Análise comparativa de linguagens tipadas e não tipadas mais populares

Belle Nerissa A. Elizeu[1], **Ian Asenjo D. Cunha**[1],
Jully Ketely A. Silva[1], **Laura Lourdes C. Rodrigues**[1], **Warley Leandro dos Anjos**[1],
Prof. Gabriel de O. Campos Pacheco[1],**Laerte** [1]

[1]Instituto de Ciências Exatas e Informática – Pontifícia Universidade
Católica de Minas Gerais (PUCMG)
Caixa Postal 30535-901 – Belo Horizonte – MG – Brasil

{belizeu,iadcunha,jkasilva}@sga.pucminas.br

{laura.rodrigues,warley.anjos}@sga.pucminas.br

{laertexavier}@pucminas.br

{gacampacheco}@gmail.com

Abstract. *Professionals such as software architects and developers are assigned the task of choosing the most suitable programming language for a given project. At the time of this decision, factors such as performance, quality, cost, whether the language is strongly typed or weakly typed, must be thoroughly analyzed and doubts arise during this process, so it is essential to be based on studies about this problem. Therefore, this work performs the mining of 800 popular GitHub repositories weighing between typed and untyped languages. From this dataset, metrics related to the quality of development and bug fixes are calculated. Finally, the results demonstrate that the comparison between typed and untyped languages is inexpressive considering these metrics.*
Keywords: *repository mining; typed; GitHub*

Resumo. *Aos profissionais como arquiteto de software e desenvolvedores, é atribuído a tarefa de escolher qual a linguagem de programação mais adequada para determinado projeto. No momento dessa decisão fatores como desempenho, qualidade, custo, e linguagem ser fortemente tipada e fracamente tipada devem ser analisados minuciosamente e surgem dúvidas durante esse processo, por isso é essencial fundamentar-se em estudos a respeito desse problema. Logo, este trabalho realiza a mineração de 800 repositórios populares do GitHub ponderando entre linguagens tipadas e não tipadas das linguagens mais populares: Javascript, Python, Typescript e Java. A partir desse conjunto de dados, são calculadas métricas relativas á qualidade de desenvolvimento e correção de bugs. Por fim, os resultados sugerem que as linguagens tipadas tendem a ter mais issues de tipo bug e maior atividade de desenvolvimento, mas o tempo médio de resolução das issues não demonstra uma vantagem clara em termos de facilidade na correção de erros.*
Palavras-chaves: *mineração de repositórios; tipadas; GitHub*

1. Introdução

Nos últimos anos, tem havido um reconhecimento crescente da importância da linguagem JavaScript (que é fracamente tipada) por várias empresas de tecnologia. Isso pode ser observado pelo lançamento de sistemas de tipos estáticos desenvolvidos por grandes empresas do setor. O Google foi uma das pioneiras nesse sentido, com o lançamento do Closure. Em seguida, a Microsoft lançou o TypeScript e, mais recentemente, o Facebook anunciou o Flow [Zhang et al. 2021]. Essas iniciativas ressaltam a necessidade crescente e a importância da escolha entre o uso de linguagens tipadas ou não tipadas no desenvolvimento de projetos de software.

O debate sobre se linguagens tipadas estaticamente são melhores para a qualidade do software do que as linguagens tipadas dinamicamente tem perdurado por um longo tempo [Bogner and Merkel 2022]. Nesse sentido, a escolha da linguagem de programação pode ter um impacto significativo no desenvolvimento de software. Com a proliferação de diversas linguagens de programação ao longo do tempo, pode ser desafiador escolher a opção mais adequada para uma determinada aplicação, especialmente para os arquitetos de software que são responsáveis pela definição dos artefatos técnicos do projeto.

Em linguagens de programação, um sistema de tipos garante que os programas calculem com valores esperados [Zhang et al. 2021]. Se tratando de algumas características, as linguagens de programação que contém instruções relativas aos tipos de dados, são chamadas de linguagens tipadas ou fortemente tipadas. Estas linguagens exigem que ao criar, seja especificado o tipo de variável (inteiro, cadeia de caracteres, número de ponto flutuante, etc.). Já as linguagens de programação não tipadas ou fracamente tipados são linguagens onde é possível declarar variáveis sem definir o seu tipo. Porém, qual seria a relevância do uso do tipo da linguagem na decisão da arquitetura de um software?

O problema a ser solucionado nesta pesquisa é a relevância na escolha de uma linguagem tipada ou não tipada, com base na qualidade de código, comunidade e eficiência na resolução de *bugs*. A partir desta pesquisa, tem-se um estudo no qual é possível compreender e definir qual o tipo da linguagem é a mais adequada a ser utilizada, aplicando um estudo comparativo nas linguagens JavaScript e Python (não tipada), e TypeScript e Java (tipada).

Tendo em vista essa questão, o objetivo desse trabalho é auxiliar arquitetos e desenvolvedores na tomada de decisão sobre o uso de linguagens tipadas ou não tipadas com uma **análise sobre os principais tópicos relacionados a essa escolha, que são: quais as diferenças em relação a fatores como qualidade de código, a participação e contribuição da comunidade e facilidade na correção de erros e *bugs*.**

Para alcançar este objetivo, foi realizado um estudo que coletou dados de 200 repositórios *open source* cada para as respectivas linguagens, Python, JavaScript, TypeScript e Java, hospedados na plataforma GitHub. Os dados coletados incluíram informações sobre a quantidade de contribuidores, contribuições, total de *issues* relatadas e tempo até o fechamento das mesmas. Essas quatro linguagens foram classificadas como populares pelo relatório GitHub Octoverse de 2022[1], o que evidencia sua ampla adoção e relevância na comunidade de desenvolvimento de software. Após a coleta dos dados foram analisados

[1] https://octoverse.github.com/2022/top-programming-languages

pela ferramenta de análise estática SonarQube[2], com o fim de obter valores dos cálculos das métricas de qualidade de *code smell* e complexidade ciclomática de cada dos projeto de código.

Conclusivamente, os resultados do experimento sugerem que as linguagens tipadas tendem a ter uma quantidade maior de *issues* relacionadas a erros de tipo em comparação com as linguagens não tipadas. No entanto, o tempo médio para resolver essas *issues* é ligeiramente menor nas linguagens tipadas, embora isso não garanta uma facilidade definitiva na correção de erros. Além disso, os resultados indicam que os repositórios de linguagens tipadas populares atraem mais contribuidores e apresentam uma atividade de desenvolvimento mais intensa do que os repositórios de linguagens não tipadas. No entanto, é importante considerar que outros fatores, como a popularidade das linguagens e a natureza dos projetos, também podem influenciar o nível de participação da comunidade.

O restante deste artigo contém quatro seções. A Seção 2 é composta pelos trabalhos relacionados. Na Seção 3 é descrito o desenho da metodologia do experimento proposto. A Seção 4 apresenta os resultados das três questões de pesquisa propostas. A Seção 5 é apresentado as discussões dos resultados. A Seção 6 está as ameaças à validade e por fim a Seção 7 está a conclusão do trabalho.

2. Trabalhos Relacionados

Nessa seção os artigos relacionados envolvem comparações entre linguagens não tipadas e tipadas e análises de qualidade entre os diferentes tipos de linguagem.

O primeiro e principal artigo relacionado é o [Bogner and Merkel 2022] que apresenta um estudo que busca comparar a qualidade softwares em aplicações de JavaScript (não tipada) e TypeScript (tipada), na qual, a base são repositórios no GitHub[3] utilizados como fonte de dados. Como resultado, geralmente, as aplicações em TypeScript apresentam uma superioridade, considerando uma complexidade inferior, menor quantidade de erros e uma superior consistência estrutural. Pode-se constatar que o TypeScript auxilia a melhora da qualidade do software, graças a tipagem estática, diminuindo a quantidade de error e facilidade para realizar manutenção no código. Diante disso, esse artigo realiza uma análise semelhante, considerando que se baseia em uma amostra especificada de repositórios para comparar ambas linguagens.

Outro artigo relacionado é o [Gao et al. 2017] que investiga o impacto ao verificar tipos estáticos para encontrar *bugs* em códigos de JavaScript (não tipada) e consequentemente, evitar a ocorrência desses *bugs*. Pode-se constatar que TypeScript ajuda a melhorar a detecção de *bugs* em bases de codificação de JavaScript, graças a utilização desse tipo de ferramenta estática.

O artigo [Zhang et al. 2021] aborda características para correção de *bugs* em diferentes tipos de linguagens de programação, focando, principalmente, na identificação de padrões e tendências no contexto de resolução de *bugs*. Com esse estudo, é possível identificar diversos dados importantes a respeito de diferentes linguagens, tanto tipadas, como não tipadas, que podem favorecer na redução de *bugs* e aprimorar as técnicas e as

[2]https://www.sonarsource.com/products/sonarqube

[3]https://github.com

melhores práticas atualmente utilizadas em suas respectivas linguagens de programação.

A respeito do artigo [Harlin et al. 2017] pode-se analisar que linguagens de programação estaticamente tipadas possibilitam a redução do tempo de desenvolvimento e fortalece na qualidade do código, consequentemente, reduzindo a quantidade de *bugs*, com o resultado final do artigo de concluir que sistemas do tipo estático permitem corrigir erros de depuração do programa com maior facilidade, apesar da falta de existência quando se trata de um sistema criado do zero.

O artigo [Merkel 2021] compara a qualidade de software de aplicações TypeScript e JavaScript, analisando 604 repositórios do GitHub. Os resultados indicam que as aplicações TypeScript possuem melhor qualidade de código e compreensibilidade. No entanto, não há evidências suficientes de que elas sejam menos propensas a *bugs* ou resolvam *bugs* mais rapidamente. A segurança de tipos no TypeScript contribui para melhor qualidade em três das quatro métricas avaliadas. A escolha do *framework* tem um impacto modesto na qualidade de software.

Por fim, o artigo [Khan et al. 2022] examina o impacto da adoção de verificadores de tipo estático, como o mypy [4], em projetos Python. Com base em 210 projetos no GitHub, descobriu-se que o mypy poderia ter prevenido 15% dos defeitos encontrados. Além disso, não houve diferença significativa na experiência dos desenvolvedores em relação a defeitos relacionados a tipos. A análise também identificou as principais causas de defeitos relacionados a tipos, como redefinição de referências, inicialização de atributos dinâmicos e tratamento incorreto de objetos nulos. Recomenda-se a integração do mypy no fluxo de desenvolvimento para prevenir defeitos e mitigar padrões indesejados.

3. Metodologia

O trabalho proposto é classificado como uma pesquisa quantitativa de natureza descritiva [Wohlin et al. 2012]. Essa característica justifica-se na intenção de compreender como a qualidade de desenvolvimento e a correção de *bugs* interage com as linguagens fortemente tipadas e fracamente tipadas de um projeto através da coleta e análise de dados obtidos pela mineração de repositórios. Portanto, esta é uma pesquisa que realiza o levantamento de dados quantitativos com o intuito de estabelecer uma relação de influência. Para tanto, são avaliadas métricas de qualidade de desenvolvimento, tamanho da participação da comunidade e facilidade de correção de *bugs*. Esta seção apresenta as etapas e tecnologias utilizadas para alcançar os objetivos propostos.

3.1. Coleta de Dados

Os procedimentos adotados por este estudo, são divididos em duas fases principais, a coleta de dados e o cálculo das métricas. A coleta de dados foi realizada através da plataforma *GitHub*. Esses dados incluem informações sobre repositórios *open source* das linguagens Python, Javascript, Typescript e Java em ordem de maior popularidade aferida pela quantidade de estrelas do projeto [Borges and Tulio Valente 2018].

Essa seleção de repositórios a serem analisados se dá a partir da API GraphQL do GitHub. A escolha se justifica por ela apresentar alto desempenho e permitir selecionar apenas campos desejados, tornando a consulta efetiva e direta. Os dados gerados são

[4]https://mypy-lang.org/

armazenados para análise dos resultados. Esse armazenamento foi realizado no formato de arquivos CSV (do inglês, *comma-separated values*).

3.2. Cálculo das Métricas

Após o armazenamento de dados da etapa anterior, foi realizado a análise de código para cada repositório clonado utilizando a ferramenta SonarQube para o cálculo das métricas de qualidade de código, como demonstrado na Figura 1.

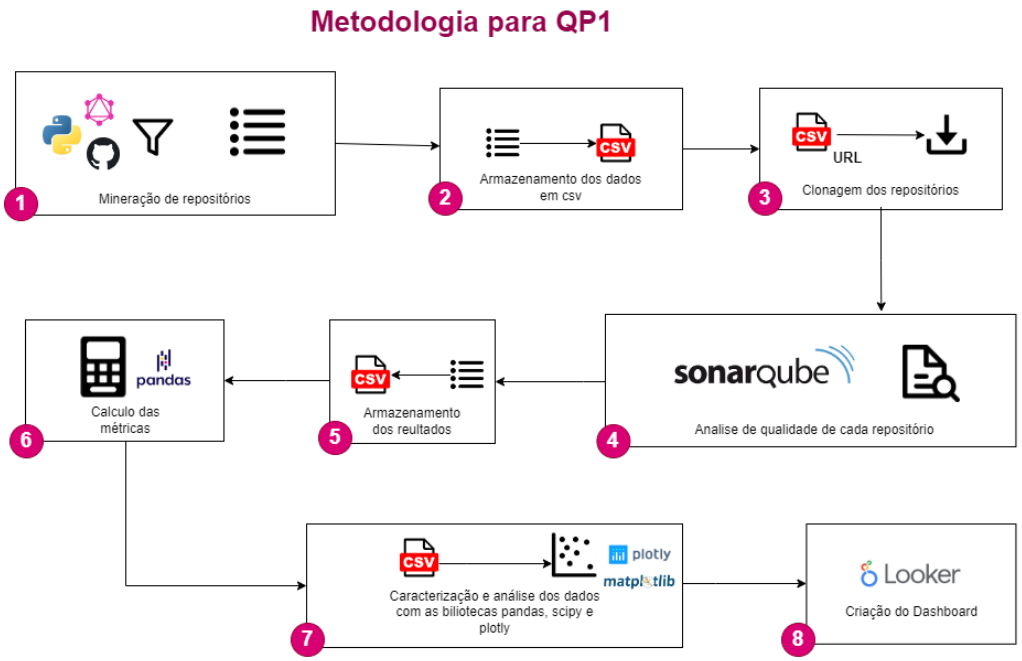

Figura 1. Fluxograma da metodologia da RQ 1

Para realizar o cálculo das métricas de tamanho da participação da comunidade e facilidade de correção de *bugs*, foram utilizados os mesmos procedimentos de coleta de repositórios e armazenamento de dados. Além disso, os dados de quantidade de colaboradores e total de *issues* com o rótulo *"bug"*, *"Type: bug"*, *"error"* e *"fix "* foram sumarizados. O tempo de fechamento dessas*issues* também foi calculado. Essas etapas estão ilustradas na Figura 2.

3.3. Hipóteses e Métricas

Nestas seções serão discorridas as perguntas de pesquisa (RQ, do inglês *Research Question*) elaboradas, suas respectivas métricas e as hipóteses levantadas. As métricas são medidas usadas para monitorar e valiar o desempenho da pesquisa e hipóteses são suposições construídas para guiar a tomada de decisão.

3.3.1. RQ1: Repositórios escritos nas linguagens tipadas possuem qualidade de desenvolvimento de código melhor comparado a não tipados?

A qualidade de desenvolvimento é um fator a se considerar no momento de escolha da linguagem de programação a ser utilizada na aplicação. Sendo assim, para analisar esse cenário foi levantada a RQ 1.

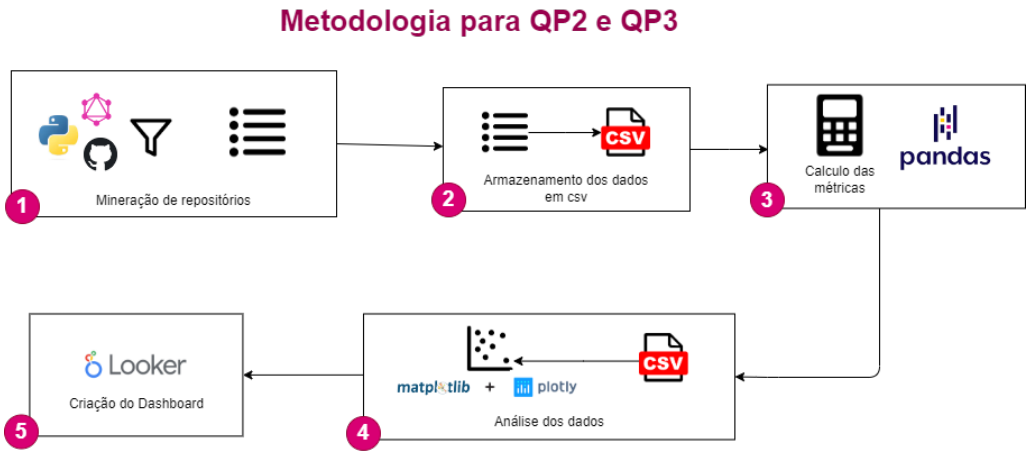

Figura 2. Fluxograma da metodologia da RQ 2 e RQ 3

Para responder essa pergunta, foram definidas duas métricas. A primeira é referente a taxa de *code smells*, o objetivo é calcular a mediana dessa taxa obtidas na análise estática de código-fonte providas pelo SonarQube de cada grupo de repositórios. A segunda métrica é a mediana dos valores da complexidade ciclomática também obtidas nessa análise estática de código-fonte do SonarQube.

A hipótese nula para a primeira métrica é de que os repositórios com linguagens fortemente tipadas apresentam taxa de *code smells* menores que os repositórios com linguagens fracamente tipadas. A hipótese alternativa é de que os repositórios com linguagens fortemente tipadas apresentam taxa de *code smells* maiores que os repositórios com linguagens fracamente tipadas.

A hipótese nula para a segunda métrica é de que os repositórios com linguagens fortemente tipadas apresentam valores da complexidade ciclomática menores que os repositórios com linguagens fracamente tipadas. A hipótese alternativa é de que os repositórios com linguagens fortemente tipadas apresentam valores da complexidade ciclomática maiores que os repositórios com linguagens fracamente tipadas.

3.3.2. RQ2: Como os repositórios de linguagens tipadas e não tipadas diferem em termos de tamanho da participação da comunidade?

Outro fator que pode contribuir no momento de de escolha da linguagem de programação a ser utilizada é o tamanho da participação da comunidade nos repositórios. Para analisar esse cenário, foi levantada a RQ2.

Para responder essa pergunta, foram definidas duas métricas. A primeira é referente a mediana da quantidade de contribuidores dos repositórios de cada grupo de linguagens. A segunda métrica é a mediana da quantidade de contribuições calculada pelo total de *commits* por total de *pull resquest* dos repositórios de cada grupo de linguagens.

A hipótese nula para a primeira métrica é de que a quantidade de contribuidores repositórios com linguagens fortemente tipadas são superiores que a quantidade de contribuidores de repositórios com linguagens fracamente tipadas. A hipótese nula para a primeira métrica é de que a quantidade de contribuidores repositórios com linguagens

fortemente tipadas são inferiores que a quantidade de contribuidores de repositórios com linguagens fracamente tipadas.

A hipótese nula para a segunda métrica é de que a quantidade de contribuições calculada pelo total de *commits* por total de *pull resquest* dos repositórios com linguagens fortemente tipadas são maiores que a quantidade de contribuidores de repositórios com linguagens fracamente tipadas. A hipótese alternativa é de que é de que quantidade de contribuições calculada pelo total de *commits* por total de *pull resquest* dos repositórios com linguagens fortemente tipadas são menores que a quantidade de contribuidores de repositórios com linguagens fracamente tipadas.

3.3.3. RQ3: Como os repositórios de linguagens tipadas e não tipadas diferem em termos de facilidade de correção de *bugs*?

O último fator discutido neste estudo que pode contribuir no momento de de escolha da linguagem de programação a ser utilizada é a facilidade de correção de *bugs*, ou seja, como é o desempenho. Esse cenário foi analisado após o levantamento da RQ3.

Para responder essa pergunta, foram definidas duas métricas. A primeira é referente a mediana do volume total de *issues* de *label bug* dos repositórios de cada grupo de linguagens. A segunda é referente a mediana do tempo médio do fechamento de uma *issue* de *label bug* dos repositórios de cada grupo de linguagens.

A hipótese nula para a primeira métrica é a mediana do volume total de *issues* de *label bug* dos repositórios de cada grupo de linguagens é inferior em repositórios linguagens fracamente tipadas se comparado a repositórios de linguagens fortemente tipadas. A hipótese alternativa é de que a mediana do volume total de *issues* de *label bug* dos repositórios de cada grupo de linguagens é superior em repositórios linguagens fracamente tipadas se comparado a repositórios de linguagens fortemente tipadas.

A hipótese nula para a segunda métrica é a mediana do tempo médio do fechamento de uma *issue* de *label bug* é inferior em repositórios linguagens fortemente tipadas se comparado a repositórios de linguagens fracamente tipadas. A hipótese alternativa para a segunda métrica é a mediana do tempo médio do fechamento de uma *issue* de *label bug* é superior em repositórios linguagens fortemente tipadas se comparado a repositórios de linguagens fracamente tipadas.

4. Resultados

Os resultados apresentados nesta seção incluem a caracterização dos dados coletados e os resultados das métricas calculadas para as RQs definidas na Seção 3.3.

4.1. Caracterização do *Dataset*

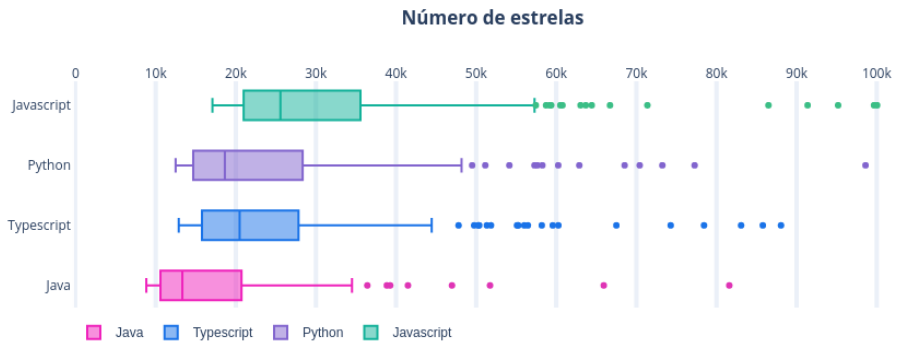

Figura 3. Distribuição do conjunto de dados sobre popularidade de linguagens tipadas e não tipadas

Com base nos resultados obtidos ao examinar o gráfico *boxplot* da Figura 3, que representa a distribuição do conjunto de dados sobre a popularidade de linguagens tipadas e não tipadas, é possível identificar uma disparidade entre os repositórios dessas linguagens. Ao analisar o eixo x, que representa a quantidade de estrelas, e o eixo y, que exibe as quatro linguagens de pesquisa em questão, observamos que as linguagens tipadas Java e TypeScript apresentaram medianas aproximadas de 13.327k e 20.4765k, respectivamente. Em contrapartida, as linguagens não tipadas Python e JavaScript exibiram medianas próximas de 18.63k e 25.571k, respectivamente.

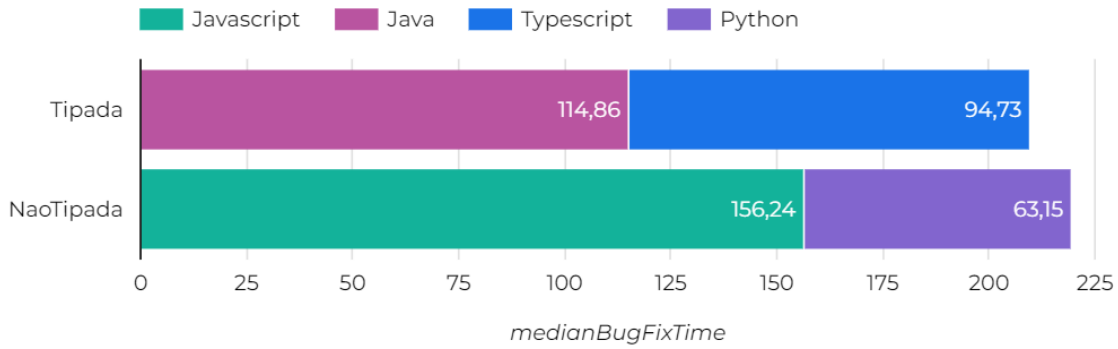

Figura 4. Total de tempo de correção em horas de *issues* de *label bug* para cada linguagem de programação e grupo

Os resultados da mineração dos dados dos conjuntos de repositórios, visualizados na Figura 4, demonstram que as linguagens não tipadas apresentam valores totais superiores comparados as tipadas. No entanto, a linguagem Javascript do conjunto de não tipadas, foi que apresentou o maior valor de mediana, com 156,24 horas totais para a correção de *issues* de tipo *bug*.

A Figura 5 apresenta os resultados visualizados em um gráfico de barras, destacando a mediana da quantidade total de contribuidores para diferentes grupos de repositórios. A análise revela algumas tendências significativas. Para repositórios tipados,

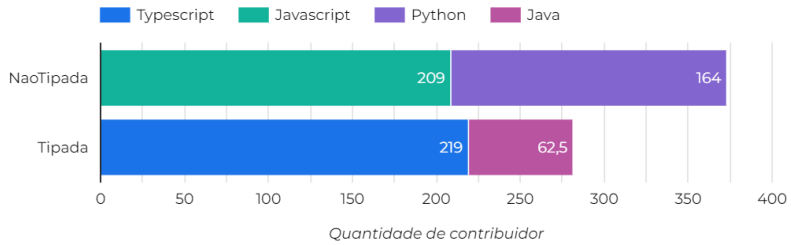

Figura 5. Total de contribuidores para cada linguagem de programação e grupo

como Java (219) e TypeScript (62.5), a mediana é de aproximadamente 282, enquanto para repositórios não tipados, como JavaScript (209) e Python (164), a mediana é de cerca de 374. É importante atentar-se que o eixo x utiliza uma escala de 1.000 para representar o número de contribuidores.

4.2. Repositórios escritos nas linguagens tipadas possuem qualidade de desenvolvimento de código melhor comparado a não tipados?

Com base na métrica M.1 e M.2, que diz a respeito do cálculo das medianas da quantidade de *Code Smells* e Complexidade Ciclomática nos diferentes grupos de repositórios das linguagens tipadas e não tipadas, em relação à RQ1, os resultados estão apresentados na Figuras fig:figura7, fig:codesmell2, fig:figura8 e fig:figura9.

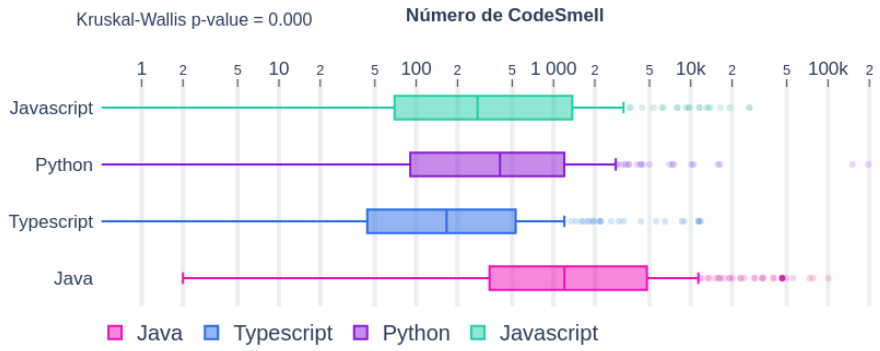

Figura 6. Distribuição do conjunto de dados das taxas de *Code Smells* de linguagens tipadas e não tipadas

A Figura 6 apresenta a visualização dos resultados em uma visão de um gráfico *boxplot*, que representa a mediana da quantidade de *Code Smells* nos diferentes grupos de repositórios, pode-se observar algumas tendências significativas. A mediana para repositórios escritos em Java é de 1201, enquanto em TypeScript é de 166,5. Já para Python, a mediana é de 408, e para JavaScript é de 280. Vale ressaltar que o eixo x utiliza uma escala logarítmica para representar o número de *Code Smells*.

A Figura 7 apresenta a visualização dos resultados em um gráfico *boxplot* com dois eixos y, representando a quantidade de *Code Smells* para os grupos de linguagens não tipadas e tipadas, em relação à taxa de *Code Smell* nos repositórios analisados. No eixo y para as linguagens não tipadas, a mediana é de 365, com um valor mínimo de 0 e máximo

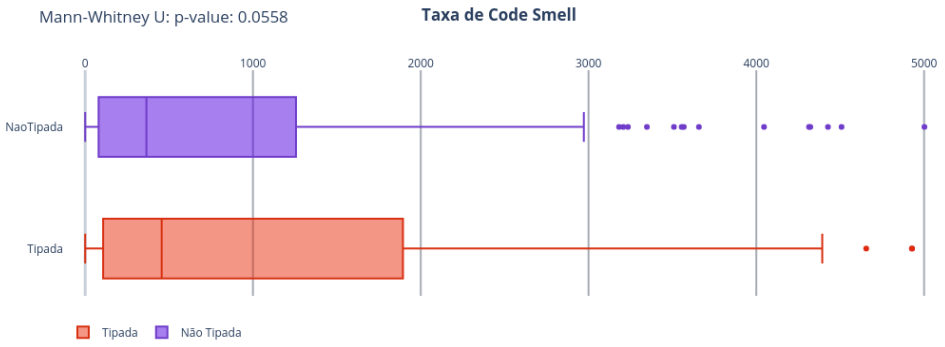

Figura 7. Medianas dos conjuntos de dados das as taxas de *Code Smells* dos repositórios analisados das linguagens tipadas e não tipadas

de 2972. Já no eixo y para as linguagens tipadas, a mediana é de 456, com um valor mínimo de 0 e máximo de 4393. O eixo x representa a taxa de *Code Smell* dos repositórios de ambas as linguagens. Ao analisar o valor do teste de Mann-Whitney U, observa-se um valor p igual a 0,0558. Isso sugere que não há uma diferença estatisticamente significativa entre os grupos de linguagens não tipadas e tipadas em relação à quantidade de *Code Smells*, embora a diferença possa ter uma certa relevância, mas não suficiente para ser considerada estatisticamente significativa.

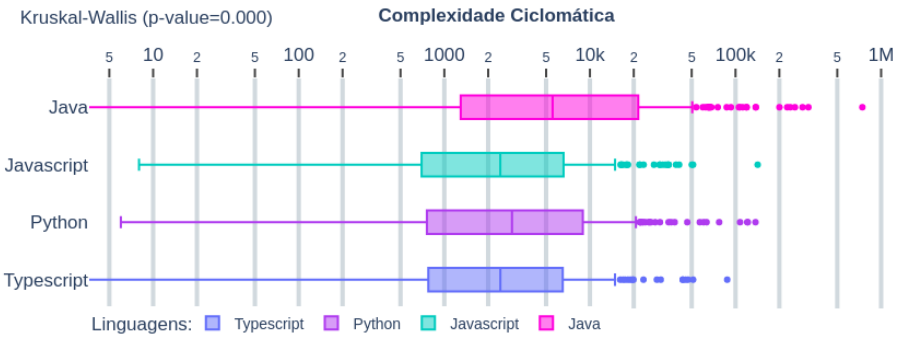

Figura 8. Distribuição do conjunto de dados das taxas de Complexidade Ciclomática de repositórios tipados e não tipados

Na visualização da Figura 8, está os dados referentes ao cálculo das medianas de complexidade ciclomática de cada uma das linguagens. Ao examinar o gráfico, observa-se que a mediana da Complexidade Ciclomática nos repositórios escritos em Java é de 5541,5, enquanto em TypeScript é de 2426. Para Python, a mediana é de 2915, e para JavaScript é de 2423,5. É importante notar que o eixo x utiliza uma escala logarítmica para representar o número de Complexidade Ciclomática.

A Figura 9 apresenta a visualização dos resultados em um gráfico *boxplot* para os grupos de linguagens não tipadas e tipadas, em relação à taxa de Complexidade Ciclomática dos repositórios analisados pela ferramenta de análise estática, SonarQube. No eixo y para as linguagens não tipadas, a mediana é igual a 2597,5, com o primeiro quartil

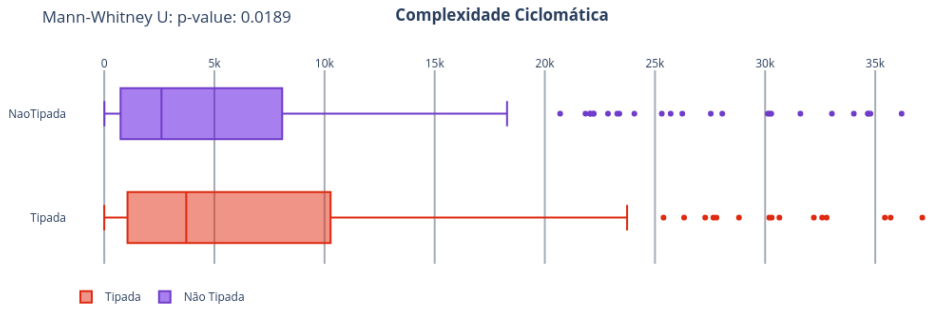

Figura 9. Medianas dos conjuntos de dados de Complexidade Ciclomática dos repositórios de linguagens tipadas e não tipadas

(Q1) em 740,5 e o terceiro quartil (Q3) em 8077. Já no eixo y para as linguagens tipadas, a mediana é de 3722, com Q1 em 1065 e Q3 em 10274. O eixo x representa a taxa de complexidade ciclomática dos conjunto de dados dos repositórios analisados divididos entre os com linguagens tipadas e não tipadas. Ao analisar o valor do teste de Mann-Whitney U, observa-se um valor p igual a 0,0189. Isso sugere que há uma diferença estatisticamente significativa entre os grupos em relação à taxa de complexidade ciclomática.

4.3. Como os repositórios de linguagens tipadas e não tipadas diferem em termos de tamanho da participação da comunidade?

Com base na métrica M.3 e M.4, que calcula a quantidade de contribuidores de cada um dos grupos de repositórios de linguagens tipadas e não tipadas, em relação às métricas da RQ2, os resultados são representados nas figuras desta sessão.

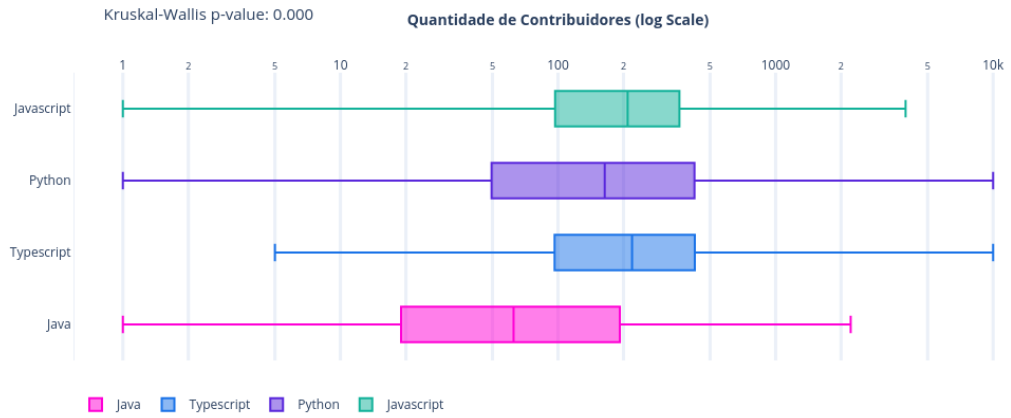

Figura 10. Distribuição do conjunto das medianas da quantidade de contribuidores de linguagens tipadas e não tipadas

A Figura 10, refere-se ao gráfico *boxplot* da mediana da quantidade de contribuidores que revelou que as linguagens tipadas, como Java e TypeScript, apresentaram medianas mais altas (Java = 62,5 e TypeScript = 219) em comparação com as linguagens não tipadas, como Python e JavaScript (Python = 164 e JavaScript = 209). O teste es-

tatístico de Kruskal-Wallis confirmou a diferença significativa entre os grupos com valor p = 0.000.

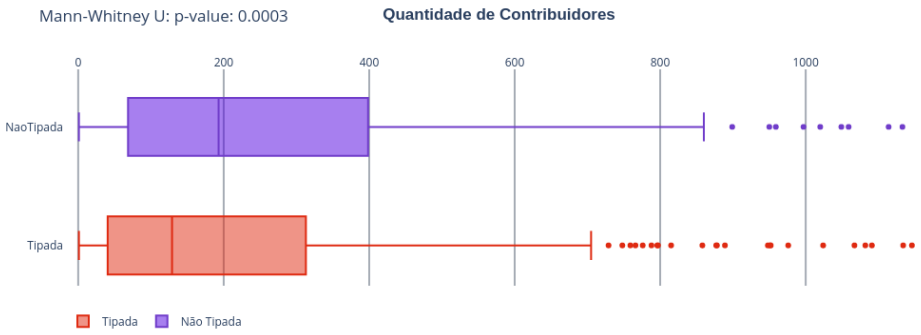

Figura 11. Medianas dos conjuntos de dados da quantidade de contribuidores de cada um dos grupos de repositórios de linguagens tipadas e não tipadas

No gráfico de *boxplot*, Figura 11, é possível visualizar a mediana da quantidade de contribuidores nos diferentes grupos do experimento, o eixo x representa o número de contribuidores, enquanto o eixo y mostra os grupos dos linguagens de pesquisa. Observa-se que a mediana do grupo dos tipados obteve valor maior, 193, comparado ao dos não tipados, 129. O grupo dos tipados apresentaram resultados de Q1 igual à 40 e Q3 igual à 313, já os não tipados apresentaram Q1 igual à 68 e Q3 igual à 398. O valor do teste de Mann-Whitney U gerou o valor de *p-value* igual à 0.0003, o que revela a existência de diferenças estatísticas entre os grupos.

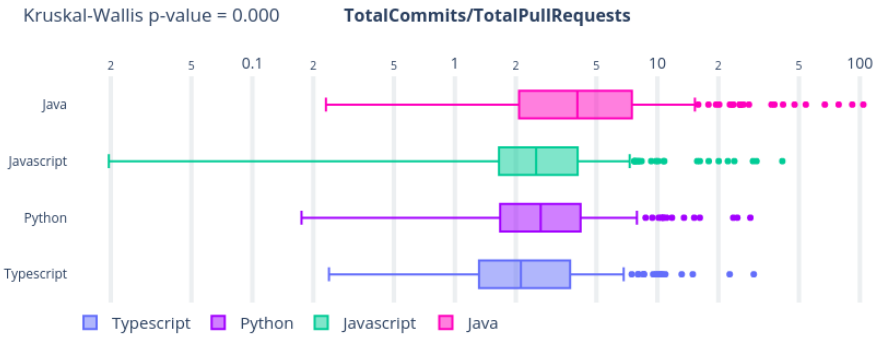

Figura 12. Distribuição do conjunto das mediana dos valores de *TotalCommits/TotalPullRequests* dos grupos de repositórios

A segunda métrica de RQ2 (Como os repositórios de linguagens tipadas e não tipadas diferem em termos de tamanho da participação da comunidade?) tem como o objetivo calcular a diferença na participação da comunidade entre repositórios de linguagens tipadas e não tipadas por meio do valor das medianas dos valores de Total de *Commits* por Total de *Pull Requests* para cada linguagem. Nesse sentido resultados obtidos, ao analisar o gráfico *boxplot* da Figura 13 da mediana dos valores de total de *commits* por

total de *pull requests*, foi possível identificar discrepâncias significativas no tamanho da participação da comunidade entre os repositórios de linguagens tipadas e não tipadas.

No eixo x, que representa a quantidade de contribuidores em escala logarítmica, e no eixo y, que exibe as quatro linguagens de pesquisa, foi observado que as linguagens tipadas Java e TypeScript apresentaram medianas aproximadas de 4,08 e 2,12, respectivamente. Em contrapartida, as linguagens não tipadas Python e JavaScript exibiram medianas próximas de 2,65 e 2,51, respectivamente. Além disso, o valor p do teste estatístico de Kruskal-Wallis (0.000) indica que há diferenças significativas entre os grupos.

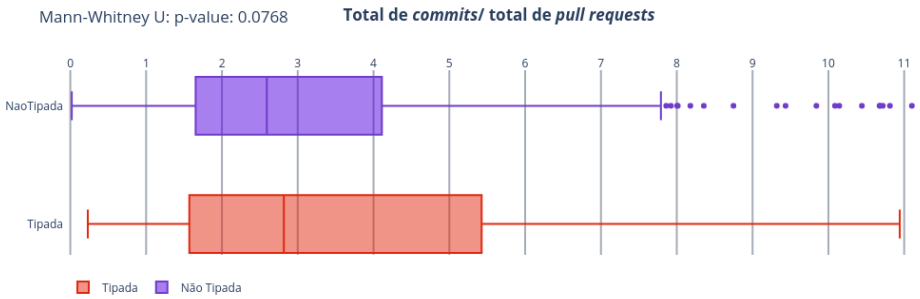

Figura 13. Medianas dos conjuntos de dados das mediana dos valores de *TotalCommits/TotalPullRequests* dos grupos de repositórios

Com base nos resultados coletados para a resposta da segunda métrica estão apresentados na Figura 13 é possível identificar discrepâncias significativas no tamanho da participação da comunidade entre os repositórios de linguagens tipadas e não tipadas. Foi observado que nas linguagens tipadas os valores do cálculo de número total de *commits* divido pelo número total de *Pull Requests* apresentou mediana de 2,82 aproximadamente, Q1 de 1,57 e Q3 de 5,43. Em contrapartida, o grupo das linguagens não tipadas exibiram medianas próximas de 2,59, Q1 de 1,64 e Q3 de 4,12. Além disso, o valor p do teste estatístico de Mann-Whitney U (0.0768) indica que não há diferenças significativas entre os grupos dessa análise.

4.4. Como os repositórios de linguagens tipadas e não tipadas diferem em termos de facilidade de correção de *bugs*?

Com base na métrica M.5 e M.6 que diz a respeito do cálculo das medianas da quantidade de mediana do volume do total de *issues* de *label "bug"* e seus respectivo tempo médio de seu fechamento nos diferentes grupos de repositórios das linguagens tipadas e não tipadas, em relação à RQ3 (Como os repositórios de linguagens tipadas e não tipadas diferem em termos de facilidade de correção de *bugs*?), os dados encontrados estão apresentados na Figura 14.

O gráfico de Função de Distribuição Empírica (ECDF, do inglês *Empirical distribution function*) da Figura 14 mostra a mediana da soma dos valores de problemas abertos e fechados relacionados a *bugs* nos repositórios de linguagens tipadas e não tipadas. No eixo x, temos o número total de problemas com a etiqueta *"bug"*, representado em escala logarítmica, e no eixo y, temos os valores ECDF. As quatro linhas representam as linguagens JavaScript, Python, TypeScript e Java. É possível observar que a maioria das

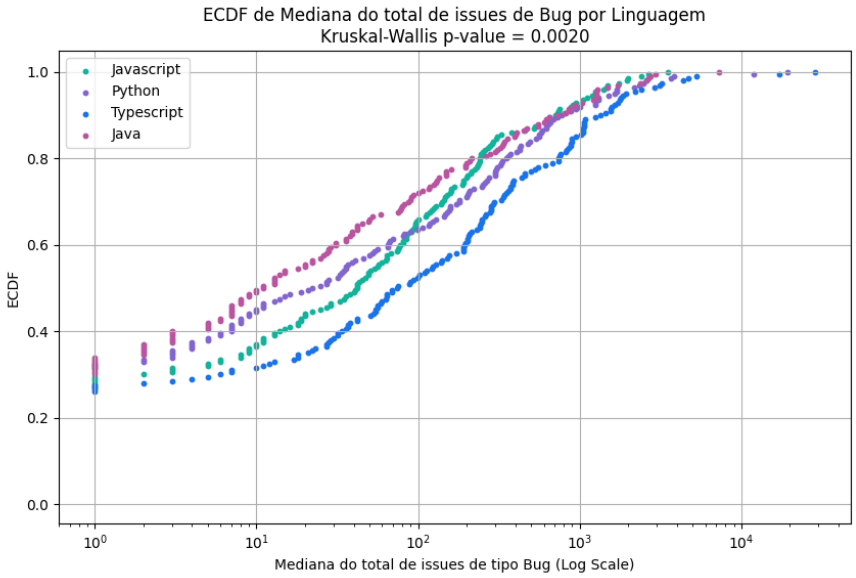

Figura 14. ECDF da mediana de *Issues* Abertas e Fechadas por Linguagem

medianas estão distribuídos nos valores de total de *issues* de 10^2 e 10^3 e ECDF de 0.5 a 0.8. O valor de p para o teste estatístico de Kruskal é 0,0020, o que indica diferenças significativas entre os grupos de linguagens tipadas e não tipadas.

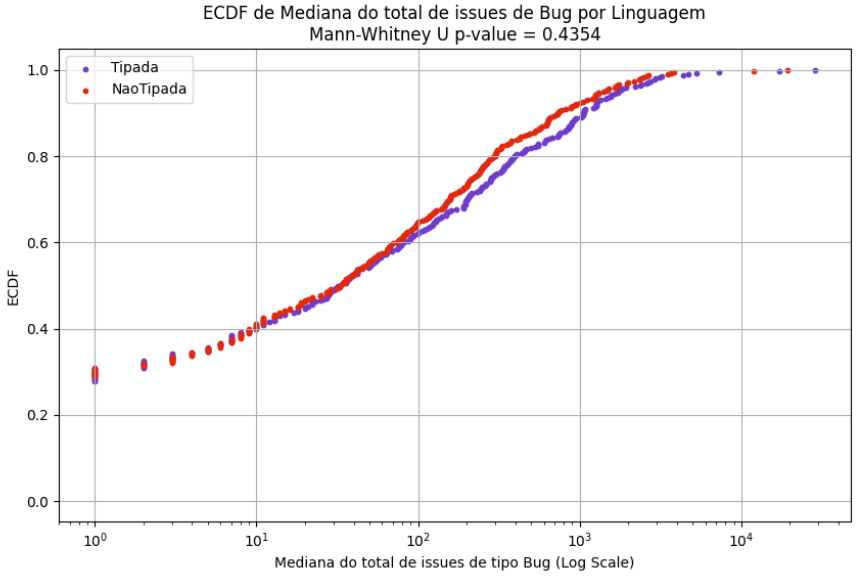

Figura 15. ECDF da mediana do total de *Issues* abertas e fechadas do grupo de tipadas e não tipadas

Na visualização da Figura 15 é possível observar com mais detalhes a diferença dos resultados entre os grupos relativos a M.5 (Desempenho de correção de *bugs*: Mediana do volume do total de *issues* de *label "bug"* de cada um dos grupos de repositórios de linguagens tipadas e não tipadas). A distribuição entre as medianas mostraram-se bastante similares, porém entre os valores de 10^2 e 10^3 os valores de mediana do grupo de

não tipados são relativamente mais altos comparados ao grupo concorrente. Além disso, o valor de *p-value* do teste de Mann-Whitney U apresentou resultado de 0,4354 (4,36% aproximado), simbolizando a diferença estatística da distribuição entre os dados dos grupos.

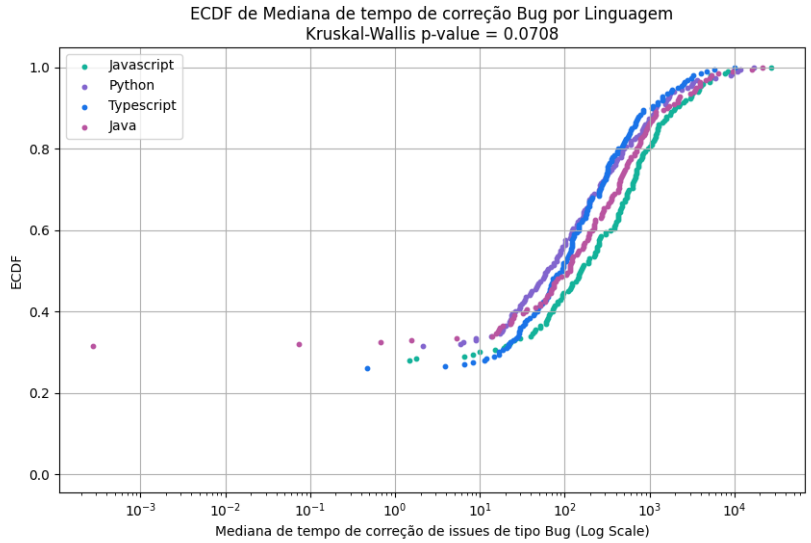

Figura 16. ECDF de mediana de tempo de correção *bug* por Linguagem

Na Figura 16 representa o gráfico de ECDF do valor das medianas do tempo de correção de *issues* de tipo *"bug"* de cada uma das linguagens selecionadas. Pode-se observar a concentração das medianas na faixa de valor de tempo de correção de 10^2 e 10^3 e da faixa de valor de ECDF de 0.4 a 0.8. O valor de p do teste estatístico de Kruskal-Wallis de 0,0708 significa que não há diferença estatisticamente significativa nos valores das medianas de tempo de correção das *issues*.

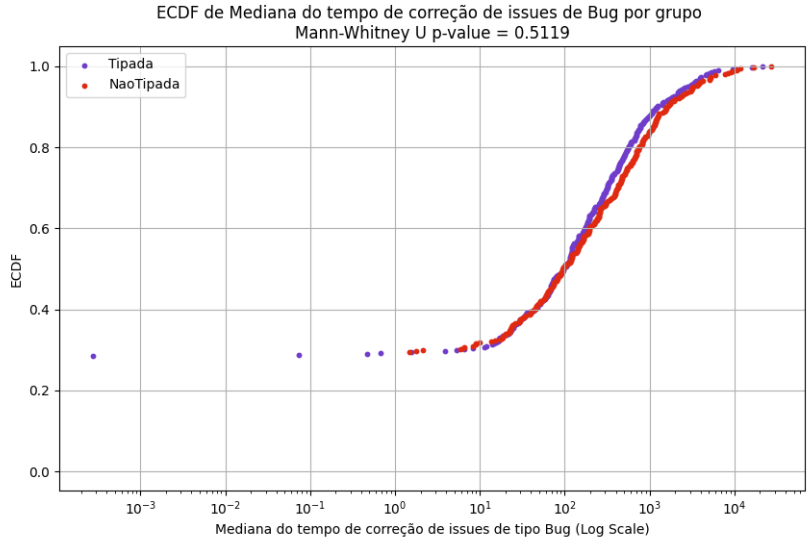

Figura 17. ECDF de mediana de tempo de correção *bug* por grupo

A Figura 17 apresenta uma visualização mais abrangente das diferenças nos resultados entre os grupos de linguagens tipadas e não tipadas em relação ao desempenho de correção de *issues* de tipo *"bug"*. Ao analisar a figura, podemos observar que as distribuições das métricas analisadas nos grupos de linguagens tipadas e não tipadas são visualmente distintas. Embora haja uma sobreposição considerável entre as distribuições, existem regiões onde as características dos grupos se destacam, novamente entre os valores de 10^2 e 10^3. Por meio de análises estatísticas, como o teste de hipótese, verificou-se que a semelhança entre a distribuição dos grupos é pouco significativa estatisticamente . O valor de p (0,5119 ou aproximados 5,12%) obtido no teste indica que a probabilidade de obter uma diferença tão significativa entre os grupos devido ao acaso é extremamente baixa.

5. Discussão dos resultados

Esta seção apresenta as implicações geradas pelas RQs respondidas neste estudo.

Repositórios escritos nas linguagens tipadas tem melhores qualidade de desenvolvimento de código comparado a não tipados. Com base na análise dos resultados referentes a RQ1, percebe-se que a mediana da taxa de *code smell* do grupo de repositórios das linguagens tipadas é maior comparada ao valor a mediana do grupo de linguagens não tipadas. Ademais, o cálculo do teste de Mann-Whitney U., aplicado em relação a análise dos dados de *code smell*, não apresentou diferença na distribuição entre os conjuntos de dados dos grupos tipados e não tipados. Apesar disso, os repositórios de linguagens tipadas apresentam uma taxa maior de *code smell*. Isso significa que, estatisticamente, não há evidências suficientes para afirmar que um grupo tem uma taxa de *code smell* de código melhor do que o outro com base apenas na taxa de *code smell*.

Enquanto que na análise da segunda métrica, a mediana da taxa de complexidade ciclomática indica um maior valor do fator no grupo das linguagens tipadas. Ao utilizar o teste de distribuição Mann-Whitney U., obteve-se um valor de 0,0189 (1,90% aproximado) que apontou diferença na disposição do conjunto dos dados. Com essa indicação e com base nos valores das medianas e dos quartis do grupo, é possível inferir que o uso de linguagem tipada pode resultar em uma complexidade ciclomática maior do código.

Portanto, conclui-se que a escolha de linguagens não tipadas pode ser considerada como um fator determinante para a qualidade do desenvolvimento de código. Porém, outros fatores como, práticas de programação, experiência da equipe e boas práticas de engenharia de software, devem ser considerados para avaliar e melhorar a qualidade do código independentemente do tipo escolhido.

Como os repositórios de linguagens tipadas e não tipadas diferem em termos de tamanho da participação da comunidade. Ao comparar os números de participação da comunidade entre os grupos, é evidente que o valor da mediana dos repositórios não tipados é maior que o valor de tipados, como também indica os valores dos primeiros e terceiros quartis. Ao aplicar o cálculo do teste de Mann-Whitney U. obteve-se o valor de 0,0003 (0,03%) indicando que há diferença entre os conjuntos.

Após analisar a segunda métrica utilizando o cálculo de Mann-Whitney U, com valor de 0,0768 (7,68%), acima do nível de significância definido em 5%, percebe-se que não há diferença estatística entre os conjuntos de tipados e não tipados. Assim, concluir-

se que tanto nos grupos tipados e não tipados há contribuição significativa nos repositórios avaliados.

Em suma, foi observado que a quantidade de contribuidores do grupo de repositórios não tipados é maior que os de tipados. Em contrapartida, a participação é equivalente entre os dois grupos, o que significa que a decisão de escolha entre linguagens tipadas e não tipadas em projetos é irrelevante em relação à participações efetiva das comunidades.

Como os repositórios de linguagens tipadas e não tipadas diferem em termos de facilidade de correção de *bugs*. Ao analisar os resultados referentes a RQ3, percebe-se que o volume de correções de *bugs* do grupo de repositórios de não tipadas é superior em comparação a distribuição das medianas do grupo de tipadas, que apresenta a concentração de ECDF no intervalo de 0.6 a 0.9, observados na Figura 15. Além disso, o valor do teste de Mann-Whitney U. igual à 0,4354 (43,54%) revela que a distribuição dos conjuntos são consideradas estatisticamente iguais.

Ao observar a distribuição dos dados em relação ao desempenho na correção de *bugs*, utilizando o teste Mann-Whitney U com valor de 0,5119 (51,19%) infere-se que a distribuição são iguais, com concentração no intervalo de ECDF de 0.58 à 0,9. Em vista disso é notável que o valor das medianas do conjunto de dados de repositórios não tipados é inferior ao conjunto dos tipados, nesse sentido é pode-se inferir que repositórios desenvolvidos em linguagens não tipadas apresentam tempo reduzido de correção de *bugs*. Essa análise indica que não há evidências de diferenças substanciais no tempo de correção de *bugs* entre os dois grupos de linguagens.

Dessa forma, concluímos que, embora os repositórios não tipados tenham um volume de correções de *bugs* maior, não há diferenças significativas no tempo de correção quando comparados aos repositórios tipados. Portanto, no contexto específico da facilidade de correção de *bugs*, não é evidente que há indícios de que o uso de linguagens tipadas ou não tipadas influencie de forma substancial na eficiência da correção.

6. Ameaças à validade

Nesta seção, são apresentadas as ameaças à validade deste estudo, assim como as estratégias adotadas para mitigá-las. Primeiramente, quanto à validade de construção, a quantidade de repositórios estudados pode não ser estatisticamente representativo. Isso ocorre, pois não há o conhecimento do tamanho da população total do universo de desenvolvedores no mundo real. Para mitigar essa ameaça, os dados coletados referem-se aos repositórios mais populares da plataforma Github, das linguagens tipadas (TypeScript e Java) e não tipadas (JavaScript e Python), a fim de avaliar os mais representativos. Segundo, a dificuldade em executar o SonarQube em projetos Java devido a problemas de integração com as bibliotecas Maven e Gradle, comprometendo a análise estática do código e a obtenção de métricas de qualidade precisas, podendo levar a resultados imprecisos e conclusões inválidas. Para mitigar essa ameaça, foi necessário resolver os problemas de compatibilidade assim, garantindo que o SonarQube esteja funcionando corretamente nos projetos Java.

Analisando a ameaça à validade interna, observa-se que pode ocorrer falha na máquina responsável pela execução dos *scripts*, falta de otimização do algoritmo de co-

leta e análise de dados ou indisponibilidade das ferramentas utilizadas, SonarQube e Looker[5]. Essas instabilidades podem impedir a coleta adequada de dados da API GraphQL do GitHub ou do cálculo da análise de qualidade, *Code Smells* e *bugs*. Para lidar com essas ameaças, foi gerado um arquivo CSV para armazenamento dos repositórios por linguagem, conforme foi progredindo a análise do *script* outros arquivos foram gerados para armazenar os resultados obtidos pela análise. Essa ação ajudou na recuperabilidade dos *scripts* para retornar ao ponto da interrupção.

Analisando a ameaça à validade externa, observa-se que pode ocorrer alguma instabilidade de rede, indisponibilidade da API ou vencimento da validade do *token*. Essas interrupções podem impedir a coleta adequada de dados da API GraphQL do GitHub e, consequentemente no cálculo da análise de qualidade, *Code Smells* e *bugs*. Para lidar com essas ameaças, foi gerado um arquivo CSV para armazenamento dos repositórios por linguagem, conforme foi progredindo a análise do *script* outros arquivos foram gerados para armazenar os resultados obtidos pela análise. Essa ação ajudou na recuperabilidade dos *scripts* para retornar ao ponto da interrupção.

Já a ameaça à validade de conclusão, a maturidade das linguagens pode afetar na análise. Desafios como disponibilidade de recursos e bibliotecas, estabilidade, suporte da comunidade, e principalmente, a popularidade podem impactar os resultados. As linguagens mais maduras (JavaScript e Java) tendem a ter mais recursos, melhor estabilidade e suporte da comunidade. Para mitigar as ameaças à validade de conclusão decorrentes da maturidade da linguagem em uma análise comparativa, foram selecionadas quatro linguagens: JavaScript (não tipada) e Java (tipada), que surgiram em 1991, e TypeScript (tipada) e Python (não tipada). Embora o TypeScript tenha surgido em 2010 e o Python tenha sido criado em 1991, sua popularização ocorreu a partir de 2010. Essa seleção permitiu equalizar os dados, considerando tanto linguagens mais maduras quanto aquelas que ganharam destaque mais recentemente.

7. Conclusão

Nessa pesquisa, propõe-se uma análise comparativa entre os conjuntos de linguagens tipadas e as não tipadas mais populares da plataforma Github, foram analisados 800 repositórios sendo 200 de cada linguagem, com o objetivo de avaliar as vantagens e desvantagens de cada dos grupos, nos requisitos qualidade de desenvolvimento, tamanho da participação da comunidade e facilidade de correção de *"bugs"*. Portanto, ao fim do experimento, os resultados obtidos sugerem que a escolha de linguagens não tipadas pode ser considerada como um fator determinante para a qualidade do desenvolvimento de código, principalmente se relevar a baixa a taxa de complexidade ciclomática. Ademais, as linguagens tipadas tem valores maiores de quantidade de *issues* de tipo *"bug"* comparado ao grupo de linguagens não tipadas. Entretanto, os valores de tempo médio de fechamento dessas mesmas *issues* demonstram ser ligeiramente menores comparadas a não tipadas, o que não pode significar uma definitiva facilidade de correção de *bugs*.

Além disso, com os resultados também foi possível inferir que os repositórios de linguagens não tipadas mais populares tendem a atrair mais contribuidores mas a atividade de desenvolvimento em comparação com os repositórios de linguagens não tipadas não foram encontradas diferenças relevantes. No entanto, é importante considerar que outros

[5]https://lookerstudio.google.com

fatores, como a popularidade das linguagens e a natureza dos projetos também podem influenciar o tamanho da participação da comunidade.

A análise desta pesquisa pode auxiliar desenvolvedores e arquitetos de software no processo de escolha da linguagem mais adequada para seus projetos, além de ajudar futuras pesquisas a respeito do tema com as observações e dados apresentados. Nesse sentido, sugere-se que trabalhos futuros explorem outras questões e fatores não exploradas nesta análise. Por exemplo, investigar o impacto dos diferentes níveis de experiência dos desenvolvedores no uso das linguagens, levando em consideração como a experiência afeta a produtividade, a qualidade do código e a eficiência do desenvolvimento.

Outra área que merece atenção em trabalhos futuros é o uso de outras ferramentas de análise de qualidade para calcular as métricas de qualidade das linguagens. Além das métricas mencionadas na análise atual, poderiam ser exploradas métricas adicionais, como a facilidade de depuração, a detecção de vulnerabilidades de segurança e a facilidade de integração com outras tecnologias.

Ao abordar essas questões e fatores adicionais, pesquisas futuras poderão fornecer conclusões mais abrangentes e precisas sobre a escolha da linguagem de programação em projetos de software. Isso permitirá que desenvolvedores e arquitetos de software tomem decisões mais embasadas e adequadas às suas necessidades específicas.

Referências

Bogner, J. and Merkel, M. (2022). To type or not to type? a systematic comparison of the software quality of javascript and typescript applications on github. In *2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR)*, pages 658–669.

Borges, H. and Tulio Valente, M. (2018). What's in a github star? understanding repository starring practices in a social coding platform. *Journal of Systems and Software*, 146:112–129.

Gao, Z., Bird, C., and Barr, E. T. (2017). To type or not to type: quantifying detectable bugs in javascript. In *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, pages 758–769. IEEE.

Harlin, I. R., Washizaki, H., and Fukazawa, Y. (2017). Impact of using a static-type system in computer programming. In *2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE)*, pages 116–119.

Khan, F., Chen, B., Varro, D., and McIntosh, S. (2022). An empirical study of type-related defects in python projects. *IEEE Transactions on Software Engineering*, 48(8):3145–3158.

Merkel, M. (2021). Do typescript applications show better software quality than javascript applications? a repository mining study on github.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M., Regnell, B., and Wesslén, A. (2012). *Experimentation in Software Engineering*. Springer, Germany.

Zhang, J. M., Li, F., Hao, D., Wang, M., Tang, H., Zhang, L., and Harman, M. (2021). A study of bug resolution characteristics in popular programming languages. *IEEE Transactions on Software Engineering*, 47(12):2684–2697.