Exploring Semantic Stability Across Reviews in the Linux Kernel

Lucas Ciziks
Universidade de São Paulo (IME-USP)
São Paulo, Brazil
luciziks@usp.br

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

Marco Aurélio Gerosa
Northern Arizona University (NAU)
Flagstaff, AZ, USA
Marco.Gerosa@nau.edu

Abstract

Code review is credited with substantially changing a patch's code between its first submission and the version that eventually lands. However, prior work typically studied only the final merged patch without comparing it to the first submission. We present a function-level measurement that tracks 10,117 trajectories (each function followed across the numbered revisions of one patch series) through the patch history of the Linux IIO subsystem, comparing similarity scores against unrelated function pairs as a baseline. A naive reading yields near-total similarity, but this is largely an artifact of composition: 75.3% of tracked trajectories are never textually modified between versions, contributing a trivial 100% similarity that inflates the headline. Restricting to the trajectories with a real edit, semantic purpose is still largely preserved (mean similarity 0.990 vs. a 0.909 baseline), but drift appears to concentrate in the first review round mainly because later rounds contain more functions that nobody touched, not because edits become more conservative over time. After controlling for it, a statistically detectable but small residual effect remains. This points to an open question: whether near-ceiling similarity reflects preserved purpose or a measurement tool that cannot detect the significance of small, localized edits. We present this work as a first look and outline next steps.

Keywords

Code review, Code embeddings, Semantic similarity, Linux kernel

1 Introduction

A long review thread on a kernel mailing list is often read as evidence that a contribution was contested. A substantial body of patch analysis work nonetheless treats the final merged commit as the primary unit of analysis [8, 9, 11], leaving aside how fixes evolve before landing. A recent large-scale study of Linux kernel bug-fix lifecycles finds that accepted repairs frequently spread across files and functions beyond where the fault was first observed, as reviewer feedback enforces constraints not visible in the initial bug report [1]. What is missing is a direct measurement of the semantic side of that evolution: does review change what a function is *for*, or does it converge on an implementation of a purpose already fixed?

We address this question in the Linux IIO (Industrial I/O) subsystem, where our own contributions to IIO drivers gave us repeated first-hand exposure to the phenomenon: v1 and vN code differ substantially line by line, even though reviewers and authors treat the series as converging on "the same fix." This gap between visible diffs and perceived semantic stability motivates our study. In the kernel's submission convention [17], a patch series progresses through numbered revisions (v1, v2, …, vN) on the mailing list before being merged or abandoned; we track each function across these revisions and pose two research questions:

RQ1. *Does a function's semantic purpose change between the first and last submitted version of a patch series, relative to what unrelated code looks like in the same embedding space?*

RQ2. *If some drift occurs, is it spread evenly across revisions, or concentrated in a specific part of the process?*

This distinction matters beyond the IIO subsystem. Tools that score a patch averaging similarity across all functions in a series, as in patch classification, fix-to-bug matching, or automated review assistants, risk treating "no edit occurred" and "review preserved purpose" as the same signal. As we show below, a large share of tracked functions in a patch series are never retouched after their first submission, contributing a similarity score of 1.0 to that aggregate. A near-ceiling score is consequently a weak guarantee: it can reflect a genuinely stable function, or simply one no reviewer touched, or a small, localized edit the embedding is not sensitive enough to register. Left unaddressed, this conflation can make semantic similarity metrics appear more reliable than they are, particularly for small, targeted edits, such as security fixes that matter most. We argue that disaggregating by whether an edit actually occurred should be a baseline requirement for this kind of measurement, not an optional refinement. Our main contribution is a **research perspective**: a function-level measurement of semantic stability across kernel patch review, with an account of what a naive reading of that measurement gets wrong and why. We also identify a limitation in whole-function cosine similarity over pretrained code embeddings and lay out a research agenda[1].

2 Related Work

Prior patch analysis work, including kernel-specific classifiers such as PatchNet [8] and broader VFC identification approaches [9, 11], operates on the landed form of a patch, leaving open whether that form is semantically representative of the fix throughout its review history. Our RQ1 addresses this gap directly. The closest related work is a large-scale study of Linux kernel bug-fix lifecycles [1], which finds that accepted repairs frequently spread across files and functions beyond where the fault was first observed, as reviewer feedback enforces constraints not visible in the initial report. Tracking a patch through its pre-integration history on a mailing list is a known hard problem. Ramsauer et al. [15] address it with a language-independent similarity method validated against a hand-built ground truth. We use a coarser, cheaper key (normalized subject line, file path, function name), adequate for within-series tracking but a natural point of comparison for follow-up work. As we note in Section 6, we plan to quantify its error rate against that ground truth.

Our data comes from the LKML5Ws dataset [13] and complements DUKS [14], a dashboard that unifies mailing-list and git-tree

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

data to visualize kernel process-level evolution, including contributor activity and commit flow between trees. That dataset lacks a code-content dimension. A validated semantic-stability metric like ours would be a natural addition.

The issue we discuss in this paper is also documented elsewhere. Raw cosine similarity in transformer embedding spaces is known to be inflated by anisotropy, so even unrelated pairs score artificially high [3]. Pretrained code encoders such as CodeBERT [5] and UniXcoder [6] share this architecture, so the same risk plausibly applies to them. We control for this with a random-pair null throughout. Nevertheless, two recent results suggest this correction is not enough. Nikiema et al. [12] systematically tested 18 similarity measures under controlled small transformations and found that embedding-based methods, cosine similarity in particular, routinely score semantically opposite code as similar, while switching from cosine to Euclidean distance on the same embeddings improves discrimination by 24 to 66%. Farhad and Dass [4] compare vulnerable and patched function pairs across multiple encoders, including UniXcoder, and find similarity near 0.99 for nearly all pairs across models regardless of whether the patch closes a real vulnerability. They interpret this not as encoder failure but as evidence that whole-snippet semantic similarity cannot detect relevant, spatially localized changes, and complement it with a structural (AST-based) signal. Both, in different corpora, describe the same problem we report below.

3 Data and Method

Corpus. We build on LKML5Ws [13], a dataset of Linux Kernel Mailing List messages, and extract function-level records for the IIO subsystem. For each patch, we fetch the pre-patch file blob via `git cat-file`, reconstruct a standalone single-file diff, and apply it with `git apply` in an isolated scratch directory to obtain the real post-patch file. Rows are dropped when (1) the blob SHA is absent from the local tree, (2) `git apply` fails because the hunk context no longer matches the fetched blob (e.g., the email's diff was generated against a slightly different base), (3) the file is newly introduced or deleted, (4) the file extension is outside `{.c, .h, .rs}`, or (5) the touched function cannot be matched unambiguously by name in both the before- and after-file. This conservative strategy yields 64,044 function-version records.

Trajectories. A trajectory is one function followed across the versions of one series, keyed by (normalized subject line, file path, function name), a key that remains stable across v1..vN, unlike the per-email message ID. Of 37,643 trajectories, 10,117 have at least two clean versions and therefore a defined endpoint pair (v_1, v_N). Of those, 4,387 have three or more. This key is a heuristic rather than a validated identity-resolution method, such as that of Ramsauer et al. [15]. Quantifying its mislinking or false-drop rate on this corpus is part of our research agenda (Section 6).

Embedding. Each function body, before and after a patch, is embedded with UniXcoder [6] (`microsoft/unixcoder-base`), using attention-mask-weighted mean pooling over the last hidden state [16], truncated to the model's 512-token pretraining limit. Functions in this corpus average ≈40 LOC, so truncation affects only the long tail; the number of truncated inputs is logged per run. Every distribution below is reported alongside a random-pair null, the similarity of embeddings for unrelated function pairs sampled from the same corpus (mean 0.910). An observed similarity is informative only to the extent it clears this floor; though, clearing it turned out to be a weaker guarantee than we initially assumed.

Data Quality. We validate function-body reconstruction directly with two checks, rather than assuming hunk-based extraction gets boundaries right by construction. First, a signature-presence check flags trajectories in which a function introduced partway through a series was attached to an unrelated code fragment. Second, we test whether every reconstructed function body is brace-balanced (with equal opening and closing braces) and string-literal-aware. This approach revealed a boundary-detection issue in an earlier version of the extractor, which we corrected before computing the results by re-embedding the full corpus against brace-balanced boundaries. In the corrected corpus, 96.5% of before- and after-bodies are brace-balanced. The remainder is attributable to macro-heavy IIO code whose braces are unbalanced inside string or preprocessor content for legitimate syntactic reasons.

Composition of the Sample. Before trusting any similarity number, we characterize the sample by comparing the *literal function-body text*—not the embedding—across consecutive and endpoint versions. Of the 10,117 endpoint trajectories, 7,616 (75.3%) have *byte-identical* v_1 and v_N bodies: the function appears in the diff context of a later revision but is never re-touched after its first submission. Hence, the encoder receives identical text for v_1 and v_N and reports a similarity of exactly 1.0, conveying no information about whether the review preserves its purpose. The remaining 2,501 trajectories (24.7%) carry a real edit between v_1 and v_N. Restricted to this subset, mean endpoint similarity is 0.990 (median 0.992, 25th percentile 0.989, minimum 0.748), still above the 0.909 null for this subset (Mann-Whitney $U = 6{,}206{,}684$, $p \approx 0$), but lower and more spread out than the pooled headline figure of 0.997, which includes the trivial cases. The same pattern holds at the transition level: of 18,656 consecutive-version transitions, 15,489 (83.0%) involve no textual change to the tracked function that round.

Measurements. We compute three quantities from the same embedding cache. *M1* is a sanity check: cosine similarity between the before and after version of a function within a single patch version, binned by lines changed, confirming the embedding tracks edit magnitude. *M2*, endpoint drift, is the cosine similarity between v_1 and v_N for each trajectory with at least two clean versions, the direct measurement for RQ1. *M3*, consecutive-transition drift, is the cosine similarity between adjacent clean versions (v_i, v_{i+1}), grouped by transition index, the direct measurement for RQ2. All comparisons use the Mann-Whitney U test [10], since similarity scores are strongly left-skewed rather than normally distributed. Given the small effect sizes found (Section 4), p-values are treated as indicative rather than confirmatory and we rely on effect size where available.

4 Results

We report results in the order of Section 3: M1 (sanity check), M2 (endpoint drift, RQ1), and M3 (consecutive-transition drift, RQ2).

M1: sanity check. Mean within-version similarity is 0.991 against a null of 0.910, and decreases monotonically as edits get larger (Figure 1), from 0.994 at a single changed line to 0.982 above 100 lines.

The embedding responds to edit size in the expected direction. Even the largest bucket (100+ lines) sits at 0.982, above the 0.910 null.

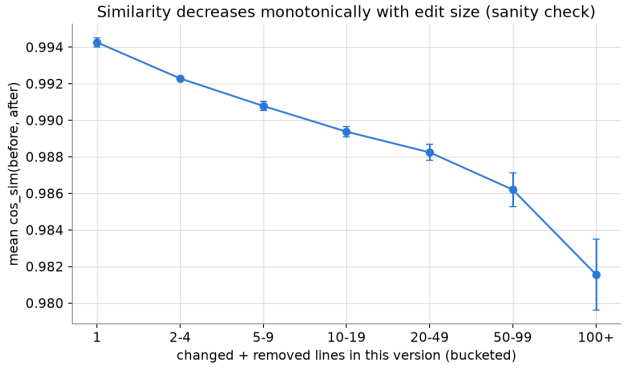

Figure 1: Mean within-version similarity by edit size, with 95% confidence intervals (M1, $n = 64{,}044$). Similarity decreases monotonically as edits increase, but remains well above the 0.910 null even in the largest bucket.

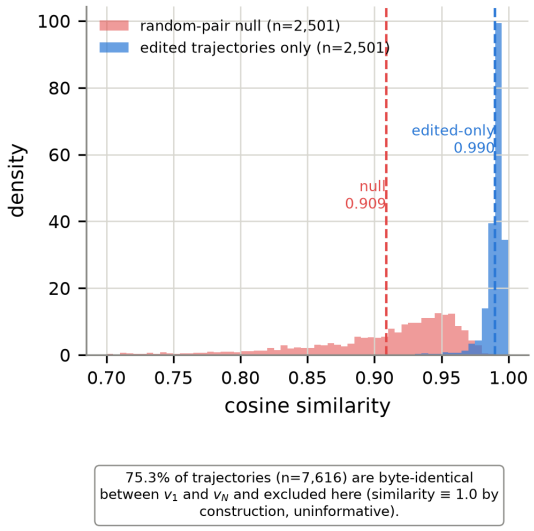

Figure 2: Distribution of $\cos(v_1, v_N)$ restricted to the 2,501 trajectories with a real edit between v_1 and v_N (mean 0.990), against a random-pair null of the same size (mean 0.909). All percentages shown derive from byte-level textual comparison, not from the embedding.

M2: endpoint drift (RQ1). Across 10,117 trajectories, mean cosine similarity between v_1 and v_N is 0.997 (median 1.000, std 0.007, minimum 0.748), against a null mean of 0.910 (Mann-Whitney $U = 102{,}154{,}592$, $p \approx 0$). This result answers RQ1: the semantic purpose is preserved end-to-end for almost every trajectory. However, three-quarters of that mass is trivial: the underlying text never changed, yielding a similarity of 1.0. Figure 2 shows the distribution restricted to the 2,501 trajectories with a real edit instead: a still-high but more modest mean of 0.990 against a matched random-pair null of the same size (mean 0.909).

A separate concern is whether 0.990 merely restates that most tracked functions receive small cumulative edits: trajectories with 500+ cumulative edited lines (1.3% of all 10,117 trajectories; $n = 134$) still average 0.996 similarity, and even the 1000+ bucket ($n = 10$) averages 0.993, far above the 0.910 null. Cumulative edit volume moves the number, but not enough to explain the ceiling effect.

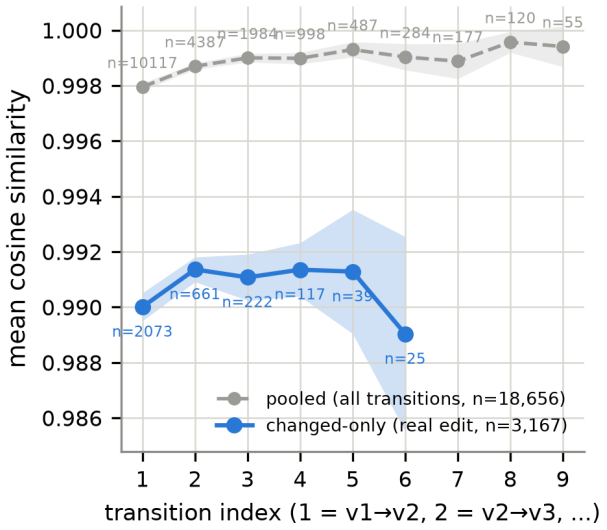

Figure 3: Mean similarity by ordinal transition index: pooled (all transitions, gray dashed) versus restricted to transitions with a real textual edit that round (blue solid). The changed-only series is plotted only through the sixth transition; beyond that point, fewer than 25 genuinely edited transitions remain and per-index estimates are unreliable.

M3: consecutive-transition drift (RQ2). Across 18,656 transitions, mean similarity in the pooled series (Figure 3, gray dashed line) rises from the first to the fifth transition (0.99795 to 0.99930) and becomes noisier at higher indices, where the sample size per transition is small. Pooling the first transition against all later ones, the first round has lower similarity (mean 0.998) than later rounds (mean 0.999, Mann-Whitney $U = 40{,}164{,}232$, $p = 1.12 \times 10^{-18}$). This difference is statistically real, but on a scale (10^{-3}) small relative to the full 0-to-1 range of cosine similarity.

This pooled result has an unanticipated confound: the fraction of transitions with no textual change rises, non-monotonically, from 79.5% at the first transition to a range of roughly 88–95% among later transitions, peaking at 95.0% by the eighth. If later rounds contain more trivial similarity-1 cases, that alone would make them appear more stable without genuine edits becoming more conservative. Restricting to transitions where the function's text changed that round (3,167 of 18,656, 17.0%) reduces the effect (Figure 3, blue solid line): first-transition mean similarity is 0.9900 (median 0.9923, $n = 2{,}073$) versus 0.9912 (median 0.9927, $n = 1{,}094$) for later transitions, still in the same direction but far weaker (Mann-Whitney $U = 1{,}073{,}784$, $p = 6.98 \times 10^{-3}$ vs. $p = 1.12 \times 10^{-18}$ unfiltered). Plotting the changed-only series at each transition makes the fragility of this comparison visible: the sample size falls from 2,073 real edits at the first transition to 25 by the sixth, and the confidence band widens accordingly. We stop plotting beyond that point because fewer than 25 genuinely edited transitions remain, and a per-index

estimate would be meaningless. To assess how much this difference matters in practice, we compute its rank-biserial correlation [2], a standard effect-size measure. The result is small ($r \approx 0.05$ on a -1 to 1 scale): the two groups' similarity distributions overlap almost completely despite the formal statistical significance. Therefore, most of what appears as review converging over rounds in the pooled M3 result is better explained by later rounds touching the function less often. The residual signal - that edits become marginally more conservative across later rounds - is statistically significant but practically small.

5 Discussion

M2 and M3 support a specific claim: **review overwhelmingly preserves a function's semantic purpose, and the little drift that exists concentrates in the first round.** This claim requires two qualifications. First, 75.3% of endpoint trajectories and 83.0% of consecutive transitions involve no textual change. The embedding cannot detect drift where none exists, inflating both the M2 headline and the apparent M3 convergence pattern. Second, once restricted to trajectories and transitions with a real edit, the RQ1 result holds (0.990 vs. a 0.909 null). However, the RQ2 answer changes substantially: the apparent concentration of drift in the first round is substantially accounted for by the rising rate of untouched transitions across the series, though a residual effect remains after controlling for it. What remains is a weaker, small-effect-size version of the same pattern ($r \approx 0.05$). This paper's most concrete contribution is to locate and quantify the extent to which a reported effect (RQ2) was an artifact of sampling composition.

A residual concern applies to the edited-only M2 subset: even heavily edited trajectories (500+ cumulative lines, $n = 134$) still average 0.996 similarity, far above the 0.910 null. The likely mechanism is dilution: mean pooling across all token representations produces a single 768-dimensional vector, redistributing the signal of a localized 2-line fix across hundreds of token representations that did not change, leaving the resulting vector nearly identical to its predecessor. The weak but real $\rho = -0.146$ correlation with edit volume confirms the model is sensitive to *some* of this signal—but the ceiling persists because localized change is diluted. This is the resolution problem: the instrument cannot distinguish a semantically neutral whitespace fix from a critical concurrency correction if both touch a similar number of tokens inside a large unchanged function. This structural insufficiency matches what Nikiema et al. [12] and Farhad and Dass [4] independently report in other corpora: **cosine similarity on whole-snippet embeddings is structurally the wrong resolution for spatially localized relevant change.** Whether near-ceiling similarity on genuinely edited functions reflects preserved purpose or this resolution limit remains the open question we cannot answer from whole-function embeddings alone.

6 A Research Agenda

Report the edited-only subset by default, not only the aggregate one. Prior work that scores patches from a single snapshot [8] does not distinguish a real edit from an untouched function. We show this distinction is not cosmetic: the untouched fraction rises across review rounds (79.5% at the first transition, higher at later ones), and pooling it in produces the apparent semantic convergence in M3; controlling for it reduces but does not eliminate the effect.

Validate the trajectory-linking heuristic and sharpen the instrument. A stratified annotation protocol has been prepared: a blinded annotation sheet covering 100+ trajectories sampled across five endpoint-similarity strata—including the ten highest-edit-volume cases—is ready for human review. Executing this annotation and reporting the mislinking rate against the validated method of Ramsauer et al. [15] is the immediate next step. In parallel, we plan to re-run M2 and M3 with Euclidean distance on the same UniXcoder vectors, following Nikiema et al. [12], and to embed the diff region itself [7] rather than the whole function, which should be more sensitive to a small edit diluted inside a large unchanged context.

Add a structural signal and validate by hand. Following Farhad and Dass [4], pairing the semantic score with an AST edit-distance signal would flag cases where the two disagree, such as a small textual edit with a large structural change, or vice versa. No automated metric substitutes for a kernel-literate reviewer confirming, on a sample of small edits, high-similarity cases, whether the edit is cosmetic or has an outsized behavioral effect, the only way to calibrate what a similarity threshold means for this corpus.

We also plan to extend the corpus beyond IIO to test whether the composition pattern reported here is specific to this subsystem or holds for kernel patch review generally.

7 Final Remarks

This paper is a first step toward a broader research perspective: measuring, rather than assuming, how much semantic purpose a function retains across a kernel patch series, at a resolution that matters, since most of what reviewers touch is a handful of lines embedded in an otherwise stable function. That resolution is the central problem. Tracking function-level trajectories through the Linux IIO subsystem with whole-function embeddings, we show that a naive, aggregate reading of semantic similarity is not trustworthy on its own: once similarity is measured against a proper null and disaggregated by how much a function was touched, the instrument's present resolution is not fine enough to reliably separate small but meaningful edits from noise. This limitation is not specific to our corpus or encoder. Independent work on vulnerability patches and controlled code transformations encounters the same limitation, which is why we treat it as a research opportunity rather than a solution.

Section 6 lays out the next steps for this direction: validating how trajectories are tracked across a series, embedding the edit rather than the whole function, pairing semantic and structural signals, and grounding similarity thresholds in human judgment. Our contribution is a **working measurement pipeline for this question, an account of where it currently falls short, and an initial basis for a line of research on semantic stability in patch review**.

Artifact Availability

The extraction pipeline, embedding scripts, derived metric tables, and analysis notebook used to produce the results in this paper are available at https://doi.org/10.5281/zenodo.21853709.

References

[1] Luyao Bai, Kenan Alghythee, Hang Zhang, and Xiaoguang Wang. 2026. Beyond Crash-to-Patch: Patch Evolution for Linux Kernel Repair. *arXiv preprint arXiv:2604.03851* (2026).

[2] Edward E. Cureton. 1956. Rank-biserial correlation. *Psychometrika* 21, 3 (1956), 287–290. doi:10.1007/BF02289138

[3] Kawin Ethayarajh. 2019. How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 55–65. doi:10.18653/v1/D19-1006

[4] Mohammad Farhad and Shuvalaxmi Dass. 2026. Residual Risk Analysis in Benign Code: How Far Are We? A Multi-Model Semantic and Structural Similarity Approach. arXiv preprint arXiv:2604.21051.

[5] Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. 2020. CodeBERT: A Pre-Trained Model for Programming and Natural Languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics, Online, 1536–1547. doi:10.18653/v1/2020.findings-emnlp.139

[6] Daya Guo, Shuai Lu, Nan Duan, Yanlin Wang, Ming Zhou, and Jian Yin. 2022. UniXcoder: Unified Cross-Modal Pre-training for Code Representation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Dublin, Ireland, 7212–7225. doi:10.18653/v1/2022.acl-long.499

[7] Thong Hoang, Hong Jin Kang, David Lo, and Julia Lawall. 2020. CC2Vec: Distributed Representations of Code Changes. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering (ICSE '20)*. ACM, New York, NY, USA, 518–529. doi:10.1145/3377811.3380361

[8] Thong Hoang, Julia Lawall, Richard J. Oentaryo, Yuan Tian, and David Lo. 2019. PatchNet: A Tool for Deep Patch Classification. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. IEEE, 83–86. doi:10.1109/ICSE-Companion.2019.00044

[9] Syful Islam and Stefano Zacchiroli. 2026. On the Use of Commit Messages for Corrective Software Maintenance: A Systematic Mapping Study. In *Proceedings of the International Conference on Evaluation and Assessment in Software Engineering (EASE)*. Preprint: arXiv:2604.16404.

[10] H. B. Mann and D. R. Whitney. 1947. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics* 18, 1 (1947), 50–60. doi:10.1214/aoms/1177730491

[11] Huu Hung Nguyen, Duc Manh Tran, Yiran Cheng, Thanh Le-Cong, Hong Jin Kang, Ratnadira Widyasari, Shar Lwin Khin, Ouh Eng Lieh, Ting Zhang, and David Lo. 2025. Mapping NVD Records to Their Vulnerability-Fixing Commits: How Hard Is It? *arXiv preprint arXiv:2506.09702* (2025). Preprint: arXiv:2506.09702.

[12] Serge Lionel Nikiema, Albérick Euraste Djiré, Abdoul Aziz Bonkoungou, Micheline Bénédicte Moumoula, Jordan Samhi, Abdoul Kader Kaboré, Jacques Klein, and Tegawendé F. Bissyandé. 2025. How Small Transformation Expose the Weakness of Semantic Similarity Measures. *arXiv preprint arXiv:2509.09714* (2025).

[13] Rafael Passos, Arthur Pilone, David Tadokoro, Laura Arakaki, and Paulo Meirelles. 2026. LKML5Ws: The What, When, Who, Where, and Why in the Linux Kernel Mailing Lists. In *2026 IEEE International Conference on Software Maintenance and Evolution (ICSME)*.

[14] Rafael Passos, Arthur Pilone, David Tadokoro, and Paulo Meirelles. 2025. Streamlining Analyses on the Linux Kernel with DUKS . In *2025 IEEE Working Conference on Software Visualization (VISSOFT)*. IEEE Computer Society, Los Alamitos, CA, USA, 125–128. doi:10.1109/VISSOFT67405.2025.00025

[15] Ralf Ramsauer, Daniel Lohmann, and Wolfgang Mauerer. 2019. The List is the Process: Reliable Pre-Integration Tracking of Commits on Mailing Lists. In *Proceedings of the 41st International Conference on Software Engineering (ICSE)*. IEEE, 807–818. doi:10.1109/ICSE.2019.00088

[16] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proc. EMNLP 2019*. doi:10.18653/v1/D19-1410

[17] The Linux Kernel Organization. 2024. Submitting patches: the essential guide to getting your code into the kernel. https://docs.kernel.org/process/submitting-patches.html. Accessed July 2026.