Como o Truck Factor afeta indivíduos e interações?*

Gustavo Cunha Teles[1], Adolfo Gustavo Serra Seca Neto[2], and Maria Cláudia Figueiredo Pereira Emer[3]

[1] UTFPR, Curitiba, Paraná, Brasil
gustavoteles@alunos.utfpr.edu.br
[2] UTFPR, Curitiba, Paraná, Brasil
adolfo@utfpr.edu.br
[3] UTFPR, Curitiba, Paraná, Brasil
mcemer@utfpr.edu.br

Abstract. Truck factor (TF) is the number of individuals, whose exit from the project will leave you in a critical state. Critical Coverage Threshold (CCT) measures domain knowledge of the project known by the members, that is, the amount of knowledge about the project, whose metric can be calculated based on commits per developer. The objective of this study is to analyze whether there is a correlation between the overfitting, persistence and rotation indicators of technology professionals in software projects and the reduction of risk of discontinuation of these projects. Several metrics were extracted, including TF, from 175 projects, 35 from 5 large organizations. HTML and Typescript languages had one of the highest TF, around 3, to the detriment of the average TF of 1.8 for programming languages. The projects with most stars have a higher average TF (3.8314) and the average TF of all projects is worth 2.44. The projects with most collaborators have a higher average TF (4.8771) and the average TF of all projects is 1.7263. Most durable projects have a higher average TF (4.3253) and the average TF of all projects is worth 3.2538. From systematic mapping, two tools were found that provided TF value of each project, one entitled Truck-Factor and another entitled BusFactor. A tool was created that would provide other metrics, in addition to TF.

Keywords: Agile Methodologies · Turnorver · Overfitting · Popularity · Duration.

1 Introdução

Segundo [11], *Truck factor* (TF) pode ser visto como uma medida da distribuição do conhecimento sobre o sistema de *software* e é abordado através da comunidade ágil, além de possuir a limitação de estar atrelado ao *critical coverage threshold* (CCT). De acordo com [9], para identificar o TF, um CCT que representa

* UTFPR.

a cobertura crítica de arquivos para o projeto deve ser definido, logo, pode-se considerar a cobertura mínima de cada arquivo, ou seja, no pior caso, os desenvolvedores detém conhecimento apenas sobre o conjunto de arquivos mais importantes.

TF pode ser importante na projeção de mensuração da duração de sistemas. Ademais, pode ser relevante na escolha de entrada, permanência ou até mesmo saída de integrantes do projeto, além de poder ajudar na tomada de decisão do uso de um sistema como provedor de informação ou até mesmo como biblioteca de terceiros.

Este estudo traz uma abordagem prática e baseada em dados sobre risco de falha em projetos *open source*[4] e com código-fonte baseado no *Github*[5]. Este artigo aborda o rodízio de integrantes da equipe de engenharia de *software*, além da falta ou existência, da disseminação de conhecimento técnico entre a equipe e a distribuição da participação dos integrantes em cada projeto, trazendo a perspectiva das metodologias ágeis.

Este artigo está organizado da seguinte forma, Seção 2 aborda conceitos relacionados à pesquisa, 3 apresenta o método de pesquisa, Seção 4 demonstra os resultados, Seção 5 discute os resultados, por fim a Seção 6 apresenta a conclusão.

2 Contexto

TF é o número de indivíduos, cuja saída do projeto o deixará em estado crítico. O CCT mede o percentual do domínio do projeto pelos membros, ou seja, a quantidade de conhecimento sobre o projeto, cuja métrica pode ser calculada a partir de commits por desenvolvedor.

O TF possui dois sinônimos na literatura, o *bus factor* (BF) e o *lottery number* (LN). Independente da nomenclatura, o termo está intrinsecamente atrelado aos temas de *software* livre e de código aberto, e de desenvolvimento de *software* em colaboração.

Vale observar que apesar de o nome sugerir a morte por atropelamento de caminhão, o motivo de saída da equipe pouco importa, pois o fator relevante é que esses desenvolvedores levam consigo o conhecimento de regras de negócio e detalhes técnicos de implementação dos arquivos. Neste artigo será utilizada apenas a nomenclatura TF para abordar o tema.

Quanto mais o TF de um sistema tender ao valor 1, mais conhecimento estará concentrado na consciência de, aproximadamente, apenas um integrante da equipe e, portanto, o projeto enfrentará um grande risco de falha, caso este desenvolvedor abandone o projeto [5].

[4] Projetos de código-fonte aberto.

[5] Plataforma de hospedagem de código-fonte com controle de versão, disponível em `https://github.com`

3 Método de Pesquisa

No início desta pesquisa foi necessário realizar o levantamento dos trabalhos já existentes nas áreas relacionadas ao tema central deste estudo, para posteriormente, avaliar o que poderia ser feito de melhoria e contribuição, dado o que fosse encontrado de mais relevante na literatura. Foi escolhido o mapeamento sistemático como método de pesquisa para embasar este artigo.

3.1 Mapeamento Sistemático

No planejamento do mapeamento sistemático foi utilizada busca automatizada através dos buscadores das bibliotecas digitais *IEEE Xplore*[6], *ACM*[7], *Springer Link*[8], e *Science Direct*[9], e o conteúdo buscado foi o da string "*truck factor*". O critério de inclusão escolhido foi "Conter a expressão *truck factor* pelo menos na Seção de *keywords*". Os critérios de exclusão escolhidos foram "Não escrito em inglês" e "Não pertencente à Engenharia de *Software*".

Foram criadas quatro questões de pesquisa, a primeira questão intitulada de QP1 é "Quantos artigos relacionados ao assunto truck factor são publicados por ano?", a segunda questão intitulada de QP2 é "Quantos artigos correlacionam truck factor e projetos open source por ano?", a terceira questão intitulada de QP3 é "Quantos artigos correlacionam truck factor e práticas ágeis por ano?" e a quarta questão intitulada de QP4 é "Quantos artigos abordam a complexidade computacional e modelagens matemáticas relacionadas ao problema em si por ano?". Os dados coletados durante a resposta de cada questão de pesquisa geraram os respectivos grupos Q1, Q2, Q3 e Q4.

Foram considerados como estudos primários, 23 dos 25 artigos, encontrados inicialmente pela busca automatizada, pois foram encontrados um tutorial e um livro na busca da *Springer Link*. Através da técnica de *snowballing* foram filtrados 5 artigos pelas referências do artigo [1] e 2 artigos pelas referências do artigo [5]. Totalizando a referência de 30 artigos para este mapeamento sistemático, cuja a extração dos artigos está descrita na Tabela 1.

Table 1: Busca Automatizada.

Biblioteca Digital	Busca Automatizada	Duplicatas Retiradas	Critérios de Exclusão	Estudos Primários	Snowballing	Total
IEEE Xplore	4	1	1	1	2	
ACM	5	5	5	5	3	
Springer Link	33	33	14	12	1	
Science Direct	44	44	5	5	1	
Total	86	83	25	23	7	30

[6] `https://ieeexplore.ieee.org`
[7] `https://dl.acm.org`
[8] `https://link.springer.com`
[9] `https://www.sciencedirect.com`

3.2 Revisão dos trabalhos realizados sobre *truck factor*

Estes estudos primários podem ser agrupados em 3 grupos para facilitar a compreensão do assunto de uma maneira mais ampla. Pode-se observar o primeiro grupo chamado de Q1 que representa o total de artigos considerados estudos primários de acordo com a Tabela 1.

Já o segundo grupo chamado de Q2 é o de artigos que abordam algoritmos de estimação do TF através de mineração de dados de plataformas como o *Github*.

O *Github* possui um sistema de controle de versionamento de *software* baseado em *git*[10], integrando diversos recursos para codificação social, como a possibilidade de bifurcação da branch atual em duas novas branchs, *master*, que seria a principal, e a secundária, conhecido como *fork*. Também é possível realizar melhorias na secundária e enviar *pull requests* para o detentor do repositório original, e caso haja aprovação do proprietário, combinar as alterações feitas na secundária com o código da *master*. Finalmente, existe o recurso de mostrar satisfação ou interesse no repositório através do botão *star*, assim como o recurso de *like* provido pelas redes sociais [2].

O terceiro grupo chamado de Q3 é o de artigos que abordam metodologias ágeis como a divisão de conhecimento em desenvolvimento de *software* para mitigar o TF.

De acordo com [5], altos valores de TF normalmente resultam de práticas de propriedade coletiva de código, normalmente pregadas por metodologias ágeis de desenvolvimento de *software*.

Por fim, o quarto grupo chamado de Q4 é o de artigos que abordam a complexidade computacional e modelagens matemáticas relacionadas ao problema em si e não às aplicações práticas.

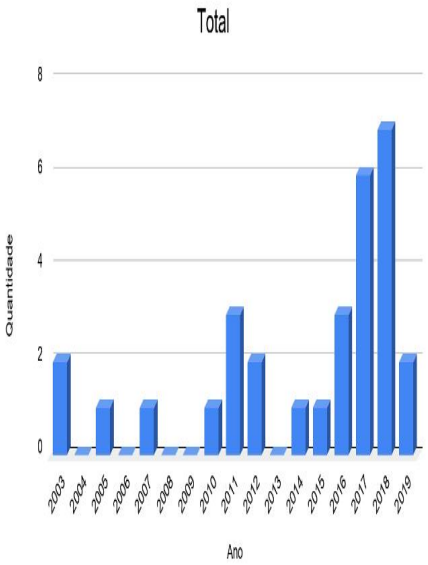
De acordo com [4], cada camada de um repositório, seja esta um arquivo, extensão de arquivo, diretório ou o próprio projeto, possui um valor respectivo de TF e, mesmo que haja uma camada com valor 1, ou seja, apenas o desenvolvedor mais experiente nesta camada está mantendo-a, mesmo que de uma forma não mais viável, todavia, o projeto não necessariamente terá o mesmo TF, e portanto, não necessariamente deixará de ser mantido e atualizado pela falta de manutenção apenas desta camada.

Para demonstrar a variação quantitativa dos artigos ao longo do tempo nos grupos Total, Open Source, Metodologias Ágeis e Modelagem Matemática, foram geradas, respectivamente, as Figuras 1a, 1b, 1c e 1d. Estes gráficos representam a tendência na pesquisa do assunto TF no geral e em relação a outros que este possua conexão, como Open Source, Metodologias Ágeis e Modelagem Matemática.

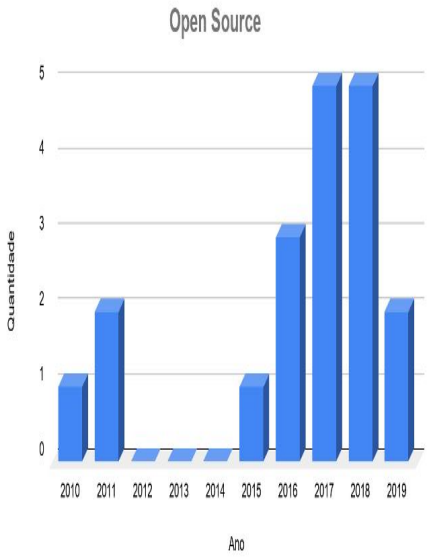
[10] Sistema de controle de versões distribuído, gratuito e de código aberto.

Fig. 1: Total, Open Source, Metodologias Ágeis e Modelagem Matemática.

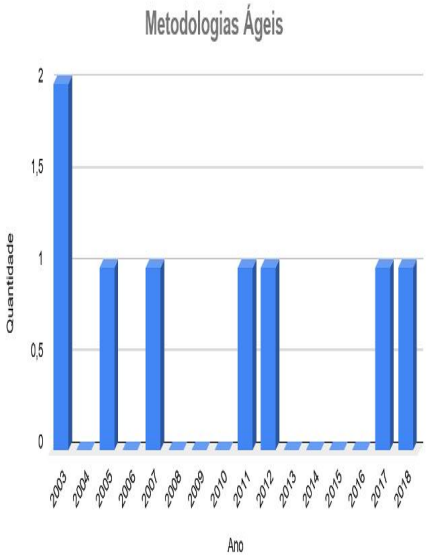
(a) Total.



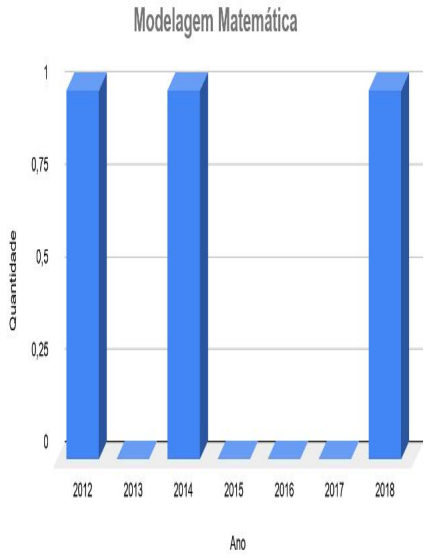
(b) Open Source.



(c) Metodologias Ágeis.



(d) Modelagem Matemática.



3.3 Análise crítica dos problemas em aberto

A questão de pesquisa Q1 mostra um gráfico de artigos por ano com alta e crescente densidade de publicações nos anos de 2016, 2017 e 2018, podendo-se fazer uma análise técnica do gráfico com uma tendência de alta para o mesmo período de 3 anos, isto é, de 2019 até 2021. Esta tendência sugere uma continuidade na pesquisa deste tema nos próximos anos.

O grupo Q2 é aquele que responde à questão de pesquisa Q2, que agrupa os artigos pelo tema "*open source*", e possui mais da metade do total dos artigos, 19 dos 30 referenciados. O gráfico possui alta e crescente densidade de publicações nos anos de 2016, 2017 e 2018, podendo-se fazer uma análise técnica do gráfico com uma tendência de alta para o mesmo período de 3 anos, isto é, de 2019 até 2021. Esta tendência sugere uma continuidade na pesquisa deste grupo nos próximos anos também.

Open Source Software (OSS) é um projeto de código aberto que permite que colaboradores espalhados em diferentes locais no globo atualizem ou melhorem o código sem a necessidade da contratação de uma organização específica, precisando apenas do acesso à Internet. A popularidade do OSS deve-se à colaboração da comunidade de desenvolvedores de *software* como um todo. Os benefícios da inovação, otimização do desempenho e suporte da comunidade são incomensuráveis [8].

O grupo Q3 é aquele que responde à questão de pesquisa Q3, que agrupa os artigos pelo tema "metodologias ágeis". O gráfico possui consistência durante o tempo, porém com uma quantidade bem menor que a de Q2, portanto não se pode fazer uma análise técnica do gráfico. Esta tendência sugere incerteza na pesquisa deste grupo nos próximos anos.

O TF mensura o compartilhamento de experiência entre os participantes do projeto, isto é, projetos com maior TF são menos sensíveis a saídas de desenvolvedores. O abandono do projeto pelo desenvolvedor resulta na redução da mão de obra, e indiretamente na perda de experiência, pois os desenvolvedores que permanecem mantendo as entregas do projeto podem não ter a experiência suficiente em determinados módulos do sistema, isto reduz consideravelmente o domínio sobre o código. O conceito de propriedade de código compartilhado, derivado do XP[11], aponta a necessidade de redução de perda de experiência do projeto, isto é, manter os desenvolvedores e disseminar o conhecimento entre eles, logo, este conceito pode ajudar no aumento do TF [6].

O grupo Q4 é aquele que responde à questão de pesquisa Q4, que agrupa os artigos pelo tema "modelagem matemática". O gráfico possui apenas 3 artigos e um grande espaço de anos entre as publicações, podendo-se fazer uma análise técnica do gráfico com uma tendência de baixa. Esta tendência sugere uma descontinuidade na pesquisa deste grupo nos próximos anos.

Segundo [7], o *turnover* de profissionais de tecnologia é a intensidade de integrantes sendo substituídos por outros indivíduos sem nenhum conhecimento

[11] Metodologia ágil com foco em *feedback* constante, abordagem incremental e comunicação.

sobre sistemas legados, os quais precisam de manutenção com certa frequência. O tempo entre novas atribuições é inversamente proporcional ao tempo do *turnover*.

3.4 Ferramenta

A apuração dos valores vinculados às métricas demonstradas na Seção 4 foi realizada a partir de uma ferramenta, a qual se conecta com o *Github* através da *API*[12] fornecida pelo próprio site e retorna tais informações, que posteriormente foram organizadas e transformadas em gráficos.

4 Resultados

Os resultados obtidos dizem respeito às quatro métricas vinculadas aos 175 projetos *open source* observados, isto é, linguagem de programação utilizada, quantidade de estrelas atribuídas ao projeto, quantidade de colaboradores e duração em anos.

Os dados indicam que projetos que utilizam linguagens de programação como *HTML*[13] e *Typescript*[14] obtiveram TF maior que a maioria, em torno de 3, enquanto a mediana foi de 1.8, logo há menor risco de descontinuação em projetos que usem estas linguagens, como é visto na Figura 2a. Após a análise da quantidade de estrelas dos projetos, observou-se que, na maioria dos casos, projetos com mais estrelas possuem maior TF médio, um valor de 3.8314, enquanto a média foi de 2.44, logo há menor risco de descontinuação em projetos com mais estrelas, como é visto na Figura 2b. Após a análise da quantidade de colaboradores dos projetos, observou-se que, na maioria dos casos, projetos com mais colaboradores possuem maior TF médio, um valor de 4.8771, enquanto a média foi de 1.7263, logo há menor risco de descontinuação em projetos com mais colaboradores, como é visto na Figura 2c. Após a análise da duração em anos dos projetos, observou-se que, na maioria dos casos, projetos com mais anos possuem maior TF médio, um valor de 4.3253, enquanto a média foi de 1.7263, logo há menor risco de descontinuação em projetos com maior duração, como é visto na Figura 2d.

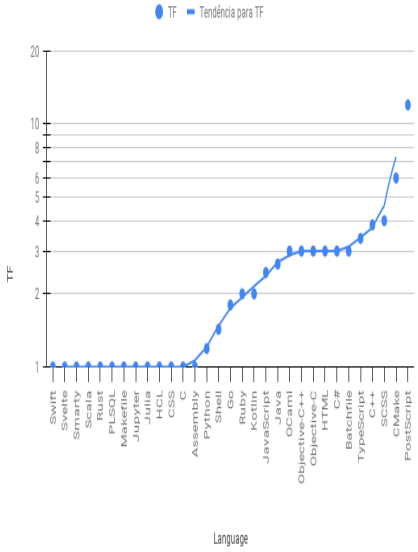
O *overfitting* é a divisão de trabalho de forma extremamente específica das equipes de tecnologia. A intenção é evitar sombreamento de funções ou tarefas durante o cotidiano profissional. Gerentes, designers, desenvolvedores e comunicadores tornam-se papéis separados, com atribuições e responsabilidades bem definidas. O problema que esta abordagem traz é a individualização dos sistemas, majorando a dependência para os projetos, e consequentemente, reduzindo o TF, isto é, aumentando o risco de descontinuação [10].

A relevância das metodologias ágeis é visível em diversas pesquisas. O aumento no TF elevados pode ser consequência de práticas coletivas de propriedade de código. O compartilhamento de informações pode reduzir as dependências.

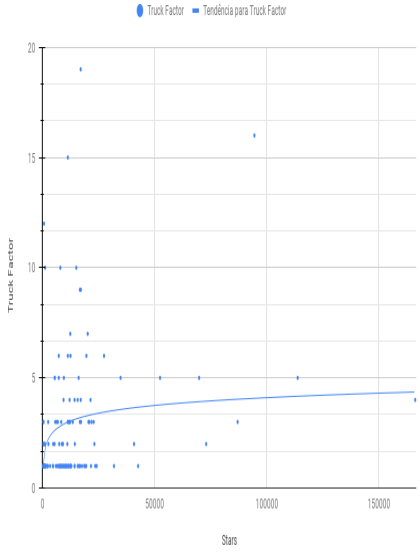
[12] Application Programming Interface.
[13] HyperText Markup Language.
[14] Superconjunto de JavaScript desenvolvido pela Microsoft.

Fig. 2: Gráficos de linguagens, estrelas, quantidade de colaboradores de cada projeto e de duração em anos de cada projeto em comparativo com seus respectivos valores de TF dos 175 projetos pesquisados neste artigo.

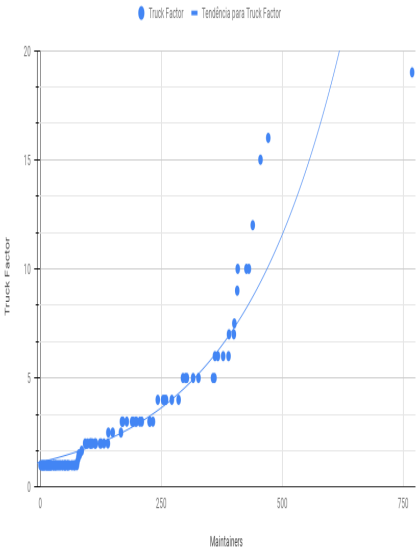
(a) Linguagens.



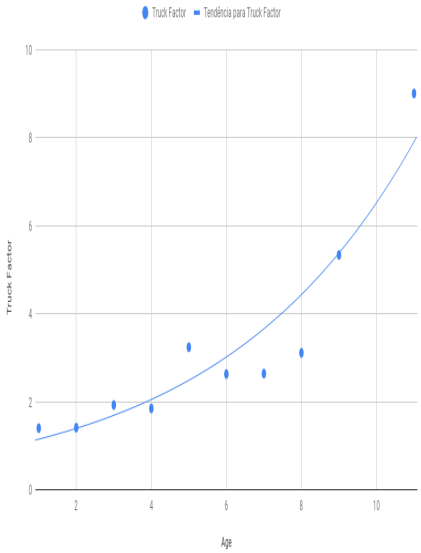
(b) Estrelas.



(c) Colaboradores.



(d) Duração.



5 Discussão

Os dados das Figuras 3a, 3b, 3c e 3d dizem respeito ao artigo [5]. Estes dados apontam para um aumento do TF em relação ao aumento das variáveis colaboradores e duração, assim como foi neste artigo anteriormente na Seção 4, mas há uma divergência na métrica de estrelas, pois enquanto o TF sobe junto com as estrelas na Seção 4, no artigo citado a quantidade de estrelas aumenta e o TF cai. Já em relação às linguagens, na Seção 4 projetos que utilizam linguagens de programação como *HTML* e *Typescript* obtiveram TF maior que a maioria, e no caso deste artigo citado, foi a vez de linguagens como *PHP*[15] e *Python*[16].

Pelo fato do artigo ter abordado 35 projetos, e os resultados obtidos na Seção 4 serem baseados na análise de 175 projetos, pode haver esse descolameto, principalmente no tocante à quantidade de estrelas. Deve-se levar em consideração também a diferença de tempo entre os estudos, pois o artigo citado data de 2017, já os resultados aqui abordados estão atualizados em 2021, um período relevante para a mudança no contexto de projetos *open source*. O fator *open source* também é fonte relevante no momento da comparação entre os conjuntos de dados, pois trata-se de um contexto volátil e simultaneamente extremamente mutável.

O conceito de "propriedade coletiva de código", cuja origem é na metodologia ágil *Extreme Programming* (XP), significa uma alta distribuição de conhecimento técnico sobre o código entre integrantes da equipe de desenvolimento, isto reflete num aumento direto no TF, pois um TF baixo significa que apenas poucos desenvolvedores detém grande parte do conhecimento do projeto, o que aumenta a dependência do projeto em relação a estes mantenedores, e consequentemente o risco de término. Estes contribuidores são denominados *Heroes*[17], pelo fato de gerenciarem e conhecerem grande parte do projeto, inclusive setores chaves do sistema. O valor de TF dá uma ideia do custo de troca dos *Heroes* [9].

A necessidade desta distribuição de conhecimento entre as partes do projeto tem extrema correlação com metodologias ágeis, pois o time precisa ter estar ciente do projeto como um todo para avançar de forma homogênea em direção aos entregáveis. O papel de tal prática é abordado e reforçado justamente por poder ser um catalisador no processo de queda de risco de descontinuação do projeto.

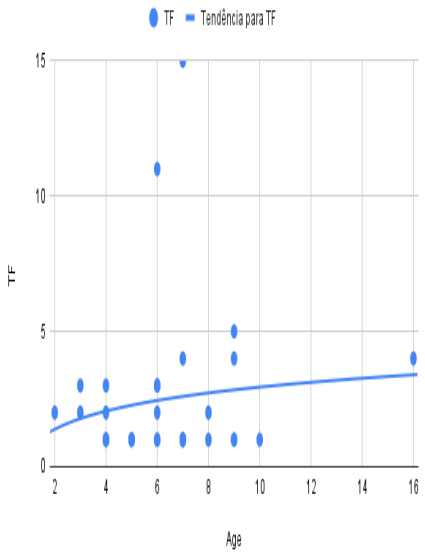
Projetos *open source* estão em transição devido à criação de uma plataforma moderna de desenvolvimento e manutenção de código-fonte aberto. O site *Github* possui cerca de 19 milhões de usuários e 52 milhões de repositórios, incluindo *forks*. A taxa de falha desses projetos vem aumentando cada vez mais. Há muito pouca pesquisa sobre as desvantagens dos projetos de código aberto [3].

A relevância do *Github* é inegável para esta pesquisa, pois todos os repositórios buscados foram obtidos a partir desta plataforma. Diversas informações sobre cada projeto podem ser buscadas, além das que são apresentadas neste artigo.

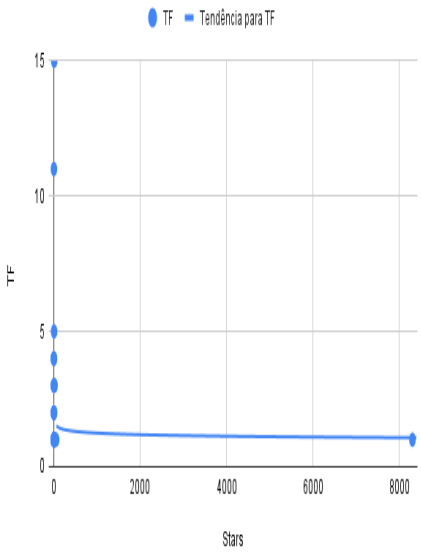
[15] Hypertext Preprocessor.
[16] Linguagem de programação de alto nível, interpretada de script, imperativa, orientada a objetos, funcional, de tipagem dinâmica e forte.
[17] Desenvolvedores chave que detém mais conhecimento do projeto.

Fig. 3: Gráficos de linguagens, estrelas, quantidade de colaboradores de cada projeto e de duração em anos de cada projeto em comparativo com seus respectivos valores de TF dos 35 projetos abordados no artigo [5].

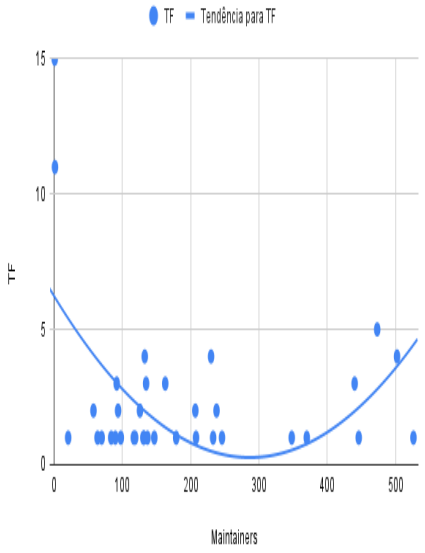
(a) Linguagens.



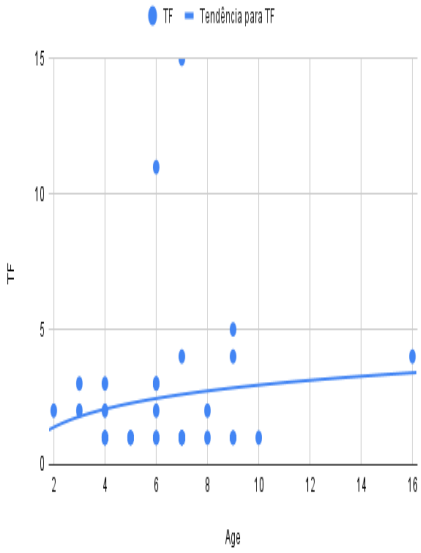
(b) Estrelas.



(c) Colaboradores.



(d) Duração.



6 Conclusão

Os resultados apontam uma correlação entre o aumento de fatores como duração, quantidade de colaboradores e de estrelas, além do uso de linguagens mais populares como boas práticas no combate à falha de projeto. É factível abordar futuramente o tema através de uma survey para angariar ainda mais dados que fortifiquem tais entendimentos sobre o assunto.

Na literatura foi amplamente analisado o viés de implementação de práticas ágeis na redução de risco de descontinuação de projetos, tal como na melhoria da qualidade das entregas destes. Espera-se que através das informações trazidas e dos resultados apresentados, possa-se interpretar melhor o tema TF e aplicar estes conhecimentos de forma prática no âmbito de metodologias ágeis.

É inteligível que existam limitações nesta pesquisa, visto que foram observados apenas 175 repositórios. Pela acessibilidade dos projetos de código aberto foram coletados dados apenas de projetos deste tipo. É possível que aumentando a base de pesquisa e agregando projetos proprietários na pesquisa possa-se conhecer mais ainda sobre o assunto, a partir das informações sobre novos conjuntos de dados mais diversificados.

References

1. AVELINO, G.: Measuring and analyzing code authorship in 1 + 118 open source projects (2019), https://www.sciencedirect.com/science/article/pii/S0167642318300388
2. BORGES, H.: What's in a github star? understanding repository starring practices in a social coding platform (2018), https://www.sciencedirect.com/science/article/pii/S0164121218301961
3. COELHO, J.: Why modern open source projects fail (2017), https://dl.acm.org/citation.cfm?doid=3106237.3106246
4. COSENTINO, V.: Assessing the bus factor of git repositories (2015), https://ieeexplore.ieee.org/document/7081864
5. FERREIRA, M.: A comparison of three algorithms for computing truck factors (2017), https://dl.acm.org/citation.cfm?id=3101439
6. HANNEBAUER, C.: Algorithmic complexity of the truck factor calculation (2014), https://link.springer.com/chapter/10.1007/978-3-319-13835-0_9
7. HOST, M.: Modeling the effects of project management strategies on long-term product knowledge (2012), https://link.springer.com/chapter/10.1007/978-3-642-31063-8_9
8. MAQSOOD, J.: Success or failure identification for github's open source (2017), https://dl.acm.org/citation.cfm?id=3034957
9. RICCA, F.: Are heroes common in floss projects? (2010), https://dl.acm.org/citation.cfm?id=1852856
10. STEGHOFER, J.P.: No silver brick: Opportunities and limitations of teaching scrum with lego workshops (2017), https://www.sciencedirect.com/science/article/pii/S0164121217301206
11. TORCHIANO, M.: Is my project's truck factor low?: theoretical and empirical considerations about the truck factor threshold (2011), https://dl.acm.org/citation.cfm?id=1985379