Um Método para Detectar Similaridade entre Sistemas baseado em Decisões de Design: um Estudo Preliminar

Marcos Dósea[1,2], **Cláudio Sant'Anna**[1]

[1]Departamento de Ciência da Computação
Universidade Federal da Bahia (UFBA) – Salvador, BA – Brazil

[2]Departamento de Sistemas de Informação
Universidade Federal de Sergipe – Itabaiana, SE – Brazil

dosea@ufs.br, santanna@dcc.ufba.br

Abstract. *Stable systems are commonly used as design model to develop, maintain, or assure the quality of other systems developed based on similar design decisions. However, finding applications developed with similar design decisions may not be a trivial task. This work proposes a new method to automatically calculate the similarity between systems. The method calculates the similarity using an abstract representation that relies on design roles identified in each system. We conducted an exploratory study applying the proposed method on fifteen real-world systems from three distinct system domains. The proposed method was able to identify a high similarity between systems developed based on similar design decisions.*

Resumo. *Sistemas estáveis são geralmente utilizados como modelo de design para desenvolver, manter ou garantir a qualidade de outras sistemas desenvolvidos com decisões de design similares. Entretanto, encontrar sistemas desenvolvidos com decisões de design similares pode não ser uma tarefa trivial. Este trabalho propõe um método automático para calcular a similaridade entre sistemas a partir de uma representação abstrata baseada nos papéis de design identificados em cada sistema. Foi conduzido um estudo exploratório aplicando o método proposto sobre quinze sistemas do mundo real de três domínios de sistemas distintos. O método proposto foi capaz de identificar alta similaridade entre os sistemas desenvolvidos com decisões de design semelhantes.*

1. Introdução

Sistemas estáveis e com qualidade reconhecida são geralmente utilizados como modelo de *design* para desenvolver, manter e garantir a qualidade de outros sistemas desenvolvidos com decisões de *design* similares. Por exemplo, considerando um sistema desenvolvido com estilo arquitetural em camadas e utilizando o *framework* Hibernate[1], é comum que o desenvolvedor busque soluções de *design* em outros sistemas desenvolvidos com decisões de *design* similares, ou seja, com estilo arquitetural, *frameworks* e bibliotecas similares [Singer et al. 2010]. Identificar sistemas desenvolvidos com decisões de *design* similares também pode auxiliar na criação de *benchmarks* usados para extrair valores limiares de métricas que levam em consideração o contexto de *design* das classes dos sistemas [Dósea et al. 2018].

[1]http://hibernate.org/orm/

A similaridade entre trechos de código, usada principalmente para busca de exemplos código e detecção de clones, é um tópico bastante explorado na literatura [Holmes and Murphy 2005, Roy et al. 2009, Wu et al. 2010]. Entretanto, identificar a similaridade de porções de código de maior granuralidade, como componentes ou o próprio sistema são tópicos ainda pouco explorados. A busca por sistemas desenvolvidos com decisões de *design* similares pode ser ainda mais complexa quando precisa ser realizada em grandes repositórios públicos de sistemas, por exemplo, o GitHub[2], ou mesmo repositórios corporativos com uma diversidade de sistemas disponíveis.

Nesse contexto, este trabalho propõe um método para calcular a similiaridade entre sistemas a partir de uma representação abstrata, criada para cada sistema, que se baseia nos papéis de *design* identificados. O papel de *design* de uma classe é um conjunto de responsabilidades assumidas pela classe para se encaixar em uma comunidade, como, por exemplo, um *framework* ou uma arquitetura corporativa [Wirfs-Brock and McKean 2003]. Em sistemas orientados a objetos, a arquitetura referência normalmente associa os papéis de *design* a uma ou mais classes através de herança, implementação de interfaces ou anotações. Essa arquitetura referência pode utilizar papéis pré-definidos em um *framework*, por exemplo, classes associadas aos papel de *design Action* no Struts. Mas também pode definir seus próprios papéis, por exemplo, criar a interface *IRepository* para associar o papel de *design Persistence*, responsável pela persistência. O método proposto baseia-se na hipótese de que sistemas implementados com os mesmos papéis de *design* devem possuir alta similaridade entre si.

Além de definir um novo método para o cálculo da similaridade, foi conduzido um estudo exploratório aplicando o método proposto para calcular a similaridade entre 15 sistemas de três domínios distintos. Foi considerado que dois sistemas pertencem a um mesmo domínio quando eles executam sobre a mesma plataforma. Os resultados mostraram que o método foi capaz de indentificar a similaridade no *design* entre a maioria dos sistemas do mesmo domínio. Mas o valor pode se tornar mais representativo se outras decisões de *design*, por exemplo, bibliotecas utilizadas, forem consideradas no cálculo.

O restante do artigo está organizado como descrito a seguir: A Seção 2 apresenta os trabalhos relacionados. A Seção 3 apresenta o método proposto para calcular a similaridade entre sistemas. A Seção 4 descreve o configuração do estudo exploratório realizado para avaliar a proposta. A Seção 5 apresenta os resultados e a Seção 6 discute as ameaças à validade. Finalmente, a Seção 7 apresenta a conclusão e os trabalhos futuros.

2. Trabalhos Relacionados

Poucos trabalhos consideram a similaridade entre porções maiores de código-fonte. Tibermacine et al. [Tibermacine et al. 2014] propõem uma abordagem para medir a similaridade de web services através de suas interfaces WSDL. O objetivo é encontrar o melhor substituto para um Web Service quando ele falhar. Al-msie'deen et al. [Al-msie'deen et al. 2013] propõem uma abordagem para minerar *features* através do cálculo da similaridade léxica e estrutural. A abordagem proposta também baseia-se em uma similiaridade estrutural dos papéis de *design* identificados. Entretanto, o método proposto utiliza um nível de abstração mais alto que permite considerar o sistema completo.

[2]https://github.com/

Nagappan et al. [Nagappan et al. 2013] propõem uma técnica para avaliar a cobertura de uma amostra de sistemas para que sejam representativos para realização de um experimento. Propõem uma função de similaridade de sistemas baseada em dimensões numéricas (ex: número de desenvolvedores e linhas de código) e dimensões categóricas (ex: principal linguagem de programação e domínio). O método proposto propõe uma nova dimensão numérica de similaridade baseada nos papéis de *design* identificados, não se limitando a informações genéricas sobre o sistema.

3. Método para Calcular a Similaridade entre Sistemas

O objetivo do método é encontrar sistemas desenvolvidos com decisões de *design* similares. O método calcula a similiaridade entre sistemas a partir de uma representação abstrata, criada para cada sistema, baseada nos papéis de *design* identificados e no percentual de linhas de código associadas a cada papel de *design*. Decisões de *design*, por exemplo, estilo arquitetural, *frameworks* e bibliotecas utilizados, acabam impacatando na representação abstrata obtida para cada sistema. O método é executado em quatro passos:

Passo 1) Identificar os papéis de *design* das classes de cada sistema: Para execução desse passo foi utilizada a ferramenta DesignRoleMiner [Dósea et al. 2018] que propõe uma heurística para identificar os papéis de *design* de cada classe do sistema. A heurística utiliza informações estruturais sobre herança, anotações e implementação de interfaces para associar um papel de *design* a cada classe. Para uniformizar o nome dos papéis de *design*, também é utilizada uma tabela de tokens que ao serem encontrados associam um papel de *design* padronizado. Por exemplo, uma classe que implementa a interface IRepository será associada ao papel de *design Persistence* porque 'Repository' é um dos tokens referentes ao papel de *design* padronizado *Persistence*. Caso esse token não existisse na tabela, a classe seria associada a um papel de *design* com o mesmo nome da interface. A heurística obteve alta precisão quando avaliada com sistemas corporativos.

Passo 2) Desconsiderar papéis de *design* que não descrevem o *design* do sistema: Quando a heurística não consegue identificar o papel de *design* da classe, um papel de *design* genérico, denominado *Undefined*, é associado. Muitos sistemas, independente do domínio, acabam tendo classes associadas ao papel *Undefined*. Esse papel foi desconsiderado porque se ele fosse utilizado na criação da representação abstrata do sistema, adicionaria algum valor de similaridade entre a maioria dos sistemas. Também foi desconsiderado o papel de *design Test*, associado a classes responsáveis pela execução de testes no código, porque essas classes não fazem parte do *design* do sistema.

Passo 3) Criar a representação abstrata do sistema: A representação abstrata proposta descreve cada sistema como um vetor que considera os papéis de *design* identificados no sistema e percentual de código-fonte associado a cada papel. Por exemplo, se a metade do número de linhas de código dos papéis de *design* considerados estão associadas ao papel de *design Service*, então foi considerado o peso de 50% para esse papel no vetor. O mesmo cálculo é realizado para os demais papéis a serem considerados. Definir a representatividade do papel de *design* a partir do percentual de linhas de código associado ao papel permite comparar sistemas com tamanhos distintos. Adicionalmente, pretende-se avaliar se esse percentual é suficiente para representar outras decisões de *design*. Por exemplo, o uso de diferentes bibliotecas e estilo de codificação para implementar o papel de *design Persistence* em sistemas distintos pode influenciar no percentual de código associado a esse papel em cada sistema.

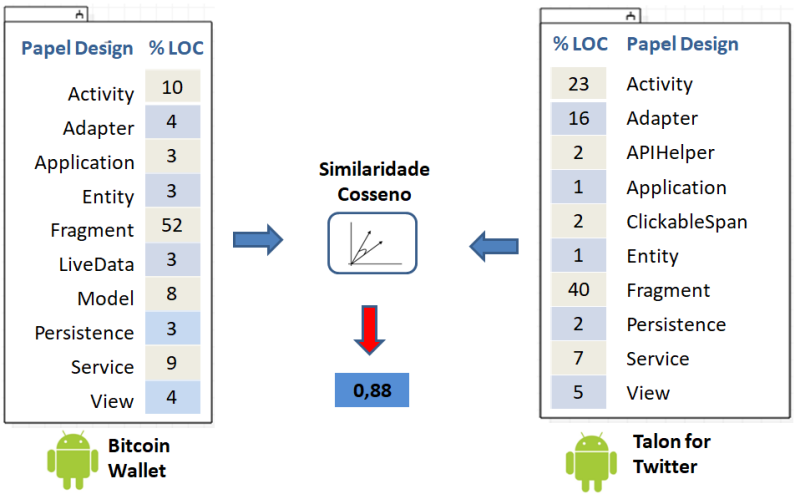

Figura 1. Cálculo da Similaridade entre Sistemas Bitcoin Wallet e Talon for Twitter

Passo 4) Calcular a similaridade das representações abstratas dos sistemas: Foi usada a medida de similaridade cosseno para comparar os dois vetores que representam o *design* de cada sistema. O valor resultante varia de 0 a 1. Quanto mais próximo de 1 mais similares são os sistemas. A medida cosseno foi utilizada por ser uma das mais eficientes funções de similaridade utilizadas em aplicações de processamento de linguagem natural e recuperação da informação [Wilkinson and Hingston 1991]. A função calcula a relevância dos tokens através do cálculo do cosseno entre dois vetores. No método proposto, foi considerado que os tokens são os papéis de *design* e a representatividade (peso) desses papéis foi definida a partir do percentual de linhas de código associado.

A Figura 1 ilustra esse processo com os sistemas Android Bitcoin Wallet e Talon for Twitter, apresentados na Seção 4. Cada sistema é representado de forma abstrata pelos papéis de *design* identificados (passo 1) e o percentual de código-fonte associado a cada papel (passo 3). A representação eliminou os papéis de *design Test* e *Undefined* (Passo 2). A partir dessas representações foi realizado o cálculo da similaridade usando a função cosseno (passo 4). O resultado (0,88) é muito próximo do valor 1, indicando que os dois sistemas possuem alta similaridade nas decisões de *design*.

4. Configuração do Estudo

O objetivo do estudo exploratório foi avaliar se o método para o cálculo de similaridade entre sistemas, proposto na Seção 3, é efetivo para encontrar sistemas desenvolvidos com decisões de *design* similares. Foram formuladas as seguintes questões de pesquisa:

QP1: *Sistemas pertencentes ao mesmo domínio possuem alta similaridade?*

QP2: *A representação abstrata do* design *do sistema proposta foi suficiente para detectar a similaridade entre sistemas?*

A primeira questão de pesquisa visa determinar se sistemas do mesmo domínio sempre irão possuir alta similaridade. Na segunda questão de pesquisa pretende-se verificar se a representação abstrata proposta para descrever o *design* do sistema foi suficiente para detectar sistemas desenvolvidos com decisões de *design* similares.

Sistemas Alvo: Foram selecionados 15 sistemas do repositório GitHub, já utili-

Tabela 1. Características dos Sistemas Alvo

Domínio	Sistema	Descrição	#classes	#métodos	#LOC	#releases	#Papéis de Design	#Data Commit
Aplicações Android	Bitcoin Wallet	Pagamentos em Bitcoin	111	1407	25K	138	9	2016-10-07
	Exoplayer	Media player	429	4532	84K	63	27	2016-10-06
	K-9 Mail	Cliente de Email	488	6913	109K	344	23	2016-04-13
	SMS Backup+	Backup de SMS	118	921	12K	78	10	2016-07-08
	Talon for Twitter	Cliente Twitter	312	3391	83K	1	12	2016-04-03
Plug-ins Eclipse	Activiti-Designer	Editor BPMN	704	3768	74K	19	51	2016-08-18
	AngularJS Eclipse	Editor do AngularJS	106	653	12K	19	14	2016-07-03
	Arduino IDE for Eclipse	IDE para Arduino	124	1180	24K	5	12	2016-04-04
	Drools and jBPM	IDE para Drools e jBPM	796	6936	107K	76	54	2016-12-09
	SonarLint Eclipse	Avalia qualidade do código	200	1178	18K	45	20	2016-12-12
Aplicações Web	BigBlueButton	Aprendizado on-line	1537	11377	176K	20	56	2016-10-18
	OpenMRS	Registro de pacientes	1066	12195	210K	115	37	2016-10-12
	Heritrix	Portal para web-crawling	567	4950	90K	2	34	2016-07-21
	Qalingo	Sistema E-commerce	956	13836	147K	4	18	2016-09-25
	LibrePlan	Gerenciamento de Projetos	1541	23278	282K	32	35	2016-11-09

zados em outros estudos [Dósea et al. 2018]. Os sistemas foram selecionados usando as strings de busca: "Eclipse plugin language:java", "android language:java" e "Web language:java". Adicionalmente, foram excluídos *frameworks* e bibliotecas porque elas raramente compartilham decisões de *design* com outros sistemas.

A Tabela 1 apresenta as características dos quinze sistemas utilizados. As colunas #classes e #métodos exibem o número de classes e métodos de cada sistema. Os sistemas possuem entre 12 e 282 mil linhas de código (coluna #LOC). A coluna #releases mostra o número de releases estáveis disponível em cada sistema. A última coluna mostra a data do *commit* correspondente ao código fonte utilizado no estudo. Finalmente, a coluna #papéis de *design* mostra o número de papéis de *design* identificados em cada sistema. Por exemplo, o sistema Android SMS Backup+ possui 10 papéis de *design* identificados. O website [3] contém detalhes sobre os papéis de *design* identificados em cada sistema.

Procedimentos do Estudo: Para responder a primeira questão de pesquisa (QP1) foi realizado o cálculo da similaridade entre os 15 sistemas estudados, totalizando 105 valores de similaridade. Em seguida, para cada sistema, foi identificado os quatro sistemas mais similares, ou seja, com os maiores valores na medida proposta de similaridade. Finalmente, o código-fonte desses quatro sistemas mais similares foi analisado para justificar o valor de similaridade encontrado. Para responder a segunda questão de pesquisa (QP2) também foi analisado o código-fonte dos quatro sistemas mais similares com o objetivo de identificar se a utilização dos papéis de *design* e percentual de linhas de código associado a cada papel foi suficiente para representar outras decisões de *design* do código, por exemplo, bibliotecas e estilo de codificação utilizado.

5. Resultados e Discussão

A Tabela 2 apresenta os valores de similaridade, usando o método proposto na Seção 3. Cada valor da tabela corresponde ao cálculo da similaridade entre os sistemas que constam na linha e na coluna. Por exemplo, a similaridade entre os sistemas K9 Mail e Bitcoin Wallet é igual a 0,41. Os valores em negrito destacam os quatro sistemas mais similares ao sistema que consta na linha. Por exemplo, os quatro sistemas mais similares ao sistema LibrePlan são OpenMRS, Heritrix, Qalingo e SMS Backup.

QP1: *Sistemas pertencentes ao mesmo domínio possuem alta similaridade?*

A hipótese inicial era que sistemas pertencentes ao mesmo domínio sempre teriam alta similaridade entre si devido a alta probabilidade de usarem os mesmos papéis

[3]https://sites.google.com/site/designdecisions2018/data/rq1

Tabela 2. Similaridade entre Sistemas

	BigBlueButton	OpenMRS	Heritrix	Qalingo	LibrePlan	Bitcoin	K9 Mail	Exoplayer	SMS Backup	Talon	Activiti	AngularJS	Arduino	DroolsJBPM	Sonarlint
BigBlueButton	1,00	0,17	0,23	0,28	0,19	0,08	0,41	0,33	0,28	0,11	0,31	**0,56**	**0,48**	**0,51**	**0,56**
OpenMRS	0,17	1,00	0,10	**0,37**	**0,34**	0,10	0,24	**0,29**	**0,31**	0,08	0,06	0,07	0,07	0,13	0,08
Heritrix	**0,23**	0,10	1,00	**0,48**	**0,28**	0,03	**0,15**	0,10	0,10	0,01	0,06	0,05	0,05	0,07	0,05
Qalingo	**0,28**	**0,37**	**0,48**	1,00	**0,64**	0,11	0,18	0,15	0,18	0,06	0,14	0,01	0,03	0,08	0,03
LibrePlan	0,19	**0,34**	**0,28**	**0,64**	1,00	0,11	0,16	0,15	**0,21**	0,06	0,06	0,02	0,03	0,08	0,03
Bitcoin	0,08	0,10	0,03	0,11	**0,11**	1,00	**0,41**	0,08	**0,16**	**0,88**	0,04	0,07	0,06	0,08	0,08
K9 Mail	**0,41**	0,24	0,15	0,18	0,16	**0,41**	1,00	0,04	**0,65**	**0,53**	0,15	0,33	0,25	0,32	0,33
Exoplayer	**0,33**	0,29	0,1	0,15	0,15	0,08	0,04	1,00	0,28	0,10	0,15	**0,34**	0,24	**0,36**	**0,35**
SMS Backup	**0,28**	**0,31**	0,10	0,18	0,21	0,16	**0,65**	**0,28**	1,00	0,03	0,03	0,03	0,03	0,09	0,04
Talon	**0,11**	0,08	0,01	0,06	0,06	**0,88**	**0,53**	0,10	**0,30**	1,00	0,03	0,08	0,07	0,08	0,10
Acitiviti	0,31	0,06	0,06	0,14	0,06	0,04	0,15	0,15	0,03	0,03	1,00	**0,42**	**0,38**	**0,47**	**0,40**
AngularJS	**0,56**	0,07	0,05	0,01	0,02	0,07	0,33	0,34	0,03	0,08	0,42	1,00	**0,77**	**0,74**	**0,89**
Arduino	**0,48**	0,07	0,05	0,03	0,03	0,06	0,25	0,24	0,03	0,07	0,38	**0,77**	1,00	**0,62**	**0,76**
DroolsJBPM	**0,51**	0,13	0,07	0,07	0,08	0,08	0,32	0,36	0,09	0,08	0,47	**0,74**	**0,62**	1,00	**0,74**
Sonarlint	**0,56**	0,08	0,05	0,01	0,03	0,08	0,33	0,35	0,04	0,10	0,4	**0,89**	**0,76**	**0,74**	1,00

de *design*. Os resultados obtidos e posterior confirmação através da análise do código-fonte, mostraram que nem sempre isso ocorre. Alguns sistemas, pertencentes ao mesmo domínio, tiveram baixo valor de similaridade devido ao uso de papéis de *design* muito diferentes dos geralmente utilizados pelos sistemas deste domínio. Adicionalmente, sistemas que não associam o papel de *design* das classses através de mecanismos de herança, interface ou anotações também geraram uma representação abstrata do *design* do sistema pouco representativa para o cálculo da similaridade.

No domínio dos sistemas Web, o sistema Qalingo obteve maior similaridade com os outros quatro sistemas do mesmo domínio. Já os sistemas Libreplan e Heritrix obtiveram maior similaridade com três sistemas do domínio, e o sistema OpenMRS com apenas dois sistemas do domínio Web. *Service*, *Entity*, *Component* e *Controller* são papéis de *design* geralmente encontrados e representativos em sistemas desse domínio. Entretanto, alguns sistemas obtiveram maior similaridade com sistemas do domínio Android devido a alguns papéis comuns aos dois domínios. Por exemplo, o sistema Android SMS Backup foi o quarto sistema mais similar ao sistema Web LibrePlan. *Persistence* e *Service* foram papéis de *design* comuns aos dois sistemas e que influenciaram no cálculo da similaridade. Entretanto, no sistema LibrePlan, *Persistence* é implementado com o Hibernate, enquanto que, no sistema Android SMS Backup, são usadas bibliotecas do Android para persistência de mensagens SMS. Ou seja, o mesmo papel (*Persistence*) é implementado com decisões de *design* distintas nos dois sistemas. Finalmente o sistema BigBlueButton acabou não tendo alta similaridade nem com os sistemas do domínio Web nem com os do domínio Android. Analisando o código desse sistema, notou-se que 45% de suas classes não tiveram um papel de *design* associado (*Undefined*), criando uma representação abstrata do sistema pouco representativa dentro do domínio Web.

No domínio dos sistemas Android, os sistemas Bitcoin, K9 Mail, Talon for Twitter tiveram três sistemas com mais alta similaridade pertencentes ao mesmo domínio. O sistema SMS Backup obteve alta similaridade com dois sistemas do mesmo domínio. Apenas o sistema Exoplayer não obteve alta similaridade com nenhum sistema Android. Assim como aconteceu com o sistema BigBlueButton, o sistema Exoplayer teve 56% das classes associadas aos papéis de *design Undefined* e *Test*, que não são considerados para representação abstrata do sistema usada no cálculo da similaridade. Adicionalmente, os papéis de *design Activity*, *Fragment*, *Service*, comuns nos sistemas do domínio Android, não existiam ou eram pouco representativos no sistema Exoplayer. Dessa forma, o cálculo da similaridade refletiu o distanciamento do *design* do Exoplayer dos sistemas do seu domínio.

Finalmente, os sistemas do domínio de plug-ins para Eclipse foram os que obtiveram os maiores valores de similaridade entre si. Esses valores são explicados pela menor variação dos papéis de *design* que podem ser utilizados. Em todos os plug-ins são bem representativos papéis de *design* como *Action*, *Dialog*, *Plugin* e *View*. Entretanto, o *design* do sistema Web BigblueButton, discutido anteriormente, acabou tendo maior similaridade com alguns plug-ins do Eclipse devido a identificação de alguns papéis de *design* comuns ao domínio dos plug-ins para Eclipse (ex. *View*, *Action* e *Dialog*). Os papéis de *design* são implementados com bibliotecas distintas nos dois domínios, porém, como o método não considera isso no cálculo, houve alta similaridade entre esses sistemas.

Em resumo, a medida de similaridade proposta conseguiu refletir a similaridade de *design* normalmente encontrada em sistemas do mesmo domínio. Também foi capaz de identificar quando o *design* do sistema se distanciava das decisões de *design* comuns ao sistemas do mesmo domínio, por exemplo, ao usar papéis de *design* diferentes.

QP2: *A representação abstrata do* design *do sistema proposta foi suficiente para detectar a similaridade entre sistemas?*

A representação proposta foi capaz de identificar a similaridade de decisões de *design* entre maioria dos sistemas pertencentes ao mesmo domínio. Entretanto, em alguns casos ocorreu alta similaridade entre sistemas pertencentes a domínios distintos. Por exemplo, o sistema Web OpenMRS obteve alta similaridade com dois sistemas pertencentes ao domínio Android (Exoplayer e SMS Backup) devido ao uso comum de alguns papéis de *design*. Por exemplo, papéis de *design* como *Persistence*, *Service*, *Entity* e *View* são comuns aos dois domínios, mas normalmente são implementados com decisões de *design* distintas em cada domínio. Ou seja, nesses casos o método encontrou alta similaridade entre sistemas que, na realidade, são implementados com outras decisões de *design* distintas, apesar de terem conjuntos de papéis de *design* semelhantes. Trabalhos futuros podem avaliar se considerar outras decisões de *design*, por exemplo, as bibliotecas usadas por cada papel de *design*, trariam melhorias para o cálculo da similaridade.

6. Ameaças à Validade

Nesta seção são discutidas as ameaças à validade do estudo preliminar conduzido e as ações que foram tomadas para minimizá-las.

Validade do Construto: Uma possível ameaça é que os sistemas escolhidos priorizem a similaridade de sistemas do mesmo domínio. Para diminuir esse viés, foram aplicados critérios bem definidos para selecionar os sistemas do repositório GitHub. Outra ameaça foi a escolha da medida de similaridade cosseno. Apesar de ser uma das medidas de similaridade mais utilizadas, esse estudo foi exploratório e trabalhos futuros podem avaliar outras medidas de similaridade.

Validade Interna: A principal ameaça é em relação a ferramenta para identificar os papéis de *design* utilizados para criação da representação abstrata do sistema. Entretanto, a ferramenta já foi utilizada em outros estudos identificando com boa precisão a maioria dos papéis de *design* dos sistemas [Dósea et al. 2018]. Apesar do papel *Undefined* não ser considerado na representação abstrata proposta, futuras melhorias na identificação dos papéis de *design* podem trazer melhorias no cálculo de similaridade.

Validade Externa: Os resultados obtidos são válidos para os quinze sistemas avaliados e três domínios. Não é sugerido a generalização desses resultados para outros

sistemas ou domínios.

7. Conclusão

Neste trabalho foi apresentado um método para detectar a similaridade entre sistemas. Sistemas similares podem ser utilizados como modelo de *design* para o desenvolvimento e manutenção de funcionalidades e para criação de benchmarks usados na extração de valores limiares para métricas. Foi realizada uma avaliação com 15 sistemas de três domínios distintos e os resultados mostraram que o método proposto foi capaz de encontrar a maioria dos sistemas desenvolvidos com decisões de *design* similares.

Como trabalhos futuros, pretende-se evoluir a medida de similaridade proposta para considerar as bibliotecas utilizadas por cada papel de *design*. Percebeu-se alguns casos onde o mesmo papel de *design*, implementado com bibliotecas distintas, aumentou equivocadamente o valor de similaridade. Adicionalmente, pretende-se conduzir estudos que avaliem se a medida proposta pode ser utilizada para detectar o distanciamento do *design* planejado durante o processo de desenvolvimento de uma aplicação.

Agradecimentos. Esse trabalho é apoiado pelo CNPq (projeto 312153/2016-3).

Referências

Al-msie'deen, R., Seriai, A. D., Huchard, M., Urtado, C., and Vauttier, S. (2013). Mining features from the object-oriented source code of software variants by combining lexical and structural similarity. In *Int. Conf. on Inf. Reuse Integration (IRI)*, pages 586–593.

Dósea, M., Sant'Anna, C., and da Silva, B. C. (2018). How do design decisions affect the distribution of software metrics? In *Proceedings of the 26th Conference on Program Comprehension (ICPC)*, pages 74–85.

Holmes, R. and Murphy, G. C. (2005). Using structural context to recommend source code examples. In *Int. Conf. on Software Engineering (ICSE)*, pages 117–125.

Nagappan, M., Zimmermann, T., and Bird, C. (2013). Diversity in software engineering research. In *Foundations of Software Engineering (FSE)*, pages 466–476.

Roy, C. K., Cordy, J. R., and Koschke, R. (2009). Comparison and evaluation of code clone detection techniques and tools: A qualitative approach. *Sci. Comput. Program.*, 74(7):470–495.

Singer, J., Lethbridge, T., Vinson, N., and Anquetil, N. (2010). An examination of software engineering work practices. In *CASCON First Decade High Impact Papers (CASCON)*, pages 174–188.

Tibermacine, O., Tibermacine, C., and Cherif, F. (2014). A Practical Approach to the Measurement of Similarity between WSDL-based Web Services. *Revue des Nouvelles Technologies de l'Information*, pages 3–18.

Wilkinson, R. and Hingston, P. (1991). Using the cosine measure in a neural network for document retrieval. In *Int. ACM SIGIR Conf. on Res. and Dev. in Inf. Retrieval (SIGIR)*, pages 202–210, New York, NY, USA. ACM.

Wirfs-Brock, R. and McKean, A. (2003). *Object design: roles, responsibilities, and collaborations*. Addison-Wesley Professional.

Wu, Y. C., Mar, L. W., and Jiau, H. C. (2010). Codocent: Support api usage with code example and api documentation. In *Int. Conf. Soft. Eng. Adv.(ICSEA)*, pages 135–140.