Caracterização do Papel Desempenhado por Desenvolvedores Responsáveis pelo Truck Factor de Projetos de Software

Thaís Mombach[1], **Mívian Ferreira**[1], **Marco Tulio Valente**[1], **Kecia Ferreira**[2]

[1]Departamento de Ciência da Computação – UFMG
Belo Horizonte – MG – Brasil

[2]Departamento de Computação – CEFET-MG
Belo Horizonte – MG – Brasil

{thaismombach,mivian.ferreira,mtov}@dcc.ufmg.br, kecia@decom.cefetmg.br

Abstract. *Truck Factor is a metric that calculates the concentration of knowledge on software project teams. Currently there are some algorithms proposed to calculate Truck Factor. These algorithms are based on commits history. In this work, we analyzed the results reported by such algorithms. We show that commit data are not sufficient to define whether a developer is part of the Truck Factor set. Moreover, we identify other factors that can turn a developer part of the Truck Factor.*

Resumo. *Truck Factor é uma métrica que calcula concentração de conhecimento em times de desenvolvimento de software. Atualmente, existem alguns algoritmos propostos para calcular Truck Factor. Esses algoritmos são baseados em histórico de commits realizados no projeto. No presente trabalho, foi realizada uma análise dos resultados reportados pelos algoritmos propostos na literatura. Os resultados indicam que as informações dos commits não são suficientes para determinar se um desenvolvedor faz parte do conjunto Truck Factor do software. Além disso, este trabalho identifica outros fatores que podem tornar desenvolvedores parte do Truck Factor.*

1. Introdução

Evolução e manutenção são atividades vitais para a continuidade de um projeto de desenvolvimento de software. A saída de desenvolvedores chave, *i.e.*, desenvolvedores que possuem grandes níveis de conhecimento sobre o código-fonte de um sistema, configura um risco para projetos de software, uma vez que pode fazer com que a realização dessas atividades seja dificultada ou até mesmo impossibilitada [Avelino et al. 2016]. Sendo assim, é necessário que os gerentes de projetos sejam capazes de identificar os desenvolvedores chave de um projeto para mitigar os riscos relacionados à saída dos mesmos.

Truck Factor (a.k.a *Bus Factor*) é uma métrica que permite identificar a concentração de conhecimento em projetos de desenvolvimento de software. Proposta pela comunidade *Agile*, *Truck Factor* é definida como o número mínimo de desenvolvedores de um projeto que devem ser atropelados por um caminhão (*i.e.*, abandonar a equipe de desenvolvimento) para que um projeto enfrente sérios problemas [Williams and Kessler 2003].

Dada a relevância da métrica, foram propostos na literatura alguns algoritmos para a identificação dos desenvolvedores chave de projetos de desenvolvimento de software: Zazworka et al. [2010], Cosentino et al. [2015], Avelino et al. [2016] e Rigby et al.[2016]. Esses algoritmos são baseados na premissa de que dados sobre a distribuição de conhecimento sobre código-fonte podem ser obtidos através da análise de informações de *commits*.

Em um estudo recente, Ferreira et al. [2017] conduziram uma avaliação comparativa entre três algoritmos para o cálculo de *Truck Factor*: Cosentino et al. [2015], Avelino et al. [2016] e Rigby et al. [2016]. Além do valor da métrica de *Truck Factor*, os algoritmos reportam o conjunto de desenvolvedores que fazem parte do *Truck Factor*. Os autores avaliaram a acurácia dos algoritmos em relação a um oráculo de *Truck Factors* de 35 sistemas de código aberto hospedados no GitHub. Os resultados indicam que o algoritmo proposto por Avelino et al. [2016] possui melhor acurácia dentre os algoritmos avaliados. Embora esse algoritmo apresente bons resultados, ele não foi capaz de identificar corretamente o conjunto *Truck Factor* de 25% dos sistemas avaliados. Isso sugere que podem existir outros aspectos que devem ser considerados na identificação de desenvolvedores chave de um projeto, além da contribuição deles no código-fonte, por meio de *commits*.

Neste trabalho pretende-se: (i) investigar a precisão da estimativa de *Truck Factor* de um sistema considerando apenas informações contidas em seus *commits*; e (ii) identificar quais outras informações, não refletidas nos *commits*, são importantes para o cálculo do *Truck Factor*. As análises realizadas neste trabalho foram baseadas nos dados dos 35 sistemas presentes no oráculo construído por Ferreira et al. [2017].

Este artigo está organizado da seguinte forma. A Seção 2 descreve o oráculo utilizado neste trabalho. A Seção 3 apresentada a metodologia utilizada neste estudo. A Seção 4 apresenta uma análise do conjunto *Truck Factor* utilizando *ranking* de *commits* dos sistemas. A Seção 5 discute outras razões para contribuidores fazerem parte do *Truck Factor*, que não são refletidas no número total de *commits*. A Seção 6 trata das ameaças à validade do estudo e a Seção 7 apresenta os principais trabalhos relacionados. Por último, a Seção 8 apresenta as conclusões deste trabalho.

2. Oráculo de *Truck Factor*

Neste trabalho, foi utilizado o oráculo de *Truck Factor* criado por Ferreira et al. [2017]. Ele é composto por 35 sistemas de código aberto hospedados no GitHub, dos quais 27 são, por sua vez, reusados do trabalho de Avelino et al. [2016] e 8 foram obtidos em um *survey* conduzido por Ferreira et al. [2017]. Os sistemas presentes no oráculo figuram dentre os mais bem avaliados (de acordo com o número de estrelas) das seis linguagens mais populares do GitHub: JavaScript, Ruby, Python, PHP, Java e C/C++. Eles possuem entre 2.863 e 59.184 estrelas, 21 e 7.265 contribuidores e 2 e 16 anos de existência.

O oráculo foi construído de acordo com a percepção dos desenvolvedores dos projetos avaliados. Ferreira et al. [2017] analisaram as respostas de desenvolvedores no *survey* conduzido inicialmente por Avelino et al. [2016] com 27 sistemas. Nesse *survey*, foi relatado aos desenvolvedores o valor do *Truck Factor* gerado pelo algoritmo e questionado acerca da concordância com o valor reportado. Na análise, foram consideradas as respostas que indicavam concordância com o *Truck Factor* apontado, bem como aquelas que indicavam discordância, mas que forneceram o valor e o conjunto *Truck Factor*

considerado correto pelo desenvolvedor. Para os demais sistemas do oráculo de Ferreira et al. [2017], os autores criaram *issues* no GitHub perguntando qual era o *Truck Factor* do sistema, na opinião dos desenvolvedores que contribuíam com aquele projeto. Foram obtidas respostas de 8 projetos, que foram incorporados ao oráculo.

Nas análises realizadas no presente trabalho, foram considerados 33 dos 35 sistemas presentes no oráculo. Os dois sistemas desconsiderados na análise foram: deis/deis e celluloid/celluloid. Para o sistema deis/deis, o oráculo não contém a lista de desenvolvedores que fazem parte do conjunto *Truck Factor* deste sistema. No sistema celluloid/celluloid, os dados do desenvolvedor indicado como *Truck Factor* não foram encontrados no repositório para análise.

3. Metodologia

Com o intuito de investigar se os componentes do conjunto *Truck Factor* são caracterizados pelo número de *commits*, foi criado um *ranking* dos contribuidores para cada um dos sistemas avaliados. A fim de evitar inconsistências na análise conduzida, foram utilizados os mesmos dados de *commits* dos sistemas utilizados para a construção do oráculo de *Truck Factors*. No *ranking* construído, os contribuidores foram classificados em ordem decrescente de acordo com o total de *commits* que realizaram no sistema. O total de *commits* de cada desenvolvedor foi obtido através da utilização de uma variação do comando `git shortlog`, que retorna uma lista ordenada de contribuidores do sistema de acordo com o total de *commits*.

A existência de *alias* pode mascarar os resultados obtidos [Wiese et al. 2016], sendo assim, foi desenvolvida uma abordagem para identificação de *aliases*. A identificação foi realizada através da busca de desenvolvedores com o mesmo nome ou com o mesmo *e-mail*; nessas situações, os desenvolvedores foram considerados como sendo um mesmo contribuidor e seus números de *commits* foram somados. Após isso, os contribuidores foram novamente ordenados de forma decrescente em relação ao total de *commits*, para se obter o *ranking* final.

Com o *ranking* para cada um dos 33 sistemas, foi realizada uma análise dos conjuntos *Truck Factor* de cada sistema para verificar se a presença de cada um desses desenvolvedores era explicada pela posição ocupada por ele no *ranking* de *commits*, o que está detalhado na Seção 4. A fim de investigar outras razões para que um contribuidor seja incluído no conjunto *Truck Factor*, foi conduzido um *survey*, sobre os papéis que desempenham no sistema, mas não são refletidos no número de *commits*, o que está detalhado na Seção 5.

4. Análise Crítica de Estimativas de *Truck Factor* via *Commits*

A fim de verificar se informações de *commits* do sistema são suficientes para identificar o conjunto *Truck Factor*, foram analisados os *rankings* de cada um dos sistemas utilizando a seguinte abordagem:

1. Para cada um dos 33 sistemas, foi verificado se os contribuidores listados no conjunto *Truck Factor* estavam entre os TF primeiros contribuidores no *ranking* de *commits*, onde TF é o valor resultante do algoritmo de *Truck Factor*.
2. Para os sistemas que atenderam essa característica, foi considerado que o conjunto *Truck Factor* pode ser explicado com base nos dados de *commits*.

3. Caso um ou mais desenvolvedores não estejam entre os TF primeiros contribuidores do *ranking*, o conjunto *Truck Factor* foi classificado como não explicado apenas pelo número de *commits*.

Com resultado dessa primeira análise, 23 (70%) dos sistemas analisados tiveram o *Truck Factor* explicado por meio da análise de *commits*. Para os 10 (30%) sistemas restantes, os dados de *commits* não se mostraram suficientes para explicar o conjunto *Truck Factor*, uma vez que todos os desenvolvedores do conjunto não estavam entre os TF primeiros do *ranking*.

Na Tabela 1, é apresentada a relação da quantidade de sistemas que tiveram ou não seu conjunto *Truck Factor* justificado pelo número de *commits*, de acordo com o tamanho do conjunto TF. É possível verificar que os 23 conjuntos *Truck Factor* que foram justificados por *commits* possuem $1 \leq TF \leq 3$. A porcentagem de projetos cujos conjuntos *Truck Factor* são explicados por *commits* é de 89%, 80% e 67%, para os conjuntos de tamanho 1, 2 e 3, respectivamente. Esse resultado sugere que à medida que o valor TF aumenta, a porcentagem de conjuntos *Truck Factor* justificados apenas pelo número de *commits* diminui.

Tabela 1. Quantidade de sistemas justificados e não justificados pelo número de *commits* de acordo com o tamanho do conjunto *Truck Factor*.

Tamanho do conjunto TF	Sistemas justificados	Sistemas não justificados
1	17	2
2	4	1
3	2	1
4	0	3
5	0	1
11	0	1
15	0	1

Observa-se também que os sistemas cujos conjuntos *Truck Factor* não são justificados apenas por *commits* possuem TF variados. Em relação a esses conjuntos TF, foi calculada a porcentagem de contribuidores do conjunto *Truck Factor* que puderam ser explicados pelo número de *commits*, ou seja, a porcentagem de desenvolvedores que fazem parte do *Truck Factor* e que estão entre os TF primeiros desenvolvedores do *ranking*. Os resultados da Tabela 2 representam a média das porcentagens calculadas nesses sistemas de acordo com tamanho do conjunto.

A porcentagem de contribuidores do conjunto explicados pelo número de *commits* é zero para sistemas com $TF = 1$ e $TF = 2$, e para sistemas com $TF \geq 3$, a taxa de acerto é de pelo menos 50%, como visto na Tabela 2. Portanto, analisando os resultados para os sistemas do oráculo, é possível concluir que apenas os dados de *commits* não permitem o cálculo correto do *Truck Factor* de todos os sistemas. Isso é ainda mais evidente nos casos em que os sistemas possuem TF maiores. Na Seção 5, serão investigados então outros aspectos que também impactam no conjunto *Truck Factor*.

Tabela 2. Porcentagem média de desenvolvedores justificados pelo número de *commits* de acordo com TF.

Valor do TF	% Média de desenvolvedores justificados
1	0%
2	0%
3	67%
4	50%
5	80%
11	73%
15	67%

Resumo: É possível concluir que apenas os dados de *commits* não permitem o cálculo correto do *Truck Factor* de todos os sistemas.

5. Outros Fatores Críticos para Estimativa de *Truck Factor*

Visando investigar a justificativa dos conjuntos *Truck Factor* dos 10 sistemas que não puderam ser explicados por dados de *commits* na análise da Seção 4, foram levantados outros aspectos da construção de sistemas que podem ter impacto no *Truck Factor*. Para isso, foi perguntado aos contribuidores indicados no *Truck Factor* dos sistemas no oráculo e que não estavam nas TF primeiras posições do *ranking* as funções desempenhadas por eles e que eles consideram que não são refletidas no número de *commits*.

O contato com os desenvolvedores foi realizado via *e-mail*, disponibilizados no GitHub. Um *e-mail* personalizado foi enviado para cada um deles, contendo as seguintes informações: nome do sistema, o valor do TF, a posição do desenvolvedor no *ranking* de *commits*, o mês e o ano nos quais o *Truck Factor* do sistema foi calculado. Uma vez que o cenário do cálculo do *Truck Factor* foi explicado, foi perguntado ao desenvolvedor quais papéis ele desempenhava no projeto que justificasse sua presença no conjunto *Truck Factor* e que não se refletem no número de *commits* do sistema.

Foram enviados *e-mails* para 20 contribuidores, de 10 sistemas diferentes, entre os dias 23 e 25 de abril de 2017. Dos 20 *e-mails* enviados, três deles não foram entregues pois o endereço de *e-mail* informado não existe. Para os 17 *e-mails* que restaram, foram obtidas sete respostas de desenvolvedores de quatro sistemas diferentes. Com isso, a taxa de resposta obtida foi de $41{,}12\%$.

As respostas recebidas foram analisadas a fim de se identificar outros fatores críticos para estimativa de *Truck Factor*. A análise dessas respostas foi realizada separadamente por dois dos autores deste trabalho, que classificaram cada uma das respostas. Posteriormente as classificações foram discutidas e sumarizadas, a fim de se obter uma única classificação dos fatores que impactam em *Truck Factor* apontados pelos desenvolvedores. Ao final, foram obtidas as seguintes classificações: interação social, *pull request*, documentação, garantia de qualidade e ferramentas de suporte ao projeto.

Interação social está relacionada a contribuidores que disseram ser ativos na comunidade do projeto, através de respostas de *e-mails*, *issues* no GitHub, questões no Stac-

kOverflow, entre outros meios, atualização de blogs ou *websites* do projeto, e também na divulgação do projeto em conferências e outros encontros. *Pull request* diz respeito a contribuidores que afirmaram que o fato de analisarem aceitações de *pull requests* ou fato de que tiveram muitos *pull requests* aceitos foi relevante para serem considerados parte do *Truck Factor* do sistema. A categoria Documentação engloba respostas nas quais o contribuidor afirmou ser um dos principais contribuidores da documentação do sistema ou que participa participa da construção dessa documentação. A categoria Garantia de Qualidade refere-se a desenvolvedores responsáveis pelos testes, identificação de falhas no projeto ou revisão de código. E por último, na classificação Ferramentas de Suporte, o contribuidor informa ser responsável por ferramentas como *plugins*, instalador e versão demo do projeto.

Uma vez que foram desconsideradas respostas baseadas apenas em aspectos relacionados a *commits*, restaram cinco respostas válidas que foram classificadas de acordo com as categorias apresentadas. Destas cinco respostas, quatro (80%) delas destacaram a interação social dos contribuidores do projeto como um fator relevantes para o cálculo do *Truck Factor*. Contribuição para documentação do projeto, *pull Request* e garantia de qualidade ocupam o segundo lugar na análise de fatores mais citados, presente em 40% das respostas. A categoria ferramentas de suporte ao projeto foi apontada em apenas uma das cinco respostas, o que corresponde a 20% dos casos.[1]

Resumo: Não só contribuição para o código do projeto deve ser considerada para se estimar *Truck Factor*, mas também aspectos como interações sociais.

6. Ameaças à Validade

6.1. Ameaças Internas

A fim de se identificar outros fatores importantes para o cálculo do *Truck Factor*, foi perguntado aos desenvolvedores quais outros papéis, que não estavam refletidos no número de *commits*, eles desempenhavam no sistema. As respostas obtidas foram manualmente categorizadas, estando assim sujeitas a interpretação pessoal do autor que as classificou. A fim de mitigar essa ameaça, a classificação das respostas foi realizada separadamente por dois dos autores deste artigo. Essas duas classificações foram comparadas para gerar a classificação final. Na presença de divergência entre as respostas durante a comparação, as mesmas foram novamente analisadas por ambos autores para obter-se um consenso.

6.2. Ameaças Externas

As análises e resultados apresentados foram obtidos através de estudos de sistemas de um oráculo de *Truck Factor*. Nesse oráculo, estão presentes sistemas: desenvolvidos nas seis linguagens mais populares no GitHub, de diferentes tamanhos, quantidades de contribuidores e tempos de existência. Embora seja um conjunto de sistemas diversificado e abrangente, não se pode garantir a generalidade dos resultados reportados neste artigo.

7. Trabalhos Relacionados

Embora a estimativa de *Truck Factor* seja importante na gestão de projetos de software, existem poucos trabalhos na literatura que buscam identificar maneiras de calcular essa

[1] Como algumas respostas mencionaram mais de um papel, a soma excede 100%.

métrica. A primeira abordagem proposta para o cálculo de *Truck Factor* foi apresentada por Zazworka et al. [2010]. Nesse trabalho, os autores assumem que todos os contribuidores que fizeram modificações em um arquivo têm conhecimento sobre ele. Ricca et al. [2011] propuseram uma ferramenta para o cálculo de *Truck Factor* baseada na definição de Zazworka et al. [2010]. A ferramenta foi avaliada em um *dataset* contendo 37 sistemas de código aberto, sendo 24 deles classificados como pequenos e 13 como grandes. Com os resultados obtidos, os autores identificaram um problema de escalabilidade para sistemas com mais de 30 contribuidores. Posteriormente, Hannebauer and Gruhn [2014] justificaram o problema de escalabilidade para abordagem baseada na ideia de Zazworka, provando que este problema é NP-Difícil.

Outras heurísticas foram propostas para calcular *Truck Factor* de sistemas de código aberto que utilizam histórico de versão. Rigby et al. [2016] propuseram um algoritmo não determinístico no qual a autoria dos arquivos é dada pelo *git-blame*. O algoritmo gera grupos aleatórios de desenvolvedores para encontrar o conjunto *Truck Factor*. Na abordagem de Cosentino et al. [2015], o *Truck Factor* é calculado por partes: primeiro, o *Truck Factor* é calculado para cada arquivo; em seguida, esses resultados são utilizados para calcular o *Truck Factor* dos módulos; por fim, os *Truck Factors* dos módulos são utilizados para se obter o valor do *Truck Factor* do sistema. O algoritmo proposto por Avelino et al. [2016] utiliza a métrica DOA (*Degree-of-Authorship*) [Fritz et al. 2014] para identificar os principais responsáveis por um arquivo e aplica uma heurística gulosa para calcular o *Truck Factor*. O trabalho de Ferreira et al. [2017] realizou uma comparação entre esses algoritmos, identificando que a proposta de Avelino et al. [2016] é mais precisa. Todavia, todos esses trabalhos são baseados na ideia da análise de *commits*. Porém, o presente trabalho revelou que as informações de *commits* não são suficientes para estimar corretamente o *Truck Factor* de todos os sistemas.

8. Conclusão

O objetivo deste estudo foi investigar se dados de *commits* são suficiente para a estimativa de *Truck Factor*. Para isso, foram utilizados 33 sistemas de um oráculo *Truck Factor* com 35 sistemas de código aberto. Os resultados obtidos indicam que 70% dos sistemas puderam ter o grupo de desenvolvedores do *Truck Factor* explicados utilizando informações de *commits*. Além disso, observou-se que o percentual de sistemas cujo conjunto *Truck Factor* pode ser explicado analisando apenas o número de *commits* diminui à medida que o TF aumenta. Os resultados obtidos neste estudo sugerem que, embora o cálculo de *Truck Factor* por meio da análise de informações relacionadas a *commits* seja eficiente em alguns casos, é necessário se atentar para o fato que o uso apenas dessa informação não é suficiente para obter um resultado correto em todos os casos.

Além disso realizou-se um *survey* com contribuidores cujo objetivo foi identificar outros fatores que podem levar um desenvolvedor a ser considerado como parte do *Truck Factor*. As respostas indicaram que os seguintes fatores também são importantes: interação social, *pull request*, documentação, garantia de qualidade e ferramentas de suporte ao projeto. A análise das respostas mostra que aspectos sociais do projeto podem ter grande impacto na definição do conjunto *Truck Factor*. Como trabalho futuro, sugere-se a investigação de algoritmos de estimativa de *Truck Factor* que considerem os fatores identificados no presente artigo.

9. Agradecimentos

Esta pesquisa é apoiada pela FAPEMIG, CAPES e CNPq.

Referências

Avelino, G., Passos, L., Hora, A., and Valente, M. T. (2016). A novel approach for estimating truck factors. In *24th International Conference on Program Comprehension (ICPC)*, pages 1–10.

Cosentino, V., Izquierdo, J. L. C., and Cabot, J. (2015). Assessing the bus factor of git repositories. In *22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, pages 499–503.

Ferreira, M., Valente, M. T., and Ferreira, K. (2017). A comparison of three algorithms for computing truck factors. In *25th International Conference on Program Comprehension (ICPC)*, pages 207–217.

Fritz, T., Murphy, G. C., Murphy-Hill, E., Ou, J., and Hill, E. (2014). Degree-of-knowledge: Modeling a developer's knowledge of code. *ACM Transactions on Software Engineering and Methodology*, 23(2):1–42.

Hannebauer, C. and Gruhn, V. (2014). Algorithmic complexity of the truck factor calculation. In *15th Product-Focused Software Process Improvement (PROFES)*, pages 119–133.

Ricca, F., Marchetto, A., and Torchiano, M. (2011). On the difficulty of computing the truck factor. In *12th Product-Focused Software Process Improvement (PROFES)*, pages 337–351.

Rigby, P. C., Zhu, Y. C., Donadelli, S. M., and Mockus, A. (2016). Quantifying and mitigating turnover-induced knowledge loss: Case studies of Chrome and a project at Avaya. In *38th International Conference on Software Engineering (ICSE)*, pages 1006–1016.

Wiese, I. S., da Silva, J. T., Steinmacher, I., Treude, C., and Gerosa, M. A. (2016). Who is who in the mailing list? comparing six disambiguation heuristics to identify multiple addresses of a participant. In *32nd Software Maintenance and Evolution (ICSME)*, pages 345–355.

Williams, L. and Kessler, R. (2003). *Pair Programming Illuminated*. Addison-Wesley.

Zazworka, N., Stapel, K., Knauss, E., Shull, F., Basili, V. R., and Schneider, K. (2010). Are developers complying with the process: an XP study. In *4th International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 1–10.