# Indicadores de Produtividade de Desenvolvedores de Software com Base em Métricas Git: Um Estudo Exploratório

**Eduardo Bandeira de Melo Guimarães** [1]

[1]Instituto de Ciências Exatas e Informática
Pontifícia Universidade Católica de Minas Gerais (PUC-Minas)
Rua Cláudio Manoel, 1162 – Savassi – Belo Horizonte – MG – Brazil – 30.140-100

ebmguimaraes@sga.pucminas.br

***Abstract.*** *Even after several decades of research and development in Software Engineering, measuring the individual productivity of developers remains a challenge. As a consequence, the assessment of this individual productivity becomes impaired, potentially leading to losses for development teams and their leaders. In this context, this work aims to investigate which Git metrics could support the analysis and research on the individual productivity of software developers. To this end, data were collected on the contributions of developers from open-source projects, examining which Git metrics show a direct relationship with those metrics typically used to measure individual productivity. The results indicate that the number of pull requests opened and the number of issues closed are metrics with potential to underpin productivity analysis, whereas the LOC metric showed no correlation with this dimension.*

***Resumo.*** *Mesmo após algumas décadas de pesquisa e desenvolvimento da Engenharia de Software, medir a produtividade individual de desenvolvedores ainda é um desafio. Como consequência, a medição dessa produtividade individual torna-se comprometida, podendo acarretar em perdas para equipes de desenvolvimento e para seus líderes. Nesse contexto, este trabalho busca verificar quais métricas Git poderiam dar corpo à análise e à pesquisa sobre produtividade individual de desenvolvedores de software. Para isso foi conduzida uma coleta de dados sobre contribuições de desenvolvedores de projetos open-source, verificando quais métricas Git apresentam relação direta com aquelas métricas tipicamente utilizadas para medir produtividade individual. Verificou-se que a quantidade de pull requests abertas e a quantidade de issues fechadas são métricas com potencial para embasarem a análise de produtividade, enquanto que a métrica LOC não apresentou correlação com essa dimensão.*

**Bacharelado em Engenharia de *Software* - PUC Minas**
**Trabalho de Conclusão de Curso (TCC)**

Orientador de conteúdo (TCC I): Aline Brito - alinebrito@pucminas.br
Orientador de conteúdo (TCC I): Leonardo Vilela - leonardocardoso@pucminas.br
Orientador acadêmico (TCC I): Cleiton Tavares - cleitontavares@pucminas.br
Orientador do TCC II: Michelle Hanne Soares de Andrade - michelleandrade@pucminas.br

Belo Horizonte, 23 de novembro de 2025.

## 1. Introdução

Sistemas de software são, tipicamente, produzidos em equipes multidisciplinares e, a depender da complexidade do sistema em questão, essas equipes podem apresentar um número considerável de pessoas. Nesse contexto, uma das variáveis essenciais de se considerar para um bom funcionamento dessas equipes é a produtividade dos desenvolvedores, agentes centrais da entrega do produto. Essa medição é fundamental uma vez que, em muitos casos, empresas não podem aumentar a velocidade, volume e qualidade da produção de software simplesmente aumentando seus times [Ruvimova et al. 2022]. Além disso, essas métricas são fundamentais para gestores monitorarem o fluxo de trabalho e, assim, serem capazes de detectar gargalos e pontos de melhoria de eficiência. Entender o que afeta a produtividade de desenvolvedores pode ajudar organizações a tomarem decisões sábias de investimento no ambiente técnico e social delas [Cheng et al. 2022], além de permitir fazer software de mais alta qualidade e de forma mais eficiente [Forsgren et al. 2021b].

Entretanto, o tema da medição da produtividade individual de desenvolvedores é bastante duvidoso. Existe um entendimento amplo de que avaliar a produtividade individual de desenvolvedores de software de forma eficiente e justa é crítico [Forsgren et al. 2021b]. O entendimento mais comum e aceito para embasar essa afirmação é o de que nem todo trabalho de um desenvolvedor pode ser medido. Exemplos clássicos incluem: o tempo dedicado a auxiliar outro desenvolvedor, a realizar *peer programming* ou mesmo a participar de sessões importantes de *brainstorming*. Koskinen (2024) afirma que quantificar entradas e saídas de um trabalho que consiste em criar soluções criativas é difícil e depende dos objetivos do ambiente de trabalho. Forsgren et al. (2021) mencionam que, mesmo após décadas de pesquisa e experiência prática, **saber medir a produtividade de desenvolvedores, ou mesmo defini-la ainda permanece elusivo**.

A produtividade de desenvolvedores de software e a qualidade do produto desenvolvido são aspectos diferentes porém relacionados no âmbito da Engenharia de Software [Storey et al. 2022]. Além disso, gerentes de times de desenvolvimento buscam, com frequência, monitorar a produtividade dos desenvolvedores para planejamento de curto prazo e para medir a produtividade de equipes no médio prazo [Shah et al. 2023]. Forsgren at al. (2021) citam ainda a possibilidade de essas métricas serem proveitosas até mesmo para os próprios desenvolvedores poderem fazer uma autoavaliação de suas produtividades individuais. Assim, entende-se a relevância de existirem estudos aprofundados que validem ou contestem de forma crível a utilização de determinadas métricas como indicadores da produtividade ou mesmo de atividade, de forma a apoiar metodologias de avaliação de produtividade. Dessa forma, a tomada de decisão por parte de gestores de projetos e a autoavaliação de desenvolvedores pode ser feita de maneira mais precisa e confiável.

Nesse contexto, este estudo tem como objetivo **investigar a relação entre métricas Git de atividade de desenvolvedores e métricas ágeis consolidadas, com o propósito de identificar possíveis indicadores complementares de produtividade na Engenharia de Software**. A partir disso, são buscados os seguintes objetivos específicos: i) caracterizar métricas de atividade disponíveis na plataforma GitHub; ii) identificar os maiores contribuidores de projetos *open-source* populares na plataforma e iii) calcular

as métricas ágeis desses desenvolvedores com base nos dados coletados e verificar a correlação delas com métricas Git de atividade, coletadas do GitHub, com o propósito de sinalizar possíveis indicadores complementares de produtividade.

Ao final do estudo, espera-se encontrar métricas Git que possam complementar os indicadores ágeis de produtividade consolidados, ou dar suporte a metodologias que utilizam métricas de atividade como insumo para avaliar a produtividade de desenvolvedores. Com isso, busca-se fornecer a gerentes de projetos e líderes de equipes um leque maior e mais robusto de métricas, permitindo, assim, uma análise possivelmente mais assertiva e segura dos resultados de produtividade. Além disso, busca-se, também, fornecer análises sólidas e robustas para sustentar trabalhos futuros que estudem a produtividade individual de desenvolvedores de software.

O restante do artigo está organizado da seguinte forma: na Seção 2 encontra-se a Fundamentação Teórica para a elaboração do estudo. Na Seção 3 são trazidos Trabalhos Relacionados ao tema. Na Seção 4 são listados os Materiais e Métodos utilizados na condução deste. A Seção 5 apresenta os Resultados do experimento e discussões. Já a Seção 6 aponta as Ameaças à Validade e, a Seção 7, a Conclusão do trabalho.

## 2. Fundamentação Teórica

Nesta seção são apresentados os principais conceitos sobre o tema do estudo realizado.

### 2.1. Produtividade

Segundo Ferreira (2010), produtividade diz respeito ao volume ou total produzido. Ainda segundo o autor, é sinônimo de 'rendimento', que significa, dentre outras definições, o resultado satisfatório obtido por um trabalhador manual ou intelectual. Entende-se portanto que o conceito define uma medida da entrega daquilo que é esperado de um trabalhador. No contexto da Engenharia de Software pode-se dizer que, de um desenvolvedor, espera-se entregas das funcionalidades que lhe foram delegadas e, como consequência, sua produtividade pode ser medida pelo volume dessas entregas, realizadas em conformidade com o grau de qualidade esperado.

### 2.2. Métricas Git

São indicadores quantitativos extraídos do histórico de um repositório Git que descrevem como o código é desenvolvido, modificado e mantido ao longo do tempo. Elas são obtidas a partir de atividades e ações realizadas dentro do ambiente Git/GitHub, como *commits*, *branches*, *pull requests*, *issues* e revisões de código, e servem para analisar aspectos do processo de desenvolvimento de software. Alguns exemplos incluem: *commits* por semana, *pull requests* por *sprint*, *pull requests* aceitas / *pull requests* propostas, etc.

### 2.3. Métricas Ágeis de Produtividade

O Scrum não define métricas de produtividade [Schwaber 1997], tampouco o Manifesto Ágil, que foi posteriormente criado para formalizar os princípios desse [Beck et al. 2001]. Entretanto, algumas métricas se tornaram consagradas pelo uso no mercado de desenvolvimento de software com o objetivo de avaliar equipes inseridas dentro dos processos guiados por metodologias ágeis. Em alguns casos essas métricas são adaptadas para se avaliar o desempenho individual de desenvolvedores. Alguns exemplos de métricas, como

levantadas por Akhiwy et al. (2025) incluem o *Throughput* e o *Cycle Time*, que medem, respectivamente, a vazão de entregas em um determinado período e o tempo gasto entre o início e o término de determinada tarefa. Em alguns casos também é utilizada a métrica de Trabalho Efetivo, que mede a proporção de mudanças aceitas em relação ao conjunto de mudanças propostas por um desenvolvedor.

## 3. Trabalhos Relacionados

Nesta seção são apresentados os trabalhos relacionados à tentativa de propor e validar métricas e indicadores de produtividade de desenvolvedores de *software*.

Através do estudo desenvolvido por Forsgren et al. (2021), os autores propuseram um *framework* para mensurar a produtividade de desenvolvedores com base em múltiplos fatores que fugissem de análises de métricas que consideravam rasas para fazer esse tipo de avaliação. Assim, a metodologia leva em consideração indicadores combinados de 5 aspectos diferentes, sendo eles: satisfação e bem-estar, performance, atividade, comunicação e colaboração e, por fim, eficiência e fluxo para mensurar essa medida de forma mais completa e fidedigna. Ele foi desenvolvido com base em décadas de pesquisa e experiência prática e foi validado por meio da aplicação em diversos cenários comuns. O modelo não determina métricas para cada aspecto, apenas lista algumas sugestões e exemplos. Nesse contexto, o presente estudo tem como objetivo complementar esse trabalho, analisando quais métricas do aspecto de atividade de desenvolvedores se relacionam mais fortemente com a produtividade e, assim, são mais recomendadas para serem escolhidas.

No trabalho de Xia et al. (2022) , os autores buscaram fazer uma análise aprofundada sobre a atividade de desenvolvedores de *software* no GitHub. Para isso, foi realizada uma análise em larga escala dos dados da plataforma 'Fluxo de Eventos do GitHub' (do inglês, 'GitHub Event Stream'), baseando-se em dados de 860 milhões de eventos registrados no ano de 2020. Os pesquisadores obtiveram diversas informações sobre distribuição geográfica dos usuários mais ativos, tempo ativo por dia e que a grande maioria segue a janela de horário comercial dos Estados Unidos e Europa. Esse estudo servirá como base para este ao propor critérios para avaliação de um usuário da plataforma como ativo ou inativo, durante a fase de limpeza dos dados e remoção dos usuários considerados inativos anteriormente à análise dos resultados.

Buffardi (2020) buscou encontrar maneiras objetivas para mensurar a contribuição de alunos em projetos de desenvolvimento de *software* em um contexto acadêmico. Para isso, o autor avaliou as contribuições de 42 alunos em 10 equipes de desenvolvimento, usando dados do histórico do sistema de controle de versão (Git) e de histórias de usuários de seus quadros de gerenciamento de projetos (Kanban). O estudo foi conduzido durante o ano letivo de 2018-2019 em um curso de Engenharia de Software na California State University, Chico. Como conclusão, o autor recomenda que se utilize *commits* e histórias de usuário como complemento às avaliações subjetivas de instrutores e pares. O presente estudo irá complementar essa pesquisa, verificando o quão confiáveis e precisas são as avaliações quantitativas de contagem de *commits* como medida de produtividade e, automaticamente, de contribuição.

Em um contexto corporativo, Storey et al. (2022) abordaram o problema da divergência entre desenvolvedores e gerentes na forma de avaliar produtividade e qualidade. No estudo, os autores buscaram encontrar uma metodologia que conciliasse ambos os

pontos de vista. Na condução do experimento, foi feita uma pesquisa exploratória por meio de questionário com 1.500 desenvolvedores e 720 gerentes da Microsoft®. Os pesquisadores concluíram que uma abordagem ideal seria a mesclagem de dois *frameworks* de avaliação de produtividade: SPACE [1] e TRUCE [2]. As conclusões dessa pesquisa deram suporte à condução deste estudo ao prover metodologias confiáveis e robustas que serviram como referência para se avaliar outras métricas de produtividade, sendo parte das métricas propostas possíveis de serem obtidas pela plataforma GitHub.

De forma semelhante, Sanwal e Deva (2024) propõem, a partir de uma abordagem multifacetada, métricas e indicadores para se avaliar a produtividade de equipes de desenvolvimento de software. Os autores validaram os produtos propostos através de um experimento com sete equipes diferentes ao longo de sete *sprints* com coletas automáticas de dados através do uso de *plugins*. Como resultado, validaram positivamente a metodologia para rastrear e melhorar o desempenho das equipes ao longo do tempo, identificando tendências que correspondiam às mudanças nas práticas das equipes. Algumas das métricas propostas servirão de base como indicadores de produtividade para viabilizar a validação de outras métricas a serem verificadas.

## 4. Materiais e Métodos

Este estudo é classificado como aplicado, quantitativo e de natureza explicativa. Nele, foram coletados e analisados dados da plataforma GitHub de forma a comparar e verificar métricas de atividade como complementares àquelas consolidadas para medição de produtividade de desenvolvedores ou como insumo para a realização dessa avaliação.

Para realizar a análise das métricas Git selecionadas, descritas na seção 4.2, foram elaboradas três questões de pesquisa (QP):

**QP1.** Existe correlação significativa entre as métricas Git de atividade de desenvolvedores e o *Throughput* desses desenvolvedores?

- $\mathbf{H_0}$ **(Hipótese nula):** Não existe correlação estatisticamente significativa entre as métricas Git de atividade de desenvolvedores e o *Throughput* desses desenvolvedores.
- $\mathbf{H_1}$ **(Hipótese alternativa):** Existe correlação estatisticamente significativa entre, pelo menos, uma das métricas Git de atividade de desenvolvedores e o *Throughput* desses desenvolvedores.

Essa questão de pesquisa avalia se alguma das métricas Git de atividade de desenvolvedores é consideravelmente sensível à métrica *Throughput* e, assim, poderia ser utilizada como possível métrica complementar para a avaliação de produtividade.

**QP2.** Existe correlação significativa entre as métricas Git de atividade de desenvolvedores e o *Cycle Time* desses desenvolvedores?

- $\mathbf{H_0}$ **(Hipótese nula):** Não existe correlação estatisticamente significativa entre as métricas Git de atividade de desenvolvedores e o *Cycle Time* desses desenvolvedores.

[1] modelo criado pela Microsoft Research para avaliar a produtividade de desenvolvedores e equipes de software de forma ampla

[2] modelo criado para avaliar produtividade de desenvolvedores a partir de métricas provenientes de ferramentas de versionamento

- **$H_1$ (Hipótese alternativa):** Existe correlação estatisticamente significativa entre, pelo menos, uma das métricas Git de atividade de desenvolvedores e o *Cycle Time* desses desenvolvedores.

Essa questão de pesquisa avalia se alguma das métricas Git de atividade de desenvolvedores é consideravelmente sensível à métrica *Cycle Time* e, assim, poderia ser utilizada como possível métrica complementar para a avaliação de produtividade ou como insumo a metodologias que utilizam métricas de atividade para fazer avaliação de produtividade.

**QP3.** Existe correlação significativa entre as métricas Git de atividade de desenvolvedores e o Trabalho Efetivo desses desenvolvedores?

- **$H_0$ (Hipótese nula):** Não existe correlação estatisticamente significativa entre as métricas Git de atividade de desenvolvedores e o Trabalho Efetivo desses desenvolvedores.
- **$H_1$ (Hipótese alternativa):** Existe correlação estatisticamente significativa entre, pelo menos, uma das métricas Git de atividade de desenvolvedores e o Trabalho Efetivo desses desenvolvedores.

Essa questão de pesquisa avalia se alguma das métricas Git de atividade de desenvolvedores é consideravelmente sensível à métricas de produtividade Trabalho Efetivo e, assim, poderia ser utilizada como possível métrica complementar para a avaliação de produtividade.

### 4.1. Arranjo Experimental

Para a condução da pesquisa, foram desenvolvidos *scripts* na linguagem de programação Python, com a utilização de bibliotecas diversas, para coletar e tratar os dados disponibilizados pela API do GitHub e, então realizar as análises estatísticas relevantes para o estudo. Na Figura 1 encontra-se a representação visual das atividades desenvolvidas ao longo do experimento. Em um primeiro momento, buscou-se os repositórios mais populares da plataforma em número de estrelas, descartando aqueles arquivados, inativos (mais de 30 dias sem atividade), aqueles nos quais o tamanho excessivo impedia que a API enviasse os dados completos, ou cujo tempo de existência não coincidia com a janela de estudo (01/01/2024 até 31/10/2025). Desses, coletou-se os primeiros 250 repositórios filtrados. Então, foi realizada outra varredura para listar os 10 maiores contribuidores para cada um desses repositórios, desconsiderando os contribuidores não humanos (*bots*). O GitHub classifica e ordena essa característica com base no número total de *commits* do desenvolvedor aceitos na *branch* 'main' do repositório.

Em posse da base de repositórios e seus respectivos contribuidores principais, iniciou-se a coleta dos dados de atividades desses contribuidores que estivessem dentro da janela temporal. Os dados coletados foram utilizados para calcular as métricas ágeis de produtividade desses desenvolvedores dentro de intervalos de quatro semanas cada. Então, por fim, esses indicadores foram comparados com as métricas de atividade coletadas, a fim de verificar quais dessas métricas são mais sensíveis às variações de produtividade. Os dados foram coletados entre os dias 24 de outubro de 2025 e 16 de novembro de 2025.

### 4.2. Métricas

Para a realização do experimento, foram determinados dois tipos de métricas: métricas Git, a serem verificadas como possíveis indicadores de produtividade ou atividade de

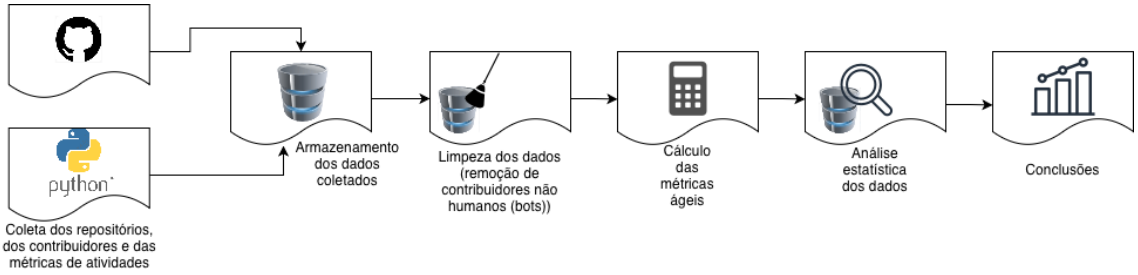


**Figura 1. Arranjo Experimental**

desenvolvedores, e aquelas consolidadas como indicadores ágeis de produtividade. A seleção desses indicadores ágeis de produtividade teve como objetivo assegurar um ponto de referência de forma a verificar uma correlação de valores. As métricas Git de atividade analisadas foram:

- Quantidade de *pull requests* abertas: número total de *pull requests* abertas nos intervalos analisados.
- Quantidade de *commits*: número total de *commits* realizados dentro de cada intervalo observado (4 semanas).
- Complexidade dos *commits*: caracterização dos *commits* realizados de acordo com a quantidade de linhas alteradas (LOC) e arquivos modificados.
- Quantidade de *issues* fechadas: número total de *issues* fechadas nos intervalos analisados.
- Quantidade de comentários: número total de comentários realizados pelo desenvolvedor nos intervalos analisados.

As métricas ágeis de produtividade adotadas foram:

- *Throughput*: é a frequência de tarefas entregues, podendo ser calculado como a quantidade de *pull requests* aceitas ou a quantidade de *issues* fechadas dentro de um intervalo de tempo. Neste estudo, foi considerada a quantidade de *pull requests* realizadas para cada intervalo de 4 semanas.
- *Cycle Time*: tempo demandado para a execução de uma tarefa, contado a partir do início do trabalho para concluí-la (momento do primeiro *commit* da *pull request* fechada) até a conclusão dela (momento do último *commit* da *pull request*).
- Trabalho Efetivo: quantidade relativa de *pull requests* dos contribuidores que foram aceitas pelos mantenedores do projeto em relação às abertas.

As métricas ágeis de produtividade, utilizadas no estudo como referência para indicadores sólidos e confiáveis de produtividade, foram concatenadas com base em diversos estudos e literatura consagrada da área da Engenharia de *Software*, como mencionado por Akhiwy et al. (2025).

Para uma validação estatística consolidada, empregou-se o teste de normalidade de Shapiro-Wilk e os testes de correlação de Pearson (para amostras normais) e de Spearman (para amostras não-normais) [Barbetta et al. 2010]. Além disso, foi adotado um nível de significância de 0,05 por se tratar de uma convenção amplamente utilizada em pesquisas empíricas, herdada da Estatística clássica a partir dos trabalhos de Fisher (1925). Embora arbitrário, esse valor é tradicionalmente empregado por equilibrar o risco de erro do tipo I e a sensibilidade dos testes estatísticos, permitindo comparabilidade com estudos prévios da literatura.

## 5. Resultados

Nesta seção são apresentados os resultados obtidos ao longo do estudo. Na subseção 5.1 encontra-se a caracterização do conjunto de dados obtidos na coleta. Na subseção 5.2 é feita uma análise e discussão dos resultados.

### 5.1. Caracterização do Conjunto de Dados

Foram analisados dados de atividades de contribuidores em 250 repositórios, estando entre os mais populares do GitHub. Podem-se citar os repositórios 'freeCodeCamp/freeCodeCamp', 'EbookFoundation/free-programming-books' e 'kamranahmedse/developer-roadmap' como exemplos notórios de repositórios analisados, ocupando eles, respectivamente, o primeiro, o quarto e o sexto lugares dentre os repositórios mais populares do GitHub.

Dentre esses repositórios analisados, foi encontrada uma média de 79.406 estrelas com desvio padrão de 46.874,5 e uma mediana de 66.789 estrelas. Os valores máximo e mínimo da quantidade de estrelas dos repositórios analisados foram de 430.237 e 44.725, respectivamente. A Figura 2 apresenta uma representação da distribuição das proporções percentuais para diferentes idades dos repositórios analisados. Observa-se que a maioria dos repositórios apresenta mais de 10 anos de criação. A média de tempo desde a criação do conjunto de repositórios analisados foi de aproximadamente 8 anos e 5 meses com desvio padrão de 4,3; enquanto que a mediana dessa grandeza foi de, aproximadamente, 9 anos e 10 meses. O repositório mais antigo analisado foi criado em 11/04/2008, enquanto que, o mais recente, em 26/09/2023.

Na Figura 3 encontra-se a representação das 10 linguagens de programação principal mais observadas entre os repositórios analisados. Nota-se que essa listagem assemelha-se muito com o último *benchmark*[3] lançado pela plataforma sobre as linguagens de programação mais populares dela. Evidencia-se também a discrepância entre o primeiro lugar (Python) em relação aos demais.

Foram analisados dados sobre contribuições de um total de 2500 desenvolvedores, sendo 10 por repositório. A Figura 4 apresenta um *boxplot* da distribuição da quantidade de *pull requests* analisadas para cada repositório selecionado. Foram analisadas um total de 241.278 *pull requests*, com uma média de, aproximadamente, 965 *pull requests* por repositório e desvio padrão de aproximadamente 1.483 *pull requests*. Nessa representação, é possível observar o dado anteriormente mencionado, de que a distribuição da quantidade de *pull requests* analisadas entre os repositórios foi bastante irregular, o que pode ser explicado por tamanhos de projetos e números de contribuições muito heterogêneas. Foi a partir dessa atividade que calculou-se o *Throughput* de cada desenvolvedor, o *Cycle Time* das atividades desempenhadas por ele, o Trabalho Efetivo, a quantidade de *commits* realizados e também a complexidades desses.

A quantidade total de *commits* analisados foi de 1.085.839, o que representou médias de, aproximadamente, 4.343 *commits* por repositório e 434 *commits* por desenvolvedor. Sobre a complexidade dos *commits* analisados, foi verificado um LOC total igual a 1.568.502.236 e um total de 11.616.608 arquivos modificados (criados + removidos). Esses dados sobre as complexidades dos *commits* representaram uma média de

[3]https://github.blog/news-insights/octoverse/octoverse-a-new-developer-joins-github-every-second-as-ai-leads-typescript-to-1/

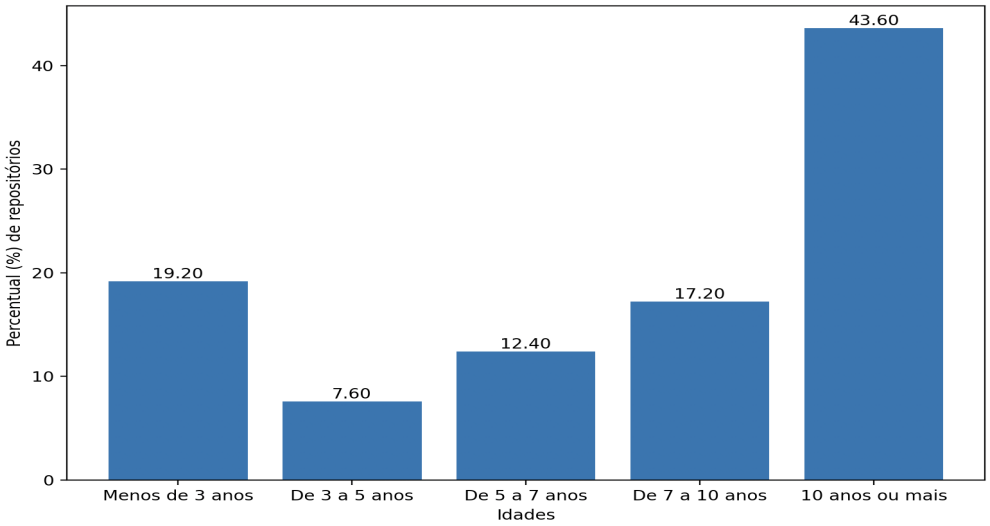


**Figura 2. Proporção dos tempos desde a criação dos repositórios analisados**

6.274.009 linhas alteradas por repositório e 627.401 linhas alteradas por desenvolvedor, ao longo do período analisado. Em relação aos arquivos modificados no mesmo período, os dados apontam para uma média de 46.466 alterações por repositório e 4.646 por desenvolvedor. Foi analisado um total de 43.442 *issues* fechadas pelos contribuidores. Por fim, foram analisados também dados relativos aos comentários realizados pelos desenvolvedores estudados, tanto em *pull requests* quanto em *issues*. O total de atividades desse tipo encontradas foi igual a 360.681. Desses comentários, 53,45% foram realizados em *pull requests* enquanto que 46,55% foram realizados em *issues*.

## 5.2. Análise dos Resultados

Esta subseção discute criticamente os resultados obtidos, relacionando-os às três questões de pesquisa e às respectivas hipóteses.

Para responder às questões de pesquisa, calcularam-se as médias das métricas Git e das métricas ágeis apresentadas por todos os desenvolvedores estudados, em cada intervalo. Depois, foram verificadas as distribuições dessas médias de forma a determinar o teste de hipótese a ser utilizado para cada par de métricas (Git x Ágil). Os *p-valores* encontrados para cada conjunto de médias estão representados na Tabela 1. Para verificar essa distribuição, foi realizado o teste de normalidade de Shapiro-Wilk sobre cada um desses conjuntos.

Com base nesses resultados e partindo-se um valor de $\alpha$ de 0,05, verificou-se que apenas os conjuntos de médias de *Cycle Times* e *commits* não apresentaram uma distribuição normal. Dessa forma, determinou-se que seria utilizado o teste de Pearson para verificação de correlação entre *Throughputs* e métricas Git (exceto para *commits*), assim como para verificar a correlação entre Trabalho Efetivo e métricas Git (exceto para *commits*). Para a verificação das correlações entre *Cycle Times* e as métricas Git, e também para as exceções anteriormente citadas, foi escolhido o teste de Spearman. Os resultados dos testes de hipótese realizados estão apresentados na Tabela 2, onde estão descritos os p-valores encontrados para cada teste de correlação estatística e o coeficiente de correlação para cada teste ($r$ quando para Pearson e $\rho$ quando para Spearman).

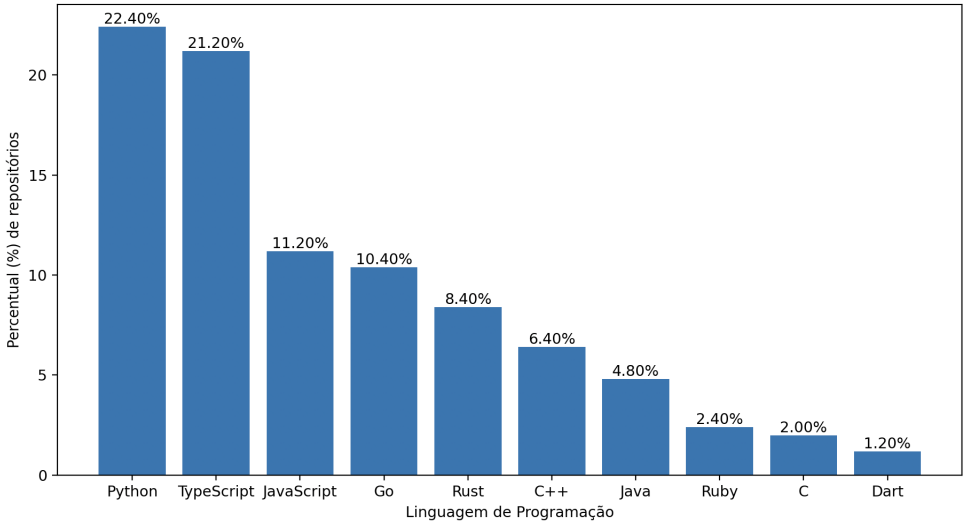


**Figura 3. Proporção das linguagens principais dos repositórios analisados**

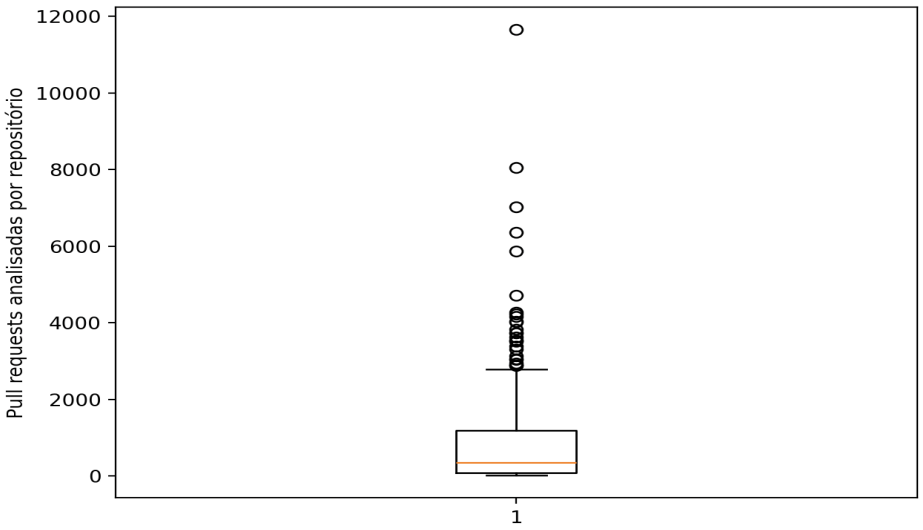


**Figura 4. Distribuição das quantidades de pull requests analisadas por repositório**

Partindo-se dos resultados encontrados, buscou-se, então, responder à QP1. Na Figura 5 encontra-se uma representação visual dos dados obtidos e descritos na Tabela 2 para a primeira coluna (*Throughput*). Ao analisar os dados apresentados nessa tabela para a coluna de *Throughput*, observa-se a existência de significância estatística para se rejeitar a hipótese nula para as seguintes métricas Git de atividade: i) quantidade de *pull requests* abertas; ii) quantidade de comentários realizados e iii) quantidade de arquivos modificados. Dessa forma, como pelo menos uma métrica Git apresentou correlação com o *Throughput*, rejeitou-se a hipótese nula da QP1. Os coeficientes de correlação para essas métricas foram, respectivamente, 0,95, 0,77 e 0,63, indicando correlação muito forte para *pull requests* abertas, e fortes para quantidade de *commits* realizados e para a quantidade de arquivos modificados.

Na Figura 6 encontra-se uma representação visual dos dados obtidos e descritos na

**Tabela 1. Resultados dos testes de distribuição**

| **Métricas** | ***p-valores*** |
|---|---|
| *Throughput* | 0,49 |
| *Cycle Time* | $2{,}28 \times 10^{-5}$ |
| Trabalho Efetivo | 0,24 |
| Quantidade de *pull requests* abertas | 0,84 |
| Quantidade de *commits* realizados | 0,04 |
| Quantidade de linhas alteradas | 0,34 |
| Quantidade de arquivos modificados | 0,06 |
| Quantidade de *issues* fechadas | 0,83 |
| Quantidade de comentários realizados | 0,92 |

**Tabela 2. Resultados dos testes de hipótese**

| **Métricas Git / Métricas Ágeis** | *Throughput* | Cycle Time | Trabalho Efetivo |
|---|---|---|---|
| Quantidade de *pull requests* abertas | $r$ = 0,95<br>*p-valor*=$2{,}37 \times 10^{-12}$ | $r$ = -0,17<br>*p-valor* = 0,42 | $r$ = 0,41<br>*p-valor* = 0,049 |
| Quantidade de *commits* realizados | $\rho$ = 0,77<br>*p-valor* =$1{,}02 \times 20^{-5}$ | $\rho$ = 0,24<br>*p-valor* = 0,26 | $\rho$ = 0,074<br>*p-valor* = 0,73 |
| Quantidade de linhas alteradas | $r$ = 0,06<br>*p-valor* = 0,77 | $r$ = 0,40<br>*p-valor* = 0,055 | $r$ = -0,39<br>*p-valor* = 0,055 |
| Quantidade de arquivos modificados | $r$ = 0,63<br>*p-valor* = 0,0009 | $r$ = 0,02<br>*p-valor* = 0,92 | $r$ = 0,16<br>*p-valor* = 0,45 |
| Quantidade de *issues* fechadas | $r$ = 0,12<br>*p-valor* = 0,58 | $r$ = -0,65<br>*p-valor* = 0,00049 | $r$ = 0,65<br>*p-valor* = 0,0006 |
| Quantidade de comentários realizados | $r$ = 0,34<br>*p-valor* = 0,09 | $r$ = 0,56<br>*p-valor*= 0,0044 | $r$ = -0,30<br>*p-valor* = 0,15 |

Tabela 2 para a segunda coluna (*Cycle Time*). Tendo como base os resultados apresentados nessa tabela, verificou-se que o teste de hipótese de correlação entre o *Cycle Time* e as métricas Git de atividade apresentou significância estatística relevante para as seguintes métricas: i) quantidade de *issues* fechadas e ii) quantidade de comentários realizados. Assim, rejeitou-se a hipótese nula da QP2. Os coeficientes encontrados para essas duas métricas evidenciaram uma correlação moderada entre elas e os resultados de *Cycle Time*, sendo uma relação inversa para a quantidade de *issues* fechadas e direta para a quantidade de comentários realizados. Esses resultados indicam que, de forma moderada, quanto maior o número de *issues* fechadas por um desenvolvedor, menor é o tempo que este leva entre o início e o fim de suas tarefas. Por outro lado, os resultados indicam que, também de forma moderada, um número maior de comentários realizados por um desenvolvedor está associado a uma demora maior na conclusão de suas tarefas.

Por fim, foram analisados os resultados dos testes de hipótese, apresentados na Tabela 2 para responder à QP3, cujos coeficientes de correlação descritos na terceira coluna (Trabalho Efetivo) estão visualmente representados na Figura 7. Ao analisá-la, foi

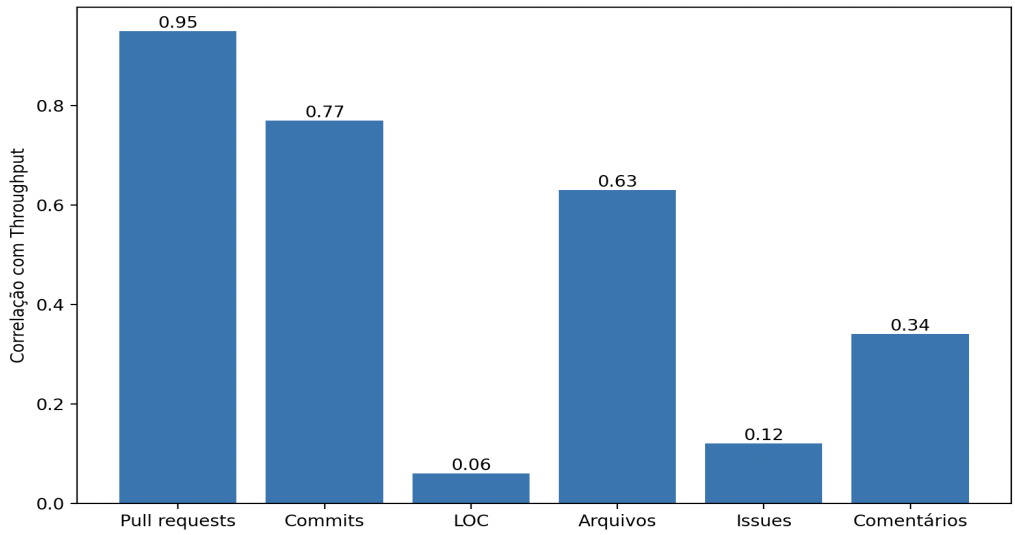


**Figura 5. Correlações entre *Throughput* e métricas Git**

possível observar que os resultados são estatisticamente relevantes para as métricas git: i) quantidade de *pull requests* abertas e ii) quantidade de *issues* fechadas. Assim, a hipótese nula apresentada por essa questão de pesquisa foi rejeitada, pois pelo menos uma métrica Git de atividade apresentou correlação com o Trabalho Efetivo alcançado pelos desenvolvedores analisados. Os coeficientes de correlação encontrados foram, respectivamente, 0,41 e 0,65. Esses valores indicam que a quantidade de *pull requests* abertas por um desenvolvedor se relaciona de forma direta e moderada com seu trabalho efetivo, enquanto que a quantidade de *issues* fechadas se relaciona de forma direta e forte com a mesma métrica de produtividade.

A partir dos resultados anteriormente enumerados, verifica-se que tanto a quantidade de *pull requests* abertas por um desenvolvedor quanto a quantidade de *issues* fechadas estão diretamente relacionadas com duas das três métricas de produtividade utilizadas como indicadores de produtividade. Além disso, nota-se que a métrica LOC não apresenta relação com nenhum dos indicadores de produtividade adotados.

## 6. Ameaças à Validade

Todo estudo empírico em Engenharia de Software está sujeito a diferentes ameaças à validade, que precisam ser explicitadas para fortalecer a confiabilidade das conclusões [Wohlin et al. 2012]. Nesta subseção, são discutidas as principais ameaças à validade dos resultados deste estudo e as estratégias adotadas para mitigá-las. As ameaças foram organizadas nas categorias: interna, externa, de construção e de conclusão, conforme a nomenclatura tradicional em estudos empíricos de Engenharia de Software.

**Ameaças à validade interna:** É possível que outros fatores não identificados e não rastreados por este estudo pudessem interferir nos resultados de produtividade utilizados como referência e, assim, causassem distúrbios nas conclusões tomadas. Para mitigar essa ameaça, foram adotadas três indicadores distintos de produtividade de desenvolvedores de software, com forte embasamento na literatura.

**Ameaças à validade externa:** Os resultados encontrados neste estudo poderiam

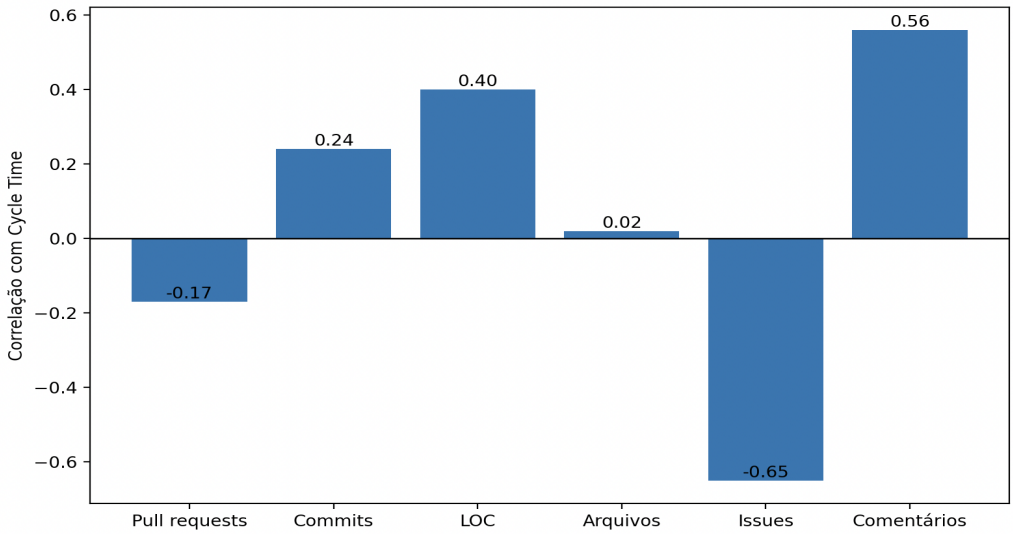


**Figura 6. Correlações entre *Cycle Time* e métricas Git**

não ser replicados para a maioria dos desenvolvedores de software. Para mitigar essa ameaça, buscou-se analisar uma amostra significativa de desenvolvedores, de forma que diferentes perfis de atividade e produtividade fizessem parte do conjunto de dados analisados.

**Ameaças à validade de construção:** Ao tomar indicadores como referências para a produtividade de desenvolvedores, era possível que esses indicadores não representassem bem essa variável. Para mitigar essa ameaça, foram utilizados três indicadores, todos consagrados pelo uso na indústria e amplamente referendados pela literatura.

**Ameaças à validade de conclusão:** Essa ameaça está associada à validade das conclusões estatísticas estabelecidas no estudo. De forma a mitigar essa ameaça, buscou-se realizar os testes de correlação estatística adequados para cada conjunto de dados, aplicando testes de distribuição para todos os conjuntos de dados analisados antes que os testes de hipótese fossem realizados.

## 7. Conclusão

Este trabalho investigou a relação entre seis métricas Git de atividade de desenvolvedores de software e três métricas ágeis de produtividade de desenvolvedores, sendo elas: *Throughput* (QP1), *Cycle Time* (QP2) e Trabalho Efetivo (QP3). Os testes estatísticos aplicados levaram à rejeição de todas as $H_0$ e ao acolhimento das $H_1$.

Em relação à métrica ágil *Throughput*, três das seis métricas Git de atividade avaliadas apresentaram correlação significativa com ela. Foram elas: i) quantidade de *pull requests* abertas, ii) quantidade de *commits* realizados e iii) quantidade de arquivos modificados ao longo de certo intervalo de tempo. Para a métrica ágil *Cycle Time*, duas das seis métricas Git apresentaram correlação direta, sendo elas: i) quantidade de *issues* fechadas e quantidade de comentários realizados. Por último, as métricas Git i) quantidade de *pull requests* abertas e ii) quantidade de *issues* fechadas mostraram indícios fortes de correlação com o chamado Trabalho Efetivo dos desenvolvedores avaliados.

É notório observar que, das métricas Git avaliadas, duas delas se relacionam com

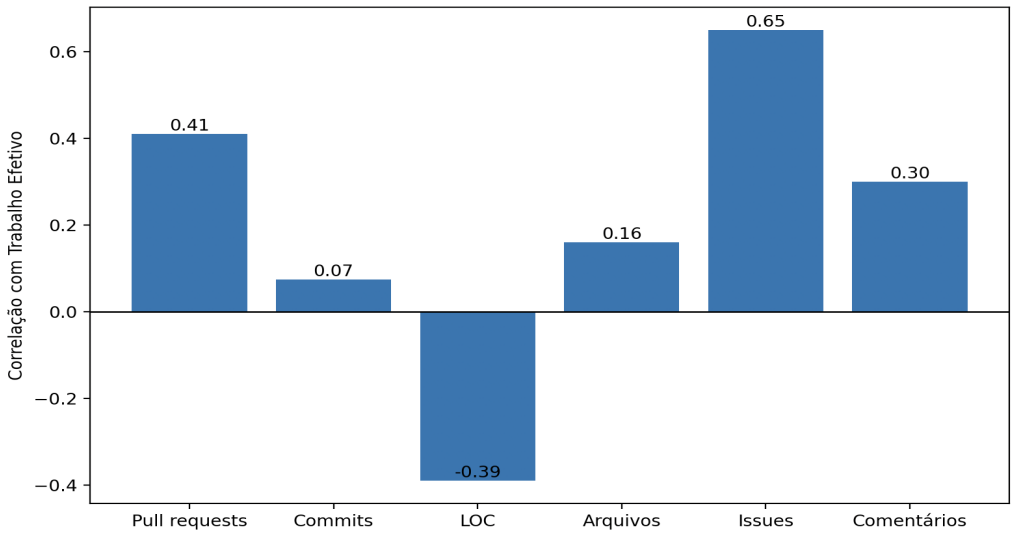


**Figura 7. Correlações entre Trabalho Efetivo e métricas Git**

mais de uma métrica ágil, sendo elas a quantidade de *pull requests* abertas e a quantidade de *issues* fechadas. No outro extremo, nota-se que a quantidade de linhas alteradas (LOC) não apresentou relação direta com nenhuma das métricas de produtividade utilizadas como base para avaliação.

Esses resultados são relevantes tanto para desenvolvedores quanto para líderes de projetos, pois fornecem dados sobre quais métricas são mais relevantes como possíveis indicadores de produtividade individual, favorecendo, assim, insumos tanto para uma autoanálise quanto para uma medição da produtividade de participantes em projetos de desenvolvimento de software. Além de possíveis indicadores de produtividade, esses resultados podem ser utilizados para embasar escolhas de métricas que são utilizadas como insumos para um cálculo mais complexo de produtividade, como descrito pelo *framework* SPACE, citado neste como trabalho-base.

Para trabalhos futuros, é recomendada a realização de uma pesquisa mais ampla e com uma base de projetos ainda mais considerável. Recomenda-se, também, avaliar outros tipos de projetos, indo além daqueles categorizados como *open-source*. Além disso, seria pertinente a avaliação de outras métricas que poderiam ser utilizadas como complementos aos indicadores de produtividade e de um número maior de contribuidores, não se limitando apenas àqueles com um elevado grau de engajamento.

## Pacote de Replicação

O pacote de replicação deste trabalho encontra-se disponível em:

`https://github.com/ICEI-PUC-Minas-PPLES-TI/plf-es-2024-2-tcci-0393100-pes-eduardo-guimaraes`

## Referências

Akhiwu, B. (2025). Operationalizing innovation at scale: Embedding agile and lean principles into enterprise technology workflows. *Department of Business Administration, University of Illinois*.

Barbetta, P. A., Reis, M. M., and Bornia, A. C. (2010). *Estatística para cursos de engenharia e informática*. Atlas, São Paulo, Brazil, 3 edition.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., and Thomas, D. (2001). Manifesto for agile software development. `https://agilemanifesto.org/`. Acessado em: 2025-11-19.

Buffardi, K. (2020). Assessing individual contributions to software engineering projects with git logs and user stories. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, SIGCSE '20, page 650–656, New York, NY, USA. Association for Computing Machinery.

Cheng, L., Murphy-Hill, E., Canning, M., Jaspan, C., Green, C., Knight, A., Zhang, N., and Kammer, E. (2022). What improves developer productivity at google? code quality. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2022, page 1302–1313, New York, NY, USA. Association for Computing Machinery.

Ferreira, A. B. d. H. (2010). *Aurélio: dicionário da língua portuguesa*. Positivo, Curitiba, 5 edition. Verbete: algoritmo.

Fisher, R. A. (1925). *Statistical Methods for Research Workers*. Oliver and Boyd, Edinburgh.

Forsgren, N., Storey, M.-A., Maddila, C., Zimmermann, T., Houck, B., and Butler, J. (2021a). The space of developer productivity. *acmqueue*, 19(1):1–29.

Forsgren, N., Storey, M.-A., Maddila, C., Zimmermann, T., Houck, B., and Butler, J. (2021b). The space of developer productivity: There's more to it than you think. *ACM Queue*, 19(1):20–48.

Koskinen, L. (2025). Setting productivity metrics for automation development teams: From one-dimensional metrics toward a holistic view. Master's thesis, Aalto University School of Business.

Ruvimova, A., Lill, A., Gugler, J., Howe, L., Huang, E., Murphy, G., and Fritz, T. (2022). An exploratory study of productivity perceptions in software teams. In *2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE)*, pages 99–111.

Sanwal, M. and Deva, I. (2024). A multi-faceted approach to measuring engineering productivity. *International Journal of Trend in Scientific Research and Development*, 8:516–521.

Schwaber, K. (1997). Scrum development process. In Sutherland, J., Casanave, C., Miller, J., Patel, P., and Hollowell, G., editors, *Business Object Design and Implementation*, pages 117–134, London. Springer London.

Shah, H. M., Syed, Q. Z., Shankaranarayanan, B., Palit, I., Singh, A., Raval, K., Savaliya, K., and Sharma, T. (2023). Mining and fusing productivity metrics with code quality information at scale. In *2023 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 563–567.

Storey, M.-A., Houck, B., and Zimmermann, T. (2022). How developers and managers define and trade productivity for quality. In *Proceedings of the 15th International Conference on Cooperative and Human Aspects of Software Engineering*, CHASE '22, page 26–35, New York, NY, USA. Association for Computing Machinery.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M., Regnell, B., and Wesslén, A. (2012). *Experimentation in Software Engineering*, pages 123–151.

Xia, X., Weng, Z., Wang, W., and Zhao, S. (2022). Exploring activity and contributors on github: Who, what, when, and where. In *2022 29th Asia-Pacific Software Engineering Conference (APSEC)*, pages 11–20.