The Governance-at-Scale Paradox: Technical Debt Variance across Java Ecosystems

Mateus Stainer Rakoski
Instituto Federal do Paraná (IFPR), Campus Paranavaí
Paranavaí, Brazil
20230006272@estudantes.ifpr.edu.br

Evanise Araujo Caldas Ruiz
Instituto Federal do Paraná (IFPR), Campus Paranavaí
Paranavaí, Brazil
evanise.ruiz@ifpr.edu.br

Abstract

Growing software organizations face a tension between team autonomy and cross-project architectural consistency, which we call the governance-at-scale paradox. This study investigates whether the strength of organization-level governance is associated with the variance of code-level technical debt across publicly available Java projects. In a pre-registered observational study, 60 projects from three governance archetypes, namely Google's standardized scaffolding (17 projects), Apache's community-process governance (24), and decentralized platform governance represented by Netflix, LinkedIn, and Uber (19), were analyzed with SonarQube, using SQALE debt density as the dependent variable. The pre-registered hypothesis predicted increasing variance from Google to Apache to the decentralized archetype and was evaluated by a conjunctive rule combining a Monte-Carlo-calibrated Brown-Forsythe test with a direct check of the predicted variance ordering. The hypothesis was rejected on both the direct and logarithmic scales: Google exhibited the highest variance, and a post-hoc bootstrap indicates that the predicted ordering rarely arises under resampling of these data. Descriptively, Apache projects show systematically higher debt density, an effect that intensifies when the comparison is restricted to similarly sized projects. Under the conditions of this study, the results suggest that cohesive local technical cultures may be associated with greater cross-project consistency than centralized governance. The pre-registered protocol, amendment history, dataset, and analysis scripts are publicly available for replication.

Keywords

Technical debt, SQALE, Software governance, Pre-registration, Brown-Forsythe, Mining software repositories

1 Introduction

Every software organization that grows large enough eventually faces the same question: how can the architectural coherence of a system built by teams working in parallel be preserved without sacrificing the autonomy that makes such scale possible? We call this tension the *governance-at-scale paradox*[1]. Granting teams freedom accelerates innovation but invites gradual architectural degradation; imposing centralized control may yield cross-system consistency at the cost of stifling experimentation and adding friction [9, 24]. The consequences of these choices accumulate in the code over time, often unnoticed. This drift away from the intended architecture, which Perry and Wolf [20] describe as architectural erosion and drift, constitutes *architectural degradation*, whose consequences for maintainability and evolution are well documented [15, 16].

[1]"Paradox" is used in the sense of Solow's "productivity paradox": a persistent tension between desirable but conflicting goals, not a logical contradiction.

The investigation is grounded in the mirroring hypothesis of MacCormack, Baldwin, and Rusnak [14], who demonstrated empirically that organizational structure is reflected in the modularity of the software produced. If governance models differ systematically, their signatures should be measurable in the code. Our conjecture is that stronger organizational constraints are associated with more stable and predictable levels of technical debt at scale, particularly when compared to ecosystems where enforcement mechanisms are absent or merely advisory [16]. This leads to the central research question:

> *Does the strength of organization-level architectural governance correlate with the variance of code-level technical debt across publicly available Java projects?*

Two hypotheses were fixed before data collection:

- **H0:** There is no statistically significant difference in code-level technical debt density, nor in its variability, across the three governance archetypes.
- **H1:** The variance of code-level debt density follows the pre-registered ordering $\sigma^2_{\text{Google}} < \sigma^2_{\text{Apache}} < \sigma^2_{\text{Decentralized}}$, mirroring the ordering of organizational governance from centralized scaffolding, to community process governance, to decentralized platform models. This ordering was fixed a priori from the conceptual centralization of each archetype (Section 3) and is not revisable post hoc.

The contribution is threefold: (i) an empirical characterization of code-level technical debt across three governance archetypes, showing that the pre-registered variance ordering does not hold in this sample and that Apache projects carry systematically higher debt density, a descriptive effect robust to project size; (ii) the operationalization of governance as an ecosystem-level independent variable, with a substantive hypothesis about *variance* rather than central tendency; and (iii) a fully pre-registered and auditable analytical protocol, a practice that mitigates hypothesizing after the results are known (HARKing [11]) and follows the pre-registration movement in empirical science [19]. Particular attention is given to the statistical machinery: the choice of the Brown-Forsythe test [2] for a variance question, and the Monte Carlo calibration of its critical value.

2 Background and Related Work

2.1 Architectural Degradation and Technical Debt

Perry and Wolf [20] characterize architectural *erosion* and *drift* as phenomena operating against architectural stability over time. Martini et al. [16] confirm empirically that these phenomena are

constitutive of the evolution of production systems, and MacCormack and Sturtevant [15] show that inter-component coupling patterns constitute a measurable proxy of architectural debt. Technical debt [4], the accumulated cost of expedient implementation decisions, is commonly split into code debt and architectural debt [16]. The SQALE method [13] operationalizes code debt as remediation cost: the SQALE Quality Index (SQI) is the sum, in minutes of work, of the remediation costs of all rule violations, and SQALE density normalizes SQI by system size.

2.2 Mirroring and Governance Archetypes

The mirroring hypothesis holds that the technical structure of an artifact tends to mirror the social structure of the organization producing it; MacCormack et al. [14] provide an empirical test on software, showing that loosely coupled organizations produce more modular products. Three governance archetypes have consolidated in large-scale Java ecosystems. **Standardized scaffolding** (Google) [24] combines centralized mechanisms (a monolithic repository [21], mandatory ownership-mediated review, binding style guides, standardized internal tooling) with top-down architectural decisions. **The Apache Way** [7] adopts a deliberately vendor-neutral model in which coordination operates through process (public mailing lists, Project Management Committees, meritocracy) [17]: centralization by process, not authority. **Decentralized platform governance**, exemplified by Netflix [25], is characterized by *paved roads* and golden paths: a central team defines recommended but non-binding practices, preserving team autonomy through philosophies such as *freedom and responsibility* [9] and the full-cycle developer [18].

2.3 Positioning

Adjacent lines of research associate organizational and community characteristics with technical debt and quality outcomes. Community smells have been linked to self-admitted technical debt in machine learning projects [5]; repository-level project characteristics have been explored as correlates of debt [6]; and pull-request governance has been studied as a quality-control mechanism in open-source communities [1]. These studies examine within-project or community-level factors. To the extent verifiable in the literature, no prior study has quantitatively compared *organizational governance archetypes* as an ecosystem-level independent variable over code-level debt metrics, with a pre-registered ordered hypothesis on *variance* and explicit confounder treatment. We measure debt at code level for reproducibility, at the cost of capturing a code-level *signature* rather than architectural consistency directly; architectural analyses via tools such as Arcan [8] are future work.

3 Methodology

3.1 Study Design and Pre-Registration

The study is an observational comparison across the three archetypes. The independent variable is the organizational archetype of the project; the dependent variable is code-level SQALE debt density, measured at a single snapshot and aggregated per project. There is no experimental intervention: publicly available Java projects under each governance model are compared as found in each ecosystem.

The protocol was frozen before collection: the ordered hypothesis H1, the primary metric, the decision rule, and the analysis scripts were fixed and committed to the repository a priori, and every subsequent methodological decision was recorded as a dated, pre-declared amendment (Git tags in the repository [22]) created *before* the data it governs was collected, preserving an auditable ordering between each decision and the observations it precedes. An initial wave of 34 projects motivated a pre-declared sample expansion after a power analysis indicated inadequate power ($\approx 9\%$) to detect the expected effect magnitude under the pre-registered rule; the expansion was declared before any new project was collected. The complete protocol, amendment chronology, collection pipeline, analysis scripts, and consolidated dataset are publicly available [22]: every table and figure in this paper is regenerated deterministically from the published CSV.

3.2 Inclusion Criteria and Archetype Classification

The sample is restricted to publicly available Java projects on GitHub satisfying criteria established before collection: at least 70% of the code recognized as Java by GitHub; at least 10,000 non-commented lines of code (NCLOC); at least 25 unique contributors; a snapshot at most 24 months old at collection time; and a feasible local build with the available infrastructure (JDK 8–21; Maven, Gradle, Ant, Bazel; Linux). The NCLOC and contributor thresholds ensure each project is a system of relevant size developed by an organizational structure rather than an isolated developer; the snapshot-age criterion prevents a confound between archetype and snapshot staleness, since maintenance-mode decentralized projects tend to have older snapshots than Apache or Google ones.

A total of 83 candidate projects were enumerated from the public repositories of the named organizations and screened against the criteria above; 23 were excluded, primarily for failing the inclusion criteria (including five with documented build incompatibilities, e.g., a macOS-only build requirement, whose reintroduction is pre-registered for an extended version), yielding the final $N = 60$. The organizations themselves were fixed a priori as the most-cited exemplars of each archetype in the practitioner and research literature (Section 2); projects from other organizations were not enumerated, a scope decision revisited in Section 4.6.

Archetype classification is determined by the repository's owning organization: `apache/` projects are Apache; `google/` and `GoogleCloudPlatform/` are Google; `Netflix/`, `linkedin/`, and `uber/` are decentralized. This operationalization captures governance as a property of the ecosystem, under the premise that each organization applies its characteristic governance model to published projects. We note explicitly that this premise was not validated at the project level (e.g., by inspecting contribution rules, review requirements, or decision processes per repository), so the operationalization conflates organizational identity with governance archetype; this is revisited in Section 4.6.

The final sample comprises $N = 60$ projects: 24 Apache, 17 Google, 19 decentralized, collected in two waves (34 on 2026-05-17 under a stable pre-2026 release-tag criterion; 26 on 2026-05-24 under a HEAD-of-main criterion, a relaxation pre-declared before the second wave). Two compositional asymmetries are declared up

front. First, 11 of the 19 decentralized projects are Netflix's, so effects attributed to the archetype may partly reflect Netflix's specific technical culture. Second, 7 of the 24 Apache projects originate from Chinese companies (Alibaba, Huawei, among others) later donated or incubated at the ASF, a source of internal heterogeneity. Apache projects are also, on average, older and larger, reflecting the population of active public Java in each ecosystem; restricting Apache to small, young projects would no longer represent Apache. These distributions are reported before any inferential analysis, and robustness analyses control for size (Section 3.6).

3.3 Collection Pipeline

Collection was executed with SonarQube Community Build 26.2.0.119303 (MQR mode), Sonar Scanner 5.0.1.3006, and the default *Sonar way* Quality Profile with no customization, a pre-registered decision to avoid rule-selection bias. The pipeline runs five steps per project: (i) cloning the target commit identified by SHA in a master spreadsheet; (ii) build detection (Maven, Gradle, Ant, Bazel) and compilation; (iii) Sonar Scanner execution; (iv) metric extraction via the SonarQube API; and (v) consolidation into a CSV. JDK selection follows a $21 \rightarrow 17 \rightarrow 11 \rightarrow 8$ cascade upon compatibility failures. Three projects required build patches; the patches are restricted to build configuration and dependency resolution, modify no analyzed source code, and are individually documented in the repository [22].

3.4 Primary Metric and Decision Rule

The primary metric is SQALE density, defined as

$$\text{density} = \frac{\text{sqale\_index}}{\text{NCLOC}}, \tag{1}$$

i.e., the average remediation cost, in minutes, per non-commented line of code. Normalizing by system size allows comparison across projects of drastically different magnitudes, since the absolute index would contaminate any cross-archetype comparison; NCLOC is adopted as the size measure, rather than total lines, because it excludes comments and blank lines and thus counts only the code subject to quality rules.

H1 predicts an ordering of the *variance* of density across projects, not of its central tendency. The choice is theoretically motivated: the governance paradox postulates that organizational enforcement mechanisms reduce the *dispersion of architectural choices across projects of the same ecosystem*, not necessarily the ecosystem's absolute debt level. A centrally governed archetype may carry much or little debt; H1 only predicts its projects will be more *consistent* with one another (lower variance) than projects from archetypes with more permissive governance.

The pre-registered decision rule, denoted $C_1 \wedge C_2$, requires two conditions to hold simultaneously for rejecting H0:

- C_1 **(significance):** the Brown-Forsythe statistic F computed on the sample exceeds a critical value F_{crit} calibrated by Monte Carlo (Section 3.5).
- C_2 **(ordering):** the observed sample variances satisfy the predicted order $s^2_{\text{Google}} < s^2_{\text{Apache}} < s^2_{\text{Decentralized}}$.

The interpretation of all four scenarios was declared a priori: $C_1 \wedge C_2$ rejects H0 in favor of H1; $C_1 \wedge \neg C_2$ constitutes evidence *against* H1 and suggests an alternative mechanism to be discussed; $\neg C_1$ is a failure to detect a difference and, under the estimated power of $\approx 9\%$, does not constitute evidence of equivalence. Although C_2 involves no significance test, the sample variances it compares remain subject to sampling variability; the uncertainty of its verdict is therefore quantified post hoc by bootstrap (Section 4.2).

3.5 Test Selection and Monte Carlo Calibration

Testing H1 requires a test of *homogeneity of variances* across $k = 3$ groups. The classical candidates are ill-suited to these data: Bartlett's test and the variance-ratio F-test assume normality within groups, and under skewed or heavy-tailed distributions their Type I error rate inflates far beyond the nominal α, manufacturing spurious "significant" variance differences. SQALE densities are exactly that kind of data: bounded below by zero, right-skewed, and approximately lognormal, as the two panels of Figure 1 show, with observations piling low under long upper tails on the direct scale and approximately symmetric distributions after the log transform. The Brown-Forsythe test [2] is therefore adopted: the statistic is the one-way ANOVA F computed on the absolute deviations of each observation from its group *median*, a centering that is insensitive to extreme values, so a single high-debt outlier project cannot inflate the statistic and the test maintains its nominal Type I error rate under skewed, heavy-tailed distributions while retaining good power. Under H0 the statistic approximately follows an $F(k-1, N-k)$ distribution.

That approximation, however, is asymptotic. With small, unbalanced groups ($n = 17, 24, 19$) drawn from lognormal populations, the true null distribution of the Brown-Forsythe statistic deviates from the theoretical $F(2, 57)$, so using the textbook critical value would silently miscalibrate the test's error rate. We therefore calibrated F_{crit} empirically by Monte Carlo: 10,000 replicates were simulated under H0 (three lognormal groups with the observed group sizes and equal variances), the Brown-Forsythe statistic was computed on each replicate, and F_{crit} was taken as the 95th percentile of the resulting null distribution. This procedure fixes the Type I error at 5% under the study's actual data-generating regime rather than under an idealized one, and it was frozen in the analysis scripts before collection. The calibrated values are $F_{\text{crit}} = 2.944$ on the direct scale and $F_{\text{crit}} = 2.811$ on the log scale. The same simulation machinery yields the design's power estimate: replicates were generated under H1 (lognormal groups at the final sizes, with variances following the predicted ordering at a ratio judged plausible a priori and specified in the pre-registered protocol [22]), and the full conjunctive rule $C_1 \wedge C_2$ was satisfied in only $\approx 9\%$ of them on the direct scale. This is the power figure cited throughout: the probability that the design would reject H0 if H1 were true at that effect magnitude.

A complementary analysis replicates Brown-Forsythe on log(density). The rationale is twofold: (i) the logarithm of a lognormal variable is approximately normal (Figure 1, right panel), placing the test in the regime where it is best calibrated; and (ii) Monte Carlo simulations indicated a substantial power gain for plausible effect sizes. Variance on the log scale measures *relative* (multiplicative) dispersion, whereas variance on the direct scale

measures absolute dispersion; both are legitimate readings of "consistency across projects." The rule $C_1 \wedge C_2$ is applied unchanged, and the log-scale results are reported as complementary evidence with its provenance declared in the protocol amendments, not as part of the frozen confirmatory plan.

3.6 Secondary Analyses and Confounder Treatment

Four pre-declared complementary analyses accompany the primary test, all non-parametric given the lognormal distribution. **Cliff's** δ [3], a rank-based effect size robust to non-normality, is reported for the three archetype pairs, with magnitudes per Romano et al. [23] ($|\delta| < 0.147$ negligible, < 0.33 small, < 0.474 medium, otherwise large). **Kruskal-Wallis** [12] tests the secondary hypothesis on central tendency (heterogeneity of medians, η^2 as effect size), protecting against the critique that variance-focused analyses omit the absolute debt level. **Jonckheere-Terpstra** [10] is an exploratory test of the a priori ordering of medians (Google ≤ Apache ≤ Decentralized), the only classical test sensitive to a pre-specified order across multiple groups.

Confounders are treated in three declared steps. First, the sample composition in NCLOC, project age, and contributors is reported before any inference. Second, a partial Spearman correlation between archetype (ordinally coded) and density is computed controlling for log(NCLOC), project age, and snapshot age, as a descriptive analysis. Third, a robustness analysis restricts the sample to the NCLOC range where all three archetypes overlap (10,000–100,000) and verifies whether the direction of effects persists. Additionally, per-organization dispersion within the decentralized archetype (Netflix, LinkedIn, Uber; Table 1, bottom rows) quantifies internal cohesion, supporting the question of whether observed consistency stems from strong local technical culture rather than the archetype itself. A decomposition of SQI by rule category (maintainability, reliability, security) is available as supplementary material [22].

4 Results and Discussion

4.1 Sample Composition and Descriptive Statistics

Table 1 and Figure 1 report the density distribution per archetype and, within the decentralized archetype, per company. Apache has the highest median debt density (0.465), followed by Decentralized (0.341) and Google (0.232). Apache and Decentralized show comparable sample variances (≈ 0.05), both substantially lower than Google's (0.084); the pattern of Google combining the lowest median with the widest spread is directly visible in the boxplots on both scales. The sample spans projects of diverse natures in all archetypes; nevertheless, whether Google's higher variance reflects its governance model or a greater diversity of project types within its public portfolio remains an open alternative explanation.

Within the decentralized archetype (Table 1, bottom rows), each company exhibits high internal cohesion (variances 0.011–0.043), with Netflix's median close to Apache's (0.44 vs. 0.47). What appears as "decentralized archetype variance" (0.048) may therefore reflect differences *between* the three companies rather than dispersion within any of them: the apparent cohesion of the archetype

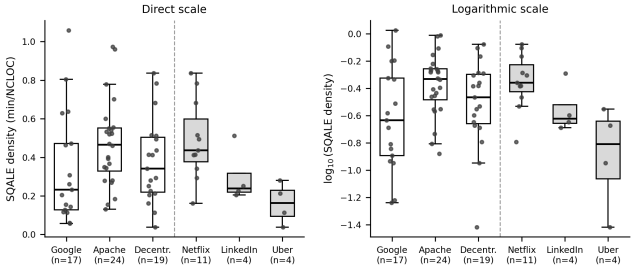

Figure 1: SQALE density per governance archetype on the direct (left) and logarithmic (right) scales, with individual projects overlaid; the three rightmost boxes split the decentralized archetype by company. Google combines the lowest median with the widest spread; Netflix sits close to Apache, while LinkedIn and Uber are low and internally tight.

Table 1: SQALE density by archetype ($N = 60$); indented rows split the decentralized archetype by organization.

Group	n	Median	Q1	Q3	Variance
Google	17	0.232	0.128	0.472	0.084
Apache	24	0.465	0.329	0.551	0.050
Decentralized	19	0.341	0.219	0.503	0.048
Netflix	11	0.436	0.377	0.599	0.043
LinkedIn	4	0.238	0.220	0.317	0.021
Uber	4	0.163	0.094	0.229	0.011

would stem not from it being decentralized, but from each company having its own cohesive internal technical culture. LinkedIn's and Uber's sample sizes ($n = 4$ each) limit inference on the individual subgroups.

4.2 Primary Analysis: The Conjunctive Rule

Table 2 applies $C_1 \wedge C_2$ on both scales. On the direct scale, both conditions fail: $F = 0.42 < F_{\text{crit}} = 2.94$ (scenario 4 of the pre-declared interpretation). On the log scale, C_1 is satisfied ($F = 2.93 > F_{\text{crit}} = 2.81$) but C_2 fails: the sample ordering is $s^2_{\text{Apache}} < s^2_{\text{Desc}} < s^2_{\text{Google}}$, placing Google, the most centrally governed archetype, at the *top* rather than the bottom (scenario 2). Both scales converge on rejecting H1.

The failure of C_2 on both scales warrants emphasis, with an important qualification. C_2 involves no significance test, but the sample variances it compares are themselves estimates from small, unbalanced, lognormal groups and are therefore subject to sampling variability. To quantify that uncertainty, a post-hoc bootstrap (10,000 resamples, resampling projects within each archetype) computed how often the predicted ordering $s^2_{\text{Google}} < s^2_{\text{Apache}} < s^2_{\text{Decentralized}}$ arises: it appears in only 4.6% of replicates on direct density and 0.4% on log density. The failure of C_2 is thus unlikely to be an artifact of sampling variability alone, although the point estimates of the variances remain uncertain at these group sizes. Although the observed ordering contradicts H1, it is consistent with a

Table 2: Application of the $C_1 \wedge C_2$ rule to SQALE density on the direct and logarithmic scales.

Statistic	Direct	Logarithmic
F	0.419	2.929
F_{crit} (Monte Carlo, 10k replicates)	2.944	2.811
s^2_{Apache}	0.050	0.269
s^2_{Google}	0.084	0.776
$s^2_{\text{Decentralized}}$	0.048	0.544
Sample order	$s_d^2 < s_a^2 < s_g^2$	$s_a^2 < s_d^2 < s_g^2$
C_1 (significance)	False	True
C_2 (predicted order)	False	False
H1	**Rejected**	**Rejected**

Table 3: Pairwise Cliff's δ; magnitudes per Romano et al. [23]. Right column: subsample restricted to 10k–100k NCLOC (n = 35).

Group X	Group Y	δ (full)	δ (10k–100k)
Apache	Google	+0.373 (medium)	+0.833 (large)
Apache	Decentralized	+0.268 (small)	+0.804 (large)
Google	Decentralized	−0.183 (small)	−0.333 (medium)

post-hoc narrative visible in Figure 1: locally cohesive technical cultures, manifest both in Apache's community governance and in the per-company cohesion of the decentralized archetype, may produce greater cross-project consistency than top-down governance.

4.3 Effect Sizes and Central Tendency

Cliff's δ (Table 3) confirms the descriptive ordering of medians: Apache exceeds Google with a medium effect (+0.373) and the decentralized archetype with a small one (+0.268). Kruskal-Wallis yields $p = 0.087$ with $\eta^2 = 0.051$ (small), a borderline result that does not reject at $\alpha = 0.05$, and Jonckheere-Terpstra on the pre-registered ordering yields $p = 0.272$ ($z = 0.606$), i.e., no support for the predicted order in central tendency either. The pre-declared partial Spearman correlation between archetype (ordinally coded by conceptual centralization) and density, controlling for log(NCLOC), project age, and snapshot age, is $r = 0.234$ ($p = 0.205$): no adjusted monotonic association between archetype and debt density is detected.

4.4 Robustness to Project Size

Restricting the analysis to the NCLOC range where the three archetypes overlap (10,000–100,000; $n = 35$) tests whether the descriptive effect "Apache > the others" is an artifact of Apache projects being roughly an order of magnitude larger. If size were the driver, the effect should attenuate among similarly sized projects. The opposite occurs (Table 3, right column): Apache vs. Google grows from +0.373 to +0.833 and Apache vs. Decentralized from +0.268 to +0.804, both large. The intensification contradicts the size explanation and highlights the elevated debt density that Apache carries, particularly in its smaller projects.

4.5 Implications for Research and Practice

For research, the results caution against equating governance archetype with organizational identity: the per-company cohesion inside the decentralized archetype suggests that *local technical culture*, not the formal governance model, may be the operative unit of consistency, and future mirroring-style studies should sample enough independent organizations per archetype to separate the two. For practitioners, two readings are warranted. First, centralized scaffolding did not deliver the expected cross-project consistency in this sample, so mandating tooling and review alone should not be assumed to homogenize debt outcomes. Second, all density readings depend on the adopted Quality Profile (default *Sonar way*): ecosystems whose conventions align less with this ruler are penalized in density, so Apache's elevated density does not by itself indicate worse engineering, and practitioners comparing their own portfolios against public baselines should calibrate the profile before drawing conclusions; the SQI decomposition by rule category in the supplementary material [22] supports inspecting this hypothesis, and a definitive diagnosis would require replication under alternative profiles.

4.6 Limitations

The statistical power to detect plausible effect sizes under $C_1 \wedge C_2$ is estimated at $\approx 9\%$ on the direct-density scale: even if H1 were true, the primary test would reject H0 in only about 9% of samples of this size. This caveat applies with full force to C_1; for C_2, the post-hoc bootstrap (Section 4.2) indicates that the predicted ordering rarely arises under resampling, but variance estimates from groups of 17–24 projects remain uncertain, and the verdict of C_2 should be read as strong sample evidence rather than a definitive population claim. Governance is operationalized by repository ownership without project-level validation, so archetype effects cannot be separated from organizational identity; in particular, Netflix's dominance of the decentralized archetype (11 of 19 projects) means effects attributed to the archetype may partly reflect Netflix's specific technical culture. Each project is measured at a single snapshot, so projects may be compared at different stages of development, stabilization, or maintenance; the snapshot-age criterion and the reported controls mitigate but do not eliminate this, and only a longitudinal design would establish temporal comparability. The two collection waves also used different snapshot criteria: snapshot type is balanced across archetypes (41–47% head-of-main per archetype), but density differs by wave ($p = 0.016$ overall; $p = 0.018$ within Apache), contributing a nuisance component to within-archetype dispersion. Finally, measurement operates at code level, not architectural level: SQALE density captures a code-level *signature* of organizational choices, and H1, formulated in terms of the governance paradox, could hold at the architectural layer even while false at code level.

4.7 Future Work

The most immediate direction is reformulating the analysis at the architectural layer: applying Arcan [8] to the same 60-project sample would test H1 where the paradox is actually formulated, in module dependency structures rather than rule-violation density. Expanding the sample, both in projects and in independent organizations

per archetype, would raise power to adequate levels and allow separating governance effects from company-specific cultures, and the pre-registered protocol [22] supports replication, including the five excluded projects. Project-level validation of governance practices (contribution rules, review requirements, decision processes) would strengthen the operationalization, complemented by qualitative work (e.g., interviews on review and decision practices) to capture mechanisms unobservable in code.

5 Conclusion

This study asked whether the strength of organization-level architectural governance correlates with the variance of code-level technical debt across publicly available Java projects. Under a pre-registered design with the conjunctive rule $C_1 \wedge C_2$ fixed a priori over the sample variances of SQALE density, H1 was rejected on both analysis scales: the ordering $\sigma^2_{\text{Google}} < \sigma^2_{\text{Apache}} < \sigma^2_{\text{Decentralized}}$ was not observed under the conditions of this study, namely the sampled public Java projects, governance operationalized by repository ownership, SQALE density under the default *Sonar way* profile, and single-snapshot measurements. A post-hoc bootstrap indicates that this failure is unlikely to be attributable to sampling variability alone. Beyond the rejection, a descriptive finding emerged: Apache exhibits systematically higher SQALE density than Google and the decentralized archetype, and Apache and Decentralized show comparably low variances, both well below Google's, motivating the post-hoc hypothesis that cohesive local technical cultures may produce greater consistency than centralized governance.

The main methodological contribution is the fully auditable pre-registration workflow: hypothesis, metric, and decision rule frozen before collection; amendments recorded as dated tags; and a Brown-Forsythe test whose critical value is Monte-Carlo-calibrated to the study's actual data regime. The consolidated dataset and analysis scripts are publicly available [22].

Artifact Availability

The collection protocol, amendment chronology (with a human-readable summary of each dated amendment in addition to the Git tags), collection pipeline, analysis scripts, build patches for the three affected projects, and consolidated dataset are permanently archived at https://doi.org/10.5281/zenodo.20597946, with development history at https://github.com/Rakoski/scripts-tcc. Data are released under CC-BY 4.0 and scripts under the MIT license. Every table and figure in this paper is regenerated deterministically from the published CSV.

Acknowledgements

In preparing this work, the authors used generative AI assistants (Claude and Gemini) for translating, adapting, revising, and editing parts of the text and for English-language refinement, and Claude Code for writing parts of the analysis scripts. The authors reviewed and verified all such content and take full responsibility for the research design, data, analysis, and conclusions.

References

[1] Adam Alami, Raúl Pardo, Marisa Leavitt Cohn, and Andrzej Wąsowski. 2021. Pull Request Governance in Open Source Communities. *IEEE Transactions on Software Engineering* 48, 12 (2021), 4838–4856. doi:10.1109/TSE.2021.3128356

[2] Morton B. Brown and Alan B. Forsythe. 1974. Robust Tests for the Equality of Variances. *J. Amer. Statist. Assoc.* 69, 346 (1974), 364–367. doi:10.1080/01621459.1974.10482955

[3] Norman Cliff. 1993. Dominance Statistics: Ordinal Analyses to Answer Ordinal Questions. *Psychological Bulletin* 114, 3 (1993), 494–509. doi:10.1037/0033-2909.114.3.494

[4] Ward Cunningham. 1992. The WyCash Portfolio Management System. *ACM SIGPLAN OOPS Messenger* 4, 2 (1992). doi:10.1145/157710.157715

[5] Shamse Tasnim Cynthia, Nuri Almarimi, and Banani Roy. 2025. How Do Community Smells Influence Self-Admitted Technical Debt in Machine Learning Projects?. In *2025 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE.

[6] Jandisson S. de Jesus and Ana C. V. de Melo. 2017. Technical Debt and the Software Project Characteristics: A Repository-Based Exploratory Analysis. In *2017 IEEE 19th Conference on Business Informatics (CBI)*, Vol. 1. IEEE. doi:10.1109/CBI.2017.24

[7] Isabel Drost-Fromm and Roman Tompkins. 2021. Open Source Community Governance the Apache Way. *Computer* 54, 4 (2021), 70–75. doi:10.1109/MC.2021.3055137

[8] Francesca Arcelli Fontana, Ilaria Pigazzini, Riccardo Roveda, Damian Tamburri, Marco Zanoni, and Elisabetta Di Nitto. 2017. Arcan: A Tool for Architectural Smells Detection. In *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*. 282–285. doi:10.1109/ICSAW.2017.16

[9] Reed Hastings and Erin Meyer. 2020. *No Rules Rules: Netflix and the Culture of Reinvention*. Penguin Press, New York, NY, USA.

[10] A. R. Jonckheere. 1954. A Distribution-Free k-Sample Test against Ordered Alternatives. *Biometrika* 41, 1/2 (1954), 133–145. doi:10.2307/2333011

[11] Norbert L. Kerr. 1998. HARKing: Hypothesizing After the Results are Known. *Personality and Social Psychology Review* 2, 3 (1998). doi:10.1207/s15327957pspr0203_4

[12] William H. Kruskal and W. Allen Wallis. 1952. Use of Ranks in One-Criterion Variance Analysis. *J. Amer. Statist. Assoc.* 47, 260 (1952), 583–621. doi:10.1080/01621459.1952.10483441

[13] Jean-Louis Letouzey. 2012. The SQALE Method for Evaluating Technical Debt. In *Proceedings of the Third International Workshop on Managing Technical Debt (MTD)*. 31–36. doi:10.1109/MTD.2012.6225997

[14] Alan MacCormack, Carliss Baldwin, and John Rusnak. 2012. Exploring the Duality between Product and Organizational Architectures: A Test of the "Mirroring" Hypothesis. *Research Policy* 41, 8 (2012), 1309–1324. doi:10.1016/j.respol.2012.04.011

[15] Alan MacCormack and Daniel J. Sturtevant. 2016. Technical Debt and System Architecture: The Impact of Coupling on Defect-Related Activity. *Journal of Systems and Software* 120 (2016), 170–182. doi:10.1016/j.jss.2016.06.007

[16] Antonio Martini, Jan Bosch, and Michel Chaudron. 2015. Investigating Architectural Technical Debt Accumulation and Refactoring over Time: A Multiple-Case Study. *Information and Software Technology* 67 (2015), 237–253. doi:10.1016/j.infsof.2015.07.005

[17] Audris Mockus, Roy T. Fielding, and James D. Herbsleb. 2002. Two Case Studies of Open Source Software Development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology* 11, 3 (2002), 309–346. doi:10.1145/567793.567795

[18] Netflix Technology Blog. 2018. Full Cycle Developers at Netflix: Operate What You Build. Netflix Tech Blog. Available at: https://netflixtechblog.com/full-cycle-developers-at-netflix-a08c31f83249.

[19] Brian A. Nosek, Charles R. Ebersole, Alexander C. DeHaven, and David T. Mellor. 2018. The Preregistration Revolution. *Proceedings of the National Academy of Sciences* 115, 11 (2018), 2600–2606. doi:10.1073/pnas.1708274114

[20] Dewayne E. Perry and Alexander L. Wolf. 1992. Foundations for the Study of Software Architecture. *ACM SIGSOFT Software Engineering Notes* 17, 4 (1992), 40–52. doi:10.1145/141874.141884

[21] Rachel Potvin and Josh Levenberg. 2016. Why Google Stores Billions of Lines of Code in a Single Repository. *Commun. ACM* 59, 7 (2016), 78–87. doi:10.1145/2854146

[22] Mateus Stainer Rakoski and Evanise Araujo Caldas Ruiz. 2026. Replication Package: The Governance-at-Scale Paradox (Protocol, Amendment Chronology, Collection Pipeline, Analysis Scripts, and Dataset). doi:10.5281/zenodo.20597946

[23] Jeanine Romano, Jeffrey D. Kromrey, Jesse Coraggio, and Jeff Skowronek. 2006. Appropriate Statistics for Ordinal Level Data: Should We Really Be Using t-test and Cohen's d for Evaluating Group Differences on the NSSE and Other Surveys? *Annual Meeting of the Florida Association of Institutional Research* (2006), 1–33.

[24] Titus Winters, Tom Manshreck, and Hyrum Wright. 2020. *Software Engineering at Google: Lessons Learned from Programming over Time*. O'Reilly Media.

[25] Yunong Xiao. 2017. The Paved Road PaaS for Microservices at Netflix. In *QCon New York*. Available at: https://www.infoq.com/news/2017/06/paved-paas-netflix/.