Um Estudo Empírico sobre Critérios de Seleção de Repositórios GitHub

Laerte Xavier[1], Jailton Coelho[1], Luciana L. Silva[2]

[1]ASERG Group – Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – Minas Gerais – Brasil

[2]ASERG Group – Instituto Federal de Minas Gerais (IFMG)
Belo Horizonte – Minas Gerais – Brasil

{laertexavier,jailtoncoelho}@dcc.ufmg.br, luciana.lourdes.silva@ifmg.edu.br

Abstract. *Mining GitHub repositories is the first step of a plenty of Software Engineering studies. Based on them, researchers assess metrics, discuss behavior, investigate patterns. However, there is no silver bullet when deciding the criteria for selecting a relevant dataset. Recent studies have characterized researches based on GitHub, while others discuss the number of stars as a measure of popularity. In this study, we perform a systematic mapping of papers published in prestigious conferences in software engineering, with the purpose of investigating (i) the criteria applied to mine repositories; (ii) the characteristics of the selected dataset. To accomplish that, we manually analyze 140 papers published in 2017 at ICSE and FSE, both considered top conferences by CSIndex. As a result, we present a list of five main selection criteria:* POPULARITY, ACTIVITY, INDIVIDUAL CRITERIA, BUGS *and* AGE. *Besides, we show that, in the median, studies involving GitHub datasets analyze 25 repositories, in 5 main programming languages.*

Resumo. *Mineração de repositórios GitHub é o primeiro passo de diversos estudos em Engenharia de Software. Baseado neles, pesquisadores avaliam métricas, discutem comportamentos, investigam padrões. Entretanto, não existe nenhuma "bala de prata" que auxilie na decisão sobre quais critérios utilizar para seleção de um dataset relevante. Neste estudo, realiza-se uma revisão sistemática da literatura de artigos publicados em conferências de alto nível de Engenharia de Software, com o propósito de investigar (i) os critérios aplicados na mineração de repositórios; (ii) as características dos datasets selecionados. Para tanto, são analisados, manualmente, 140 artigos publicados nas conferências ICSE e FSE no ano de 2017, ambas conferências consideradas de alto nível pelo CSIndex. Como resultado, é apresentada uma lista de cinco principais critérios de seleção:* POPULARIDADE, ATIVIDADE, CRITÉRIOS PRÓPRIOS, BUGS *e* IDADE. *Além disso, observou-se que, na mediana, trabalhos envolvendo datasets provenientes do GitHub analisam 25 repositórios, em 5 principais linguagens de programação.*

1. Introdução

Minerar repositórios do GitHub[1] representa, atualmente, um importante passo no desenvolvimento de diversos estudos na área de Engenharia de Software. A plataforma, cri-

[1]https://github.com/

ada com o objetivo de auxiliar no desenvolvimento e armazenamento descentralizado de código aberto, tornou-se também um importante ambiente social, onde criadores, mantenedores e contribuidores interagem tanto em atividades de desenvolvimento, quanto de manutenção e evolução de sistemas. Dentre as principais vantagens de utilizar o GitHub como fonte de dados para trabalhos científicos, destacam-se importantes características da própria plataforma, tais como a disponibilização, via API (*Application Programming Interface*), de dados de repositórios, histórico de mudanças e *report* de *bugs*.

Dessa forma, existem diversos trabalhos, em variados contextos, que sustentam suas conclusões em *datasets* minerados da plataforma. Tratam, por exemplo, do ciclo de vida de repositórios [Coelho et al. 2018], refatoração de código [Silva and Valente 2017], ou *breaking changes* de APIs [Brito et al. 2018]. Entretanto, não existe *bala de prata* a respeito de qual seja a melhor estratégia para obtenção de um *dataset* relevante: quantos sistemas devem ser analisados? Qual linguagem deve ser escolhida? Qual o melhor critério de seleção?

Nesse contexto, existem trabalhos que caracterizam a utilização das *features* disponíveis no GitHub como instrumento para o desenvolvimento de pesquisa científica [Penzenstadler et al. 2014]. No entanto, tais estudos não analisam aspectos relacionados à aplicação dessas *features* na definição de *datasets*. Assim, neste trabalho, foram analisados 140 artigos de Engenharia de Software, publicados nas duas conferências mais relevantes da área (ICSE e FSE), com o objetivo de (i) elicitar os critérios utilizados por esses estudos para minerar repositórios no GitHub; (ii) caracterizar os *datasets* obtidos. Especificamente, o objetivo deste trabalho é o de investigar empiricamente às seguintes questões de pesquisa:

- ***RQ#1.*** Como são selecionados os repositórios GitHub estudados por artigos de Engenharia de Software?
- ***RQ#2.*** Quais as principais características dos *datasets* selecionados?

Dessa forma, as principais contribuições deste estudo são (i) fornecer um panorama de trabalhos com *datasets* oriundos do GitHub, publicados em conferências de prestígio de Engenharia de Software; (ii) prover uma lista de *cinco* critérios utilizados para seleção dos *datasets* destes trabalhos: POPULARIDADE, ATIVIDADE, CRITÉRIOS PRÓPRIOS, BUGS e IDADE; e (iii) descrever as principais características destes *datasets*, em termos de número de repositórios estudados e linguagens de programação.

O restante deste artigo está organizado da seguinte forma: a Seção 2 apresenta a metodologia dos estudos realizados. Na Seção 3 são descritos os resultados obtidos. Na Seção 4 são discutidas as ameaças à validade. Por fim, as conclusões obtidas neste estudo são apresentadas na Seção 5.

2. Metodologia

Com o objetivo de responder às questões de pesquisa propostas, foi desenvolvida uma revisão sistemática da literatura de artigos relevantes da Engenharia de Software. Inicialmente, foram selecionadas conferências de prestígio na área (Seção 2.1). Em seguida, foram analisados, manualmente, os artigos aceitos nas trilhas principais dessas conferências (Seção 2.2), para posterior codificação dos mesmos, conforme descrito na Seção 2.3.

2.1. Seleção de Conferências

Como primeiro passo, foram selecionadas conferências relevantes na área de Engenharia de Software. Para tanto, utilizamos a classificação provida pelo CSINDEX [Valente and Paixao 2018], um sistema de indexação da produção científica brasileira em ciência da computação. Neste sistema, as conferências são classificadas em *top* e *near-to-top*, de acordo com o número de artigos submetidos e o h5-index da conferência. O h5-index é uma métrica autoral que busca medir a produtividade e o impacto (em termos de número citações) de publicações científicas. Dessa forma, de acordo com o CSINDEX, são consideradas *top*, as conferências tais que: *(i) o número de submissões seja maior que 180; e (ii) o h5-index da conferência seja maior que 40.*

A Tabela 1 apresenta as *cinco* conferências mais relevantes em Engenharia de Software, de acordo com o CSINDEX, ordenadas pelo critério de julgamento dos desenvolvedores da ferramenta. Dentre elas, duas conferências são classificadas como *top*: ICSE (*International Conference on Software Engineering*), com 415 submissões em 2017 e h5-index igual a 68; e FSE (*Symposium on the Foundations of Software Engineering*), com 295 submissões no mesmo ano e h5-index, 43. Dessa forma, foram selecionadas para estudo estas top-conferências, no ano de 2017 (período de análise da ferramenta).

Tabela 1. A cinco conferências mais relevantes em Engenharia de Software, de acordo com o CSINDEX. Em negrito aquelas classificadas como *top* pelo sistema

Conferência	#Submissões	#Aceitos	Taxa de Aceitação	h5-index
ICSE	**415**	**68**	**16,40%**	**68**
FSE	**295**	**72**	**24,40%**	**43**
ASE	314	65	20,70%	31
MSR	121	37	30,60%	39
ISSTA	118	31	26,30%	31

2.2. Filtragem de Artigos

Após a definição do primeiro critério de seleção, que resultou em 140 artigos aceitos nas trilhas principais (*Research Track*) do ICSE e do FSE em 2017, o segundo passo deste estudo foi a análise manual de cada um destes trabalhos. Dessa forma, com o objetivo de filtrar aqueles que utilizaram *datasets* provenientes do GitHub, foram analisados os *abstracts* e as seções de *study design* de cada um deles. Assim, identificaram-se 19 artigos (13,57% do total analisados) cujos *datasets* foram minerados da plataforma. Destes, 11 artigos foram publicados no FSE e 8 no ICSE, os quais foram todos utilizados nas análises deste trabalho. A Tabela 2 sumariza estes resultados.

Tabela 2. Filtragem de artigos cujos *datasets* foram minerados do GitHub

Conferência	#Aceitos	#GitHub
FSE	72	11 (15,27%)
ICSE	68	8 (11,76%)
Total	**140**	**19 (13,57%)**

2.3. Análise dos Artigos

Por fim, após a seleção dos 19 artigos cujos *datasets* foram minerados do GitHub, a análise de cada um destes estudos se deu de forma manual, com o objetivo de:

- Codificar temas relacionados ao critério utilizado na filtragem dos repositórios analisados (RQ#1);
- Identificar a *linguagem de programação* dos repositórios analisados (RQ#2);
- Identificar a *quantidade* de repositórios selecionados (i.e., o tamanho dos *datasets* estudados) (RQ#2).

3. Resultados

RQ#1. Como são selecionados os repositórios GitHub estudados por artigos de Engenharia de Software?

Com o objetivo de responder a esta questão de pesquisa, os *abstracts* e as seções de *study design* (ou equivalentes) foram analisadas em cada um dos 19 artigos selecionados nesta revisão. Para cada um deles foram atribuídos códigos, ou critérios, que permitissem o agrupamento desses estudos. Assim, *cinco* critérios de seleção principais foram elicitados: POPULARIDADE, ATIVIDADE, CRITÉRIOS PRÓPRIOS, BUGS e IDADE. A Tabela 3 detalha cada um deles, bem como apresenta a quantidade de artigos que os aplica.

Tabela 3. Critérios de seleção de repositórios no GitHub

Critério	Descrição	#Ocorrências
POPULARIDADE	Nº de estrelas ou *forks*	10
ATIVIDADE	Qtde. de *commits* e *pushs*	3
CRITÉRIOS PRÓPRIOS	Particularidades do estudo	3
BUGS	Qtde. de *bugs* reportados	2
IDADE	Data de criação	1

POPULARIDADE. O critério de seleção mais recorrente entre os estudos de Engenharia de Software está relacionado à quantidade de pessoas que interagem com os repositórios. Dessa forma, os autores consideram relevante selecionar para o *dataset* dos seus estudos repositórios que possuem, de alguma forma, impacto dentro da comunidade do GitHub. Neste critério, foram observados três abordagens diferentes: seleção por número de estrelas (5 artigos), por notoriedade na literatura (3 artigos) e por número de *forks* (2 artigos).

Conforme discutido por [Borges et al. 2016], o número de estrelas é de fato utilizado como critério de seleção de repositórios em estudos de Engenharia de Software [Coelho and Valente 2017, Floyd et al. 2017, Long et al. 2017, Bocić and Bultan 2017, Xiong et al. 2017]. Nestes casos, os autores estão interessados em selecionar repositórios que possuem maior reconhecimento dentro da própria plataforma do GitHub, uma vez que as estrelas funcionam como ferramenta de qualificação. Outra abordagem recorrente é o de selecionar repositórios famosos na literatura e hospedados na plataforma (e.g., JUNIT-TEAM/JUNIT4 e FACEBOOK/REACT) [Ma et al. 2017, Khatchadourian and Masuhara 2017, Padhye and Sen 2017]. Nestes casos, os repositórios funcionam como *subjects* de experimentos voltados para avaliação de ferramentas ou novas abordagens propostas pelos autores. Por fim, o número de *forks* também é utilizado como critério de seleção de repositórios populares [Brown et al. 2017, Hellendoorn and Devanbu 2017], uma vez que mensuram a quantidade de desenvolvedores trabalhando paralelamente em diferentes instâncias daquele código.

ATIVIDADE. O segundo critério mais recorrente de seleção de repositórios refere-se às atividades registradas no repositório [Labuschagne et al. 2017, Garbervetsky et al. 2017,

Wittern et al. 2017]. Neste caso, são utilizados como filtro métricas tais como *número de pushs*, *número de commits* e, num caso bastante peculiar, *número de requisições na rede*. Nestes *datasets*, é relevante observar que os estudos possuem o foco em analisar o comportamento dos desenvolvedores de sistemas *open-source*, bem como as interações que acontecem entre eles. Isso requer que os repositórios sejam ativos para fornecer dados relevantes. Assim, não são raras as definições de limiares que sirvam como base para essas filtragens e que delimitem as escolhas dos repositórios em análise (e.g., pelo menos um *commit* no último mês). Por fim, percebe-se também que este critério é comumente associado a outros, como o de POPULARIDADE, por exemplo, provendo um refinamento maior aos *datasets* estudados [Coelho and Valente 2017].

CRITÉRIOS PRÓPRIOS. Em três dos artigos analisados foi observada a adoção de critérios bastante específicos, relacionados à natureza dos experimentos desenvolvidos. Por exemplo, em [Abdalkareem et al. 2017] o *dataset* selecionado respeitou o seguinte critério definido pelos autores:

"We choose non-forked applications with more than 100 commits and more than 2 developers."

Neste caso, além de um critério de ATIVIDADE, relacionado à quantidade de *commits*, os autores decidiram filtrar os repositórios com pelo menos 2 desenvolvedores. O objetivo é obter uma base de contribuidores relevantes para o survey desenvolvido no estudo. Decisões particulares como esta são encontradas também em [Sadeghi et al. 2017, Linares-Vásquez et al. 2017].

RQ#2. Quais as principais características dos *datasets* selecionados?

Nesta questão de pesquisa são analisadas as principais características dos *datasets* encontrados nos 19 artigos revisados. Para tanto, durante a fase de análise (Seção 2.3), observou-se a linguagem de programação dos repositórios selecionados, bem como a quantidade de sistemas que reunem. Destaca-se que tais características são as mais relevantes neste contexto, uma vez que são frequentemente utilizadas para caracterização de *datasets* [Penzenstadler et al. 2014].

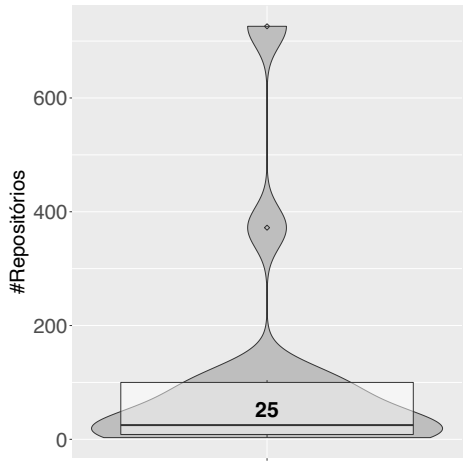

Figura 1. Quantidade de repositórios analisados por artigo

A Figura 1 apresenta a distribuição da quantidade de repositórios selecionados por artigo. Observa-se que o primeiro quartil é igual a 8.5; a mediana, 25; e o terceiro quartil, 100 repositórios. Nesse contexto, destacam-se como *outliers* estudos em larga escala, tais como [Linares-Vásquez et al. 2017, Long et al. 2017], que avaliam 726 e 372 repositórios.

A Tabela 4 descreve as linguagens de programação dos datasets avaliados, a quantidade de artigos que os estudam e o total de repositórios que reunem. Nesse caso,

observa-se que JAVA é a linguagem estudada no maior número de artigos, oito no total, enquanto que JAVASCRIPT é a linguagem com maior número de repositórios avaliados (mais de 38K). Essa discrepância se dá pela existência de um *outlier* entre os *datasets* estudados: em [Abdalkareem et al. 2017] foram avaliados 38.807 repositórios JAVASCRIPT, com o objetivo de entender as motivações dos desenvolvedores de sistemas nesta linguagem utilizarem pacotes triviais da mesma.

Tabela 4. Principais linguagens selecionadas nos *datasets*

Linguagem	#Artigos	#Repositórios
JAVA	8	14.567
JAVASCRIPT	5	38.807
C	2	150
PYTHON	1	7
RAILS	1	25
TODAS	2	107

Além disso, observa-se que os estudos que avaliam repositórios utilizando o critério de POPULARIDADE, filtrados pela notoriedade da literatura [Ma et al. 2017, Khatchadourian and Masuhara 2017, Padhye and Sen 2017], possuem os menores *datasets* observados, com 5, 5 e 7 repositórios, respectivamente. Isto pode ser explicado pelo caráter subjetivo da seleção, umas vez que envolvem conhecimento da literatura e não são minerados automaticamente via API. Tais datasets reúnem repositórios de linguagens tais como JAVA e PYTHON, respectivamente.

Por fim, observa-se que em dois casos [Coelho and Valente 2017, Garbervetsky et al. 2017] não houve pre-definição da linguagem de programação dos repositórios minerados. Nestes casos, os *datasets* foram filtrados utilizando critérios convenientes (POPULARIDADE e ATIVIDADE, respectivamente), sem estabelecer a de linguagem de programação de interesse. Entretanto, a análise mais aprofundada destes artigos revela que, apesar de terem minerado repositórios sem pre-definição das suas linguagens de programação, seus *datasets* reunem, majoritariamente, sistemas escritos em JAVA, JAVASCRIPT e OBJECTIVE-C.

4. Ameaças à Validade

Validade de Conclusão. As conclusões apresentadas neste estudo são ameaçadas pelo tamanho da amostra de artigos selecionados. É importante destacar que uma amostra de apenas 19 artigos pode não ser representativa o suficiente para que se estabeleça uma relação de causa e efeito verdadeira. Entretanto, com o objetivo de diminuir esta ameaça, foram selecionados artigos de alto impacto, publicadas em conferências de prestígios, das mais diversas áreas da Engenharia de Software.

Validade Externa. A ameaça à validade externa diz respeito à capacidade de generalização dos resultados apresentados neste estudo. Como podemos observar, as conclusões obtidas neste trabalho dizem respeito à revisão de um conjunto de 19 artigos, publicados em apenas duas conferências de Engenharia de Software. Dessa forma, a lista de critérios apresentadas não pode ser generalizada para todos os trabalhos da área. Entretanto, para mitigar esta ameaça foram escolhidas conferências com publicações de alto nível, com alta taxa de submissões e alto número de citações.

Validade Interna. Além de ter seus resultados restritos a um pequeno conjunto de artigos, podemos destacar como ameaça à validade interna deste trabalho a avaliação manual realizada sobre os estudos revisados. Apesar de todo rigor de protocolo e execução desprendidos na fase de avaliação, destaca-se que ela foi executada por apenas um autor deste artigo, e sua natureza subjetiva pode representar um risco para as conclusões apresentadas. Entretanto, os resultados obtidos foram confrontados com outros trabalhos da literatura [Penzenstadler et al. 2014, Borges et al. 2016], destacando a conformidade dos critérios apresentados.

5. Conclusão

Neste trabalho foram analisados 19 artigos publicados nas conferências ICSE e FSE no ano de 2017, com a finalidade de investigar em detalhes como se dá o processo de seleção de repositórios do GitHub para estudos em Engenharia de Software. Dessa forma, cinco principais critérios de seleção foram elicitados: POPULARIDADE, ATIVIDADE, CRITÉRIOS PRÓPRIOS, BUGS e IDADE. Além disso, observou-se que os *datasets* selecionados possuem, na mediana, 25 repositórios, escritos nas linguagens JAVA, JAVASCRIPT e C, principalmente. Por fim, observou-se também que a linguagem com maior número de repositórios analisados é JAVASCRIPT.

Referências

Abdalkareem, R., Nourry, O., Wehaibi, S., Mujahid, S., and Shihab, E. (2017). Why do developers use trivial packages? an empirical case study on npm. In *11th Joint Meeting on Foundations of Software Engineering (FSE)*, pages 385–395.

Bocić, I. and Bultan, T. (2017). Symbolic model extraction for web application verification. In *39th International Conference on Software Engineering (ICSE)*, pages 724–734.

Borges, H., Hora, A., and Valente, M. T. (2016). Understanding the factors that impact the popularity of GitHub repositories. In *32nd IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 334–344.

Brito, A., Xavier, L., Hora, A., and Valente, M. T. (2018). Why and how Java developers break APIs. In *25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 255–265.

Brown, D. B., Vaughn, M., Liblit, B., and Reps, T. (2017). The care and feeding of wild-caught mutants. In *11th Joint Meeting on Foundations of Software Engineering (FSE)*, pages 511–522.

Coelho, J. and Valente, M. T. (2017). Why modern open source projects fail. In *11th Joint Meeting on Foundations of Software Engineering (FSE)*, pages 186–196.

Coelho, J., Valente, M. T., Silva, L. L., and Hora, A. (2018). Why we engage in FLOSS: Answers from core developers. In *11th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, pages 1–8.

Floyd, B., Santander, T., and Weimer, W. (2017). Decoding the representation of code in the brain: An fmri study of code review and expertise. In *39th International Conference on Software Engineering (ICSE)*, pages 175–186.

Garbervetsky, D., Zoppi, E., and Livshits, B. (2017). Toward full elasticity in distributed static analysis: The case of callgraph analysis. In *11th Joint Meeting on Foundations of Software Engineering (FSE)*, pages 442–453.

Hellendoorn, V. J. and Devanbu, P. (2017). Are deep neural networks the best choice for modeling source code? In *11th Joint Meeting on Foundations of Software Engineering (FSE)*, pages 763–773.

Khatchadourian, R. and Masuhara, H. (2017). Automated refactoring of legacy Java software to default methods. In *39th International Conference on Software Engineering (ICSE)*, pages 82–93.

Labuschagne, A., Inozemtseva, L., and Holmes, R. (2017). Measuring the cost of regression testing in practice: A study of Java projects using continuous integration. In *11th Joint Meeting on Foundations of Software Engineering (FSE)*, pages 821–830.

Linares-Vásquez, M., Bavota, G., Tufano, M., Moran, K., Penta, M. D., Vendome, C., Bernal-Cárdenas, C., and Poshyvanyk, D. (2017). Enabling mutation testing for Android apps. In *11th Joint Meeting on Foundations of Software Engineering (FSE)*, pages 233–244.

Long, F., Amidon, P., and Rinard, M. (2017). Automatic inference of code transforms for patch generation. In *11th Joint Meeting on Foundations of Software Engineering (FSE)*, pages 727–739.

Ma, W., Chen, L., Zhang, X., Zhou, Y., and Xu, B. (2017). How do developers fix cross-project correlated bugs?: A case study on the GitHub scientific python ecosystem. In *39th International Conference on Software Engineering (ICSE)*, pages 381–392.

Padhye, R. and Sen, K. (2017). Travioli: A dynamic analysis for detecting data-structure traversals. In *39th International Conference on Software Engineering (ICSE)*, pages 473–483.

Penzenstadler, B., Raturi, A., Richardson, D., Calero, C., Femmer, H., and Franch, X. (2014). Systematic mapping study on software engineering for sustainability (se4s). In *18th International Conference on Evaluation and Assessment in Software Engineering (EASE)*, pages 1–14.

Sadeghi, A., Jabbarvand, R., and Malek, S. (2017). Patdroid: Permission-aware gui testing of Android. In *11th Joint Meeting on Foundations of Software Engineering (FSE)*, pages 220–232.

Silva, D. and Valente, M. T. (2017). RefDiff: Detecting refactorings in version histories. In *14th International Conference on Mining Software Repositories (MSR)*, pages 1–11.

Valente, M. T. and Paixao, K. (2018). CSIndexbr: Exploring the Brazilian scientific production in Computer Science. *arXiv*, abs/1807.09266.

Wittern, E., Ying, A. T. T., Zheng, Y., Dolby, J., and Laredo, J. A. (2017). Statically checking web API requests in JavaScript. In *39th International Conference on Software Engineering (ICSE)*, pages 244–254.

Xiong, Y., Wang, J., Yan, R., Zhang, J., Han, S., Huang, G., and Zhang, L. (2017). Precise condition synthesis for program repair. In *39th International Conference on Software Engineering (ICSE)*, pages 416–426.