. (1994). A metrics suite for object oriented design. *IEEE Trans. Softw. Eng.*, 20:476–493.

Counsell, S., Swift, S., and Crampton, J. (2006). The interpretation and utility of three cohesion metrics for object-oriented design. *ACM Trans. Softw. Eng. Methodol.*, 15:123–149.

Czibula, I. G. and Czibula, G. (2008). Clustering based automatic refactorings identification. In *Proceedings of the 2008 10th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, pages 253–256, Washington, DC, Estados Unidos. IEEE Computer Society.

Ferreira, K. A. M. (2006). *Avaliação de Conectividade em Sistemas Orientados por Objetos*. Dissertação de Mestrado - DCC/UFMG., Belo Horizonte, Brasil.

Ferreira, K. A. M. (2011). *Um Modelo de Predição de Amplitude da Propagação de Modificações Contratuais em Software Orientado por Objetos*. Tese de Doutorado - DCC/UFMG., Belo Horizonte, Brasil.

Ferreira, K. A. M., Bigonha, M. A. S., Bigonha, R., Mendes, L. F. O., and Almeida, H. C. (2009). Reference values for object-oriented software. In *XXIII Brazilian Symposium on Software Engineering*, pages 62–72, Fortaleza, Ceará, Brazil.

Fowler, M. (1999). *Refactoring - Improving the Design of Existing Code*. Addison Wesley.

Gyimothy, T., Ferenc, R., and Siket, I. (2005). Empirical validation of object-oriented metrics on open source software for fault prediction. *IEEE Transactions on Software Engineering*, 31:897–910.

Hitz, M. and Montazeri, B. (1995). Measuring coupling and cohesion in object-oriented systems. In *Int. Symposium on Applied Corporate Computing*, Monterrey, México.

Jancke, S. (2010). *Smell Detection in Context*. Diploma thesis. University of Bonn. Bonn, Germany., http://dirkriehle.com/computer-science/research/dissertation/.

Kessentini, M., Vaucher, S., and Sahraoui, H. (2010). Deviance from perfection is a better criterion than closeness to evil when identifying risky code. In *Proceedings of the IEEE/ACM International conference on Automated Software Engineering*, ASE '10, pages 113–122, New York, NY, Estados Unidos. ACM.

Kitchenham, B. (2009). What's up with software metrics? - a preliminary mapping study. *The Journal of Systems and Software*, 83:37–51.

Lincke, R., Lundberg, J., and Löwe, W. (2008). Comparing software metrics tools. In *ISSTA '08: Proceedings of the 2008 International Symposium on Software Testing and Analysis*, pages 131–142, New York, NY, Estados Unidos. ACM.

Marcus, A. and Poshyvanyk, D. (2005). The conceptual cohesion of classes. In *Proceedings of the 21st IEEE International Conference on Software Maintenance*, pages 133–142, Washington, DC, Estados Unidos. IEEE Computer Society.

Marinescu, R. (1998). Using object-oriented metrics for automatic design flaws detection in large scale systems. In *Workshop ion on Object-Oriented Technology*, ECOOP '98, pages 252–255, London, UK. Springer-Verlag.

Mäkelä, S. and Leppänen, V. (2007). Cliente based object-oriented cohesion metrics. In *IEEE 31st Annual International Computer Software and Applications Conference*, pages 743–748, Beijing.

Myers, G. J. (1975). *Reliable software through composite design*. Petrocelli/Charter, Nova York, 2 edition.

Olague, H. M., Etzkorn, L. H., Gholston, S., and Quattlebaum, S. (2007). Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly iterative or agile software development processes. *IEEE Trans. Softw. Eng.*, 33:402–419.

Olbrich, S., Cruzes, D. S., Basili, V., and Zazworka, N. (2009). The evolution and impact of code smells: A case study of two open source systems. In *ESEM '09: Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*, pages 390–400, Washington, DC, Estados Unidos. IEEE Computer Society.

Pressman, R. S. (2006). *Engenharia de Software*. MacGraw Hill, Rio de Janeiro, 6 edition.

Riehle, D. (2000). *Framework Design: A Role Modeling Approach*. Dissertation No. 13509, ETH Zürich., http://dirkriehle.com/computer-science/research/dissertation/.

Singh, Y., Kaur, A., and Malhotra, R. (2010). Empirical validation of object-oriented metrics for predicting fault proneness models. *Software Quality Journal*, 18:3–35. 10.1007/s11219-009-9079-6.

Zhou, Y. and Leung, H. (2006). Empirical analysis of object-oriented design metrics for predicting high and low severity faults. *IEEE Transactions on Software Engineering*, 32(10):771–789.