Corporate Truck Factor: Firm-Level Knowledge Concentration in Linux Kernel Subsystems

Ellian Carlos Costa
Universidade de São Paulo (IME-USP)
São Paulo, Brazil
elliancarlos@gmail.com

Gabriel Braga Lagrotaria
Universidade de São Paulo (IME-USP)
São Paulo, Brazil
gabrielblo@ime.usp.br

Ricardo Hideki Hangai Kojo
Universidade de São Paulo (IME-USP)
São Paulo, Brazil
ricardo.kojo@ime.usp.br

Arthur Pilone
Universidade de São Paulo (IME-USP)
São Paulo, Brazil
arthurpilone@ime.usp.br

Paulo Meirelles
Universidade de São Paulo (IME-USP)
São Paulo, Brazil
paulormm@ime.usp.br

Abstract

Knowledge concentration in Free/Libre and Open Source Software (FLOSS) is usually measured per developer, through metrics such as the Truck Factor. However, paid employees write most of the Linux Kernel code, so one company's departure can leave behind far more knowledge than any per-developer metric would reveal. We present a firm-level, token-based concentration model that joins token authorship from the `cregit` tool with e-mail-domain-to-company resolution, market-concentration metrics from economics, and our proposed corporate Truck Factor, across three Linux subsystems (`amdgpu`, `net`, and `iio`), chosen to span known regimes. Counting by firm rather than by person exposes an employment fold: authorship that looks broadly shared among individuals collapses onto a few employers. The starkest case is `amdgpu`: by individual Truck Factor, it looks safer than `iio` (35 individuals against 31), yet by firm, it is the most fragile of the three (1 company), a reversal that only the firm-level view makes measurable.

Keywords

Truck Factor, Knowledge Concentration, Company Contributions, Linux Kernel, Free Software, Mining Software Repositories

1 Introduction

We propose a model of knowledge concentration in the Linux Kernel that measures the distribution of corporate contributions and the inequality between corporate and individual contributions, using market concentration metrics from economics. Knowledge concentration has been dominated by the Truck Factor (also known as the Bus Factor), the minimal set of developers whose departure would leave a project stranded. The concept traces back to the Agile community [5], which raised concerns about knowledge being held by too few people, thereby supporting practices such as pair programming and collective ownership.

To quantify this phenomenon, the reference algorithm is the Degree-of-Authorship (DoA) greedy removal by Avelino et al. [2]. This algorithm was widely applied in knowledge concentration analysis for software [3, 9–13, 19] and it was directly related to software quality and maintenance [6, 23, 26, 28]. However, these applications have remained entirely at the individual level.

Consequently, existing work on authorship concentration in the Linux Kernel flags a real risk when knowledge is concentrated in too few individuals, but fails to capture the full picture. On large FLOSS projects, the individual Truck Factor understates turnover loss: their subsystems have enough developers that the Truck Factor signals no risk from contributor turnover. However, because many of those developers are paid by an IT company [8, 27], the true risk is far greater if a company leaves. The scale of this risk can be significant given that corporate participation in FLOSS is extensive and, across the literature, is almost always measured by activity rather than knowledge. Linux Foundation reports, for instance, that paid contributions account for above 85% of kernel commits [8]. Similarly, the OpenStack line of work by Zhang et al. imports a dominance measure from economics and finds domination in a majority of repositories, five dominance patterns, and a negative relationship with project survival [31, 34–37]. In the same ecosystem, more than half of the companies that joined a given version later withdrew [32], and the Mozilla/Rust case served as a real example of anchor withdrawal [24]. Coopetition [25], commercial-participation surveys [20], inter-organizational collaboration [29], and developer disengagement [18, 22] round out the strand, but none measure the knowledge left orphaned in the code that remains.

To elevate this analysis to the organizational level, other approaches and tools are required. In this context, affiliation and identity resolution are the primary methods that enable firm-level analysis. For instance, per-commit email-domain affiliation with near-perfect identity merging was used on Rust [33], and enterprise-domain mining reached 89% identification accuracy at scale [30]. Furthermore, `cregit` provides token-level blame over the full git history [14]. Despite these methodological advances, a critical gap in the literature emerges at the intersection of these studies: they rarely cite each other, and none aggregates authorship-based knowledge metrics (DoA, Truck Factor, Rigby-style loss) at the firm level. Specifically, a corporate truck factor, such as the Elephant factor [4], a quantitative indicator of a project's dependence on a small set of corporate contributors, has not been well explored. N'emeth and Wachs explicitly identified this gap and called for research into the systemic effects of firm withdrawal [24]. Our proposed model addresses this call by introducing a corporate Truck Factor metric that, unlike the Elephant factor, is fundamentally token-based [1].

[1]We thank the São Paulo Research Foundation — FAPESP (2025/26679-7), and the São Paulo State Data Analysis System Foundation – SEADE (FAPESP 2023/18026-8).

2 Methodology

We propose an approach to measuring Linux Kernel contributions by extracting tokens from git history, collected via the `cregit` [14], which performs token-level authorship attribution. A token is a sequence of non-blank characters meaningful for a specific language; `cregit` processes the full git history and generates a collection of SQLite databases that map tokens to authors, commit of origin, and prior token versions, allowing us to walk through the history commit by commit at token granularity and attribute each surviving token to the author who last wrote it. German et al. reported an accuracy between 94.5% and 99.2% in identifying which commit introduced a token [14]. Although `cregit` runs over the entire commit history, in this work we consider only the surviving tokens.

Mapping. We used an enriched version of `gitdm`'s kernel affiliation map[2] with a few manual additions for gaps we detected to map each contribution to a company. In total, we mapped 887 e-mail domains to 729 distinct companies. Two non-company labels are kept separate. The *"independent"* label aggregates known personal e-mail domains such as `gmail.com` or `kernel.org`, domains that are in the map but do not denote a company. The *"unknown"* label denotes domains that are not present in the map at all, whose affiliation we could not determine, and that may hide either corporate contributors we failed to map or independent contributors. Both labels are excluded from the firm-level concentration metrics. Because of this *"independent"* and *"unknown"* residual, all corporate-contribution percentages in this paper are lower-bound metrics.

Affiliation is resolved per commit from the author's e-mail domain, so each token is attributed to the company the author held at the time that token was authored. When a developer changes employers, only their subsequent contributions carry the new affiliation, while the tokens they wrote earlier retain the affiliation held when written. Corporate acquisitions are not aggregated. Because the company resolution is coupled to the e-mail domain, a firm that was later acquired remains attributed to its original name.

Accordingly, we ran `cregit` over three subsystems inside the Linux Kernel, chosen to span a spectrum of corporate concentration: the `amdgpu` subsystem (`drivers/gpu/drm/amd`), the `net` subsystem (`net`), and the `iio` subsystem (`drivers/iio`). The `amdgpu` subsystem was chosen as a clearly single-vendor extreme, where essentially one company authors the code. The `net` subsystem was chosen as the opposite: a distributed case in which contributions are spread across many companies, with no single dominant firm. `iio` was chosen as an intermediate case: an actively developed subsystem maintained by multiple consultancy firms alongside silicon vendors. Across the three subsystems, our analysis covers 12.6 million surviving content tokens over 4,600 source code files, including comments, since the Linux Kernel's migration to git in 2005. We drop structural tokens such as punctuation and language keywords, so that a holder's mass reflects the identifiers and expressions they authored rather than the syntax shared by all code. For `amdgpu`, we also drop the machine-derived register headers (suffixes `_enum.h`, `_offset.h`, `_sh_mask.h`): 313,358 tokens, about 6% of the subsystem and essentially all `amdgpu`-authored. Their removal shifts `amdgpu`'s share by under 0.1 pt and does not change our concentration metrics.

[2] https://repo.or.cz/w/git-dm.git

Model. We joined market-structure analysis from microeconomics with knowledge-concentration analysis to characterize how the contributions to each subsystem are distributed. The unit of measurement throughout our analyses is the surviving tokens of each subsystem, attributed to the commit that last authored them.

Let a subsystem have n holders (individuals or firms). Let t_i be the number of surviving tokens attributed to holder i on a given subsystem, and $T = \sum_{j=1}^{n} t_j$ the total attributed mass, holder i's share is $s_i = t_i/T$. At the firm grain, T is the known-affiliation mass: the non-organizational buckets (Unknown and Independent) are excluded from the firm shares, and the corporate coverage is reported alongside so that concentration is read against how much of the subsystem is corporate at all.

$$\mathrm{HHI} = \sum_{i=1}^{n} s_i^2, \qquad N_{\mathrm{eff}} = \frac{1}{\mathrm{HHI}}, \tag{1}$$

We use the Herfindahl-Hirschman Index (HHI) [16, 17] to measure how token-level contributions are distributed across the holders of a subsystem, and from it the effective number of firms, its numbers-equivalent [1], where s_i is the share of the known-affiliation token mass held by holder i. The HHI ranges from $1/n$ when all holders contribute equally to 1 when a single holder owns every token. N_{eff} reads the same value as the count of equal-sized firms that would produce the same concentration.

As a complementary inequality view, we report the Gini coefficient [15] and Lorenz curves [21] at both grains. We also adapt the Truck Factor of Avelino et al. [2] to our setting. Their method identifies the experts of a file through the Degree of Authorship (DoA); since our data records token ownership directly, we replace that test with a token-share threshold: a holder is an expert of a file when it owns at least a fraction τ of its surviving tokens, with $\tau = 0.05$ matching the 5% ownership threshold. For instance, Bird et al. [6] use it to distinguish major from minor contributors, and swept from 0.02 to 0.20 as a robustness check. When no holder of a file clears τ, we treat its single largest holder as the expert, ensuring that every file starts with at least one expert owner and that none is orphaned before any removal. A file becomes orphaned once all of its expert owners have been removed. **Our corporate Truck Factor** TF **is the minimum number of holders whose removal orphans more than half of the files:**

$$\mathrm{TF} = \min\left\{ k \;:\; \frac{|\text{files orphaned after removing top } k \text{ holders}|}{|\text{files}|} > 0.5 \right\}. \tag{2}$$

We compute it by greedily removing the holder that is the sole expert of the largest number of still-covered files, and the non-organizational buckets are never removable. TF indicates how many individual authors and how many companies would have to withdraw before the majority of the subsystem is left without an expert owner. The gap between the two is the withdrawal analog of the collapse from individual to firm counts.

3 Results

Affiliation coverage was 59.3% for the `net` subsystem, 61.8% for `iio`, and 97.9% for `amdgpu`, as lower-bound values. Considering the excluded affiliations, the upper bound for the same coverage rises to 86.0% for `net`, 73.0% for `iio`, and 98.9% for `amdgpu` (Figure 1).

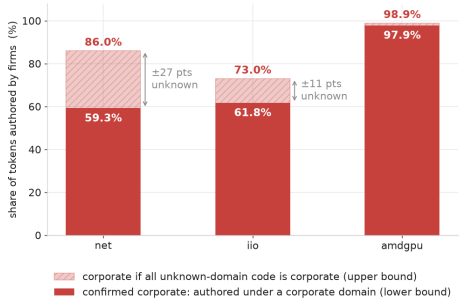

Figure 1: Corporate share of surviving code tokens.

This data is consistent with previous work [8], which shows that employees make most kernel contributions today. The amdgpu coverage is much higher than the others, since it is maintained by AMD employees using their AMD-based e-mail addresses to commit code. The gap between the lower and upper bound estimates is higher in the net subsystem than iio. This follows from a known limitation of domain-based affiliation: many kernel developers commit from personal or vanity domains rather than a corporate address, which is why affiliation tools maintain per-developer overrides on top of the domain map [7, 30]. Inspecting net's unknown bucket, its largest unmapped domains are precisely such personal domains of senior maintainers who are in fact employed, so net's true corporate share sits toward the upper bound.

Once we know how much of each subsystem is corporately maintained, we ask how concentrated that corporate mass is and how the picture changes when we count by firm rather than by person. Figure 2 reports both at each grain for corporate committed code only. By HHI (Subfigure 2.A), amdgpu is a one-company distribution while iio and net are far more dispersed; the effective holders (Subfigure 2.B) restate this as an intuitive count of about 13 firms for net, 8 for iio, and 1 for amdgpu. iio is lower than net in effective firms for two complementary reasons. Its top company, "Analog Devices", holds 30% of the surviving tokens while net's top company, "Intel", holds only 17.7%, so iio's effective count is pulled down by a larger dominant share. At the same time, net has more firms, 185 against 85, which raises its effective count further.

Figure 2 makes the employer-employee fold clear: the effective number of individuals is large in every subsystem, but recounting by employer folds them into far fewer firms, by a factor that grows as a single firm captures the subsystem and reaches 37 for amdgpu. The raw head-counts give the underlying team sizes: net is authored by 1,738 corporate employees across 185 firms, iio by 358 across 85, and amdgpu by 720 across 37, a mean of about 9, 4, and 20 authors per firm. However, the median firm contributes only 2 to 3 authors, and roughly a third contribute just one, so a long tail of drive-by corporate contributors sits behind a few large in-house teams.

Figure 3 shows that even where the effective-firm count suggests authorship is distributed, the distribution is far from even: every Lorenz curve bows steeply, so a few companies author most tokens even on iio and net, where the HHI reads as "competition." All Gini coefficients range from 0.77 to 0.97.

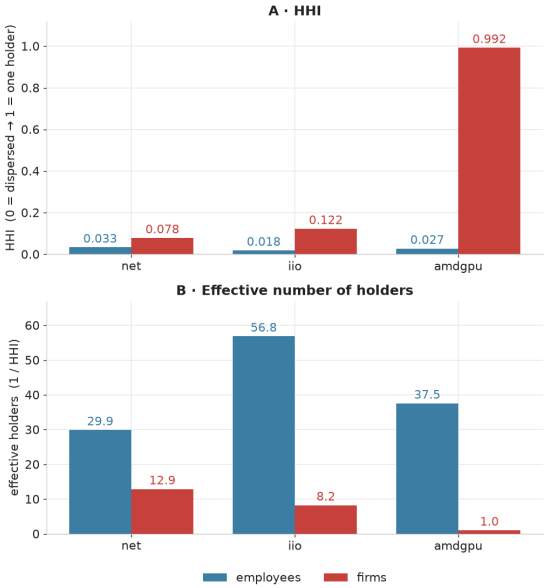

Figure 2: Concentration on corporate code.

Our corporate Truck Factor identified the risk of withdrawal. At the firm grain, the number of companies whose removal orphans more than half of the files is 17 for net, 7 for iio, and 1 for amdgpu. The three values describe a spectrum rather than a uniform alarm. amdgpu is a single point of failure: removing its top firm alone leaves 98% of its files orphaned, so the subsystem is left with almost no expert owner once that company withdraws. iio is moderate at 7, and net is robust at 17, so at the firm grain net tolerates the departure of many companies before it is stranded.

Table 1: Corporate Truck Factor

τ	amdgpu	iio	net
0.02	1	8	23
0.03	1	7	20
0.05 (default)	1	7	17
0.075	1	6	14
0.10	1	6	12
0.15	1	5	10
0.20	1	5	9

As it depends on the expert threshold τ, we recompute our corporate Truck Factor by sweeping τ from 0.02 to 0.20 (Table 1). The ranking amdgpu $\ll$ iio $<$ net holds at every threshold, and the three subsystems never cross; amdgpu remains at 1 across the entire range, since a single firm owns more than half the files regardless of the cut, while iio and net decline monotonically as a higher bar leaves fewer firms qualified as experts. A 95% file-bootstrap confirms the separation: the intervals do not overlap at any threshold (for example, at $\tau = 0.05$, amdgpu $[1, 1]$, iio $[6, 7]$, net $[16, 18]$).

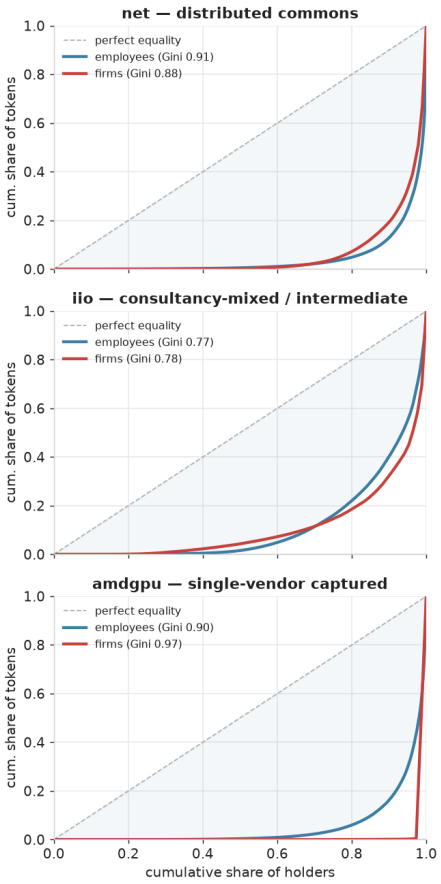

Figure 3: Lorenz curves of the surviving token mass, one panel per subsystem, for both individual and firms.

Figure 4 traces the full withdrawal trajectory at both grains: the cumulative share of files orphaned as holders are removed most-expert first. The firm curves separate the regimes, with `amdgpu` jumping to 98% on the first removal, `iio` crossing half its files at the seventh firm, and `net` climbing gradually and only crossing near the seventeenth. The gap between the two grains is the sharpest single statement of capture: `amdgpu` needs 35 individual authors removed to orphan half its files, but only 1 firm, whereas `net` needs 85 individuals against 17 firms, and `iio` 31 against 7. Where the two Truck Factors stay close, the people-level diversity carries to the firm level, with each expert sitting at a different employer. Where the firm value collapses far below the individual one, as in `amdgpu` (35 against 1), a single employer stands in for the whole expert workforce. This inverts the `amdgpu`–`iio` ordering across grains: at the individual grain `amdgpu` appears the safer of the two (35 experts against `iio`'s 31), yet at the firm grain it is by far the most fragile of all three (1, against `iio`'s 7 and `net`'s 17), while `net` remains the most robust at both grains (85 individuals, 17 firms).

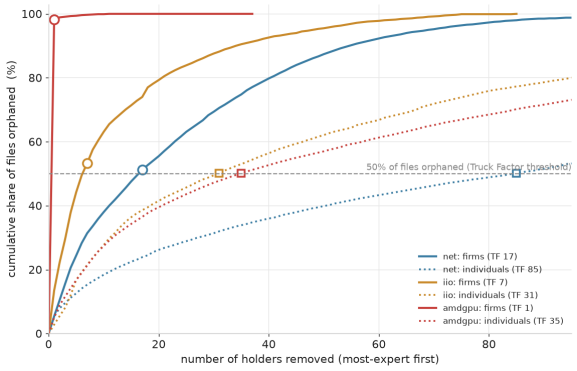

Figure 4: Withdrawal trajectory at both grains

4 Discussion

Both Truck Factor and DoA answer one question well: how many holders must leave before the project is stranded. However, that answer is threshold-based and enumerable. The Gini coefficient, the HHI, and the effective number of firms are continuous and share-based, so they distinguish subsystems that share a Truck Factor yet differ in ownership balance and compare across subsystems of different sizes. They show that **a subsystem can appear widely shared among individual authors yet be held by very few firms, which individual-level metrics cannot reveal**; `amdgpu`'s cross-grain inversion, robust by people yet fragile by firm, is the most severe case. **As threats to validity**, the domain-based affiliation might not reflect the real employee-to-employer relationship. Also, the choice of the three subsystems might not be representative of the entire kernel. Moreover, employees' contributions might be much higher than measured, as the "unknown" could be entirely made up by companies. The analysis of surviving tokens, rather than token history, and, altogether, the use of tokens as a knowledge proxy may only partially reflect contributor knowledge.

5 Conclusion and Future Work

The contribution of this work is a corporate Truck Factor and the market concentration model that underpins it, which together distinguish the risk of knowledge loss between companies and individuals and identify which subsystems are diversified and which are captured, rather than flagging every subsystem as equally at risk. For future work, this model can be replicated across the entire Linux Kernel source code, and the analysis can be made temporal, considering both dead and surviving tokens. This analysis raises research questions such as how the departure or merger of companies has affected the kernel's knowledge base over time, and what happens to the code they leave behind.

Artifact Availability

The scripts that produce every figure and table, along with the affiliation map, are publicly available[3]. Anonymized token level data per subsystem is also publicly available [4].

[3]https://github.com/EllianCarlos/cbsoft-vem2026-corporate-truck-factor
[4]https://zenodo.org/records/21302469

References

[1] M. A. Adelman. 1969. Comment on the "H" Concentration Measure as a Numbers-Equivalent. *The Review of Economics and Statistics* 51, 1 (1969), 99–101. doi:10.2307/1926955

[2] Guilherme Avelino, Leonardo Passos, Andre Hora, and Marco Tulio Valente. 2016. A Novel Approach for Estimating Truck Factors. In *Proceedings of the 24th IEEE International Conference on Program Comprehension (ICPC)*. 1–10. doi:10.1109/ICPC.2016.7503718

[3] Guilherme Avelino, Leonardo Passos, Andre Hora, and Marco Tulio Valente. 2017. Assessing Code Authorship: The Case of the Linux Kernel. In *Open Source Systems: Towards Robust Practices (OSS 2017), IFIP AICT 496*. 151–163. doi:10.1007/978-3-319-57735-7_15

[4] Elizabeth Barron, Matt Germonprez, Kevin Lumbard, Yash Prakash, Georg Link, and Maalik Peculiar C. Umeh. 2026. *Metric: Elephant Factor*. CHAOSS (Community Health Analytics Open Source Software). https://chaoss.community/?p=3940 A Linux Foundation project. Accessed on Aug 10th 2026..

[5] Kent Beck and Cynthia Andres. 2004. *Extreme Programming Explained: Embrace Change* (2nd ed.). Addison-Wesley.

[6] Christian Bird, Nachiappan Nagappan, Brendan Murphy, Harald Gall, and Premkumar Devanbu. 2011. Don't Touch My Code! Examining the Effects of Ownership on Software Quality. In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering (ESEC/FSE)*. 4–14. doi:10.1145/2025113.2025119

[7] Jonathan Corbet. [n. d.]. gitdm: The "git data miner". https://git.lwn.net/gitdm.git. Affiliation via a domain-to-employer EmailMap, with per-developer `user@domain` overrides and dated entries for developers who keep personal e-mail across employers.

[8] Jonathan Corbet and Greg Kroah-Hartman. 2017. *2017 Linux Kernel Development Report*. Technical Report. The Linux Foundation.

[9] Valerio Cosentino, Javier Luis Cánovas Izquierdo, and Jordi Cabot. 2015. Assessing the Bus Factor of Git Repositories. In *Proceedings of the 22nd IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER)*. 499–503. doi:10.1109/SANER.2015.7081864

[10] Otávio Cury and Guilherme Avelino. 2024. Knowledge Islands: Visualizing Developers' Knowledge Concentration. In *Anais do XXXVIII Simpósio Brasileiro de Engenharia de Software (SBES)*. SBC, 789–795. doi:10.5753/sbes.2024.3610

[11] Ahmed El Cheikh Ammar, Sukru Eraslan, and Yeliz Yesilada. 2025. Predicting the Truck Factor in a Software Repository Using Machine Learning. *Information and Software Technology* 184 (2025), 107765. doi:10.1016/j.infsof.2025.107765

[12] Mivian Ferreira, Thaís Mombach, Marco Tulio Valente, and Kecia Ferreira. 2019. Algorithms for Estimating Truck Factors: A Comparative Study. *Software Quality Journal* 27, 4 (2019), 1583–1617. doi:10.1007/s11219-019-09457-2

[13] Mivian Ferreira, Marco Tulio Valente, and Kecia Ferreira. 2017. A Comparison of Three Algorithms for Computing Truck Factors. In *Proceedings of the 25th IEEE International Conference on Program Comprehension (ICPC)*. 207–217. doi:10.1109/ICPC.2017.35

[14] Daniel M. German, Bram Adams, and Kate Stewart. 2019. cregit: Token-Level Blame Information in Git Version Control Repositories. *Empirical Software Engineering* 24, 4 (2019), 2725–2763. doi:10.1007/s10664-019-09704-x

[15] Corrado Gini. 1912. *Variabilità e Mutabilità: Contributo allo Studio delle Distribuzioni e delle Relazioni Statistiche*. Tipografia di Paolo Cuppini, Bologna.

[16] Orris C. Herfindahl. 1950. *Concentration in the Steel Industry*. PhD dissertation, Columbia University.

[17] Albert O. Hirschman. 1964. The Paternity of an Index. *The American Economic Review* 54, 5 (1964), 761–762.

[18] Giuseppe Iaffaldano, Igor Steinmacher, Fabio Calefato, Marco Gerosa, and Filippo Lanubile. 2019. Why Do Developers Take Breaks from Contributing to OSS Projects? A Preliminary Analysis. arXiv:1903.09528 [cs.SE]

[19] Elgun Jabrayilzade, Mikhail Evtikhiev, Eray Tüzün, and Vladimir Kovalenko. 2022. Bus Factor in Practice. In *Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. 97–106. doi:10.1145/3510457.3513082

[20] Xuetao Li, Yuxia Zhang, Cailean Osborne, Minghui Zhou, Zhi Jin, and Hui Liu. 2025. A Systematic Literature Review of Commercial Participation in Open Source Software. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 34, 2, Article 33 (2025), 31 pages. doi:10.1145/3690632

[21] M. O. Lorenz. 1905. Methods of Measuring the Concentration of Wealth. *Publications of the American Statistical Association* 9, 70 (1905), 209–219. doi:10.2307/2276207

[22] Courtney Miller, David Gray Widder, Christian Kästner, and Bogdan Vasilescu. 2019. Why Do People Give Up FLOSSing? A Study of Contributor Disengagement in Open Source. In *Open Source Systems (OSS 2019), IFIP AICT 556*. 116–129. doi:10.1007/978-3-030-20883-7_11

[23] Mathieu Nassif and Martin P. Robillard. 2017. Revisiting Turnover-Induced Knowledge Loss in Software Projects. In *Proceedings of the 33rd IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 261–272. doi:10.1109/ICSME.2017.64

[24] Brigitta Németh and Johannes Wachs. 2025. Anchor Sponsor Firms in Open Source Software Ecosystems. arXiv:2502.09060 [cs.SE]

[25] Anh Nguyen-Duc, Daniela S. Cruzes, Terje Snarby, and Pekka Abrahamsson. 2019. Do Software Firms Collaborate or Compete? A Model of Coopetition in Community-Initiated OSS Projects. *e-Informatica Software Engineering Journal* 13, 1 (2019), 37–62. doi:10.5277/e-Inf190102

[26] Olivier Nourry, Masanari Kondo, Shinobu Saito, Yukako Iimura, Naoyasu Ubayashi, and Yasutaka Kamei. 2024. Myth: The Loss of Core Developers Is a Critical Issue for OSS Communities. arXiv:2412.00313 [cs.SE]

[27] Dirk Riehle, Philipp Riemer, Carsten Kolassa, and Michael Schmidt. 2014. Paid vs. Volunteer Work in Open Source. In *Proceedings of the 47th Hawaii International Conference on System Sciences (HICSS)*. IEEE, 3286–3295. doi:10.1109/HICSS.2014.407

[28] Peter C. Rigby, Yue Cai Zhu, Samuel M. Donadelli, and Audris Mockus. 2016. 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 (ICSE)*. 1006–1016. doi:10.1145/2884781.2884851

[29] Roland Robert Schreiber and Thomas Wieland. 2026. Inter-organizational Collaborations in Open-Source Software Ecosystems. *Journal of Systems and Software* 235 (2026), 112765. doi:10.1016/j.jss.2025.112765

[30] Diomidis Spinellis, Zoe Kotti, Konstantinos Kravvaritis, Georgios Theodorou, and Panos Louridas. 2020. A Dataset of Enterprise-Driven Open Source Software. In *Proceedings of the 17th International Conference on Mining Software Repositories (MSR)*. 533–537. doi:10.1145/3379597.3387495

[31] Yuxia Zhang, Hao He, and Minghui Zhou. 2022. Commercial Participation in OpenStack: Two Sides of a Coin. *Computer* 55, 2 (2022), 78–84. doi:10.1109/MC.2021.3133052

[32] Yuxia Zhang, Hui Liu, Xin Tan, Minghui Zhou, Zhi Jin, and Jiaxin Zhu. 2022. Turnover of Companies in OpenStack: Prevalence and Rationale. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 31, 4, Article 75 (2022), 24 pages. doi:10.1145/3510849

[33] Yuxia Zhang, Mian Qin, Klaas-Jan Stol, Minghui Zhou, and Hui Liu. 2024. How Are Paid and Volunteer Open Source Developers Different? A Study of the Rust Project. In *Proceedings of the 46th International Conference on Software Engineering (ICSE)*. 195:1–195:13. doi:10.1145/3597503.3639197

[34] Yuxia Zhang, Klaas-Jan Stol, Hui Liu, and Minghui Zhou. 2022. Corporate Dominance in Open Source Ecosystems: A Case Study of OpenStack. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*. 1048–1060. doi:10.1145/3540250.3549117

[35] Yuxia Zhang, Xin Tan, Minghui Zhou, and Zhi Jin. 2018. Companies' Domination in FLOSS Development: An Empirical Study of OpenStack. In *Proceedings of the 40th International Conference on Software Engineering: Companion (ICSE Companion)*. 440–441. doi:10.1145/3183440.3195047

[36] Yuxia Zhang, Minghui Zhou, Audris Mockus, and Zhi Jin. 2021. Companies' Participation in OSS Development: An Empirical Study of OpenStack. *IEEE Transactions on Software Engineering* 47, 10 (2021), 2242–2259. doi:10.1109/TSE.2019.2946156

[37] Yuxia Zhang, Minghui Zhou, Klaas-Jan Stol, Jianyu Wu, and Zhi Jin. 2020. How Do Companies Collaborate in Open Source Ecosystems? An Empirical Study of OpenStack. In *Proceedings of the 42nd International Conference on Software Engineering (ICSE)*. 1196–1208. doi:10.1145/3377811.3380376