Análise comparativa de linguagens tipadas e tipadas mais populares

Belle Nerissa A. Elizeu[1], Ian Asenjo D. Cunha[2], Jully Ketely A. Silva[1], Laura Lourdes C. Rodrigues[3], Warley Leandro dos Anjos[4], Prof. Gabriel de O. Campos Pacheco[5], Prof. José Laerte P. Xavier Junior

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

{falreis, laertexavier}@pucminas.br

Abstract. *Through the collection of quantitative metrics of code quality, the research proposed in this work seeks to carry out a comparative study between repositories of typed and untyped languages. The 200 most popular open source repositories of Python, Javascript, Typescript and Java languages will be collected from the GitHub source code hosting platform, considering the number of stars received. The extracted repositories will be analyzed by the Sonarqube Scan static analysis tool.*

Resumo. *A pesquisa proposta neste trabalho busca realizar um estudo comparativo entre repositórios de linguagens tipadas e não tipadas. Por meio da coleta de dados, foram aplicadas métricas quantitativas de qualidade de código. Os dados foram coletados da plataforma de hospedagem de código-fonte, GitHub, minerando os 200 repositórios Open Source de cada linguagem: Python, Javascript, Typescript e Java com maior popularidade, considerando a quantidade de estrelas recebidas. Os repositórios extraídos serão analisados pela ferramenta de análise estática SonarQube Scan.*

1. Introdução

A escolha da linguagem de programação pode ter um grande impacto no desenvolvimento de software. Com o surgimento de diversas linguagens de programação ao longo do tempo, pode ser desafiador escolher a melhor opção para uma determinada aplicação.

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 são linguagens onde é possível declarar variáveis sem definir o seu tipo.

Neste artigo é apresentado os resultados de uma análise comparativa entre as linguagens tipadas e as não tipadas mais populares. Além de avaliar as diferenças, vantagens e desvantagens de cada uma delas, este estudo será uma contribuição à literatura, auxiliando desenvolvedores e arquitetos de software na escolha da linguagem mais adequada

para seus projetos, considerando, fatores como qualidade de desenvolvimento, tamanho da participação da comunidade e facilidade de correção de *bugs*.

Foram feitas três perguntas neste artigo:

RQ.1: Repositórios escritos nas linguagens tipadas têm melhores qualidade de desenvolvimento de código comparado a não tipados? A intenção é investigar possíveis ganhos das linguagens tipadas em relação às não tipadas quando comparado às taxas de *code-smell*.

RQ.2: Como os repositórios de linguagens tipadas e não tipadas diferem em termos de tamanho da participação da comunidade? Com essa segunda pergunta, a intenção é comparar se os valores das medianas da complexidade ciclomática de linguagens tipadas, são inferiores ou maiores que as linguagens não tipadas.

RQ.3: Como os repositórios de linguagens tipadas e não tipadas diferem em termos de facilidade de correção de *bugs*? A intenção com essa última pergunta é obter uma comparação do desempenho das linguagens tipadas e não tipadas, avaliando *issues* de *label bug*.

O restante deste artigo contém quatro Seções. A Seção II é composta pelos trabalhos relacionados. Na Seção III é descrito o desenho do experimento proposto. A Seção IV apresenta os resultados das duas questões de pesquisa propostas. Por fim, a Seção V conclui o artigo.

2. Trabalhos Relacionados

Nesta Seção são abordados os artigos encontrados na literatura, usados para guiar a pesquisa do presente trabalho. Os trabalhos relacionados foram separados em quatro categorias: (a) estudos sobre qualidade de software; (b) estudos sobre a participação da comunidade do GitHub; (c) estudos sobre correção de *bugs*; e (d) estudos sobre linguagens tipadas e não tipadas.

2.1. Qualidade de software

[Parnas and Lawford 2003] explicam em seu estudo que devido à complexidade do código, o software é lançado com muitos erros e a inspeção é a única maneira de melhorar a qualidade do software. A inspeção é um exame realizado para avaliar a qualidade do produto de software. Como resultado os autores concluem que os métodos de inspeção podem ser mais eficazes, mas o sucesso depende de um procedimento sólido e sistemático para conduzir a inspeção. Além disso, tanto os profissionais de software quanto os pesquisadores de software precisam melhorar a reputação do software e a única maneira de fazer isso é melhorar a qualidade do software. Esse artigo foi utilizado para embasamento nas discussões relativas a qualidade de software.

2.2. Estudos participação da comunidade do GitHub

Segundo [Borges et al. 2016] a popularidade do software é uma informação valiosa para os desenvolvedores de código aberto modernos, que constantemente querem saber se seus sistemas estão atraindo novos usuários, se novos lançamentos estão ganhando aceitação ou se estão atendendo às expectativas do usuário. O GitHub fornece uma maneira explícita para os usuários manifestarem sua satisfação com um repositório hospedado, o botão

stargazers. Em outro estudo, [Borges and Tulio Valente 2018] realizaram uma pesquisa sobre métricas mais úteis para medir a popularidade dos projetos do GitHub com usuários do Stack Overflow. Os resultados mostraram que as estrelas são vistas pelos praticantes como a medida de popularidade mais útil no GitHub, seguido por *forks* e por último observadores.

2.3. Estudos correção de *bugs*

[Zhang et al. 2021] apresenta em seu artigo um estudo em larga escala que investiga as características de resolução de *bugs* entre projetos populares do Github escritos em diferentes linguagens de programação. Nesse estudo analisaram dados de resolução de *bugs* de aproximadamente 70 milhões de linhas de código-fonte, extraídas de 3 milhões de confirmações para 600 projetos do GitHub, escritos principalmente em 10 linguagens de programação. Esse artigo inicia um debate sobre a importância da tipagem estática, e contribui nas discussões de métricas de resolução de *bugs*.

2.4. Estudos linguagens tipadas e não tipadas

[Bogner and Merkel 2022] realizaram em seu estudo uma comparação sistemática da qualidade de software de aplicações em JavaScript e TypeScript. Utilizando o SonarQube e a *Appllication Programming Interface (API)* do GitHub, coletaram e analisaram quatro métricas de qualidade de software: a) qualidade do código (taxa de *code smells* por *Lines of Code (LoC)*), b) compreensibilidade do código (complexidade cognitiva por *LoC*), c) propensão a *bugs* (taxa de confirmação de correção de *bugs*) e d) tempo de resolução de *bugs* (tempo médio em que um problema de *bug* é aberto). Sendo assim, o artigo foi utilizado para discussão de quais métricas de qualidade de software deveriam ser usadas na coleta de dados.

3. Metodologia

O estudo foi realizado a partir da coleta de métricas quantitativas de qualidade de código. O objetivo é comparar os repositórios mais populares de linguagens tipadas e linguagens não tipadas, e responder às seguintes perguntas de pesquisa:

QP.1 Repositórios escritos nas linguagens tipadas tem melhores qualidade de desenvolvimento de código comparado a não tipados?

M.1 Taxa de *Code Smells*: Mediana das taxas de *code smell* obtidas na análise estática de código-fonte providas pelo SonarQube de cada grupo de repositórios.

- H.0 Os repositórios de linguagens tipadas apresentaram taxa de *code-smell* menores que não-tipadas, ou seja uma qualidade superior em relação aos repositórios de linguagens não tipadas
- H.1 Os repositórios de linguagens tipadas apresentaram taxa de code-smell maiores que não-tipadas, ou seja uma qualidade inferior em relação aos repositórios de linguagens não tipadas

M.2 Complexidade Ciclomática: Mediana dos valores de Complexidade Ciclomática obtidos na análise estática de código-fonte providas pelo SonarQube de cada grupo de repositórios

H.0 Os repositórios de linguagens tipadas apresentaram valor de complexidade - ciclomática menores que não-tipadas, ou seja complexidade superior em relação aos repositórios de linguagens não tipadas.

H.1 Os repositórios de linguagens tipadas apresentaram valor de complexidade - ciclomática maiores que não-tipadas, ou seja complexidade inferior em relação aos repositórios de linguagens não tipadas

QP.2 Como os repositórios de linguagens tipadas e não tipadas diferem em termos de tamanho da participação da comunidade?

M.3 Quantidade de contribuidores: mediana da quantidade de contribuidores de cada um dos grupos de repositórios.

H.0 A quantidade de contribuidores em repositórios de linguagens tipadas será superior do que as não tipadas.

H.1 A quantidade de contribuidores em repositórios de linguagens tipadas será inferior do que as não tipadas.

M.4 Quantidade de contribuições: mediana da quantidade de contribuições calculada pelo total de *commits* por total de *pull request* de cada um dos grupos de repositórios.

H.0 A quantidade de contribuições em repositórios de linguagens tipadas será superior do que as não tipadas.

H.1 A quantidade de contribuições em repositórios de linguagens tipadas será inferior do que as não tipadas.

QP.3 Como os repositórios de linguagens tipadas e não tipadas diferem em termos de facilidade de correção de *bugs*?

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

H.0 A mediana do volume do total de *issues* de *label "bug"* do grupo de repositórios não tipados será inferior a de tipados.

H.1 A mediana do volume do total de *issues* de *label "bug"* do grupo de repositórios não tipados será superior a de tipados.

M.6 Desempenho da correção de *bugs*: Mediana do tempo médio do fechamento de uma *issue* com *label "bug"* um dos grupos repositórios de linguagens tipadas e não tipadas.

H.0 A mediana do tempo médio do fechamento de *issues* de *bug* no grupo de de linguagens tipadas será inferior do que as não tipadas.

H.1 A mediana do tempo médio do fechamento de *issues* de *bug* no grupo de repositórios de linguagens tipadas será superior do que as não tipadas.

Para responder às questões de pesquisa propostas, foram coletados da plataforma de hospedagem de código-fonte GitHub os 200 repositórios *open source* das linguagens Python, Javascript, Typescript e Java com maior popularidade, considerando a quantidade de estrelas recebidas. Os repositórios extraídos foram analisados pela ferramenta de análise estática Sonarqube.

As linguagens Python, JavaScript, TypeScript e Java foram escolhidas para um projeto de pesquisa quantitativa com base no relatório GitHub Octoverse 2022. Essas quatro linguagens foram classificadas entre as mais populares, indicando sua ampla adoção e relevância na comunidade de desenvolvimento de software.

As linguagens Python, Javascript, Typescript e Java foram escolhidas para realizar essa pesquisa devido a suas posições no rank do github de linguagens mais utilizadas em 2021 e por serem denominadas tipadas e não tipadas. Python e Javascript são linguagens de programação não tipadas (ou dinamicamente tipadas), o que significa que as variáveis não têm um tipo fixo e podem ser reatribuídas com diferentes tipos de dados durante a execução do programa. Typescript e Java, por outro lado, são linguagens de programação tipadas (ou estaticamente tipadas), o que significa que as variáveis têm um tipo fixo e precisam ser declaradas com um tipo específico antes de serem usadas.

A seleção da ferramenta SonarQube foi realizada considerando a gratuidade de uso para projetos *open source* e suporte às métricas mais discutidas em artigos científicos relacionadas ao tema de análise estática de código e por ser compatível com a análise de códigos das linguagens selecionadas para essa pesquisa.

3.1. Comparação da qualidade

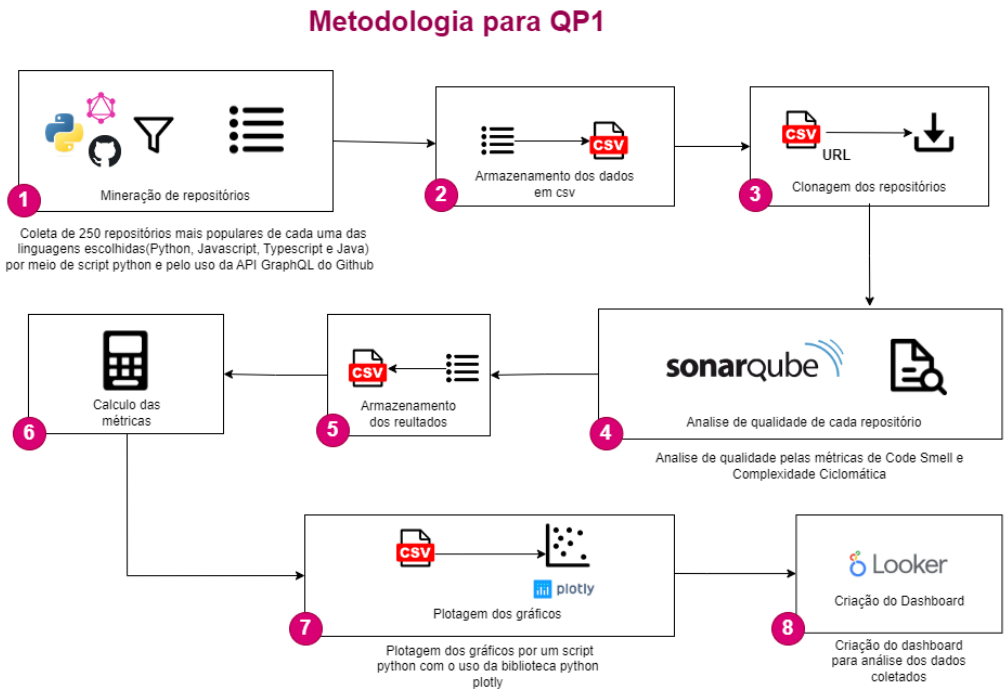

Figura 1. Fluxograma da metodologia da QP 1

A Figura 1 apresenta uma ilustração das etapas propostas para execução da metodologia de pesquisa e análise referentes à primeira questão principal deste trabalho. Foram executados os seguintes passos:

- Coleta de 200 repositórios mais populares de cada uma das linguagens escolhidas(Python, Javascript, Typescript e Java) por meio de *script python* e pelo uso da *API* GraphQL do Github.

- Armazenamento dos dados coletados na etapa anterior em um arquivo *CSV*.

- Clonagem dos repositórios coletados por meio do Git, utilizando a URL armazenada dos repositórios minerados.

- Análise de código para cada repositório clonado, a ferramenta *SonarQube* é executada para o cálculo das métricas de qualidade de código de complexidade ciclomática e *code smell*.

- Para cada um dos resultados retornados pelas ferramentas, será realizado uma sumarização dos dados, utilizando para isso medidas de mediana.

- Os dados sumarizados na etapa anterior serão armazenados em arquivos .CSV.

- A criação dos gráficos, dos dados coletados, foi feita utilizando o *Looker* para criar gráficos de barras. Além disso, também foi utilizado um *script* em Python com a biblioteca *Plotly* para criar gráficos do tipo *boxplot*.

- A criação do *dashboard* para a visualização e análise dos dados foi realizada por meio da ferramenta de *BI Looker* da *Google*.

3.2. Comparação de contribuição e *bugs*

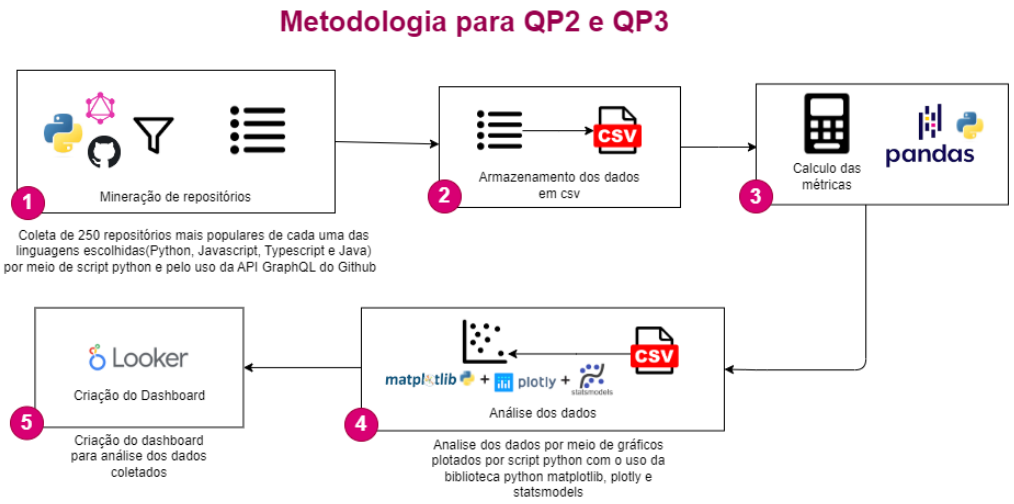

Figura 2. Fluxograma da metodologia da QP 2 e QP 3

A Figura 2 apresenta uma ilustração das etapas propostas para execução da metodologia de pesquisa e análise referentes à segunda questão principal deste trabalho. Foram executados os seguintes passos:

- Utilizado a mesma coleta e armazenamento de dados dos 250 repositórios de cada linguagem como descritos anteriormente

- Para cada um dos resultados retornados pela mineração, foi realizado uma sumarização dos dados de quantidade de colaboradores, colaborações e o total de *issues* de *label* "*bug*" assim como o cálculo do tempo de fechamento de *issues* de *bug*.

- Os gráficos de barras foram criados pelo *Looker*, enquanto os gráficos de Função de distribuição empírica (ECDF) e *boxplot* foram construídos com Python usando as bibliotecas *Matplotlib*, *Statsmodels* e *Plotly*.

- A criação do *dashboard* para a visualização e análise dos dados foi realizada por meio da ferramenta de *BI Looker da Google*.

4. Resultados

A seguir, serão apresentados os resultados que respondem as questões de pesquisa (QP.1, QP.2 e QP.3) e os resultados obtidos referentes os cálculos de cada métrica definidas para cada questão.

4.1. Caracterização do *Dataset*

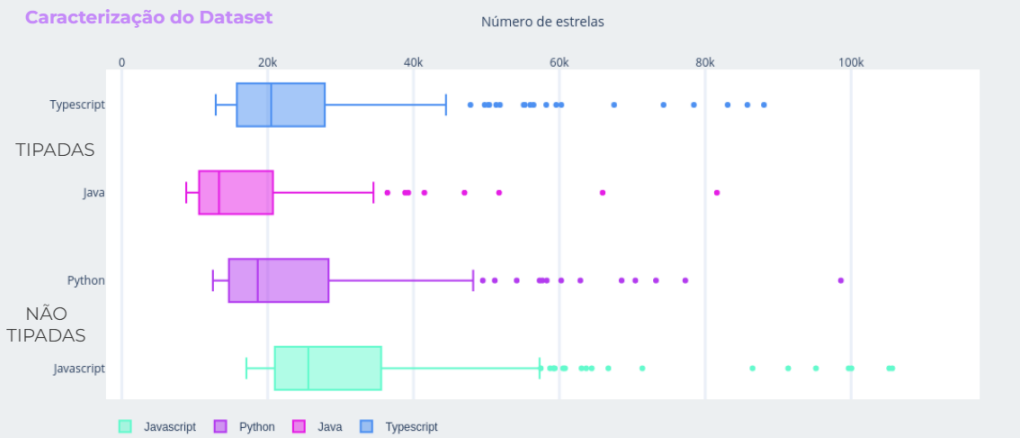

Figura 3. Distribuição do conjunto de dados sobre popularidade de linguagens tipadas e não tipadas

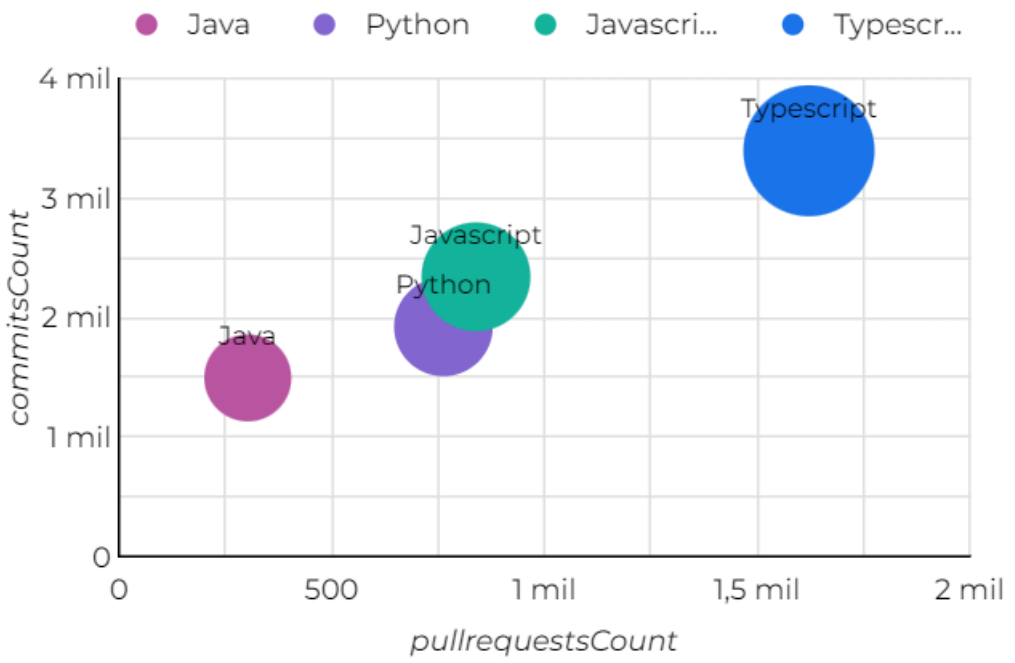

Figura 4. Quantidade de *Commits* efetuados sobre a quantidade de *Pull Requests* por linguagem de programação.

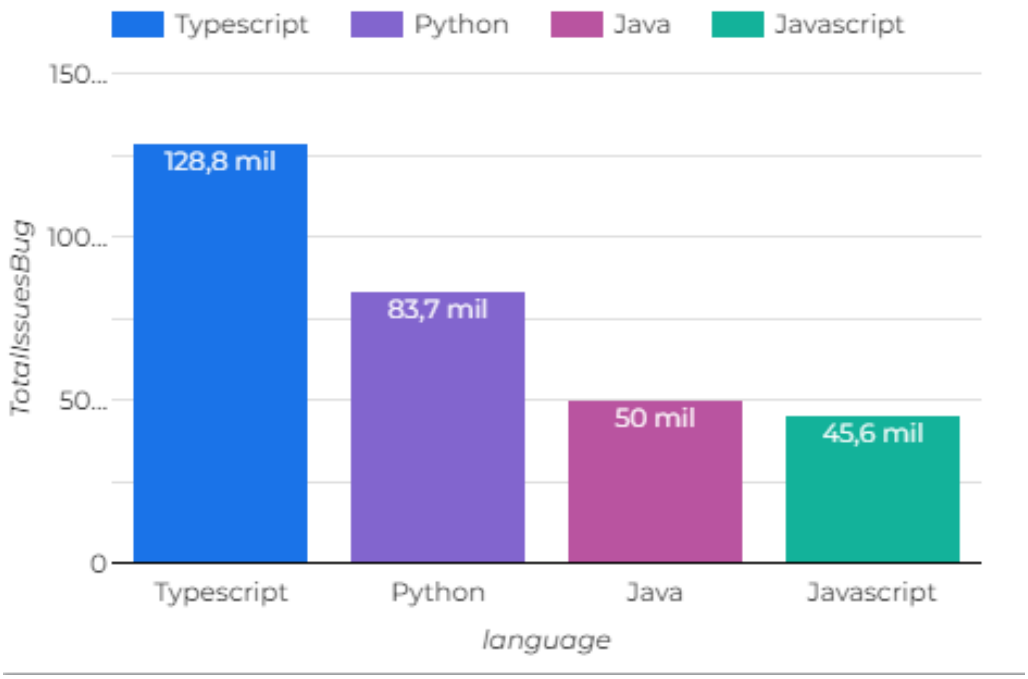

Figura 5. Total de *Issues* de label *bug* para cada linguagem de programação

4.2. Repositórios escritos nas linguagens tipadas tem melhores qualidade de desenvolvimento de código 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 QP.1, os dados encontrados estão apresentados na Figura 6 a seguir.

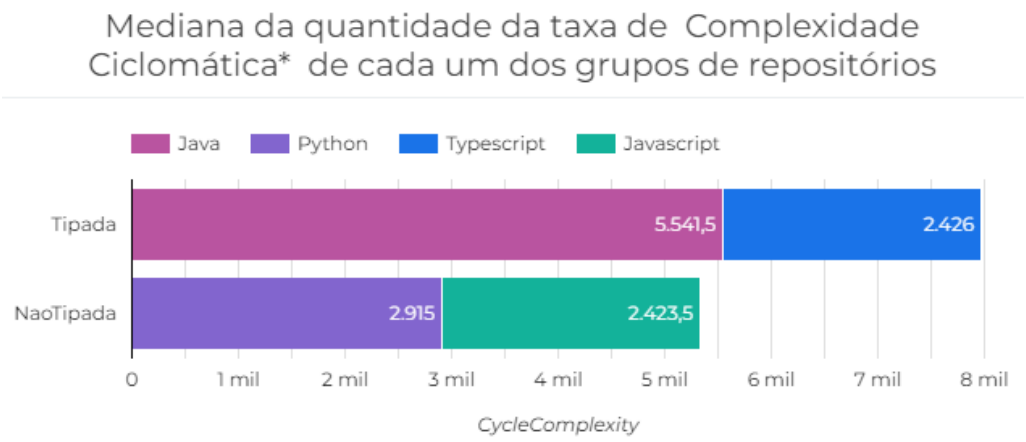

Figura 6. Mediana da taxa de complexidade ciclomáticas da linguagens tipadas e não tipadas

O gráfico de barras composto na Figura 6 apresenta a mediana da quantidade de *Code Smells* nos diferentes grupos de repositórios das linguagens tipadas e não tipadas

escolhidas para a pesquisa. Pode-se observar uma tendência significativa. A barra correspondente aos repositórios escritos em linguagens tipadas (Java e TypeScript) exibe uma mediana numericamente maior em comparação com as linguagens não tipadas (Python e JavaScript), representadas na segunda barra. Isso indica uma provável relação entre o uso de linguagens tipadas e uma maior incidência de *Code Smells* nos repositórios analisados.

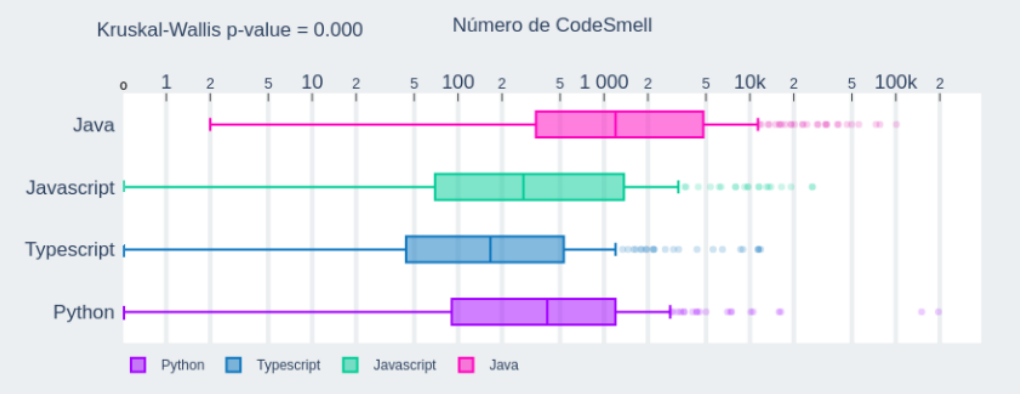

Figura 7. Distribuição do conjunto de dados sobre *Code Smells* de linguagens tipadas e não tipadas

A figura 7 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, podemos 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*. Além disso, o valor do teste de Kruskal-Wallis, indicado pelo valor p igual a 0.000, comprovando que há diferenças estatisticamente significantes entre os grupos de linguagens estudados.

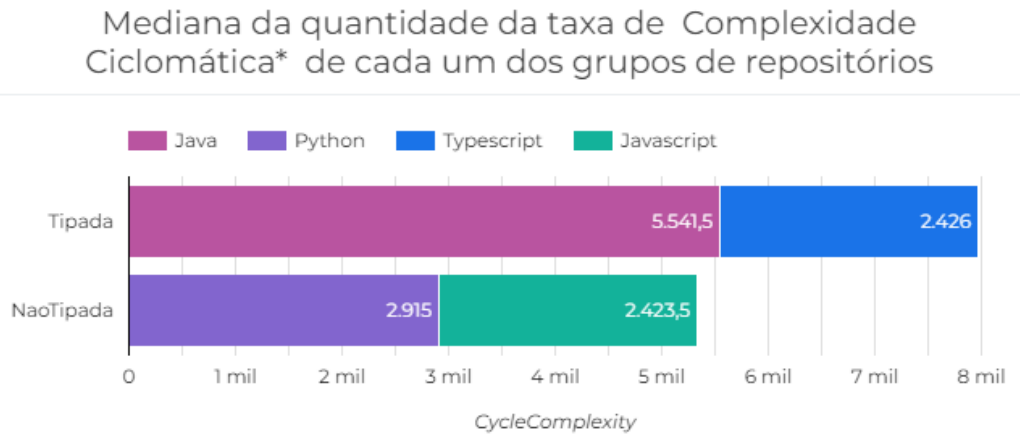

Figura 8. Mediana da taxa de Complexidade Ciclomática de repositórios tipados e não tipados

A Complexidade Ciclomática média calculada em diferentes grupos de arquivos das linguagens analisadas é representada por um gráfico de barras composto acima [Figura 8]. Ao analisar esses dados, algumas informações relevantes ficam claras. A barra

mediana representa os repositórios de linguagens tipadas (Java e TypeScript) que são numericamente mais altos do que as linguagens não tipadas (Python e JavaScript) apresentadas na segunda barra. Além disso, é essencial observar que a métrica usada para calcular a complexidade ciclomática mede o número mínimo de casos de teste necessários para a cobertura total do teste.

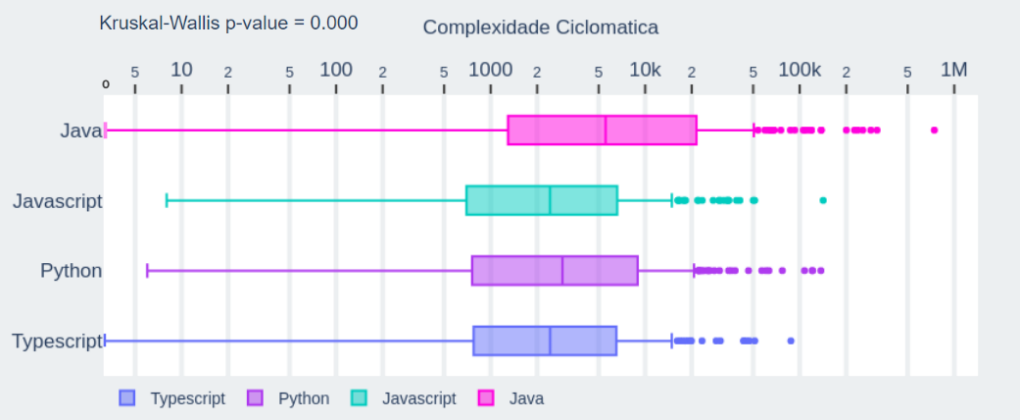

Figura 9. Distribuição do conjunto de dados sobre Complexidade Ciclomática de linguagens tipadas e não tipadas

Na visualização [Figura 9] dos dados referentes ao cálculo das medianas de complexidade ciclomática de cada um dos grupos, também foi realizado o gráfico *boxplot*. 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 e o valor do teste de Kruskal-Wallis, com um valor p igual a 0.000, indica que existem diferenças estatisticamente significativas entre os grupos de linguagens analisadas.

Em suma, com base nessas observações, pode-se inferir que os repositórios escritos em linguagens tipadas tendem a apresentar uma taxa maior de *Code Smells* e Complexidade Ciclomática. Essa tendência pode ser atribuída a uma possível estrutura de código mais complexa nessas linguagens, o que consequentemente os torna mais propensos a ter problemas de qualidade no código. Em comparação, os repositórios de linguagens não tipadas parecem ter uma qualidade superior, com uma menor incidência de *Code Smells* e 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 à métrica QP.2, os resultados são representados nas figuras abaixo.

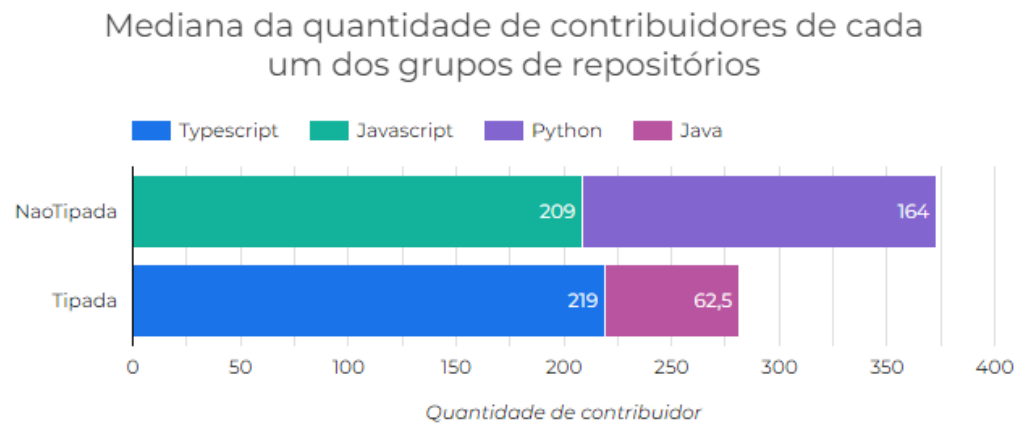

Figura 10. Mediana da quantidade de *CodeSmells* de cada um dos grupos de repositórios

No gráfico de barras composto acima, Figura 10, é possível visualizar a mediana da quantidade de contribuidores nos diferentes grupos de repositórios, o eixo x representa o número de contribuidores em escala logarítmica, enquanto o eixo y mostra as quatro linguagens de pesquisa. Observa-se que as medianas para os repositórios escritos em Java e TypeScript estão juntas no eixo das linguagens tipadas, com valores de mediana de 62,5 e 219, respectivamente. Já para Python e JavaScript, as medianas estão juntas no eixo das linguagens não tipadas, com valores de mediana de 164 e 209, respectivamente.

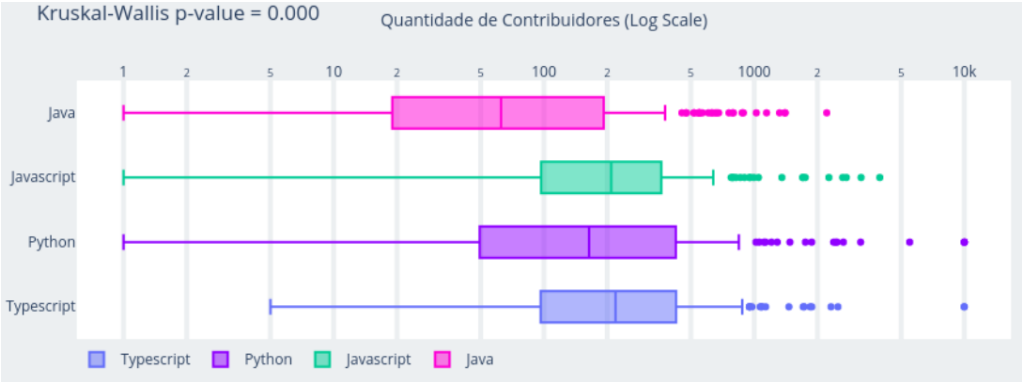

Figura 11. Distribuição do conjunto das medianas da quantidade de contribuidores de linguagens tipadas e não tipadas

A [Figura 11], 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 estatístico de Kruskal-Wallis confirmou a diferença significativa entre os grupos com valor p = 0.000.

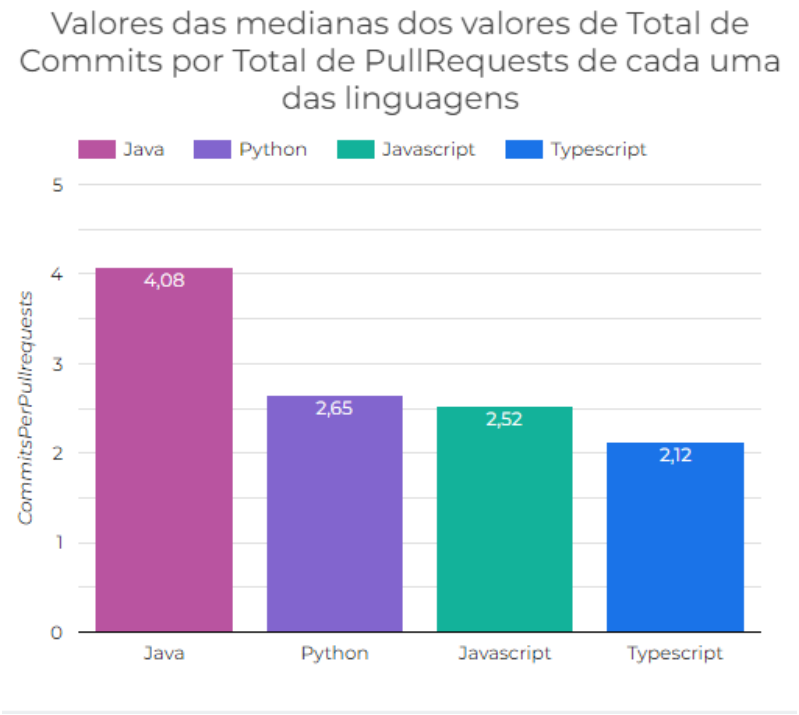

Figura 12. Mediana dos valores de *TotalCommits/TotalPullRequests* dos grupos de repositórios

A segunda métrica de QP.2 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. Para isso, um gráfico de barras da [Figura 12] foi utilizado para mostrar as medianas dos valores. Os resultados revelaram que as linguagens tipadas, como Java e TypeScript, apresentaram medianas mais altas (Java = 4,08 e TypeScript = 2,12) em comparação com as linguagens não tipadas, como Python e JavaScript (Python = 2,65 e JavaScript = 2,52).

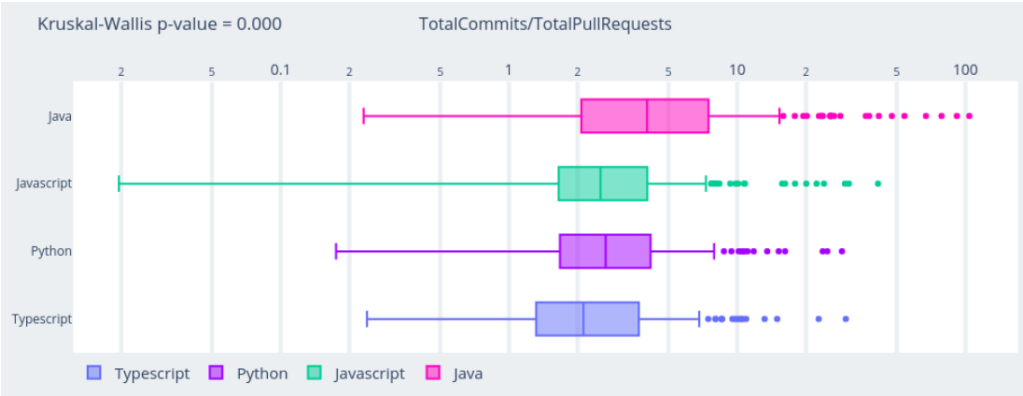

Figura 13. Distribuição do conjunto das mediana dos valores de *TotalCommits/TotalPullRequests* dos grupos de repositórios

Com base nos 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 iden-

tificar 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.

Portanto, esses resultados sugerem que os repositórios de linguagens tipadas tendem a atrair mais contribuidores e a ter uma maior atividade de desenvolvimento em comparação com 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 tamanho da participação da comunidade.

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 QP.3, os dados encontrados estão apresentados na Figura a seguir.

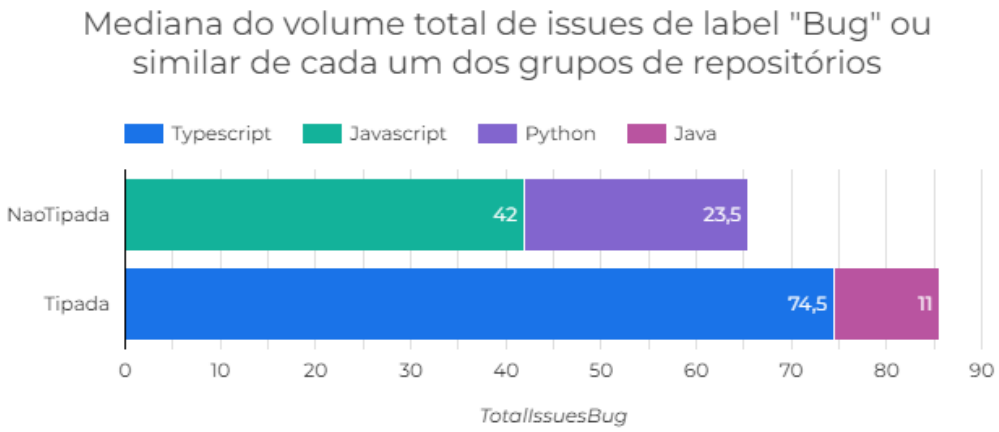

Figura 14. Mediana do volume total de *issues* de *label "Bug"* de cada um dos grupos de repositórios

Observando os resultados do gráfico de barra da [Figura 14] das medianas dos volumes totais de problemas relacionados a *bugs* nos repositórios de linguagens tipadas e não tipadas, pode-se perceber que as linguagens tipadas, como Java e TypeScript, possuem medianas maiores em relação ao número de problemas relacionados a *bugs*, indicando uma maior dificuldade da correção desses *bugs*. Para o Java, a mediana é de 11, enquanto para o TypeScript é de 74,5. Por outro lado, as linguagens não tipadas, como Python e JavaScript, apresentam medianas mais baixas, indicando uma menor da incidência de problemas relacionados a *bugs*. Para o Python, a mediana é de 23,5, enquanto para o JavaScript é de 42.

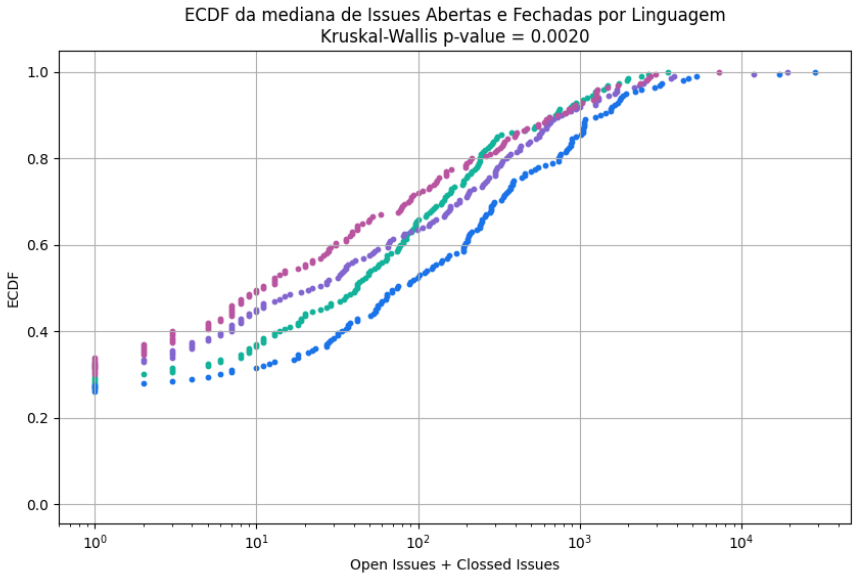

Figura 15. ECDF da mediana de *Issues* Abertas e Fechadas por Linguagem

O gráfico de função de distribuição cumulativa empírica (ECDF) da [Figura 15] 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 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.

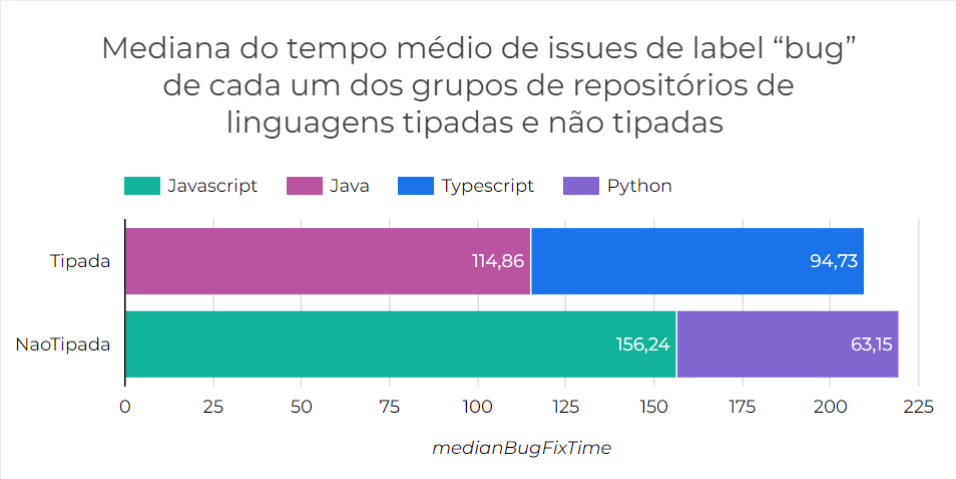

Figura 16. Mediana do tempo médio de *issues* de *label "bug"* de cada um dos grupos de repositórios de linguagens tipadas e não tipadas

Na figura 16 está o gráfico da mediana do tempo médio de *issues* de *label "bug"* de cada um dos grupos de repositórios de linguagens tipadas e não tipadas. Analisando

os resultados, podemos observar que as linguagens tipadas, como Java e TypeScript, possuem medianas menores, indicando um tempo médio de correção de *bugs* mais curto. Para o Java, a mediana é de 114,86, enquanto para o TypeScript é de 94,73. Por outro lado, as linguagens não tipadas, como Python e JavaScript, apresentam medianas mais altas, sugerindo um tempo médio de correção de *bugs* mais longo. Para o Python, a mediana é de 63,15, enquanto para o JavaScript é de 156,24.

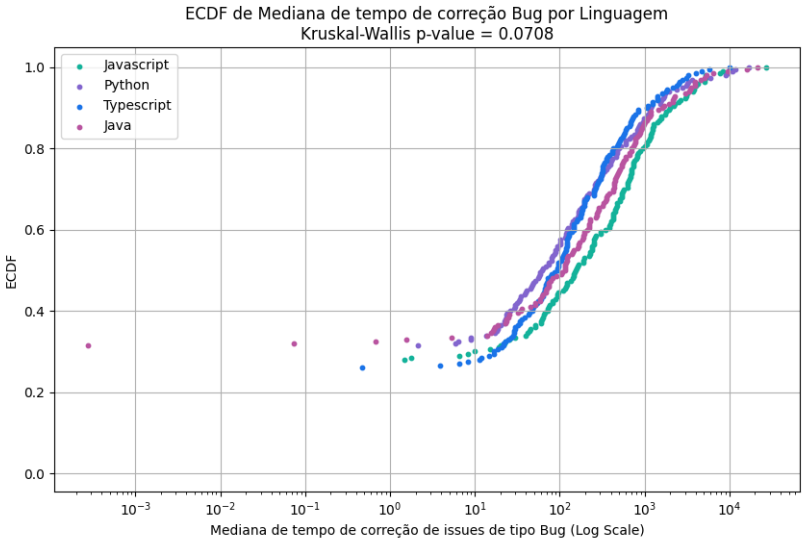

Figura 17. ECDF de Mediana de tempo de correção *Bug* por Linguagem

Por fim, a Figura 17 representa de função de distribuição cumulativa empírica (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*.

Em resumo dos resultados, é possível inferir que 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*.

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

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.

Parnas, D. and Lawford, M. (2003). The role of inspection in software quality assurance. *IEEE Transactions on Software Engineering*, 29(8):674–676.

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.