Automated use case similarity detection can aid early cohesion and coupling assessment

Abstract

The abstract goes here.

1 Introduction

A manutenção de software sempre foi tema de diversos estudos ao longo dos anos. Algumas conclusões sugerem que quanto mais rápida for a descoberta de um erro, menos custoso será sua correção [1]. Este custo pode estar relacionado a questão monetária e/ou prazo. Segundo Pressman, a fase de manutenção de um software é importante porque é larga e custosa. Manutenção consome de 40% a 80% dos custos de um software [2]. Portanto, qualquer atividade que facilite a descoberta de erros ou antecipe possíveis problemas futuros pode auxiliar a atividade de construção de software.

Partindo da ideia de que quanto mais cedo for descoberto um problema, menos custoso é a sua manutenção, podemos pressupor que, se a identificação de erros acontecer nas fases iniciais do processo de engenharia de software, tal como a etapa de análise, mais fácil será a correção deste problema.

Dentre as atividades da etapa de análise, temos a diagramação de casos de uso. Estes diagramas utilizam a notação UML (*Unified Modeling Language*) e mostram as atividades de sistema por uma visão externa. Estes casos de uso devem descrever as funções providas por sistema assim como os resultados recebidos por um ator. Um ator é uma entidade que interage com o sistema (Exemplo: um usuário, outro sistema). Para a descrição dos casos de uso é utilizada a linguagem natural [3].

Este artigo apresenta uma correlação entre as métricas de software CBO, LCOM e WMC, propostas por Chidamber e Kemerer, em 1994, com o grau de similaridade entre itens de casos de uso de um software comercial. Primeiramente, nós faremos a escolha de um dos diversos métodos de análise do grau de similaridade entre *Strings* e motivaremos nossa escolha com base em atividades realizadas por seres humanos especialistas. Então, levantaremos as métricas do estudo de caso e correlacionaremos os valores da similaridade entre casos de uso com os valores das métricas escolhidas neste trabalho.

A organização do restante do artigo é a seguinte: Na próxima seção é discutido a literatura base para a realização do presente trabalho. São abordados tópicos de análise de similaridade entre *Strings*, métricas de software e detecção de clones. Na Seção 3, nós apresentamos como foi feita a coleta de dados, extração das métricas e a correlação entre as métricas e o grau de similaridade entre casos de uso. Na Seção 5 são exibidos os resultados deste artigo e nas Seções 6 e 7 são analisadas as ameaças a validade do atigo e as conclusões e trabalhos futuros.

2 Referencial Teórico

2.1 Similaridade entre Textos em Casos de Uso

Para a realização deste artigo, foi necessário encontrar similaridade entre itens de um diagrama de casos de uso. Esta atividade poderia ser feita de duas maneiras: utilizando especialistas humanos ou utilizando técnicas automatizadas.

A primeira forma traz alguns benefícios importantes, tais como precisão mais elevada, interpretação das características do contexto e utilização da experiência na análise. Só que, por ser subjetivo, alguns problemas podem aparecer durante esta análise. O cenário citado anteriormente é útil quando a experiência e conhecimento do contexto estão bem definidos. Se estas características não forem atendidas, o reconhecimento de textos similares terá sua eficiência comprometida. Um exemplo disso é, num caso de uso, palavras técnicas devem ser bem estabelecidas no contexto dos especialistas.

A segunda forma é utilizar algoritmos automatizados para efetuar esta análise. Só que encontrar similaridade entre textos de forma automatizada não é uma tarefa trivial. Vários problemas podem aparecer, principalmente quando relacionados ao contexto da aplicação das palavras [4]. Existem vários modelos para análise de similaridade entre textos [5]. O presente trabalho focou em dois deles: *Jaro-*

Winkler [6] e *Levenshtein* [7]. Mas algumas vantagens podem ser observadas, principalmente as relacionadas ao aumento de produtividade.

Com base nessa última afirmação, este trabalho encontrou um valor de similaridade que possua a maior relação de precisão/recall de acordo com a análise de especialistas humanos. Com isto, foi possível automatizar o processo de descoberta de anomalias e poderão surgir novas atividades para validar tal número, conforme descrito na Seção 3.

2.2 Clone Detecction

A reutilização de fragmentos de código-fonte pela técnica "copiar e colar" é uma atividade comum no desenvolvimento de software. Como resultado, softwares frequentemente possuem partes de código que são muito similares, chamadas de *códigos clonados* [8]. Várias análises já foram feitas sobre os malefícios e benefícios de clonagem de código. Um dos malefícios é a descoberta de um problema numa parte de código clonada. Todas as outras partes clonadas também devem ser analisadas para corrigir este problema [9].

Portanto, a previsão da ocorrência de clonagem de código nas etapas anteriores ao desenvolvimento se torna útil para uma melhoria no índice de manutenção de um software. Só este elemento já nos motiva a pesquisar sobre a ligação entre casos de uso e a detecção de clones em código-fonte.

2.3 Software Metrics

Vários estudos já foram realizados mostrando que um componente importante na melhoria do processo de software é por meio da habilidade em metrificar este processo [10]. Várias métricas, ao longo dos anos, foram criadas, tentando abranger diversos problemas [10–12]. Dentre essa diversidade de métricas, o presente trabalho utiliza as métricas propostas por Chidamber e Kemerer, em 1994. Este conjunto de métricas, chamadas de suite de métricas CK, possuí seis métricas relacionadas a características do paradigma orientado a objetos [13]. São elas:

- **WMC - *Weighted Methods Per Class* (Métodos Ponderados por Classe):** O WMC é uma medida que é definida pela soma das complexidades dos métodos da classe. Segundo [13], a definição de complexidade não foi feita por Chidamber e Kemerer, mas foi sugerida que fosse uma unidade (um número natural), deixando esse aspecto livre. A WMC proporciona uma visão sobre alguns pontos, tais como o número de métodos, sendo que a complexidade desses pode indicar quanto tempo seria necessário para desenvolver e dar manutenção em uma determinada classe (quanto maior for o número de métodos numa classe, maior será o impacto sobre suas classes filhas). Classes com muitos métodos geralmente são de aplicações específicas, diminuindo a possibilidade de reuso [10].

- **CBO - *Coupling between object classes* (Acoplamento entre Classes):** A métrica CBO representa a quantidade de classes na qual ela está acoplada. Uma classe está acoplada a outra quando pelo menos um método de uma delas utiliza métodos e/ou variáveis de instância da outra. Por meio dessa métrica é possível identificar o nível de reusabilidade de uma classe e seu grau de modularidade. Quanto mais baixo for o valor de CBO, mais modularizada a classe fica e, consequentemente, maior é a possibilidade de reusabilidade. Ainda é possível analisar o rigor do teste de software, seja ele em nível de importância, tempo dispendido ou quantidade de testes. Isto acontece porque, quanto maior for o valor, mais complexo será o teste [13].

- **LCOM - *Lack of Cohesion in Methods* (Ausência de Coesão de Métodos):** A métrica LCOM é a medida que quantifica a similaridade dos métodos dentro de uma classe. Dois ou mais métodos são coesos quando compartilham atributos de uma classe. Quando o valor de LCOM é alto, significa que a classe não possui uma funcionalidade bem definida, uma vez que vários métodos modificam os mesmos atributos. Por outro lado, se este número for baixo, os métodos da classe serão coesos, e portanto, menos similares [13].

As métricas podem indicar diversas características dentro do processo de desenvolvimento de software, tais como propensão a erros [14–16], identificação de bad smells [17], análise do impacto na refatoração de sistemas [18] e caracterização do índice de manutenibilidade no processo de desenvolvimento de software [1]. Estes trabalhos utilizaram como foco de estudo as métricas de Chidamber e Kemerer. Com isto, é possível notar que diversos estudos utilizaram as métricas de CK, dando suporte necessário para a escolha dessas métricas no presente trabalho.

3 Model and Hyphotesis

Para concretizar o objetivo deste artigo, foi feito um trabalho sistêmico na coleta de dados, modelagem dos dados, execução das atividades e análise dos resultados, conforme descrição e ordem a seguir.

1. Escolha do estudo de caso;
2. Transformação do diagrama de casos de uso em arquivo .csv;
3. Geração de um arquivo .csv com todos os itens dos casos de uso relacionados em tuplas (a, b);

4. Aplicação da técnica Jaro-Winkler para analisar o grau de similaridade entre dois itens do diagrama de casos de uso;

5. Definição das características da metodologia, conforme Tabela 2;

6. Análise e associação dos casos de uso as classes do sistema;

7. Levantamento das métricas da suite CK das classes do sistema;

8. Levantamento do grau de clonagem de código-fonte nas classes do sistema;

9. Comparativo entre o grau de similaridade dos itens do diagrama de casos de uso com as métricas WMC, CBO e LCOM e com o grau de clonagem de código-fonte das classes.

A escolha do estudo de caso foi baseada nas seguintes características: o estudo de caso é um software comercial; a linguagem de programação é C#; o software possui documentação do diagrama de casos de uso; os funcionários da empresa possuem disponibilidade para o auxílio na realização da pesquisa; a empresa disponibiliza o código-fonte, a documentação e a mão-de-obra para pesquisas.

Assim que o software foi escolhido, iniciou-se a etapa de análise associação dos itens do diagrama de casos de uso às classes do software. Este processo foi realizado com base em especialistas humanos, que participaram do desenvolvimento e documentação do software. Esta equipe de especialistas era composta por quatro profissionais, todos com pelo menos três anos de experiência na área de desenvolvimento e participaram ativamente da construção do software.

Então, após concluir as etapas anteriores, o diagrama de casos de uso foi desmembrado em forma de tuplas de tamanho dois. Para a construção das tuplas, todos os casos de uso foram intercalados entre si, exceto repetições, já que a ordem dos itens da tupla não interessa. Com as tuplas já construídas, foi aplicado o algoritmo de Jaro-Winkler [6] nos itens das tuplas, gerando um valor de 0 a 1 de acordo com a similaridade entre os títulos dos casos de uso, onde 0 é não-similar e 1 é totalmente similar.

Validating Similarity Using Human Specialist

Para validar o grau de similaridade fornecido pela métrica *Jaro-WInkler*, foi realizado um estudo controlado com especialistas humanos que participaram do projeto de desenvolvimento do software que é estudado por este artigo.

Foram utilizados quatro profissionais da área de desenvolvimento, com 2, 7, 8 e 10 anos de experiência na área de tecnologia da informação. Os especialistas receberam o diagrama de casos de uso do software em questão e receberam as seguintes regras para a realização da atividade.

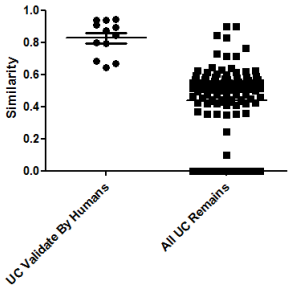

Figure 1. Gráfico comparativo entre os valores de similaridade entre casos de uso e os valores de similaridade entre casos de uso validados por humanos

1. Entendam por *similaridade* os títulos dos itens do diagrama de casos de uso que possuem uma certa semelhança entre si;

2. Entendam por *semelhança*, textos que possuam uma semântica parecida e caracteres comuns;

2. Relacionem estes casos de uso em duplas.

Após a explanação, cada um dos especialistas humanos analisaram o diagrama de casos de uso e fizeram as devidas associações. Em seguida, estes especialistas se reuniram e comentaram suas decisões. A partir disto, eles puderam mudar de opinião sobre as associações, se achassem necessário.

Para obter, de forma automatizada, valores que indiquem similaridade entre casos de uso, foi utilizado a métrica *Jaro-Winkler* [6] por meio da ferramenta *Simmetrics*[1]. A nossa escolha é sustentada pela metodologia a seguir.

Esta atividade resultou numa relação entre os valores de similaridade obtidos pela métrica *Jaro-Winkler* e a análise dos especialistas humanos. A Figura 1 exibe os valores da métrica *Jaro-Winkler* para os casos de uso que foram taxados como similares pelos especialistas humanos e os demais casos de uso que não foram considerados na análise por especialistas humanos. Este gráfico ajuda a validar o valor proposto como threshold para a característica "alta similaridade".

Ainda nesta análise, foi possível identificar um valor de similaridade da métrica *Jaro-Winkler* que possuía a melhor precisão e recall sobre a avaliação dos especialistas humanos. Esse resultado está descrito na Tabela 1. O threshold possui um recall de 70% e uma precisão de 90%.

A Tabela 2 retrata como foram tratadas algumas características da metodologia do presente trabalho. *X* é o valor da métrica Jaro-Winkler numa tupla de casos de uso e *Y* é

[1] http://sourceforge.net/projects/simmetrics/

Table 1. Precision and Recall of Jaro-Winkler based on Validation by Human Specialist

Feature	Value
Similarity	> 0.6409
Recall	60,00
95% CI	36.05% to 80.88%
Precision	87,93
95% CI	80.58% to 93.24%
Likelihood	**4.97**

a quantidade de linhas de código (LOC - *Lines of Code*) compartilhadas entre casos de uso.

Table 2. Caracterizaçªo das propriedades da metodologia

Característica	Valor
Não similares	X < 0.6409
Similares	X >= 0.6409
Compartilhamento de Classes	Y > 0
Não compartilhamento de classes	Y = 0

Para medir as métricas da suite CK, foi utilizado um software, criado durante a construção deste artigo, para o levantamento e cálculo destas métricas. O software calcula três das seis métricas de CK. São elas: WMC, CBO e LCOM. Foi utilizado como base para a realização do cálculo das métricas a biblioteca *open source Mono.Cecil*[2]. Para validar o software, as informações geradas pelo calculador de métricas de CK foram comparadas com o software comercial *nDepend* [3].

3.1 Hipóteses

O principal foco da nossa pesquisa é analisar o impacto da similaridade entre itens de um diagrama de casos de uso no processo de desenvolvimento de um software. Esta similaridade será associada a quantidade de linhas de código, métricas de acoplamento, complexidade e coesão de um conjunto de classes que se relacionam com casos de uso.

Uma segunda análise que será analisada é o compartilhamento de classes entre casos de uso. Este grau de compartilhamento será medido pela soma da quantidade de linhas de código de um caso de uso compartilha com outro caso de uso. De acordo com o grau de compartilhamento, análises serão feitas com base nas métricas de CK.

[2]http://www.mono-project.com/Cecil

[3]http://www.ndepend.com

Nós acreditamos que, de acordo com a similaridade entre casos de uso, poderão ser descobertos possíveis problemas na construção de um software. A motivação para este estudo é que, quanto mais cedo for a detecção de problemas no software, menor serão os custos envolvidos para a correção dos problemas. Custos que podem ser monetários e/ou temporais.

Hipótese 1 (H1). *Em casos de uso similares, as classes que os implementam tem maior percentual de linhas de código (LOC - Lines of Code) compartilhadas.*

Quanto mais similar for uma tupla (a, b) de itens de um diagrama de casos de uso, onde a *a* seja um caso de uso e *b* seja outro caso de uso aleatório não repetido, mais linhas de código os casos de uso compartilharão.

Hipótese 2 (H2). *Em casos de uso similares, as classes compartilhadas que os implementam possuem baixa coesão se comparadas as demais classes do sistema (LCOM).*

As classes de casos de uso que possuem similaridade e compartilham classes possuem uma maior probabilidade de ter baixa coesão, já que resolvem o problemas similares e tendem a ser mais genéricas.

Hipótese 3 (H3). *Em casos de uso não similares, as classes compartilhadas que os implementam possuem baixa coesão se comparadas as demais classes do sistema (LCOM).*

As classes de casos de uso que não possuem similaridade e compartilham classes possuem uma maior probabilidade de ter baixa coesão, já que, para atender o mesmo caso de uso, deve possuir uma responsabilidade maior, já que deve resolver diferentes problemas.

Hipótese 4 (H4). *Em casos de uso não similares, as classes compartilhadas que os implementam possuem uma maior complexidade (WMC) se comparadas as demais classes do sistema.*

As classes compartilhadas de casos de uso que não possuem similaridade realizam diferentes atividades, e portanto, devem possuir uma maior complexidade(WMC) perante ao sistema, de acordo com o acúmulo de responsabilidade.

Hipótese 5 (H5). *Em casos de uso similares, as classes compartilhadas que os implementam possuem um baixo acoplamento(CBO) se comparadas as demais classes do sistema.*

As classes compartilhadas de casos de uso similares realizam atividades semelhantes e, portanto, devem utilizar algumas classes em comum para resolver os problemas pertinentes a classe.

Hipótese 6 (H6). *Em casos de uso não similares, as classes compartilhadas que os implementam possuem um alto acoplamento(CBO) se comparadas as demais classes do sistema.*

As classes compartilhadas de casos de uso que não possuem similaridade devem resolver diferentes requisitos, e

portanto, devem possuir um alto acoplamento com as demais classes do sistema, já que, para resolver estes requisitos, várias outras classes do sistema também devem ser necessárias.

Hipótese 7 (H7). *A coesão(LCOM) de classes compartilhadas por casos de uso não similares é menor do que a coesão de classes compartilhadas por casos de uso similares.*

Classes que resolvem requisitos de casos de uso não similares devem realizar diferentes tarefas. Já classes que são utilizadas por casos de uso similares devem possuir uma coesão maior, já que resolvem problemas similares.

3.2 Estudo de caso

3.3 Casos de uso

O diagrama de casos de uso do aplicativo estudado por este artigo está exibido na Tabela 3.

Table 3. Lista de casos de uso

UserCaseID	Descrição
uc1	Lançar Dados da Banca
uc2	Visualizar TCC Postados
uc3	Disponibilizar Visualização do TCC no Site do Uniaraxá
uc4	Manter título do TCC
uc5	Impressão da ata de defesa
uc6	Impressão da lista de presença
uc7	Impressão da lista de alterações/correções
uc8	Lançar nota(status) do TCC
uc9	Escolha dos alunos que farão TCC
uc10	Postar TCC
uc11	Incluir título do TCC
uc12	Visualizar nota(status) do TCC
uc13	Lançamento do Status TCC para a disciplina
uc14	Relatório Gráfico por curso: alunos aprovados vs todos
uc15	Relatório de alunos aprovados/reprovados por curso
uc16	Lançamento da disciplina de TCC
uc17	Visualização do título do TCC

3.4 Aplicativo

O aplicativo utilizado pelo presente estudo de caso é um software comercial, desenvolvido em três meses, utilizando a metodologia Scrum [19]. O grupo de desenvolvedores foi composto por dois analistas, com seis e dois anos de experiência. Durante a construção do software, alguns artefatos foram gerados, tais como:

- diagrama de casos de uso;
- documento de requisitos definido pela metodologia RUP;
- mapeamento do processo utilizando a notação BPMN;

As classes utilizadas no estudo de caso, presentes na Tabela 4, foram escolhidas de acordo com o seguinte requisito: as classes deveriam estar relacionadas aos casos de uso. Com isso, num total de 34 classes, 30 foram selecionadas. As demais classes são de suporte ao sistema, tais como classe de conexão com o banco de dados, de geração de arquivos em PDF e configuração de procedimentos de execução em banco de dados.

Foi feito um levantamento das métricas CBO, LCOM e WMC da suite de Chidamber e Kemerer [10] sobre as classes, conforme tabela 4. O motivo da escolha destas métricas está relacionado ao conjunto de hipóteses do artigo, descritos na Seção 3.1.

Table 4. Lista de métricas das classes envolvidas no estudo de caso

Variable	N	Mean	SD	Min	1Q	2Q	3Q	Max
LOC	34	101,5	112,7	14	30	58	127,3	596
WMC	34	23,5	15,4	4	12,3	19	28,5	72
CBO	34	5,4	5,5	0	0	4	11,3	20
LCOM	34	16,1	13,6	0	7	11	25,5	64

Após o levantamento das classes e dos casos de uso, foi feita uma associação entre os casos de uso às classes. Esta associação foi feita com base no conhecimento de especialistas que pertencem a instituição onde o software utilizado no estudo de caso foi desenvolvido e participaram ativamente da construção deste software. Para realizar esta associação, os especialistas separaram o caso de uso por características e procuraram no código-fonte as classes e métodos que resolveram os requisitos. As ameaças a esta atividade estão descritas na Seção 6.1

3.5 Coleta e armazenamento de dados

Para facilitar a coleta e armazenamento dos dados do estudo de caso, foi criado um banco de dados para manipular as informações necessárias para a realização do artigo. A Figura 3 mostra o mapeamento objeto-relacional das tabelas utilizadas na construção da base de dados.

O diagrama entidade-relacionamento (DER) foi obtido de informações extraídas do diagrama de casos de uso (*Actors* e *UserCases*). A tabela *UserCasesCombination* foi construída a partir dos dados levantados na Seção 3.3. A tabela *UserCases-Classes* foi obtida após a avaliação dos especialistas sobre qual(is) classe(s) pertenciam a qual(is) casos de uso. E, por fim, a tabela *Classes* recebeu os dados das classes obtidos por meio de um software de análise estática das métricas de CK.

Com as informações necessárias para realizar o estudo de caso já inseridas no banco de dados, foi possível manipular os dados de acordo com as hipóteses propostas neste

Table 5. Relaçªo entre casos de uso e classes

UserCaseID	ClassID
uc1	cl02
	cl06
	cl07
uc2	cl02
	cl06
	cl07
	cl12
	cl13
	cl14
uc3	cl12
	cl13
	cl14
uc4	cl02
	cl06
	cl07
uc5	cl02
	cl06
	cl07
	cl18
	cl19
	cl20
	cl21
	cl22
	cl23
uc6	cl02
	cl06
	cl07
	cl18
	cl19
	cl20
uc7	cl02
	cl06
	cl07
	cl18
	cl19
	cl20
uc8	cl02
	cl06
	cl07
	cl32
	cl33
	cl34
uc9	cl32
	cl33
	cl34
uc10	cl02
	cl06
	cl07
	cl12
	cl13
	cl14
uc11	cl02
	cl06
	cl07
uc12	cl32
	cl33
	cl34
uc13	cl27
	cl28
	cl29
uc14	cl02
	cl06
	cl07
	cl32
	cl33
	cl34
uc15	cl02
	cl06
	cl07
	cl32

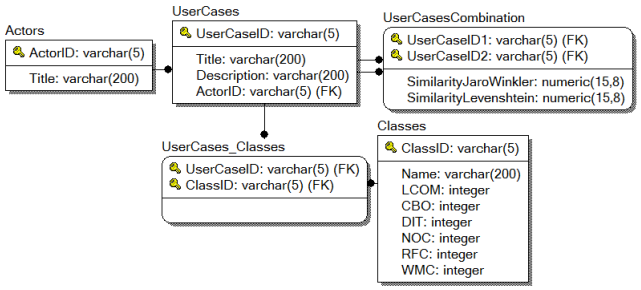

Figure 2. DER utilizado para a realizaçªo do estudo de caso

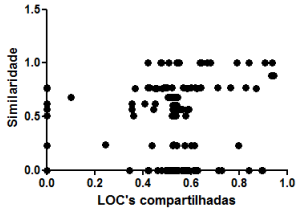

Figure 3. Similaridade entre casos de uso vs *LOC's* compartilhadas

trabalho.

4 Resultados

Para obter os resultados, foi utilizado o método estatístico *Mann-Whitney U* para analisar se os valores de uma métrica de CK de um grupo de dados são superiores aos valores de um outro grupo de dados.

Hipótese 1 (H1). *Em casos de uso similares, as classes que os implementam tem maior percentual de linhas de código (LOC - Lines of Code) compartilhadas.*

Table 6. Correlaçªo entre similaridade e *LOC's* compartilhadas

Number of XY Pairs	192
Spearman r	0,1891
95% confidence interval	0.04460 to 0.3259
P value (two-tailed)	0,0086
P value summary	**
Exact or approximate P value?	Gaussian Approximation
Is the correlation significant? (alpha=0.05)	Yes

Hipótese 2 (H2). *Em casos de uso similares, as classes compartilhadas que os implementam possuem baixa coesão se comparadas as demais classes do sistema (LCOM).*

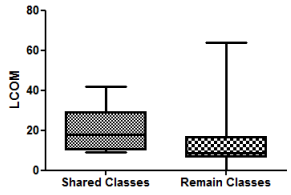

Figure 4. Graphical analysis of H2 results

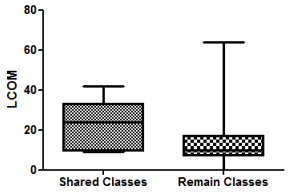

Figure 5. Graphical analysis of H3 results

O Gráfico 4 foi construído a partir das classes que possuem similaridade (conforme Tabela 1) e compartilham classes. O primeiro item do eixo *X* representa o LCOM das classes similares que compartilham classes e o segundo item do eixo *Y* representa o LCOM das demais classes.

A falta de coesão das classes similares é maior do que no restante das classes do sistema. Se existe uma falta de coesão, então a coesão de classes compartilhadas por casos de uso não similares é menor do que a coesão das demais classes do sistema, comprovando nossa hipótese *H2*.

Table 7. Numerical analysis of H2 results

Feature	Value
P value	0,0333
One or tow-tailed P value?	two-tailed
Are medians signif. different? (P < 0.05)	Yes
Mann-Whitney U	**72,50**

Hipótese 3 (H3). *Em casos de uso não similares, as classes compartilhadas que os implementam possuem baixa coesão se comparadas as demais classes do sistema (LCOM).*

Classes compartilhadas por casos de uso não similares são menos coesas (portanto possuem um LCOM maior) do que as demais classes do sistema. A diferença dessa hipótese para a *H2* é que os valores encontrados na H2 devem ser menores do que na hipótese *H3*. Esta comparação é feita em outra hipótese (H7).

Hipótese 4 (H4). *Em casos de uso não similares, as classes compartilhadas que os implementam possuem uma maior complexidade (WMC) se comparadas as demais classes do sistema.*

Table 8. Numerical analysis of H3 results

Feature	Value
P value	0,0635
One or tow-tailed P value?	two-tailed
Are medians signif. different? (P < 0.05)	No
Mann-Whitney U	**64,50**

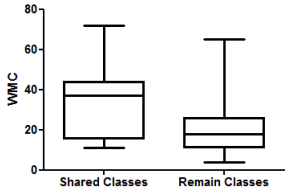

Figure 6. Graphical analysis of H4 results

Quando uma ou mais classes são compartilhadas por casos de uso não similares, estas classes tendem a ser mais complexas (WMC) que as demais, já que resolvem problemas diferentes. O Gráfico 6 mostra esta conclusão.

Table 9. Numerical analysis of H4 results

Feature	Value
P value	0,0385
One or tow-tailed P value?	two-tailed
Are medians signif. different? (P < 0.05)	Yes
Mann-Whitney U	**59,00**

Hipótese 5 (H5). *Em casos de uso similares, as classes compartilhadas que os implementam possuem um baixo acoplamento(CBO) se comparadas as demais classes do sistema.*

Classes compartilhadas de casos de uso similares não possuem um baixo acoplamento se comparada as demais classes do sistema.

Apesar desta hipótese ter sido invalidada, mas alguns pontos devem ser considerados, tais como a arquitetura do sistema e nível de experiência dos engenheiros de software com orientação a objetos.

Hipótese 6 (H6). *Em casos de uso não similares, as classes compartilhadas que os implementam possuem um maior acoplamento(CBO) se comparadas as demais classes do sistema.*

Classes compartilhadas por casos de uso não similares não apresentaram um maior acoplamento se comparadas as demais classes do sistema.

Após os resultados das hipóteses *H6* e *H7*, é possível deduzir que o acoplamento das classes compartilhadas é independente da similaridade entre os casos de uso. Esta de-

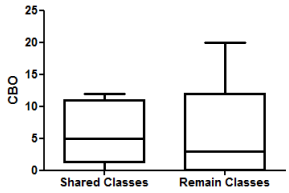

Figure 7. Graphical analysis of H5 results

Table 10. Numerical analysis of H5 results

Feature	Value
P value	0,6180
One or tow-tailed P value?	two-tailed
Are medians signif. different? (P < 0.05)	No
Mann-Whitney U	**118,0**

dução deve ser estudada mais profudamente, para que algumas ameaças como nível de experiência da equipe com orientação a objetos e arquitetura do software não interfiram de maneira relevante no resultado.

Hipótese 7 (H7). *A coesão(LCOM) de classes compartilhadas por casos de uso não similares é menor do que a coesão de classes compartilhadas por casos de uso similares.*

Diferentemente do esperado, a coesão de classes compartilhadas por casos de uso não similares não é significantemente menor do que a coesão de classes compartilhadas por casos de uso não similares.

5 Discussion

6 Ameaças a Validade

6.1 Ameaças internas a validade

Uma ameaça interna a validade deste estudo é a utilização de conhecimento e expertise dos seres humanos que

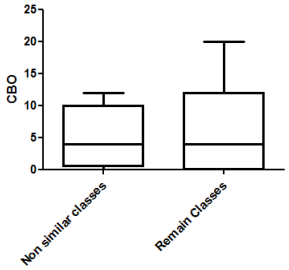

Figure 8. Graphical analysis of H6 results

Table 11. Numerical analysis of H6 results

Feature	Value
P value	0,9362
One or tow-tailed P value?	two-tailed
Are medians signif. different? (P < 0.05)	No
Mann-Whitney U	**110,0**

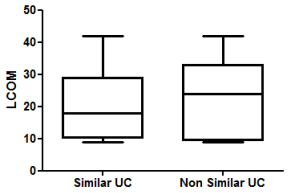

Figure 9. Graphical analysis of H7 results

participaram da avaliação da etapa de relacionar os casos de uso às classes do sistema. Por ser um processo manual, algumas falhas podem ter ocorrido. Dependendo do nível de conhecimento dos analistas envolvidos no sistema do estudo de caso, algumas classes podem ter sido relacionadas a casos de uso de forma equivocada.

Esta ameaça pode ter sido mitigada pela seleção dos participantes do estudo de caso, que foram os reais desenvolvedores do sistema utilizado no estudo de caso. Os parcipantes trabalharam diretamente das etapas de análise, projeto, desenvolvimento e teste do sistema, obtendo um bom nível de conhecimento sobre o aplicativo.

6.2 Ameaças externas a validade

Uma ameaça externa ao presente trabalho é o tamanho do aplicativo escolhido como estudo de caso. Por possuir um tamanho relativamente pequeno, este estudo de caso pode não abranger diversas áreas de aplicabilidade do software. A generalização do presente estudo também pode ter sido prejudicada, uma vez que o software foi construído por uma equipe de uma empresa privada.

Table 12. Numerical analysis of H7 results

Feature	Value
P value	1,0000
One or tow-tailed P value?	two-tailed
Are medians signif. different? (P < 0.05)	No
Mann-Whitney U	**53,50**

7 Conclusões e Trabalhos Futuros

A FAZER

Alguns trabalhos futuros poderão incrementar positivamente o trabalho proposto. Para aprimorar a análise de similaridade entre casos de uso outros métodos devem ser levados em consideração, principalmente métodos que levam em o contexto e possuem dicionário de dados.

Outros tipos de software também devem ser analisados utilizando a metodologia proposta. Estudos com outras linguagens de programação (Java e C++) e diferentes formas de desenvolvimento (*open source*).

References

[1] S. K. Dubey and A. Rana, "Assessment of maintainability metrics for object-oriented software system," *SIGSOFT Softw. Eng. Notes*, vol. 36, no. 5, pp. 1–7, Sep. 2011.

[2] S. R. Pressman, *Software Engineering - A Practitioner's Approach*, 7th ed. McGraw Hill, 2005.

[3] B. Bruegge and A. H. Dutoit, *Object-Oriented Software Engineering*. Boston: Prentice Hall, 2010.

[4] I. Dagan, L. Lee, and F. C. Pereira, "Similarity-based models of word cooccurrence probabilities," *Machine Learning*, vol. 34, pp. 43–69, 1999.

[5] W. W. Cohen, P. Ravikumar, and S. E. Fienberg, "A comparison of string distance metrics for name-matching tasks," 2003, pp. 73–78.

[6] W. E. Winkler, "The state of record linkage and current research problems," Statistical Research Division, U.S. Census Bureau, Tech. Rep., 1999.

[7] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions," and reversals. Technical Report 8, Tech. Rep., 1966.

[8] C. K. Roy, J. R. Cordy, and R. Koschke, "Comparison and evaluation of code clone detection techniques and tools: A qualitative approach," *Science of Computer Programming*, vol. 74, pp. 470–495, feb 2009.

[9] Z. Li, S. Lu, S. Myagmar, Y. Zhou, "Cp-miner: Finding copy-paste and related bugs in large-scale software code," *IEEE Transactions on Software Engineering*, vol. 32, no. 3, pp. 176–192, 2006.

[10] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design," *Software Engineering, IEEE Transactions on*, vol. 20, no. 6, pp. 476–493, jun 1994.

[11] M. Lorenz and J. Kidd, *Object-Oriented Software Metrics: A Practical Guide*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1994.

[12] F. B. Abreu and R. Carapuça, "Object–oriented software engineering: Measuring and controlling the development process," in *Object–Oriented Software Engineering: Measuring and Controlling the Development Process*, McLean, VA, USA, oct 1994, pp. 1–8.

[13] R. Harrison, S. Counsell, and R. V. Nithi, "An Investigation into the Applicability and Validity of Object-Oriented Design Metrics," *Empirical Software Engineering*, vol. 3, no. 3, pp. 255–273, 1998.

[14] R. Subramanyam and M. S. Krishnan, "Empirical analysis of ck metrics for object-oriented design complexity: Implications for software defects," *IEEE Trans. Softw. Eng.*, vol. 29, no. 4, pp. 297–310, Apr. 2003.

[15] R. Shatnawi and W. Li, "The effectiveness of software metrics in identifying error-prone classes in post-release software evolution process," *Journal of Systems and Software*, vol. 81, no. 11, pp. 1868–1882, 2008.

[16] T. G. Nair and R. Selvarani, "Defect proneness estimation and feedback approach for software design quality improvement," *Information and Software Technology*, vol. 54, no. 3, pp. 274–285, 2011.

[17] S. Olbrich, D. S. Cruzes, V. Basili, and N. Zazworka, "The evolution and impact of code smells: A case study of two open source systems," in *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 390–400.

[18] S. Singh and K. S. Kahlon, "Effectiveness of refactoring metrics model to identify smelly and error prone classes in open source software," *SIGSOFT Softw. Eng. Notes*, vol. 37, no. 2, pp. 1–11, Apr. 2012.

[19] K. Schwaber and J. Sutherland. (2011) Guia do scrum. [Online]. Available: