A characterization study of testing contributors and their contributions in open source projects.

Hugo Henrique Fumero De Souza
hugohfsouza@gmail.com
Universidade Tecnológica Federal do Paraná
Campo Mourão, Paraná, Brasil

Igor Scaliante Wiese
igor@utfpr.edu.br
Universidade Tecnológica Federal do Paraná
Campo Mourão, Paraná, Brasil

Igor Steinmacher
igorfs@utfpr.edu.br
Universidade Tecnológica Federal do Paraná
Campo Mourão, Paraná, Brasil

Reginaldo Ré
reginaldo@utfpr.edu.br
Universidade Tecnológica Federal do Paraná
Campo Mourão, Paraná, Brasil

ABSTRACT

Even though open source projects have some different characteristics from projects in the industry, the commitment of maintainers and contributors to achieve a high level of software quality is constant. Therefore, tests are among the main practices of the communities. Thus, motivating contributors to write new tests and maintain regression tests during testing activities is essential for the project's health. The objective of our work is to characterize testers and their contributions to open source projects as part of a broad study about testers' motivation. Thus, we conducted a study with 3,936 repositories and 7 different and important programming languages (C, C++, C#, Java, Javascript, Python, and Ruby), analyzing a total of 4,409,142 contributions to classify contributing members and their contributions. Our results show that test-only contributors exist, regardless of programming language or project. We conclude that, despite the unfavorable scenario, there are contributors who feel motivated and dedicate their time and effort to contribute to new tests or to the evolution of existing tests.

1 INTRODUÇÃO

Alcançar um alto nível de qualidade é um dos constantes objetivos buscados por projetos OSS (do inglês, *Open Source Software*) para que o software seja considerado confiável [17]. Em verdade, existem peculiaridades em projetos OSS que os diferenciam de projetos tradicionais [27, 34]. Muitos projetos OSS não possuem um processo de software estritamente definido pela comunidade para garantir a condução de atividades que melhoram a qualidade do software, dentre elas o teste de software [2, 6, 29, 33, 39, 45]. Justamente buscando aumentar a qualidade de software por meio de evolução no uso de boas práticas, grande parte das contribuições para projetos OSS hospedados no GitHub[1] atualmente são feitas enfatizando teste de software e revisão de código. Inclusive, a presença de testes em contribuições — frequentemente realizadas por meio de *Pull Requests* (PRs) [24] – é geralmente verificada durante a revisão de código para aceitação da contribuição, e aumenta as chances de que o PR seja aceito [35]. Além disso, algumas comunidades sequer aceitam contribuições que não acompanhem testes [3].

Apesar do teste de software ser reconhecidamente uma atividade essencial para a qualidade de projetos OSS [30, 35], estes apresentam problemas para realizar tal atividade e incorporá-la em seu processo de desenvolvimento [1, 2, 4, 33]. Por exemplo, um estudo feito com base em 50 mil projetos OSS mostrou que apenas 42.66% possuíam testes; e, destes, apenas 10% possuíam mais de 100 casos de testes [31]. Além disso, a qualidade do teste pode variar consideravelmente nos projetos OSS [44].

Não obstante os problemas técnicos relacionados à deficiência ou inadequação dos testes, aspectos motivacionais dos desenvolvedores chamam a atenção. Estudos mostram que desenvolvedores consideram a atividade de teste como sendo uma atividade destrutiva e tediosa durante o desenvolvimento e, até mesmo, uma atividade secundária, executada por colaboradores de "segundo nível" [38]. Especificamente, tais estudos mostram que: o testador percebe que a sua atividade é considerada destrutiva [12, 28]; as atividades de teste podem prejudicar a relação com outros desenvolvedores [14, 15, 38]; testadores auto-declaram que não possuem conhecimento adequado [15]; o teste é uma atividade tediosa [12, 14–16]; ou, que os testadores sofrem pressão para entregas rápidas [15, 25, 38, 47]. Em contrapartida, existem aspectos motivadores que influenciam os profissionais a entrarem e permanecerem como testadores: a vontade de melhorar a qualidade do software, a variabilidade de trabalho, e a oportunidade de aprendizado [12, 15, 25, 28, 38, 47].

Atrair e manter contribuidores é essencial para o sucesso de projetos OSS [22] não apenas do ponto de vista de sustentabilidade do projeto [10], mas também de qualidade [18]. É sabido que os fatores motivacionais influenciam a entrada e a permanência de contribuidores de projetos OSS [32, 42]. Por exemplo, estudos mostram que profissionais que fazem contribuições em projetos OSS iniciam com o objetivo de corrigir algum defeito ou compartilhar com a comunidade [23, 36, 42]. Essas são tipicamente motivações extrínsecas com recompensa direta, ou seja, a vontade de contribuir para receber algo em troca (um software com menos erros ou marketing próprio) [25]. Além dessas, também existem motivações intrínsecas como altruísmo, diversão, sentimento de *kinship* [46].

Porém, para permanecer contribuindo, é necessário que o profissional esteja satisfeito com o projeto. Esta satisfação ocorre quando o contribuidor se mantém feliz, o que afeta positivamente sua saúde mental ao longo da sua participação no projeto [20, 21]. Portanto, contribuidores são atraídos por projetos OSS por meio de motivações iniciais (intrínsecas ou extrínsecas), e eles se mantém por satisfação, que pode estar ligada a novas ou diferentes motivações [23]. Apesar da literatura tratar historicamente o que se refere

[1] https://github.com/

à motivações para contribuir para OSS, não existem até o momento estudos que relacionem motivação às atividades de teste.

O presente trabalho tem por objetivo dar subsídios, por meio da caracterização de contribuidores e suas contribuições, para entender melhor como aspectos motivacionais dos testadores se relacionam com suas contribuições de teste em projetos OSS. E, apesar do cenário posto, em que encontramos na literatura muito mais fatores de desmotivação do que de motivação, ao mesmo tempo em que existe a necessidade tanto de melhorar a qualidade quanto de manter a sustentabilidade dos projetos OSS, encontramos durante nossos estudos um fenômeno contraditório frente ao panorama não propício: colaboradores de projetos que contribuem apenas com testes. Mesmo a atividade de teste muitas vezes ser relegada a um segundo plano e considerada destrutiva, fazendo com que os testadores se sintam como desprestigiados e fiquem desmotivados [38], existem membros que encontram motivação para contribuir apenas com testes. Ainda considerando projetos OSS, mesmo com todos os desmotivadores, os contribuidores empregam seu tempo e esforço para contribuir com novos cenários de teste ou com a evolução de cenários já existentes.

Assim, consideramos que entender as motivações de tais membros contribuidores pode ser bastante benéfico, tanto para projetos OSS, como para projetos na indústria. Especificamente neste trabalho buscamos caracterizar quantitativamente os contribuidores e suas contribuições de teste em projetos OSS. Primeiramente queremos entender os contribuidores com tais características nos diferentes projetos e linguagens de programação, e em qual proporção se dá as contribuições de código de produção e de código de teste. Para nortear nossos trabalhos, queremos especificamente saber:

- QP. Qual a proporção de contribuidores e como se dão suas contribuições de código de produção e código de teste nos diferentes repositórios de maneira geral e segundo diferentes linguagens de programação?

Para responder nossa questão de pesquisa mineramos dados de repositórios do GitHub classificando contribuidores e suas contribuições para quantificar os contribuidores de apenas teste em diferentes linguagens de programação nos diferentes repositórios. Nossos resultados, obtidos quantitativamente e com análises estatísticas, mostram que esse tipo de contribuidor pode ser encontrado nos diversos repositórios. Nossa contribuição, então, está na caracterização detalhada de como se dão as contribuições de apenas teste, no pacote de replicação com o método usado, e no conjunto de dados construído, que fornece subsídios para estudos mais aprofundados em teste de software, especificamente sobre contribuidores de teste e suas motivações.

2 TRABALHOS RELACIONADOS

É necessário entendermos que a palavra motivação é usada como um termo genérico para definir comportamentos que não são necessariamente o desejo ou vontade de um profissional a executar seu trabalho de forma efetiva e com qualidade [21]. De maneira a compreeender melhor como a motivação afeta o trabalho no desenvolvimento de software, alguns estudos buscaram compreender o significado deste conceito dentro da engenharia de software [19]. Podemos compreender a motivação como uma vontade interna de executar seu trabalho de uma forma mais intensa, por longos períodos e com uma ótima qualidade. [19]. O indivíduo está motivado intrinsecamente quando executa determinada atividade apenas pela satisfação de executá-la, e não por alguma recompensa que a atividade pode trazer. Por outro lado, pode-se definir a motivação como sendo extrínseca quando o indivíduo executa uma atividade com o objetivo de alcançar alguma consequência de sua execução [40].

Paralelo a isto, estudos mostram que a atividade de testes no processo de engenharia de software é considerada uma atividade destrutiva, tediosa e uma atividade secundária durante o desenvolvimento [38]. Por conta das definições de motivações e as características das atividades de teste, estudos foram conduzidos para identificar os fatores motivacionais e desmotivações que levam o profissional de teste a executar sua atividade de forma mais adequada, pois segundo Herzberg [26] a motivação tem forte influência no trabalho entregue nas empresas.

Por se tratar de investigações de características humanas, os trabalhos encontrados sobre o tema aplicaram questionários ou conduziram entrevistas para identificar os fatores motivacionais e desmotivacionais das atividades de teste. Os trabalhos abordaram três perspectivas diferentes: testadores de software [15, 41], desenvolvedores [7, 14] e futuros profissionais [11, 12, 47]. Os trabalhos que tratam especificamente de teste de software [14, 15, 41] aplicaram os questionários na indústria, em empresas previamente escolhidas pelos pesquisadores, diferentemente deste artigo, com foco na motivação de contribuidores de projetos OSS.

Além dos diversos fatores identificados na literatura como motivadores, também podem ser encontrados trabalhos que apresentam detalhes da influência das tarefas recorrentes dos desenvolvedores em sua motivação. Por exemplo, a execução de testes manuais desmotiva mais o profissional do que a utilização de testes automatizados [7, 41]. Além disso, escrever testes automáticos faz com que o testador se empenhe em aumentar a cobertura de testes e a executar mais vezes os testes criados afim de identificar novos em novas versões do sistema [15, 41]. O tempo destas execuções também tem influência na motivação do profissional. Para os testadores, o tempo de execução da suíte completa e o tempo de evolução dos cenários criados influencia na motivação de executar suas atividades [7].

Em contrapartida, existem trabalhos que apontam detalhes das desmotivações. Como já mencionado, é comum os testadores verem suas atividades como de segunda classe, ou seja, não essencial ao desenvolvimento [7, 12, 15]. Além disso, existem desenvolvedores que acreditam que há um menor desenvolvimento de carreira e baixo retorno financeiro [7, 12, 15].

Os testadores que se motivam entendem que, ao testar uma aplicação, estão contribuindo com a qualidade do software que será liberado e se sentem desafiados a executarem novos cenários. Além disto, eles entendem que há uma variação maior de trabalho e que requer um conhecimento maior no domínio do nicho explorado pelo software. Quando observamos profissionais que desejam iniciar na carreira de teste, as principais motivações estão relacionadas a facilidade de iniciar na atividade e a facilidade da execução do trabalho [11, 14, 47].

Deak e colegas [15] buscaram identificar quais eram os fatores motivacionais e os desmotivacionais que influenciavam os testadores nas suas atividades e quais ações as empresas poderiam executar afim de incentiva-los. Para identificar os fatores, eles realizaram

uma entrevista guiada por um questionário em 12 empresas, entrevistando 39 pessoas. O nível de reconhecimento, problemas técnicos, pouco tempo para a atividade e atividade tediosa foram os principais fatores que desmotivam os testadores. Já os seguintes fatores foram identificados como motivacionais: atividades desafiadoras, foco na melhoria da qualidade, variedade de trabalho e bom gerenciamento.

A Tabela 1 apresenta um compilado dos fatores motivacionais e desmotivacionais encontrados na literatura. Os fatores motivacionais como "gestão eficaz", "maior reconhecimento financeiro" e "proximidade com usuários finais" são dificilmente aplicáveis no contexto de projetos OSS devido a suas características primárias [9]. Similarmente, alguns fatores desmotivacionais, tais como "Poucos benefícios monetários", "gestão ineficaz", "pouco tempo disponibilizado para a atividade" e "trabalho estressante", também são dificilmente presentes na realidade de projetos OSS.

Tabela 1: Principais aspectos motivadores e desmotivadores de contribuidores testadores elencados na literatura.

Motivações	Desmotivações
Trabalho desafiador	Baixo desenvolvimento da carreira
Foco em melhorar a qualidade	Complexidade
Variedade de trabalho	Atividade tediosa
Ganho de conhecimento	Perda de contato com desenvolvimento
Gestão eficaz	Poucos benefícios monetários
Trabalho criativo	Atividade considerada destrutiva
Conhecimento	Gestão ineficaz
Trabalho bem definido	Problemas técnicos
Maior reconhecimento financeiro	Falta de organização
Proximidade com usuários finais	Pouco tempo disponibilizado para a atividade
	Baixa relação com os desenvolvedores
	Ignorância sobre qualidade de software
	Trabalho estressante

3 ESTUDO DE CARACTERIZAÇÃO DOS CONTRIBUIDORES DE TESTE E SUAS CONTRIBUIÇÕES

Os passos conduzidos no estudo são ilustrados detalhadamente na Figura 1. De maneira geral, existem dois conjuntos de passos: a obtenção e tratamento dos dados, que tem por objetivo produzir um conjunto de dados adequado e útil para vários estudos; e, o estudo de caracterização, cujo objetivo é produzir resultados a respeito de contribuidores de teste e suas contribuições.

3.1 Obtenção e Tratamento dos Dados

A obtenção e tratamento dos dados contém passos que visam a dar suporte ao estudo de caracterização por meio de atividades típicas de mineração de dados. Escolhemos a estratégia de usar PRs (do inglês *Pull Requests*) como átomos de trabalho de contribuição, uma vez que são mais adequados para estudos comportamentais em projetos OSS [8].

3.1.1 Obter Dados dos Repositórios

Escolher Linguagens de Programação.

O primeiro passo de nosso trabalho para obter dados dos repositórios no GitHub foi definir quais linguagens principais dos repositórios utilizar. Para isso, analisou-se a popularidade das linguagens de acordo com a pequisa TIOBE[2], que se baseia na quantidade de resultados obtidos nos mais diversos buscadores considerando a quantidade de profissionais qualificados que usam as linguagens e de cursos disponíveis. Em janeiro de 2022 a pesquisa com desenvolvedores de software mostrou que Python, C, Java, C++, C#, Ruby e JavaScript são as 7 linguagens mais populares em desenvolvimento de software. Por esse motivo, escolhemos tais linguagens como objeto de estudo desta pesquisa.

Minerar Dados dos Repositórios.

Utilizamos então a API GitHub GraphQL [3] para coletar os dados do GitHub. Esta ferramenta possibilitou consultas agrupadas e mais específicas, diminuindo assim a faixa de repositórios com as características requeridas. Por ser mais moderna e flexível nas consultas, em contrapartida a API REST do GitHub [4], a API GraphQL se mostrou eficiente com relação ao tempo de obtenção dos dados dos projetos. É importante ressaltar que a coleta considerou apenas a linguagem principal do repositório—já que cada repositório pode hospedar códigos em múltiplas linguagens. Além disso, selecionamos apenas projetos não arquivados e com mais de 3.000 estrelas, como forma de minimamente considerar projetos com alguma relevância e visibilidade.

Ao total foram selecionados 5.238 repositórios. A Tabela 2 apresenta uma listagem com a quantidade de repositórios coletados para cada linguagem escolhida. Pode-se constatar que grande maioria, 75,45%, é de projetos implementados em Javascript, Python e Java, com destaque para Javascript com 37,08%.

Tabela 2: Listagem de repositórios recuperados por linguagem de programação e segundo os parâmetros selecionados.

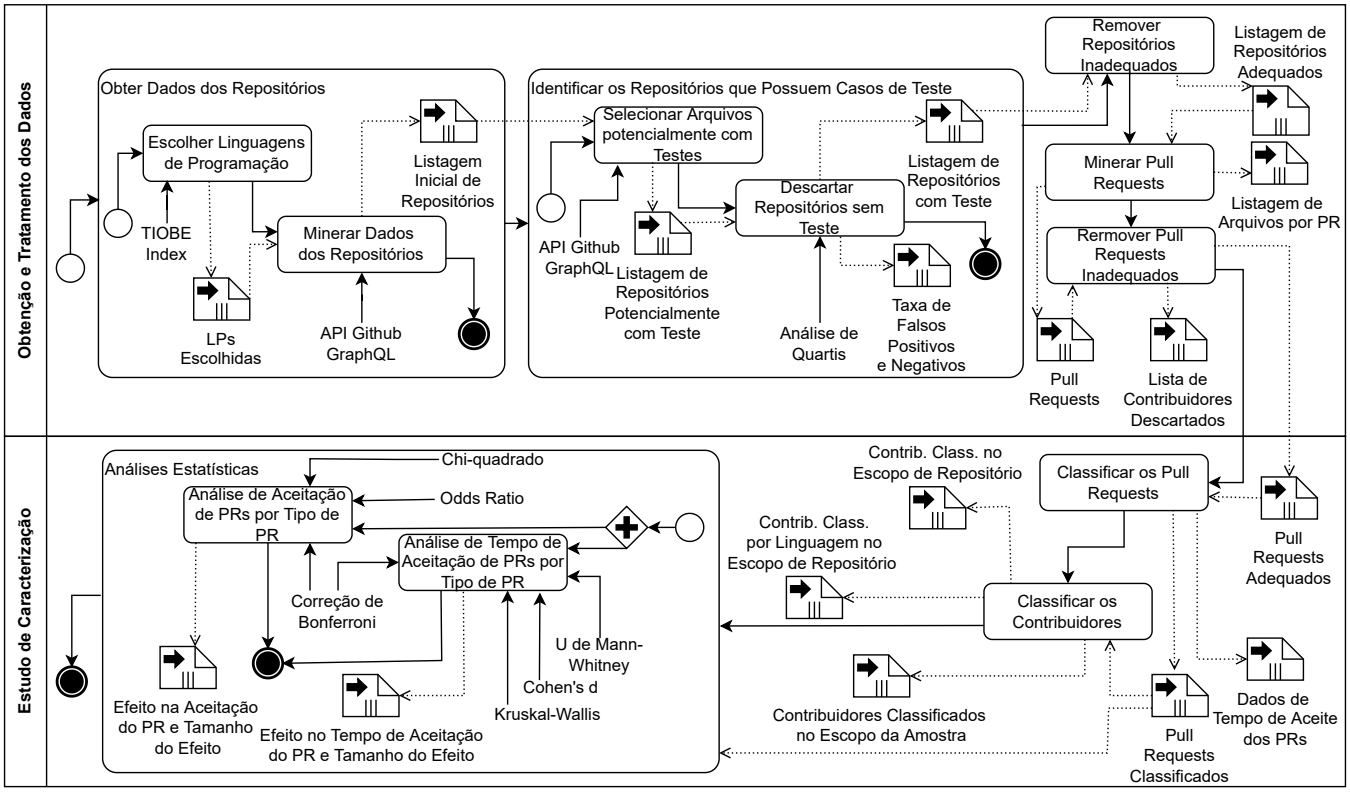
Linguagem	# Repositorios	% Repositorios
Geral	5238	100,00
C	369	7,04
C#	220	4,20
C++	471	8,99
Java	819	15,64
Javascript	1942	37,08
Python	1191	22,74
Ruby	226	4,31

3.1.2 Identificar os Repositórios que Possuem Casos de Teste

Utilizando a lista de repositórios coletados na etapa anterior, conduzimos outra busca usando a API GitHub GraphQL para identificar

[2] https://www.tiobe.com/tiobe-index/
[3] https://docs.github.com/pt/graphql
[4] https://docs.github.com/en/rest

Figura 1: Passos executados na condução do estudo.



a subpalavra "test" no nome dos arquivos ou em seu conteúdo e armazenamos a quantidade de arquivos encontrados. Dentre os 5.238 repositórios, identificamos que 4.047 (76,50%) potencialmente apresentam casos de teste por possuírem mais de uma menção à subpalavra "test".

Descartar Repositórios Sem Teste.

Salvar localmente e analisar o conteúdo de cada arquivo seria bastante custoso, assim optamos por usar a estratégia de buscar a subpalavra "test" diretamente por meio da API GitHub GraphQL. No entanto, para minimizar a quantidade de arquivos escolhidos de maneira incorreta, uma vez que a busca talvez resultasse em muitos falsos positivos, fomos conservadores e fizemos particionadamente para cada linguagem de programação uma análise de quartis no total de menções à subpalavra "test" e descartamos todos os repositórios do primeiro quartil. O resultado pode ser visto na Tabela 4.

Para mensurar os efeitos de nossa estratégia conservadora, fizemos uma verificação manual de falsos negativos presentes no primeiro quartil e falsos positivos nos 2^o, 3^o e 4^o quartis. Na análise de falsos negativos, com nível de confiança de 95% e margem de erro de 5%, obtivemos um percentual de 8,59% (25 projetos de um total de 291) que, apesar de descartados, possuem casos de teste. Na análise de falsos positivos, com nível de confiança de 95% e margem de erro de 5%, obtivemos um percentual de 5,70% (20 projetos de um total de 351) que, apesar de mantidos, não possuem casos de teste. Não fizemos qualquer tratamento para retirada de *outliers*. A Tabela 3 apresenta detalhadamente a distribuição dos falsos positivos e negativos segundo as linguagens de programação.

Tabela 3: Detalhamento da Taxa de Falsos Positivos e Negativos.

Linguagem	# FP	% FP	# FN	% FN
Geral	20	5,70	25	8,59
C	2	7,69	2	8,70
C#	0	0,00	2	22,22
C++	0	0,00	4	16,00
JAVA	4	8,00	5	9,80
Javascript	9	7,38	9	8,33
Python	5	5,43	3	4,92
Ruby	0	0,00	0	0,00

3.1.3 Remover Repositórios Inadequados

Identificamos em um análise inicial a existência de projetos cujo principal objetivo é conter exemplos de código fonte, como por exemplo, repositórios de cursos ou repositórios de estudo. Para que esse fator não influencie nos resultados da pesquisa, utilizando a saída da etapa anterior, fizemos uma análise manual em todos os repositórios que possuíam arquivos com a subparlavra *test*. Identificamos 112 (2,13%) projetos com essas características.Vale ressaltar que muitos projetos com fins educacionais possuíam testes. Como saída deste passo do processo, identificamos um total de 3.936 repositórios que não são educacionais e possuem testes.

Tabela 4: Caracterização dos projetos com relação à presença ou ausência de teste antes e depois da remoção de repositórios inadequados.

		Inadequados		Adequados	
Linguagem	**Categoria**	**# Rep.**	**% Rep.**	**# Rep.**	**% Rep.**
Geral	Com teste	4.048	77,28	3.936	76,79
	Descart.	1.190	22,72	1.190	23,21
C	Com teste	277	75,07	273	74,79
	Descart.	92	24,93	92	25,21
C#	Com teste	165	75,00	163	74,77
	Descart.	55	25,00	55	25,23
C++	Com teste	355	75,37	351	75,16
	Descart.	116	24,63	116	24,84
JAVA	Com teste	633	77,29	600	76,34
	Descart.	186	22,71	186	23,66
Javascript	Com teste	1.508	77,65	1.472	77,23
	Descart.	434	22,35	434	22,77
Python	Com teste	937	78,67	906	78,10
	Descart.	254	21,33	254	21,90
Ruby	Com teste	173	76,55	171	76,34
	Descart.	53	23,45	53	23,66

3.1.4 Minerar *Pull Requests*

Para que fosse possível identificar os contribuidores e a quantidade de contribuições de código de produção e código de teste, recuperamos todos os PRs abertos e fechados. Em seguida, recuperamos todos os arquivos alterados em cada PR. Levantamos um total de 4.682.412 PRs divididos da seguinte maneira: 353.404 PRs em projetos C, 290.970 PRs em projetos C#, 912.095 PRs em projetos C++, 671.875 PRs em projetos JAVA, 1.158.466 PRs em projetos Javascript, 1.032.581 PRs em projetos Python; e, 263.021 em projetos Ruby.

3.1.5 Remover Pull Requests Inadequados

Primeiramente foram excluídos 23.471 PRs que não fazem alterações em arquivos do repositório. Posteriormente, em uma análise inicial, identificamos a existência de PRs criados por *bots*. Para que isso não influenciasse os resultados, fizemos uma busca manual através dos nomes dos usuários. Dentre os 472.321 contribuidores, identificamos analisando os nomes dos usuários e posteriormente pelos PRs 70 contribuidores *bot* nos repositórios. Todos os 249.799 PRs destes usuários e também os usuários foram excluídos. Identificamos também que um usuário chamado *"ghost"* concentrou os PRs de todos os usuários que já foram excluídos no Github. Decidimos mantê-los pois em algum momento da vida dos projetos houve um usuário real solicitando um PR, dessa forma consideramos todos os PRs de usuários reais, mesmo se o usuário não existir mais. Ao final, identificamos 4.432.613 PRs criados por 472.251 diferentes contribuidores nos repositórios selecionados.

3.2 Resultados de Caracterização

Após a obtenção e tratamento dos dados classificamos os projetos, as contribuições feitas, e os contribuidores: os projetos foram classificados pela presença ou ausência de testes; as contribuições foram classificadas pela presença apenas de testes, ou pela presença apenas de código de produção, ou pela presença de ambos, teste e código de produção; e, similarmente às contribuições, os contribuidores foram classificados pela presença de teste, de código de produção, ou por ambos.

Após a classificação, caracterizamos as contribuições e os contribuidores de maneira unicamente quantitativa, e conduzimos algumas análises estatísticas que ajudam a explicar o comportamento dos contribuidores de teste e suas contribuições. Com isso, construímos um conjunto de dados que pode apoiar estudos qualitativos, do ponto de vista da influência dos fatores motivacionais e desmotivacionais de contribuidores de testes em OSS na qualidade das contribuições de teste.

3.2.1 Classificar os *Pull Requests*

Para classificar os PRs adicionamos três marcadores indicando se o PR possui teste (`hasTest`), possui código (`hasCode`) e/ou outros tipos de contribuição (`hasOthers`). Para isso, executamos um *script* que calcula as seguintes características para cada PR: 1) número de arquivos de testes alterados; 2) o número de arquivos de código alterados; 3) quantidade de linhas adicionadas e removidas; 4) quantidade de outros tipos de arquivos. A atribuição dos marcadores nos PRs seguem as seguintes regras:

> `hasTest`: O PR possui arquivos adicionados ou alterados, cujo conteúdo é código fonte escrito na linguagem principal do repositório (C, C++, C#, ...), e tais arquivos possuem a palavra "test" em seu nome ou possui menções à palavra *test* em seu conteúdo;
> `hasCode`: O PR possui arquivos adicionados, alterados ou excluídos, cujo conteúdo é código fonte escrito na linguagem principal do repositório (C, C++, C#, ...), e tais arquivos não possuem a palavra "test" nem existem menções à palavra *test* em seu conteúdo;
> `hasOthers`: O PR possui arquivos adicionados, alterados ou excluídos, cujo conteúdo é outro qualquer que não código fonte escrito na linguagem principal do repositório (C, C++, C#, ...);

De tal maneira, pudemos classificar os PRs em AT (Apenas Teste), AC (Apenas Código) e CT (Código e Teste), usando as seguintes regras:

- PR AT: (`hasTest && !(hasCode || hasOthers)`)
- PR AC: (`(hasCode || hasOthers) && !hasTest`)
- PR CT: (`(hasCode || hasOthers) && hasTest`)

Novamente, fomos bastante conservadores, e tentamos classificar um PR como sendo AT de maneira que qualquer outra contribuição que não for estritamente de teste seja atribuída a AC ou CT. Por exemplo, uma combinação pouco usual mas que pudemos constatar que existe, é quando um PR tem arquivos de teste alterados e também arquivos que não são código fonte (documentação) da linguagem principal do repositório. Neste caso específico, o PR é classificado como CT.

Além de classificar os PRs em AT, AC e CT, classificamos os PRs em aceitos e não aceitos, o que nos possibilita comparar se existe diferença na aceitação dos PRs AT, AC e CT. Do ponto de vista de aceitação, cujos detalhes da classificação executada são mostrados na Tabela 5, pode-se observar que 75,88% dos PRs AT são aceitos, frente a 72,32% dos PRs CT e 73,38% de PRs AC.

Considerando todos os PRs, aqueles classificados como AT (112.969) representam somente 2.56% do total de PRs dos repositórios, enquanto que todos PRs que possuem teste, ou seja, AT e CT, representam 22,53% do total de PRs dos repositórios. Uma distribuição similar é observada se considerarmos unicamente os PRs aceitos: PRs AT representam 2,65%, enquanto que os PRs que possuem teste, ou seja, AT e CT, representam 22,38% do total de PRs aceitos dos repositórios.

Tabela 5: Caracterização dos PRs com relação ao tipo de contribuição – AT, AC e CT – e aceite. Resultados em relação ao total de projetos sem considerar linguagens e, também, considerando cada uma das linguagens.

		Pull Requests		
Language	**Categ.**	**# Total**	**# accepted**	**% accepted**
Geral	AC	3.415.626	2.506.450	73,38
	AT	112.969	85.722	75,88
	CT	880.547	636.846	72,32
C	AC	319.142	224.379	70,31
	AT	3.861	2.710	70,19
	CT	26.185	17.254	65,89
C#	AC	223.802	184.688	82,52
	AT	3.987	3.551	89,06
	CT	30.775	25.465	82,75
C++	AC	805.645	593.614	73,68
	AT	9.207	6.521	70,83
	CT	73.824	50.982	69,06
JAVA	AC	389.620	277.889	71,32
	AT	30.374	22.514	74,12
	CT	221.507	152.593	68,89
Javascript	AC	804.408	553.350	68,79
	AT	21.257	13.833	65,08
	CT	197.998	139.864	70,64
Python	AC	666.690	521.387	78,21
	AT	37.891	31.696	83,65
	CT	293.458	227.013	77,36
Ruby	AC	206.319	151.143	73,26
	AT	6.392	4.897	76,61
	CT	36.800	23.675	64,33

Ainda, para caracterizar melhor os PRs, refinamos os resultados apenas dos PRs aceitos, mostrando a média e a mediana do tempo de aceite nas categorias AC, AT, CT. Os resultados detalhados são apresentados na Tabela 6.

Tabela 6: Caracterização dos PRs aceitos com relação ao tempo de aceite em horas – AT, AC, CT. Resultados em relação ao total de projetos e também considerando cada uma das linguagens de programação.

	Média em horas			Mediana em horas		
Linguagem	**PR AC**	**PR AT**	**PR CT**	**PR AC**	**PR AT**	**PR CT**
Geral	202,18	119,66	302,34	14,84	8,32	34,76
C	224,65	171,31	452,97	17,82	16,12	71,56
C#	182,93	77,31	207,96	18,83	8,39	37,69
C++	192,66	178,60	307,00	17,18	12,43	41,82
JAVA	154,49	97,16	219,53	10,45	8,67	25,07
Javascript	253,08	139,88	293,05	14,97	7,51	29,57
Python	169,61	108,40	335,07	11,55	8,00	42,54
Ruby	243,51	162,41	558,88	12,80	4,39	29,81

3.2.2 Classificar os Contribuidores

Classificamos os contribuidores em três categorias de acordo com a categoria dos PRs processados no passo anterior: MC AT (Membros Contribuidores de Apenas Teste), são aqueles que contribuíram com PRs que contém apenas código de teste; MC AC (Membros Contribuidores de Apenas Código) são aqueles que contribuíram com PRs que contém código de produção ou outros arquivos do repositório que não são destinados a teste; e MC CT (Membros Contribuidores de Código e Teste) que contribuíram com código e teste e, eventualmente, outros arquivos dentro do repositório, inclusive, por exemplo, um PR AT e um outro PR AC.

Ao todo identificamos 468.007 membros contribuidores nos repositórios selecionados. Destes, 128.868 membros (27,54%) contribuíram em mais de 1 repositório, enquanto a grande maioria dos contribuidores, ou seja 339.139 usuários (72,46%), contribuíram apenas em um repositório. Por isso, além de considerar o tipo de contribuição para classificar os contribuidores como MC AT, MC AC, MC CT, decidimos analisa-los em dois escopos distintos: o primeiro escopo, que chamamos de Escopo de Repositório, considera individualmente cada repositório, de maneira que, por exemplo, um usuário pode ser MC AT em um repositório e MC AC ou MC CT em outros repositórios; o segundo escopo, que chamamos de Escopo da Amostra, considera toda nossa amostra de 3.936 repositórios adequados ao estudo. Com isso, por exemplo, para um contribuidor ser considerado um MC AT, ele deve ter apenas PRs AT em toda a amostra.

A Tabela 7 apresenta a quantidade de repositórios por MC no escopo de repositório. Considerando o escopo de repositório, dos 468.007 MCs, 1,61% (7.524) são MC ATs, enquanto que 37,74% (176.616) são MCs que fazem teste (AT ou CT). Nas três classificações, AC, AT e CT, percebe-se uma concentração de contribuidores em apenas um repositório, respectivamente 75,32%, 97,11% e 82,42%. Considerando os MC AC, apesar da concentração em apenas um repositório, existe um certo espalhamento de contribuidores, em relação à quantidade de repositórios, quando comparados aos MC AT e MC CT. Por sua vez, considerando apenas MC AT, percebe-se que os contribuidores estão mais concentrados em apenas 1 repositório do que os contribuidores AC e CT, mas existem peculiaridades, como por exemplo um contribuidor AT que contribuiu em 55 diferentes repositórios. Porém, é possível notar que tanto os MC ATj quanto os MC CT possuem espalhamento menor do que os MC AC.

Segmentamos a caracterização dos MC AT (do escopo de repositórios) segundo as linguagens de programação, cujos resultados são mostrados na Tabela 8. Vale ressaltar que os dados mostrados nessa tabela (em qualquer uma das colunas) foram conjuntos não disjuntos de contribuidores: um mesmo contribuidor pode ser AT em repositórios C, C++. C#, Java, Javascript, Python ou Ruby. Conforme pode ser observado na tabela com os detalhes para as diferentes linguagens, mantém-se a tendência de apenas uma colaboração nos MC que colaboram com PRs AT. Em algumas linguagens isso é mais facilmente observado, por exemplo, C#, C++, em outras existe um espalhamento maior, como, Java e Python.

Considerando o escopo da amostra, cujos detalhes são mostrados na Tabela 9 dos 468.007 MCs, 0,69% (3.212) são MC ATs, enquanto que 28,55% (133.626) são MCs que fazem teste (AT ou CT). De tal forma, para detalhar as contribuições AT, calculamos os PRs feitos

Tabela 7: Caracterização dos contribuidores segundo o escopo do repositório (% MC >= 0,01). As contribuições (PRs abertas) também são classificadas segundo seu tipo (AT, AC e CT).

Apenas Código			Apenas Teste			Código e Teste		
# MC	% MC	# Rep.	# MC	% MC	# Rep.	# MC	% MC	# Rep.
294.433	75,32	1	6.850	97,11	1	106.082	82,42	1
55.248	14,13	2	147	2,08	2	14.837	11,53	2
19.533	5,00	3	26	0,37	3	4.239	3,29	3
8.730	2,23	4	8	0,11	4	1.648	1,28	4
4.603	1,18	5	6	0,09	5	815	0,63	5
2.569	0,66	6	2	0,03	6	371	0,29	6
1.609	0,41	7	5	0,07	7	245	0,19	7
1.065	0,27	8	1	0,01	8	145	0,11	8
746	0,19	9	2	0,03	9	86	0,07	9
512	0,13	10	1	0,01	11	58	0,05	10
390	0,10	11	1	0,01	13	56	0,04	11
247	0,06	12	1	0,01	14	28	0,02	12
201	0,05	13	1	0,01	19	28	0,02	13
155	0,04	14	1	0,01	27	17	0,01	14
132	0,03	15	1	0,01	28	7	0,01	15
97	0,02	16	1	0,01	55	11	0,01	16
100	0,03	17	–	–	–	–	–	–
66	0,02	18	–	–	–	–	–	–
57	0,01	19	–	–	–	–	–	–
48	0,01	20	–	–	–	–	–	–
35	0,01	21	–	–	–	–	–	–
36	0,01	22	–	–	–	–	–	–
28	0,01	23	–	–	–	–	–	–
30	0,01	24	–	–	–	–	–	–
23	0,01	25	–	–	–	–	–	–

pelos contribuidores. Conforme o esperado, existe uma concentração de contribuidores AT em apenas um PR. Tal concentração diminui gradativamente nos contribuidores AC e CT. Existem, portanto, 2.708 contribuidores AT que contribuem com um único PR (84,31% dos contribuidores).

3.2.3 Análises Estatísticas

Análise de Aceitação de PRs por Tipo de PR.

Executamos o teste chi-quadrado nos PRs para verificar se existe diferença nas aceitações considerando se o PR é AC, AT ou CT. O resultado do valor de p foi menor que 0,05, indicando que há evidências o suficiente para mostrar a não homogeneidade da aceitação dos PRs nas diferentes categorias – existe, portanto, um efeito de associação entre aceitação e o tipo de contribuição. Analisando os valores residuais, pudemos constatar que os PRs CT são os que mais influenciam na não aceitação de um PR e, os PRs AT são os que mais influenciam a aceitação de um PR. Usamos *odds ratio* para medir o tamanho do efeito da associação: os PRs AC possuem uma razão de chance de 1,05 de serem aceitos em relação aos PRs CT; e, os PRs AT possuem uma razão de chance de 1,20 de serem aceitos em relação aos PRs CT. Assim, segundo Cohen [13] o tamanho do efeito é considerado muito pequeno.

Com o intuito de investigar o mesmo questionamento mas para cada linguagem de programação, aplicamos o teste chi-quadrado usando correção de Bonferroni. Para todas as linguagens também existe efeito de associação entre aceitação do PR e o tipo de contribuição. Novamente, usamos *odds ratio* para medir o tamanho do efeito da associação em cada uma das linguagens, conforme pode ser visto na Tabela 10. *Odds ratio* é uma medida de tamanho de feito relativa, ou seja, ela mostra a razão de chances de um fator em relação a um outro fator. No nosso caso, que usamos três fatores, as razões de chance de PRs AC e PRs AT são relativas às razões de chance de PRs CT. Por exemplo, para cada uma chance de um PR CT na linguagem C ser aceito, existe 1,23 chances de um PR AC ser aceito e 1,22 chances de um PR AT ser aceito. Assim, pode-se verificar que o tamanho de efeito é muito pequeno ou pequeno para todas as linguagens de programação.

Observando os resultados das razões de chances pode-se perceber que o efeito de associação entre o tipo de PR e aceitação varia segundo a linguagem de programação, por exemplo: em C, as razões de chance de PRs AC e AT são quase iguais; em C#, as razões de chance de PRs AT são maiores que os de PRs AC; em C++, as razões de chance de PRs AC são maiores que os de PRs AT; e, em Javascript, PRs AC e AT são menores do que a referência, os PRs CT.

Análise de Tempo de Aceitação de PRs por Tipo de PR.

O teste de Kruskal-Wallis para verificar se existe diferença no tempo de aceitação de PRs nas três categorias independentemente da linguagem resultou um valor de p igual a zero, o que nos permite descartar a hipótese nula de que não há diferença no tempo de aceite entre os tipo de PR. O tamanho do efeito foi computado usando Cohen's d, e em todas as combinações de tipos de PR e foi encontrado um tamanho de efeito muito pequeno [13]. Como o valor de p do teste de Kruskal-Wallis foi menor do que 0,05, executamos o teste U de Mann-Whitney par-a-par com os tipos de PR, e o valor p em todas as combinações foi menor que 0,05.

Também verificamos se existe diferença no tempo de aceitação de PRs nas três categorias para cada linguagem de programação. Os detalhes da executação do teste U de Mann-Whitney par-a-par com correção de Bonferroni nos tipos de PR e segundo as linguagens de programação podem ser analisados por meio da Tabela 11. Apenas em um caso não há evidências suficientes para concluir que existe diferença significa, ou seja, valor de p maior que 0,05: na comparação dos tempos de aceitação de PR AC e de PR AT usando C como linguagem de programação. Nos outros casos o valor de p é menor que 0,05. Novamente calculamos o tamanho do efeito usando Cohen's d, e os resultados são em todos os casos muito pequeno ou pequeno [13].

4 DISCUSSÃO DOS RESULTADOS

Os resultados que obtivemos mostram um aumento significativo na quantidade de projetos com algum tipo de teste desde uma pesquisa feita a 8 anos atrás com projetos Java por Kochhar et al. [31], constatou que em uma massa de 50.000 projetos OSS JAVA, dos quais 42,66% continham um ou mais casos de testes. Hoje, com base nos dados levantados, observamos que independentemente da linguagem de programação a porcentagem de projetos com teste é maior, variando entre 75,00% a 78,67%, mesmo considerando a taxa de falsos positivos que foi de 5,70% e falsos negativos de 8,59%. Além disso, vale observar que 22,37% dos PRs aceitos contém testes. Podemos inferir que houve um aumento da cultura do teste em projetos OSS, conforme discute Pham et al. [35]. Tal crescimento de projetos com testes tende a aumentar os padrões de qualidade nos produtos que são disponibilizados de forma open-source.

QP. Qual a proporção de contribuidores e como se dão suas contribuições de código de produção e código de teste nos diferentes repositórios de maneira geral e segundo diferentes linguagens de programação?

Tabela 8: Caracterização dos MC AT segundo o escopo do projeto em todas as Linguagens.

Apenas Teste																				
C			C#			C++			JAVA			Javascript			Python			Ruby		
# MC	% MC	# Rep.	# MC	% MC	# Rep.	# MC	% MC	# Rep.	# MC	% MC	# Rep.	# MC	% MC	# Rep.	# MC	% MC	# Rep.	# MC	% MC	# Rep.
147	97,35	1	99	99	1	443	99,55	1	1.440	96,58	1	2.441	98,03	1	1.746	96,95	1	608	97,59	1
2	1,32	2	1	1	2	2	0,45	2	25	1,68	2	45	1,81	2	37	2,05	2	13	2,09	2
1	0,66	3	–	–	–	–	–	–	9	0,60	3	1	0,04	3	10	0,56	3	2	0,32	3
1	0,66	4	–	–	–	–	–	–	4	0,27	4	1	0,04	5	1	0,06	4	–	–	–
–	–	–	–	–	–	–	–	–	4	0,27	5	1	0,04	8	2	0,11	6	–	–	–
–	–	–	–	–	–	–	–	–	1	0,07	6	1	0,04	17	1	0,06	7	–	–	–
–	–	–	–	–	–	–	–	–	3	0,20	7	–	–	–	1	0,06	8	–	–	–
–	–	–	–	–	–	–	–	–	1	0,07	9	–	–	–	1	0,06	9	–	–	–
–	–	–	–	–	–	–	–	–	1	0,07	11	–	–	–	1	0,06	13	–	–	–
–	–	–	–	–	–	–	–	–	1	0,07	19	–	–	–	1	0,06	37	–	–	–
–	–	–	–	–	–	–	–	–	1	0,07	27	–	–	–	–	–	–	–	–	–
–	–	–	–	–	–	–	–	–	1	0,07	28	–	–	–	–	–	–	–	–	–

Tabela 9: Caracterização dos contribuidores – AT, AC, CT – segundo o escopo da amostra (20 contribuições mais relevantes quando considerados o número de MCs).

Apenas Código			Apenas Teste			Código e Teste		
# MC	% MC	# PR	# MC	% MC	# PR	# MC	% MC	# PR
196.743	58,84	1	2.708	84,31	1	32.182	24,68	1
58.823	17,59	2	348	10,83	2	19.233	14,75	2
26.384	7,89	3	94	2,93	3	12.617	9,67	3
14.443	4,32	4	24	0,75	4	8.841	6,78	4
9.596	2,87	5	8	0,25	5	6.613	5,07	5
6.005	1,80	6	10	0,31	6	5.180	3,97	6
3.987	1,19	7	5	0,16	7	4.061	3,11	7
2.725	0,81	8	4	0,12	8	3.405	2,61	8
2.203	0,66	9	3	0,09	9	2.744	2,10	9
1.681	0,50	10	1	0,03	10	2.294	1,76	10
1.320	0,39	11	2	0,06	11	1.980	1,52	11
1.080	0,32	12	2	0,06	12	1.702	1,31	12
905	0,27	13	1	0,03	14	1.513	1,16	13
736	0,22	14	1	0,03	18	1.288	0,99	14
650	0,19	15	1	0,03	41	1.201	0,92	15
571	0,17	16	–	–	–	1.077	0,83	16
462	0,14	17	–	–	–	934	0,72	17
430	0,13	18	–	–	–	829	0,64	18
335	0,10	19	–	–	–	789	0,60	19
325	0,10	20	–	–	–	734	0,56	20

Tabela 10: Resultado de teste estatístico de associação entre tipos de PRs e aceitação ou não dos PRs e tamanhos de efeito segundo cada linguagem de programação.

		Odds Ratio			Interpretação	
Linguagem	Valor p	AC	AT	CT	Int. AC	Int. AT
C	0,00	1,23	1,22	1,00	M.Peq.	M.Peq.
C#	0,00	0,98	1,70	1,00	M.Peq.	Peq.
C++	0,00	1,25	1,09	1,00	M.Peq.	M.Peq.
JAVA	0,00	1,12	1,29	1,00	M.Peq.	M.Peq.
Javascript	0,00	0,92	0,77	1,00	M.Peq.	M.Peq.
Python	0,00	1,05	1,50	1,00	M.Peq.	Peq.
Ruby	0,00	1,52	1,82	1,00	Peq.	Peq.

Tabela 11: Resultado de teste estatístico entre tipos de PRs e tempo de aceitação segundo cada linguagem de programação: "V.p" é o valor de p; "C.'d" é o valor de Cohen's d; e, "Int. C.'d" é a interpretação de Cohen's d.

	PR AC x PR AT			PR AT x PR CT			PR AC x PR CT		
Ling.	V. p	C.'d	Int. C.'d	V. p	C.'d	Int. C.'d	V. p	C.'d	Int. C.'d
C	0,80	––	––-	0,00	0,21	Peq.	0,00	0,21	Peq.
C#	0,00	0,14	M.Peq.	0,00	0,19	M.Peq.	0,00	0,03	M.Peq.
C++	0,00	0,02	M.Peq.	0,00	0,12	M.Peq.	0,00	0,12	M.Peq.
JAVA	0,00	0,07	M.Peq.	0,00	0,15	M.Peq.	0,00	0,08	M.Peq.
Javascript	0,00	0,10	M.Peq.	0,00	0,13	M.Peq.	0,00	0,03	M.Peq.
Python	0,00	0,07	M.Peq.	0,00	0,19	M.Peq.	0,00	0,16	M.Peq.
Ruby	0,00	0,07	M.Peq.	0,00	0,19	M.Peq.	0,00	0,23	Peq.

Os contribuidores AT podem ser encontrados de maneira consistente entre os diversos repositórios e linguagens de programação. Comparado com as PRs AC e CT, suas contribuições AT são acentuadamente mais concentradas em apenas um repositório e apenas um PR, embora existam casos peculiares.

Considerando como escopo o repositório, dos 468.007 contribuidores, 7.054 contribuidores são AT (1,51%) e 128.710 são CT (27,50%), totalizando 135.764 que contribuem de alguma maneira com testes (29,01%). Considerando o escopo da amostra, dos 468.007 contribuidores, 3.212 são contribuidores AT (0,69%) e 130.414 são CT (27,87%), totalizando 138.976 contribuidores que colaboram com teste (29,70%). Considerando o escopo da amostra, das 4.409.142 PRs, 112.969 são PRs AT (2,56%) e 880.547 são PRs CT (19,97%), totalizando 993.516 PRs que contém teste (22,53%). Nas PRs aceitas, de um total de 3.229.018 de PRs, 85.722 PRs são AT (2,65%) e 636.846 (19,72%) são CT, totalizando 722.568 PRs que contém testes (22,38%).

Proporção de Projetos e PRs por Membro Contribuidor.

Observamos que há uma concentração expressiva de contribuidores que participam em um único projeto OSS e, também, de contribuidores com apenas um único PR. Na literatura existem diversas barreiras para entrar em projetos OSS, tais como problemas na comunicação com os mantenedores, habilidades necessárias, orientações quanto a o que é necessário ser produzido e até mesmo barreiras por questões culturais [43]. A grande maioria de MC ATs (97,11%) fizeram contribuições em apenas um repositório, e com apenas um PR (84,31%). Contribuidores CT também são concentrados em apenas um repositório (82,42%) embora de uma maneira mais branda, mas com um espalhamento maior no número de PRs, apenas 24,68% em com apenas um PR. Por sua vez, contribuidores AC são menos concentrados ainda nos repositórios (75,32%) e ainda com número de contribuições menos espalhadas em diferentes PRs (58,84%) comparado com contribuidores CT.

Além dos problemas para iniciar em um novo projeto, há também barreiras para se continuar contribuindo com um projeto. Os usuários que contribuem uma única vez são chamados de contribuidores episódicos ou casuais [5, 37]. Na maioria dos casos, a comunicação entre os mantenedores e os contribuidores fazem com que haja apenas um envolvimento do usuário em todo o projeto. Além disso, podemos encontrar também a recusa de novas ideias, falta de tempo dos mantenedores e até mesmo elitismo [32]. Somados a estes problemas, estão as desmotivações ligadas aos testes já citadas, fazendo com que haja mais PRs únicos do que contribuições de um mesmo usuário ao longo do tempo.

Considerando PRs abertas *versus* PRs aceitas, como pode ser observado na Tabela 5, há um número expressivo de PRs que não possuem testes sendo aprovadas durante o procedimento de revisão de código. Esse padrão se mantém em todas as linguagens, sendo

que em Javascript o percentual de PRs aceitas é o menor (68,79%). Nos projetos utilizando Python, Java, Ruby e C# os PRs categorizados como AT tiveram uma maior taxa de aprovação. Observando de maneira geral, essa taxa chega a 75,88% de aprovação contra 73,38% (PRs AC) e 72,32% (PRs CT). Essa percepção de diferença na aceitação ou não de PRs com teste foi confirmada pelos resultados das análises estatísticas, no entanto, pelos testes de tamanho de efeito podemos concluir que o tipo de PR influencia de maneira irrelevante, tanto na aceitação ou não de um PR, quanto no tempo de aceitação de um PR.

4.1 Limitações do estudo

Uma limitação importante do estudo é a detecção de repositórios com teste e, posteriormente as PRs que contém algum arquivo com teste. Usamos, como uma decisão de diminuir o esforço, a estratégia de minerar os dados em dois passos por meio da API do Github. Isso porque queríamos analisar uma grande amostra, como foi o caso, de mais de 4 milhões de PRs. Identificar os arquivos com teste em diferentes linguagens de programação que podem usar diferentes *frameworks* de teste com uma amostra desse tamanho em um tempo razoável é um desafio considerável. Por isso reportamos a taxa de falsos positivos e verdadeiros negativos, o que torna o estudo mais transparente. Tal ameaça seria minimizada com técnicas mais custosas, como por exemplo, uma análise qualitativa dos PRs.

Outra limitação do estudo é a detecção e classificação dos contribuidores. Como foi mencionado, existem usuários não humanos. A maioria desses usuários foram removidos em uma análise superficial para minimizar tal efeito e um deles permaneceu no estudo, o usuário "ghost". Como ele possui muitos PRs (25.001) e representa usuários reais, optamos por mantê-lo juntamente com suas PRs. O efeito negativo de tal decisão é que no escopo da amostra ele e seus PRs foram classificamos como CT, o que representa 0,57% das PRs CT.

Também, podemos conjecturar que quanto mais contribuições um contribuidor fizer, e portanto por mais tempo ele permanecer contribuindo, maior é a tendência em ele contribuir não apenas com código de produção, mas também com código de teste. Aceitando tal conjectura, com o tempo, muitos MC ATs e MC AC se tornam contribuidores CT. Então, a proporção de PRs AT, PRs AC e PRs CT varia, fazendo com o tempo seja uma dimensão importante para a classificação dos contribuidores em AT, AC, e CT.

Entender o que motiva tais usuários a fazer mais PRs AT do que AC ou CT traria uma contribuição relevante. Por exemplo, um contribuidor pode seguir o caminho não tradicional de contribuição que é: corrigir defeito encontrado ou propor uma melhoria que deseja por meio de um PR. Tal contribuidor, ao tentar aumentar o número ou a cobertura de testes de um projeto, encontra algum defeito e, pela natureza de projetos OSS, o próprio testador faz a correção e submeta no mesmo PR. Em tal cenário, ele é classificado como contribuidor CT, porém possui características dominantes de um contribuidor AT.

Por fim, usamos o número de PRs como a base do nosso estudo, sem considerar a qualidade dos PRs, tanto de código como de teste. Novamente, uma validação qualitativa dos PRs traria maior confiabilidade aos resultados, com o ônus do custo de tal atividade.

5 CONCLUSÕES

Nosso trabalho buscou caracterizar os contribuidores e suas contribuições a respeito de um fenômeno ligado à motivação no contexto do desenvolvimento de software: contribuidores que se sentem motivados a contribuir apenas com testes em projetos OSS. Consideramos que entender as motivações de tais membros contribuidores pode ser bastante benéfico, tanto para projetos OSS, como para projetos na indústria. Assim, conduzimos um estudo com 3.936 repositórios de projetos OSS e 7 diferentes e importantes linguagens de programação (C, C++, C#, Java, Javascript, Python e Ruby), analisando um total de 4.409.142 PRs, classificando membros contribuidores e suas contribuições. Usamos uma abordagem bastante conservadora para classificar os contribuidores e suas contribuições como apenas teste (quanto presente apenas código de teste), apenas código (quando presente apenas código de produção) e código e teste (quando presente código de produção e código de teste) em duas diferentes dimensões de tamanho: escopo de repositório e escopo da amostra.

Nossos resultados mostram que os contribuidores que fazem PRs de apenas teste existem, independentemente da linguagem de programação. Eles estão concentrados em contribuir em apenas um projeto e apenas um PR. De maneira similar, os outros tipos de contribuição também estão concentrados em apenas um projeto e apenas um PR, mas de maneira menos acentuada.

Contrastando nossos resultados com a literatura, algumas questões podem ser levantadas: porque contribuições AT são ainda mais concentradas em apenas um projeto que contribuições CT porque as barreiras são maiores? porque contribuições AT não possuem um nível maior de aceitação uma vez que os projetos ainda precisam de testes e os PRs de teste são apenas aproximadamente 22% do total de PRs? Mesmo com tais questionamento, podemos concluir que existem contribuidores motivados a contribuir com teste, confirmando a existência do fenômeno aqui estudado de contribuições de apenas teste, independentemente do projeto ou de linguagem de programação. Novos estudos futuros, especialmente qualitativos, devem ser conduzidos para caracterizar as motivações de tais contribuidores.

DISPONIBILIDADE DOS ARTEFATOS

Disponibilizamos o pacote de replicação contendo todos os *scripts* usados para obtenção de dados, tratamento de dados e classificação e análises estatísticas, juntamente com o conjunto de dados em https://zenodo.org/record/5138290.

AGRADECIMENTOS

Agradecemos a UTFPR, CNPQ e Fundação araucária pelo apoio a realização desta pesquisa. CNPq/MCTI/FNDCT (grant #408812/2021-4), e MCTIC/CGI/FAPESP (grant #2021/06662-1)

REFERÊNCIAS

[1] Tamer Abdou, Peter Grogono, and Pankaj Kamthan. 2012. A Conceptual Framework for Open Source Software Test Process, In 2012 IEEE 36th Annual Computer Software and Applications Conference Workshops. *Proceedings - International Computer Software and Applications Conference*, 458–463.

[2] Mark Aberdour. 2007. Achieving Quality in Open-Source Software. *IEEE Software* 24, 1 (2007), 58–64.

[3] Adam Alami, Marisa Leavitt Cohn, and Andrzej Waisowski. 2020. How Do FOSS Communities Decide to Accept Pull Requests?. In *Proceedings of the Evaluation and Assessment in Software Engineering* (Trondheim, Norway) *(EASE '20)*. Association for Computing Machinery, New York, NY, USA, 220–229.

[4] Adam Alami, Yvonne Dittrich, and Andrzej Wasowski. 2018. Influencers of Quality Assurance in an Open Source Community. In *Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering* (Gothenburg, Sweden) *(CHASE '18)*. Association for Computing Machinery, New York, NY, USA, 61–68.

[5] Ann Barcomb, Klaas-Jan Stol, Brian Fitzgerald, and Dirk Riehle. 2022. Managing Episodic Volunteers in Free/Libre/Open Source Software Communities. *IEEE Transactions on Software Engineering* 48, 1 (2022), 260–277.

[6] Adina Barham. 2013. The Emergence of Quality Assurance Practices in Free/Libre Open Source Software: A Case Study. In *Open Source Software: Quality Verification*, Etiel Petrinja, Giancarlo Succi, Nabil El Ioini, and Alberto Sillitti (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 271–276.

[7] Moritz Beller, Georgios Gousios, Annibale Panichella, and Andy Zaidman. 2015. When, how, and why developers (do not) test in their IDEs. *2015 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE 2015 - Proceedings* (2015), 179–190.

[8] Marcus Vinicius Bertoncello, Gustavo Pinto, Igor Scaliante Wiese, and Igor Steinmacher. 2020. Pull Requests or Commits? Which Method Should We Use to Study Contributors' Behavior?. In *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 592–601.

[9] Andrea Capiluppi, Patricia Lago, and Maurizio Morisio. 2003. Characteristics of open source projects. 317– 327.

[10] Andrea Capiluppi and Martin Michlmayr. 2007. From the Cathedral to the Bazaar: An Empirical Study of the Lifecycle of Volunteer Community Projects. In *Open Source Development, Adoption and Innovation*, Joseph Feller, Brian Fitzgerald, Walt Scacchi, and Alberto Sillitti (Eds.). Springer US, Boston, MA, 31–44.

[11] Luiz Fernando Capretz, Daniel Varona, and Arif Raza. 2015. Influence of personality types in software tasks choices. *Computers in Human Behavior* 52 (2015), 373–378.

[12] Luiz Fernando Capretz, Pradeep Waychal, Jingdong Jia, Daniel Varona, and Yadira Lizama. 2019. Studies on the Software Testing Profession. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. 262–263.

[13] Jacob Cohen. 1988. *Statistical power analysis for the behavioral sciences* (2nd edition ed.). Routledge. 567 pages.

[14] Anca Deak. 2014. A comparative study of testers' motivation in traditional and agile software development. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 8892 (2014), 1–16.

[15] Anca Deak, Tor Stålhane, and Guttorm Sindre. 2016. Challenges and strategies for motivating software testing personnel. *Information and Software Technology* 73 (2016), 1–15.

[16] Anca Deak, Tor Stålhane, and Daniela Cruzes. 2013. Factors Influencing the Choice of a Career in Software Testing among Norwegian Students. *IASTED Multiconferences - Proceedings of the IASTED International Conference on Software Engineering, SE 2013* (03 2013).

[17] Vieri Del Bianco, Luigi Lavazza, Sandro Morasca, Davide Taibi, and Davide Tosi. 2010. An investigation of the users' perception of OSS quality. *IFIP Advances in Information and Communication Technology* 319 AICT (2010), 15–28.

[18] Matthieu Foucault, Marc Palyart, Xavier Blanc, Gail C. Murphy, and Jean-Rémy Falleri. 2015. Impact of Developer Turnover on Quality in Open-Source Software. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering* (Bergamo, Italy) *(ESEC/FSE 2015)*. Association for Computing Machinery, New York, NY, USA, 829–841.

[19] A. César C. França and Fabio Q.B. Da Silva. 2009. An empirical study on software engineers motivational factors. *2009 3rd International Symposium on Empirical Software Engineering and Measurement, ESEM 2009* (2009), 405–409.

[20] César França, Fabio QB Da Silva, and Helen Sharp. 2018. Motivation and satisfaction of software engineers. *IEEE Transactions on Software Engineering* 46, 2 (2018), 118–140.

[21] César França, Helen Sharp, and Fabio Silva. 2014. Motivated software engineers are engaged and focused, while satisfied ones are happy. *International Symposium on Empirical Software Engineering and Measurement* (09 2014).

[22] Felipe Fronchetti, Igor Wiese, Gustavo Pinto, and Igor Steinmacher. 2019. What Attracts Newcomers to Onboard on OSS Projects? TL;DR: Popularity. In *Open Source Systems*, Francis Bordeleau, Alberto Sillitti, Paulo Meirelles, and Valentina Lenarduzzi (Eds.). Springer International Publishing, Cham, 91–103.

[23] Marco Gerosa, Igor Wiese, Bianca Trinkenreich, Georg Link, Gregorio Robles, Christoph Treude, Igor Steinmacher, and Anita Sarma. 2021. The Shifting Sands of Motivation: Revisiting What Drives Contributors in Open Source. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. 1046–1058.

[24] Georgios Gousios, Martin Pinzger, and Arie van Deursen. 2014. An Exploratory Study of the Pull-Based Software Development Model. In *Proceedings of the 36th International Conference on Software Engineering* (Hyderabad, India) *(ICSE 2014)*. Association for Computing Machinery, New York, NY, USA, 345–355.

[25] A. Hars and S. Ou. 2001. Working for free? - Motivations of participating in open source projects. *Proceedings of the Hawaii International Conference on System Sciences* 00, c (2001), 163.

[26] Frederick Herzberg, Bernard Mausner, and Barbara Snyderman. 1959. *The motivation to work, 2nd ed.* John Wiley, Oxford, England. xv, 157–xv, 157 pages.

[27] Samoladas I, Gousios G, Spinellis D, and Stamelos I. 2008. The SQO-OSS quality model: measurement based open source software evaluation. Open source development, communities and quality. *The International Federation for Information Processing IFIP* 275 (2008), 237–48.

[28] Tanjila Kanij, Robert Merkel, and John Grundy. 2015. An Empirical Investigation of Personality Traits of Software Testers. In *Proceedings of the Eighth International Workshop on Cooperative and Human Aspects of Software Engineering* (Florence, Italy) *(CHASE '15)*. IEEE Press, 1–7.

[29] Atieh Khanjani and Riza Sulaiman. 2011. The process of quality assurance under open source software development. In *2011 IEEE Symposium on Computers Informatics*. 548–552.

[30] Foutse Khomh, Bram Adams, Tejinder Dhaliwal, and Ying Zou. 2015. Understanding the Impact of Rapid Releases on Software Quality. *Empirical Softw. Engg.* 20, 2 (April 2015), 336–373.

[31] Pavneet Singh Kochhar, Tegawendé F. Bissyandé, David Lo, and Lingxiao Jiang. 2013. Adoption of software testing in open source projects - A preliminary study on 50,000 projects. *Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR* (2013), 353–356. https://doi.org/10.1109/CSMR.2013.48

[32] Amanda Lee, Jeffrey C. Carver, and Amiangshu Bosu. 2017. Understanding the Impressions, Motivations, and Barriers of One Time Code Contributors to FLOSS Projects: A Survey. *Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering, ICSE 2017* (2017), 187–197.

[33] Sandro Morasca, Davide Taibi, and Davide Tosi. 2011. OSS-TMM: Guidelines for Improving the Testing Process of Open Source Software. *Int. J. Open Source Softw. Process.* 3, 2 (April 2011), 1–22.

[34] Anh Nguyen-Duc, Daniela S. Cruzes, and Reidar Conradi. 2015. The impact of global dispersion on coordination, team performance and software quality-A systematic literature review. *Information and Software Technology* 57, 1 (2015), 277–294.

[35] Raphael Pham, Leif Singer, Olga Liskin, Fernando Figueira Filho, and Kurt Schneider. 2013. Creating a shared understanding of testing culture on a social coding site. In *2013 35th International Conference on Software Engineering (ICSE)*. 112–121.

[36] Gustavo Pinto, Igor Steinmacher, and Marco Aurélio Gerosa. 2016. More Common Than You Think: An In-depth Study of Casual Contributors. In *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, Vol. 1. 112–123.

[37] Gustavo Pinto, Igor Steinmacher, and Marco Aurélio Gerosa. 2016. More Common Than You Think: An In-depth Study of Casual Contributors. In *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, Vol. 1. 112–123.

[38] Muh Riansyah. 2015. What motivate software tester ? A Systematic Literature Review. November (2015), 1–5.

[39] Peter Rigby, Brendan Cleary, Frederic Painchaud, Margaret-Anne Storey, and Daniel German. 2012. Contemporary Peer Review in Action: Lessons from Open Source Development. *IEEE Softw.* 29, 6 (Nov. 2012), 56–61.

[40] Richard M. Ryan and Edward L. Deci. 2000. Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions. *Contemporary Educational Psychology* 25, 1 (2000), 54–67.

[41] Ronnie Edson De Souza Santos, Cleyton Vanut Cordeiro De Magalhaes, Jorge Da Silva Correia-Neto, Fabio Queda Bueno Da Silva, Luiz Fernando Capretz, and Rodrigo Souza. 2017. Would You Like to Motivate Software Testers? Ask Them How. *International Symposium on Empirical Software Engineering and Measurement* 2017-Novem (2017), 95–104.

[42] Bianca Shibuya and Tetsuo Tamai. 2009. Understanding the process of participating in open source communities. *Proceedings of the 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development, FLOSS 2009* (2009), 1–6.

[43] Igor Steinmacher, Tayana Uchôa Conte, Marco Aurélio Gerosa, and David F. Redmiles. 2015. Social barriers faced by newcomers placing their first contribution in open source software projects. *CSCW 2015 - Proceedings of the 2015 ACM International Conference on Computer-Supported Cooperative Work and Social Computing* October 2014 (2015), 1379–1392.

[44] Valerio Terragni, Pasquale Salza, and Mauro Pezzè. 2020. Measuring Software Testability Modulo Test Quality. In *Proceedings of the 28th International Conference on Program Comprehension* (Seoul, Republic of Korea) *(ICPC '20)*. Association for Computing Machinery, New York, NY, USA, 241–251.

[45] Christopher Thompson. 2017. Large-Scale Analysis of Modern Code Review Practices and Software Security in Open Source Software. *Proceedings of the 13th International Conference on Predictive Models and Data Analytics in Software Engineering* (2017), 83–92.

[46] Georg Von Krogh, Stefan Haefliger, Sebastian Spaeth, and Martin W. Wallin. 2012. Carrots and Rainbows: Motivation and Social Practice in Open Source Software Development. *MIS Q.* 36, 2 (June 2012), 649–676.

[47] Pradeep Kashinath Waychal and Luiz Fernando Capretz. 2016. Why a testing career is not the first choice of engineers. *ASEE Annual Conference and Exposition, Conference Proceedings* 2016-June, June (2016).