Measuring Software Resilience Using Socially Aware Truck Factor Estimation

Alexis Butler
Information Security Group
Royal Holloway University of London
London, United Kingdom
alexis.butler.2023@live.rhul.ac.uk

Dan O'Keeffe
Department of Computer Science
Royal Holloway University of London
London, United Kingdom
daniel.okeeffe@rhul.ac.uk

Santanu Kumar Dash
Department of Computer Science
University of Surrey
Guildford, United Kingdom
s.k.dash@surrey.ac.uk

***Abstract*—Continued timely maintenance is a key aspect of project security, but typically requires in-depth knowledge of a project's code base. Truck Factor is a metric that aims to represent how vulnerable a project is to losing this knowledge through the attrition of key contributors. However, the accuracy of existing Truck Factor estimators scales poorly with project size since they tend to ignore influential team members in managerial roles, which are more common in large projects.**

This work proposes SNet, a novel socially aware Truck Factor estimator based on social network analysis. SNet uses network centrality measures and social signals such as GitHub Issue interactions to estimate Truck Factor and identify Truck Factor contributors. We evaluate SNet against an existing ground truth comprised of twenty-six open source projects. Our social network analysis approach achieves superior contributor classification performance (Median F1 score = 0.8) while reducing computation time by over 2x compared to state-of-the-art estimators.

***Index Terms*—Software Engineering, Software Maintenance, Software Security, Truck Factor, Open Source Software**

I. Introduction

Truck Factor is a well-known metric that measures how vulnerable a project is to dropping below the knowledge threshold required for effective development and maintenance [1], [2]. As defined by Williams et al. [3], Truck Factor aims to understand *"How many or few people would have to be hit by a truck (or quit) before the project is incapacitated?"* In addition to project resilience, Truck Factor is a strong indicator of the security risk associated with a given project, since effective security maintenance also depends on the organization having sufficient knowledge of the project's internals [4].

The majority of state-of-the-art Truck Factor algorithms focus on attributing knowledge based on file change data contained within the version control system (VCS) of the target project [5], [6]. However, recent work on self organizing structures in open-source software (OSS) suggests that those with the most knowledge are not necessarily well represented in change log data [7]. One explanation for this is that long serving members of a project often move into more managerial roles, directing the contributions of others over contributing code directly themselves [8]. This suggests that social signals, such as GitHub interactions, may be stronger indicators of knowledge ownership compared to commit frequency alone, as experienced developers guide others through issues and architectural discussions.

The Truck Factor estimator of Avelino et al. [5] can be considered the state of the art for classical Truck Factor estimators, i.e., estimators that exclusively rely on source code change data. However, Ferreira et al. find that its accuracy falls off when applied to larger projects [9]. More recently, multiple efforts have been made to extend the classical approach. These extensions incorporate additional signals from project management tools such as pull requests (PRs), meeting attendance, and file importance. However, these efforts have been limited to industrial efforts on single language internal projects [10] [6] using non-standard project management tooling. As such, their results have limited generalizability. Both the classical estimators and later extensions suffer from significant runtime costs that arise from their dependence on grouping commit data by user and by file - this data transformation cost becomes significant when operating on long-lived projects.

This work proposes SNet, a novel *socially aware* Truck Factor estimator based on social network analysis. We refer to our proposed estimator as socially aware as it captures the important role of the social side of OSS development, as demonstrated in prior work on OSS project structures [11]. A key benefit of SNet is its ability to identify critical project members whose contributions are not evident from source-code change data. SNet also avoids the computationally expensive task of commit grouping required by current estimators. As a result, it lends itself to tasks such as assessing the resilience of large scale multi-project ecosystems. We benchmark SNet against existing state-of-the art estimators using previously collected ground truth Truck Factor scores [5], [9]. As a further point of comparison, we extend an existing file-based estimator with additional social signals such as GitHub PR and issue creation and interactions.

Our initial results demonstrate that a project's social network can be used for rapid Truck Factor estimation. Not only do these findings highlight the value of social factors in supply chain risk analysis, they also enable ecosystem scale Truck Factor surveys. To support our initial findings, we provide both the replication package[1] and dataset[2].

[1]https://github.com/Abutler101/socially-aware-truck-factor
[2]https://zenodo.org/records/15223467

2643-1572/25/$31.00 ©2025 IEEE
DOI 10.1109/ASE63991.2025.00318

II. Background and Related Work

Truck Factor estimation can be considered a two-step process. First a map is created to represent which developers have knowledge of each file. Then through one mechanism or another, authors are ablated and the knowledge coverage of the project re-computed. Two common approaches to knowledge attribution are Degree of Authorship (DoA) and Degree of Knowledge (DoK). DoA is a combination of three factors: first authorship, number of commits, and number of commits by others [12]. DoK extends DoA by introducing the idea of developer interest, captured via file access patterns [12].

The most common approach to Truck Factor estimation is to constrain the attribution of authorship/knowledge to only make use of file change data stored in the VCS. For example, Avelino et al. propose a degree of authorship (DoA) approach where file authors are identified by applying a threshold to a DoA score [5]. Avelino use previously published weights identified through empirical experiments [13] to produce a DoA score per-author-per-file.

With the advancement of software development environments, more data on how developers work has become available. Researchers have begun to explore including these additional data sources into Truck Factor estimators. The work of Jabrayilzade et al. [10] and Haratian [6] extends the work of Avelino by including metrics such as meeting attendance and file importance. In both cases, the authors observed improved estimator performance when tested on internal projects.

The only comparison of multiple Truck Factor estimators against a common dataset that we are aware of is the work of Ferreira et al. [9]. The authors extend the validation dataset created by Avelino et al. and then proceed to evaluate estimators by Cosentino [14], Avelino [5], and Rigby [15]. They find that Avelino performs the best, but with a notable decrease in performance as project size increases (as captured by the true Truck Factor).

Prior work has looked at the self organizing nature of OSS communities and movements of developers within these organizational structures [16]. Bird et al. used social networks constructed from mailing list data to demonstrate the presence of multiple subcommunities within a project that are bridged by a management class [17]. Joblin et al. extend this methodology to classify core and peripheral developers through the use of calculating degree centrality over the produced social network [7]. More recently, Joblin et al. used their classification of developers to study the evolution of OSS project structures [11]. Joblin et al. found that projects frequently consist of a hierarchical managerial team on top of a non-hierarchical group of irregular contributors.

III. Methodology

A. Social Network Driven Estimator (SNet)

To overcome the limitations of existing estimators, we design and implement SNet, a novel social network based Truck Factor estimator. This approach identifies Truck Factor contributors in terms of their global positioning within the target project's social structure.

Taking inspiration from Zanetti et al. [16], the core of this estimator is a social network of project contributors. Each node in the social network represents an individual, and each weighted directed edge captures cumulative social engagement with the destination contributor. We consider multiple different types of social interactions, as illustrated in Table I.

TABLE I: SNet's social network interaction edge directions

Interaction Type	Edge Definition
PR review request	Creator → Reviewer
PR review	Reviewer → Creator
PR / Issue assignment	Creator → Assignee
PR / Issue comment	Commenter → Creator

Based on the resulting social network we next identify key developers using network centrality measures, in line with recent work on the organizational structure of OSS projects [7], [11]. We evaluate several normalized centrality measures to capture contributor importance including degree and betweenness centrality [18], Page Rank [19], and node degree. Our initial results indicate degree centrality performs the best.

Once node importance is calculated, our estimator first applies an absolute threshold where all contributors with an importance score higher than the threshold are counted. An additional filter is then applied where only the top X% of contributors whose scores are above the threshold value. The additional filter attempts to account for smaller OSS projects where the hierarchy tends to be flatter if at all present. Initial optimization for classification accuracy yielded an importance threshold of 0.75 and a percentage filter of 55%.

We note that GitHub Issue and PR comments can present a distorted view of project interactions due to the presence of bots [20]. As such, if either party involved in an interaction is identified as a bot, we exclude the edge from the network. We also exclude self interactions and interactions involving deleted users. We currently weight each interaction type equally to avoid the introduction of bias towards a specific communication pattern. Only once an understanding of each interaction type's role in knowledge transfer is established can these weights be fine-tuned. We leave the exploration of individual weights for each of the social interactions, along with the age-based decay of these weights for future work.

B. Dataset Collection

In order to validate SNet, ground truth Truck Factor values alongside commit, Pull Request (PR), and Issue information for multiple projects was required. Previously, Avelino et al. used GitHub Issues to solicit ground truth Truck Factor scores for a number of GitHub projects [5]. This list of projects was subsequently expanded by Ferreira et al. [9]. However, both authors only published the Truck Factor values, without archiving a snapshot of the targeted repositories. Consequently, collection of the relevant estimator inputs for each project was required in order to make use of their ground truth values.

The issue threads used by Avelino et al. to solicit ground truth values are still present on each of the target repositories, providing a timestamp for when the ground truth values were

collected. However, Ferreira et al.'s extension of the Avelino list, did not leave any indicators as to when the ground truth was captured. As such, we assume these values to be collected four months prior to the publication date of Ferreira et al.'s work. This yields a cutoff date of January 22^{nd} 2017.

For each target project Commits, Issues and PRs were collected from repository creation to the identified ground truth timestamp. This data was gathered via the GitHub API and stored in JSON files, this data and the scripts to collect it are made available as part of this work's replication package. One project, sass/sass - a CSS extension, had to be removed from the dataset as the commit history seems to have been significantly pruned since Avelino et al.'s study. We suspect this because the earliest remaining commit is now December 12^{th} 2015 - several months post ground truth collection. By storing the GitHub data alongside the projects' Truck Factors, we prevent further dataset degradation going forward.

The final dataset consists of twenty-six projects from Avelino [5], along with an additional eight from Ferreira [9].

C. Extended Degree of Knowledge Estimator (EDoK)

To serve as a point of comparison between the exclusively social SNet estimator and existing state-of-the-art estimators, we propose an additional extended degree of knowledge (EDoK) estimator that adds social signals to the Avelino estimator. This extension is done by tracking GitHub Issue and PR engagement.

Our EDoK estimator defines Degree of Knowledge (DoK) as the sum of Degree of Authorship (DoA) and Degree of Interest (DoI), as in Fritz et al.'s work [13]. DoA retains the same definition and internal weights identified by Avelino. Specifically, the DoA of developer d on file f combines a first authorship check (returning 1 or 0), number of changes, and number of changes made by others. In contrast to Fritz, we define DoI as a weighted sum of four social signals: issue creation count, issue comment count, PR creation count, and PR comment count. We establish both the internal DoI weights and DoK weights for our EDoK estimator through empirical optimization, with classification accuracy as the target.

D. Estimator Evaluation

We evaluate SNet against the EDoK estimator along with two state-of-the-art baseline estimators: Avelino et al.'s estimator (AVE) with default configuration, and Haratian et al. using degree centrality (H-DC) as this was the top performing configuration. We use the project list of Ferreira [9] to evaluate both the estimators' performance and computational cost.

1) Classification of Truck Factor Contributors: Of the two functions of a Truck Factor estimator, identifying the Truck Factor contributors is the most valuable capability. This is because knowing the key knowledge holders in an organization can enable more effective knowledge sharing efforts within the project [21]. Following Ferreira et al.'s evaluation methodology, we calculate the precision, recall, and F1 scores for each estimator on each target project. We consider, a True Positive to be a correctly identified Truck Factor contributor and a False Negative to be a contributor incorrectly excluded from the contributors list.

An additional aspect of the applicability of Truck Factor estimators to projects is the associated compute cost. Execution time was one of the main criticisms raised when prior work evaluated the feasibility of automating Truck Factor estimation [22]. With this in mind, we capture the total execution time for the estimators. The recorded execution time for each estimator includes all transformations performed on the input VCS data.

2) Truck Factor Estimation: Alongside the classification functionality, estimators can also be seen as regressors where the result is the Truck Factor estimate. This purely numerical output is valuable for indicating the risk to a project's long term viability. However, evaluation of estimators based purely on the regression task can be misleading. For example, an estimator may correctly produce a Truck Factor of 1 while incorrectly identifying who that single contributor is.

Drawing from the methodology of Haratian et al. [6], Truck Factor score performance was evaluated with Normalized Mean Absolute Error (NMAE), where NMAE is the mean average of estimators' NAE (Equation 1) scores from each project.

$$\text{NAE} = \frac{|TF_{\text{ground}} - TF_{\text{estimate}}|}{TF_{\text{ground}}} \tag{1}$$

We choose NMAE as it captures the role of the ground truth score in the significance of an error. For example, an off by one error is insignificant with a ground truth of one hundred, but significant in the case of a ground truth of one. We replicate Haratian et al.'s evaluation methodology as it appears the most robust of the regression evaluations in the literature ([5], [9]).

IV. Results

A. Classification of Truck Factor Contributors

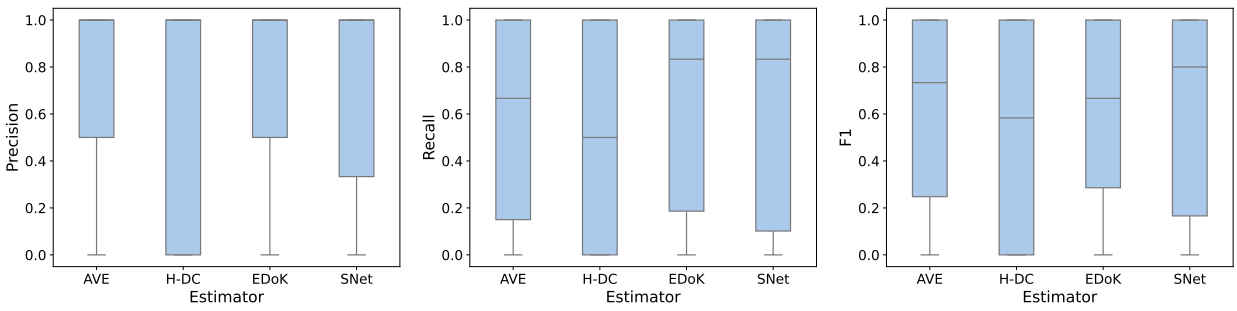
Figure 1 presents box plots for the precision, recall, and F1 scores of each of the four estimators, applied to the task of identifying the Truck Factor contributors. With median recalls of > 0.8, both the EDoK and SNet estimators outperform AVE and H-DC on the average project. However, SNet's Q1 Precision of < 0.4 indicates a stronger tendency towards false positives than the AVE or EDoK estimators.

Combining recall and precision scores into the F1 metric, we see that in the average case SNet is the overall top performer ($F1_{med} = 0.8$) when identifying Truck Factor contributors. This initial result demonstrates the viability of social network based estimation. However, SNet having the second-largest interquartile range indicates a need for further tuning.

TABLE II: Mean execution time in seconds by project size

Proj Size	AVE	SNet	H-DC	EDoK
TF=1	2.697	1.753	17.384	8.953
$2 \geq$ TF ≤ 5	8.794	3.992	233.153	47.383
TF ≥ 6	106.636	63.891	537.682	1142.756

Table II shows the average execution time for the estimators. We see that SNet has the lowest execution times across all



(a) Precision (b) Recall (c) F1

Fig. 1: Classification metrics for, EDoK, SNet, and baseline estimators, identifying the list of Truck Factor contributors for projects in the Ferreira dataset. Whiskers show $1.5 \times IQR$. Median shown when not coinciding with a quartile boundary.

project size groups. This is due to the estimator not needing to perform the costly grouping transformations required by the baseline and EDoK approaches. This dramatic reduction in computation cost represents a significant improvement for practical applications such as ecosystem-wide analysis or real time risk monitoring within an organization.

B. Truck Factor Estimation

The Normalized Mean Absolute Error scores for each of the estimators over the whole dataset (Table III), show SNet outperforming the Haratian estimator [6] but not Avelino et al.'s [5] approach (AVE). Interestingly, the EDoK estimator has the lowest performance on this metric. However, the scores across all four approaches are fairly close. Additional investigation indicated that the cause of SNet's lower NMAE performance was the difference in error profiles between AVE and SNet, with AVE having frequent off-by-one errors, and SNet making infrequent but larger errors where there was insufficient data for social network construction. Considering the SNet errors in the context of its superior classification performance (Median F1=0.8), SNet appears the stronger estimator for larger more complex projects. However, it is clear SNet needs to be refined when operating on smaller projects. A potential solution would be to fall back to a conventional estimation approach where the project's social network is below some threshold size.

TABLE III: Ascending estimator NMAE scores

Estimator	NMAE
AVE	0.3086
SNet	0.3389
H-DC	0.3511
EDoK	0.3701

TABLE IV: NMAE scores broken down by project size

Proj Size	AVE	SNet	H-DC	EDoK
TF=1	0.1579	0.2632	0.3158	0.2632
$2 \geq$ TF ≤ 5	0.4250	0.3458	0.3083	0.4250
TF ≥ 6	0.7973	0.7906	0.7455	0.8276

From Table IV, it can be seen that the SNet estimator performs worse on smaller projects than the AVE estimator. However, this is not the case for projects with more than one Truck Factor contributor. This breakdown of NMAE scores supports the idea that SNet's performance is tied to the size of the project, with a preference for larger projects. This bias towards larger projects is also seen in the scores for the Haratian estimator. An explanation for both cases is the requirement of sufficient data for graph construction.

V. Discussion & Future Work

Our novel SNet estimator has comparable or better performance than two state-of-the-art estimators when evaluated across multiple projects. This demonstrates social networks can be used to estimate Truck Factor and that this approach has distinct advantages over existing methods when applied to larger, more complex projects.

Moving forward, we plan to collect a larger, more diverse, Truck Factor dataset following the same collection methodology as Avelino et al. [5]. This expanded dataset will enable further tuning of SNet's weights and also provide a basis for further studies of the impact of social interactions on Truck Factor. Additionally, we aim to evaluate the role of social interactions in knowledge transfer. Specifically, we aim to assess the knowledge transfer capacity of each of the interactions captured by SNet and use these insights to inform non-uniform interaction weights.

Beyond improving SNet's performance, we plan to explore the role of Truck Factor in the modern software supply chain. This will involve collating recent case studies of Truck Factor incidents and identifying common indicative factors. Our overarching goal is to combine these factors along with the rate of change of Truck Factor to predict project abandonment.

VI. Acknowledgment

This work was funded in full by the UKRI CDT scheme, through the Centre for Doctoral Training in Cyber Security for the Everyday at Royal Holloway University of London.

References

[1] M. McLay, "If Guido was hit by a bus?" Mailing List, Jun. 1994. [Online]. Available: https://legacy.python.org/search/hypermail/python-1994q2/1040.html

[2] J. O. Coplien and D. C. Schmidt, Eds., *Pattern languages of program design*. USA: ACM Press/Addison-Wesley Publishing Co., 1995. [Online]. Available: https://archive.org/details/patternlanguages0000unse/mode/2up

[3] L. Williams and R. Kessler, *Pair Programming Illuminated*. USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

[4] C. Wang, Y. Li, L. Chen, W. Huang, Y. Zhou, and B. Xu, "Examining the effects of developer familiarity on bug fixing," *Journal of Systems and Software*, vol. 169, p. 110667, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121220301266

[5] G. Avelino, L. Passos, A. Hora, and M. T. Valente, "A novel approach for estimating truck factors," in *2016 IEEE 24th International Conference on Program Comprehension (ICPC)*, 2016, pp. 1–10.

[6] V. Haratian, M. Evtikhiev, P. Derakhshanfar, E. Tüzün, and V. Kovalenko, "Bfsig: Leveraging file significance in bus factor estimation," in *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2023. New York, NY, USA: Association for Computing Machinery, 2023, p. 1926–1936. [Online]. Available: https://doi.org/10.1145/3611643.3613877

[7] M. Joblin, S. Apel, C. Hunsen, and W. Mauerer, "Classifying developers into core and peripheral: An empirical study on count and network metrics," in *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, 2017, pp. 164–174.

[8] C. Jergensen, A. Sarma, and P. Wagstrom, "The onion patch: migration in open source ecosystems," in *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*, ser. ESEC/FSE '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 70–80. [Online]. Available: https://doi.org/10.1145/2025113.2025127

[9] M. Ferreira, M. T. Valente, and K. Ferreira, "A comparison of three algorithms for computing truck factors," in *Proceedings of the 25th International Conference on Program Comprehension*, ser. ICPC '17. Buenos Aires, Argentina: IEEE Press, 2017, p. 207–217. [Online]. Available: https://doi.org/10.1109/ICPC.2017.35

[10] E. Jabrayilzade, M. Evtikhiev, E. Tüzün, and V. Kovalenko, "Bus factor in practice," in *Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice*, ser. ICSE-SEIP '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 97–106. [Online]. Available: https://doi.org/10.1145/3510457.3513082

[11] M. Joblin, B. Eckl, T. Bock, A. Schmid, J. Siegmund, and S. Apel, "Hierarchical and hybrid organizational structures in open-source software projects: A longitudinal study," *ACM Trans. Softw. Eng. Methodol.*, vol. 32, no. 4, May 2023. [Online]. Available: https://doi.org/10.1145/3569949

[12] T. Fritz, J. Ou, G. C. Murphy, and E. Murphy-Hill, "A degree-of-knowledge model to capture source code familiarity," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1*, ser. ICSE '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 385–394. [Online]. Available: https://doi.org/10.1145/1806799.1806856

[13] T. Fritz, G. C. Murphy, E. Murphy-Hill, J. Ou, and E. Hill, "Degree-of-knowledge: Modeling a developer's knowledge of code," *ACM Trans. Softw. Eng. Methodol.*, vol. 23, no. 2, Apr. 2014. [Online]. Available: https://doi.org/10.1145/2512207

[14] V. Cosentino, J. L. C. Izquierdo, and J. Cabot, "Assessing the bus factor of git repositories," in *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, 2015, pp. 499–503.

[15] P. C. Rigby, Y. C. Zhu, S. M. Donadelli, and A. Mockus, "Quantifying and mitigating turnover-induced knowledge loss: case studies of chrome and a project at avaya," in *Proceedings of the 38th International Conference on Software Engineering*, ser. ICSE '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1006–1016. [Online]. Available: https://doi.org/10.1145/2884781.2884851

[16] M. S. Zanetti, I. Scholtes, C. J. Tessone, and F. Schweitzer, "The rise and fall of a central contributor: Dynamics of social organization and performance in the gentoo community," in *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*. IEEE, May 2013, pp. 49–56.

[17] C. Bird, D. Pattison, R. D'Souza, V. Filkov, and P. Devanbu, "Latent social structure in open source projects," in *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ser. SIGSOFT '08/FSE-16. New York, NY, USA: Association for Computing Machinery, 2008, p. 24–35. [Online]. Available: https://doi.org/10.1145/1453101.1453107

[18] J. Zhang and Y. Luo, "Degree centrality, betweenness centrality, and closeness centrality in social network," in *Proceedings of the 2017 2nd International Conference on Modelling, Simulation and Applied Mathematics (MSAM2017)*. Atlantis Press, 2017, pp. 300–303. [Online]. Available: https://doi.org/10.2991/msam-17.2017.68

[19] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web," Tech. Rep., 1998. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.31.1768

[20] M. Golzadeh, D. Legay, A. Decan, and T. Mens, "Bot or not? detecting bots in github pull request activity based on comment similarity," in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, ser. ICSEW'20. New York, NY, USA: Association for Computing Machinery, 2020, p. 31–35. [Online]. Available: https://doi.org/10.1145/3387940.3391503

[21] P. Chantamit-O-Pas, "Knowledge acquisition, representation and transfer in the metamodel for software development methodologies," in *2019 11th International Conference on Information Technology and Electrical Engineering (ICITEE)*, 2019, pp. 1–6.

[22] F. Ricca, A. Marchetto, and M. Torchiano, "On the difficulty of computing the truck factor," in *Product-Focused Software Process Improvement*, D. Caivano, M. Oivo, M. T. Baldassarre, and G. Visaggio, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 337–351.