Beyond CVEs: A Practical Approach to Interpreting Structural OSS Risk in the AI Era

Irlan de Alvarenga Cidade
Independent Researcher, Creator of Biguá Analyzer
March 23, 2026

Abstract

Application security teams increasingly depend on open source software, but most dependency evaluation still focuses on vulnerabilities, advisories, or static posture indicators. At the same time, AI-assisted development is making repository activity look more regular, more uniform, and sometimes more mature than it may actually be. This creates a practical problem: which repository signals still help AppSec teams judge structural risk?

This paper presents the Biguá Analyzer, an original open-source framework developed by the author to assess OSS repositories through structural indicators such as contribution concentration, contributor redundancy, turnover, activity dynamics, and inferred AI influence. Using a curated set of widely used repositories, the analysis compares fast recent-window screening with full-history structural analysis in repositories showing unusually large deviations.

The results suggest that AI-related patterns are now widespread, but they do not replace the classical structural indicators that matter most for resilience. Contribution concentration, redundancy, and long-tail participation remain stronger indicators of fragility or robustness than AI-adjacent signals alone. The study also shows that short-window analysis can overestimate AI influence and obscure historical structural dependency in mature projects. For AppSec teams, the practical message is to use recent-window screening for prioritization, but to escalate high-impact or high-deviation dependencies to deeper structural review before making trust decisions.

1. Introduction

Open source software underpins critical parts of the modern software supply chain. For application security teams, that makes repository health relevant not only to engineering quality, but also to dependency governance, resilience, and third-party risk assessment.

At the same time, AI-assisted development tools have introduced a new interpretive challenge. Code assistants, generative systems, and AI-aware workflows can make recent repository activity look cleaner, faster, and more uniform, which complicates how AppSec teams interpret surface-level signals when evaluating open source dependencies.

Much of the current discussion focuses on whether AI is present. This paper argues that the more useful question for AppSec programs is whether AI changes the underlying structural properties of OSS projects, or whether it mainly changes how recent activity appears from the outside.

Accordingly, we ask:

> **Has the transition from pre-AI to AI-assisted development changed the structural signals that define OSS project health and risk?**

To investigate this question, we developed the Biguá Analyzer, a repository analysis framework that measures structural indicators such as contribution inequality, bus factor, turnover, contributor base breadth, activity cadence, and inferred AI influence. We then used these metrics to compare repositories across different structural profiles and to test whether AI-related signals explain structural variation better than classic sociotechnical metrics.

Our findings suggest that the strongest structural distinctions remain better explained by classical sociotechnical metrics than by AI-related signals alone. AI appears widespread, but its main effect in this dataset is to change the interpretability of some observable signals rather than the structural foundations of project resilience.

Why does this matter to AppSec teams?

Because dependency review often overweights what is easy to see. Vulnerabilities, advisories, and posture checks matter, but they do not fully capture maintainer concentration, continuity risk, or historically narrow participation patterns that can shape how a project responds under stress.

That means a repository can look healthy in a recent window while still carrying structural dependency risk. Strong posture or a favorable snapshot does not automatically imply contributor redundancy, sustainable maintenance, or robust historical distribution of responsibility.

The practical value of this approach is to add a structural layer to OSS security review. It helps teams distinguish between repositories that merely look healthy and repositories that are structurally resilient across time.

The suggested practice is a tiered model: use fast analysis for prioritization, but require full validation for high-impact or high-deviation decisions such as critical dependencies, onboarding reviews, or sensitive supply chain choices.

Contributions

This paper makes four contributions:

1. Introduces Biguá Analyzer as a structural OSS assessment framework;
2. Proposes a reproducible AI-influence inference layer;
3. Distinguishes recent-window screening from full-history structural characterization;
4. Argues that AI changes signal interpretability more than underlying OSS structure.

2. Background and Motivation

Open source ecosystem health has long been studied through signals such as contributor activity, release frequency, project longevity, participation volume, and maintainer continuity. Prior work on OSS health emphasizes that ecosystem robustness depends not only on visible output, but also on how responsibility, expertise, and participation are distributed across the project. Research on truck factor/bus factor similarly frames project resilience as a function of knowledge concentration rather than simple activity volume.

The arrival of AI-assisted development complicates this landscape. AI tooling may increase output speed, smooth stylistic variation, standardize commit artifacts, and change contribution patterns in ways that are visible in repository data but not directly attributable from authorship alone. Recent empirical work on AI coding assistants shows measurable effects on developer workflows and code production, but it does not establish that AI changes the deeper structural properties of projects themselves.

This creates two research gaps.

First, many analyses remain **code-centric** instead of **structure-centric**. They ask whether code is AI-generated, not whether ecosystems become more or less resilient.

Second, even when AI influence is discussed, it is often treated as a direct risk factor. This may obscure the possibility that AI is now an environmental baseline rather than a meaningful discriminator of project health.

The motivation for this study is therefore to test a stronger hypothesis:

> **The AI era changes the interpretability of some repository signals, but not the structural foundations of OSS resilience.**

3. Methodology

3.1 Repository Set

We analyzed a curated set of widely adopted OSS repositories spanning web frameworks, infrastructure tools, programming languages, developer tooling, and other high-impact open source projects. The repository set was intentionally heterogeneous in age, size, contributor count, and release cadence in order to observe whether structural patterns remain stable across different project types.

Criteria used

We have analyzed 42 repositories in the study. They should satisfy the following conditions:

- Language: Go, JS, JAVA, Python and Rust
- ≥ 1000 stars
- ≥ 50 contributors
- ≥ 3 years of history

GitHub Repository Data

Our primary source of ecosystem signals is github repository data. We collected via GitHub API.

Examples of extracted data:

- commits

- contributors
- pull requests
- issues
- releases
- repository metadata

Example API endpoints:

/repos/{owner}/{repo}
/repos/{owner}/{repo}/commits
/repos/{owner}/{repo}/contributors
/repos/{owner}/{repo}/issues
/repos/{owner}/{repo}/releases

3.2 Structural Metrics

The Biguá Analyzer computes a set of sociotechnical metrics intended to capture project structure rather than code semantics alone.

Contribution concentration

- **Gini coefficient**: measures inequality in contribution distribution across contributors.
- **Top contributor share**: proportion of commits attributable to the single most active contributor.

Contributor redundancy

- **Bus factor (50th percentile)**: minimum number of contributors needed to account for half of analyzed contribution activity.
- **Bus factor (75th percentile)**: minimum number needed to account for three quarters of analyzed contribution activity.

Activity and maintenance

- **Release cadence** and **recent release cadence**
- **Commit volatility**
- **Contributor inactivity windows**

Turnover

- **Developer turnover**: estimated churn across the analyzed contributor base. In this work, turnover is interpreted carefully because it can reflect either instability or healthy episodic participation depending on project context. Turnover here refers to contributor-base churn across the analyzed window, not organizational attrition in the HR sense.

Repository process signals

- Presence or absence of files such as `SECURITY.md`, `CODEOWNERS`, workflow configuration, and dependency automation metadata.

3.3 AI Influence Metrics

AI influence is not directly observed; it is **inferred reproducibly** from repository-level signals computed from the analyzed commit sample. The analyzer aggregates multiple sub-signals into a composite AI influence score. These include:

- **AI commit pattern score**: commit-shape and output regularity cues.
- **AI temporal anomaly score**: unusual temporal concentration patterns suggestive of tooling-assisted bursts.
- **AI style uniformity score**: stylistic consistency patterns that may indicate assisted generation or homogenized output.
- **AI metadata signal score**: metadata-level cues associated with AI-aware workflows.
- **Temporal adoption prior**: a moderating factor that increases interpretive caution for recent periods in which AI-assisted tooling became more plausible.
- **Historical constraint / legacy variance protection**: safeguards intended to reduce false AI inflation in older or legacy-heavy repositories.

These signals are combined into an **AI Influence Score** and an **effective SDLC mode** classification (for example, human, hybrid, or AI-influenced/human-dominant interpretations depending on the version of the analyzer).

This is an inference layer, not a claim of direct authorship attribution. The method is nevertheless deterministic, documented, and reproducible across runs with the same inputs.

Biguá analyzer AI inference:Interpretation notes (worth mentioning)

- Very young repositories have their burstiness signal downscaled to avoid startup-phase false positives.

- Repositories dominated by pre-2022 activity are constrained by a historical feasibility rule so they are not labeled as AI-driven without strong non-temporal evidence.
- Large, long-lived repositories receive legacy variance protection to avoid confusing structural evolution with AI influence.
- Confidence reflects evidence coverage and repository size; small or sparse repositories should not produce highly confident AI judgments.

3.4 Fast and Full Analysis Modes

The analyzer supports two distinct modes.

Fast mode is a recent-signal screening strategy. It begins with a one-year window and uses time-bucketed sampling. When recent data is weak, the analysis window may expand automatically to two or three years to strengthen the sample.

Full mode evaluates the repository using its complete accessible history.

These two modes do not answer exactly the same question:

- **Fast mode** asks: *How does the repository look in recent observable behavior?*
- **Full mode** asks: *What is the repository's structural profile across its historical lifetime?*

This distinction became especially important in mature repositories with comparatively modest recent activity relative to their full history.

3.5 Traffic Light Classification

A critical clarification in this paper is that the **traffic light is not a direct risk class for the repository**. It is a class for the **quality and interpretability of the observed signal**.

The analyzer uses four classes:

- **Red**: insufficient recent data or metadata anomaly
- **Orange**: weak recent signal, very low/low confidence, or reliability warning
- **Yellow**: moderate signal, moderate confidence, or expanded window
- **Green**: strong signal with no reliability alerts

In other words, traffic light answers:

> **How much trust should we place in this run as a structural reading of the repository?**

This is especially important when interpreting fast-mode results.

4. Results

Table 1 - Fast analysis for all repositories

Repos analyzed	42
% in each traffic-light class	green 78,57% yellow 16,67% orange 4,76% red 0,00%
Median AI influence score	0,452632
Reruns in full mode (due to bad traffic light classification - confidence)	2
High-deviation structural outliers	4

4.1 Structural Metrics Remain the Strongest Observed Explanatory Layer

Across the analyzed repositories, the clearest structural relationship remains the interaction between contribution concentration and contributor redundancy. Repositories with higher contribution inequality tend to exhibit lower effective redundancy, while more distributed repositories sustain broader contributor coverage and higher bus factors. This is consistent with established OSS health literature and suggests that the strongest predictors of structural resilience remain sociotechnical rather than AI-specific.

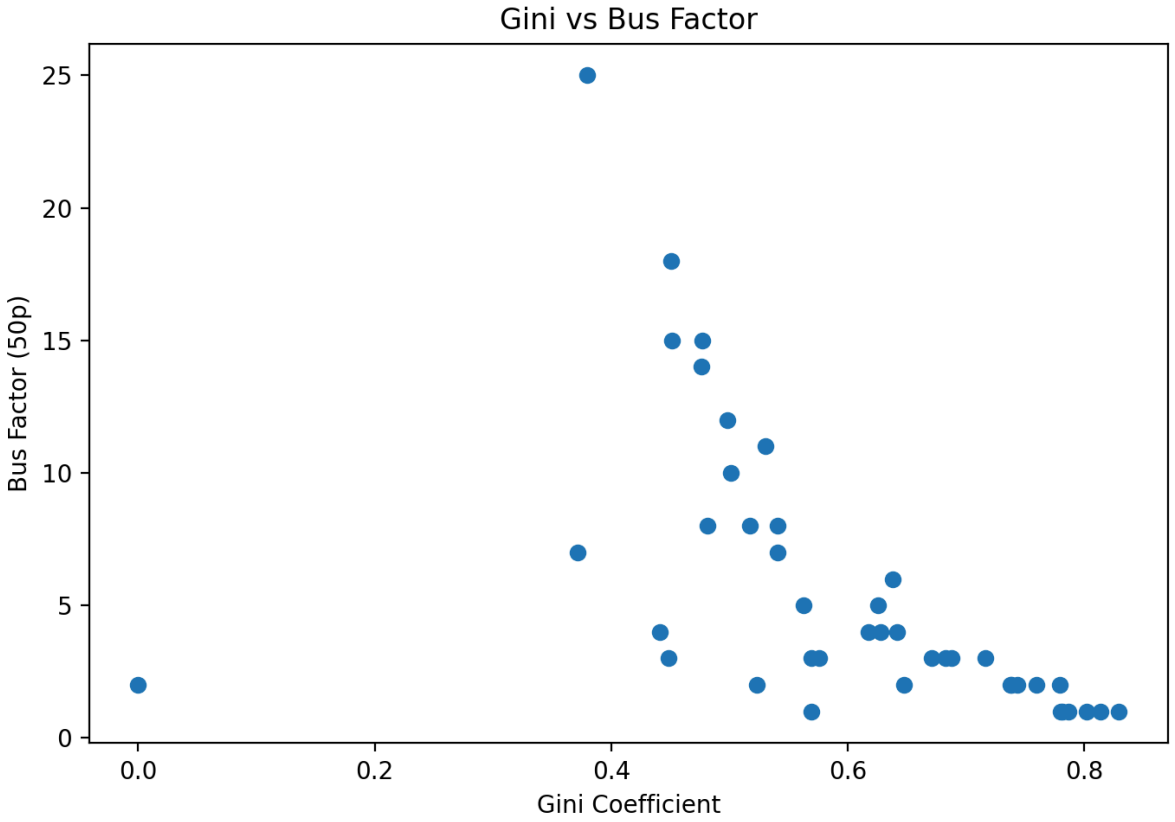

Figure 1. Gini coefficient versus bus factor (50th percentile). Higher contribution inequality is generally associated with lower contributor redundancy.

4.2 AI Influence Is Present but Not Structurally Dominant

The observed AI influence signals suggest that AI-related patterns are now present in modern OSS workflows. Most consistent structural explanatory layer in this dataset. The major distinctions still emerge from concentration, redundancy, contributor breadth, and historical participation patterns.

This leads to a key interpretation:

> **AI behaves more like an ambient capability layer than a primary structural driver.**

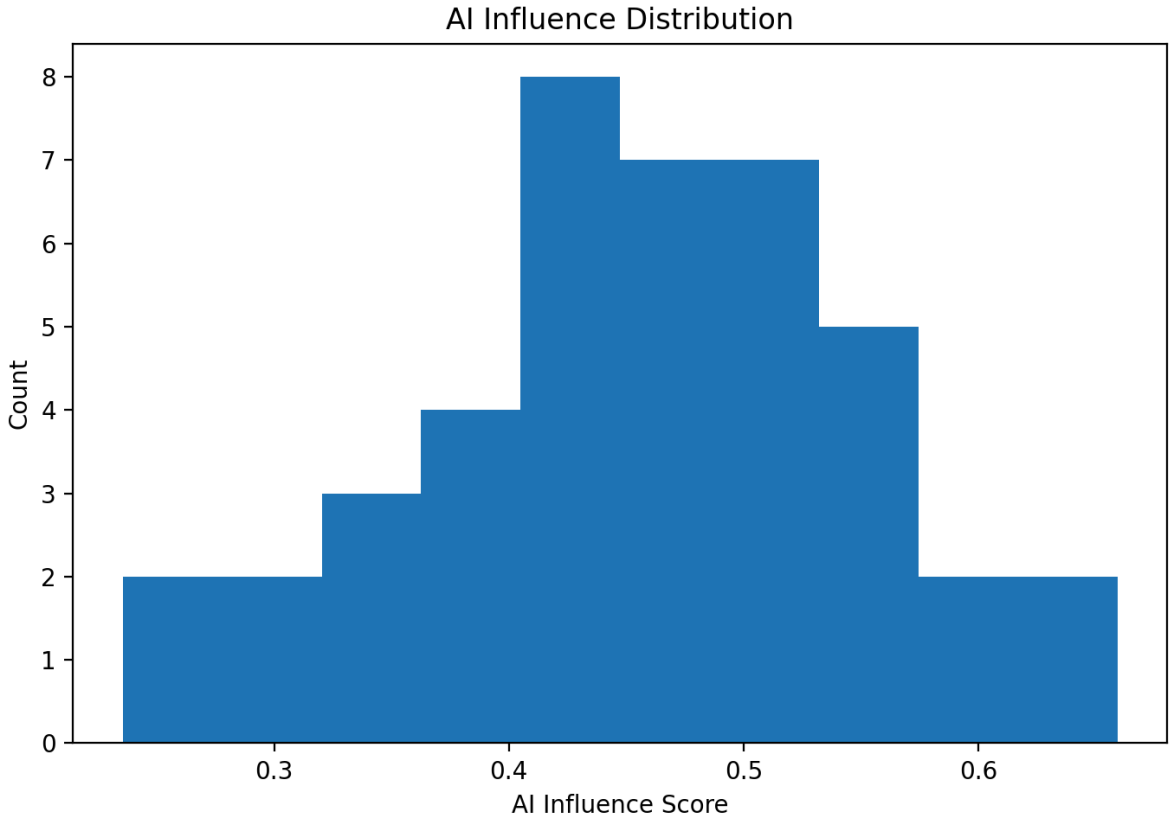

Figure 2. The AI Influence Scores across the analyzed repositories are tightly clustered within a narrow band. This concentration suggests that AI's role is a widespread background factor in development, not a strong structural differentiator between projects.

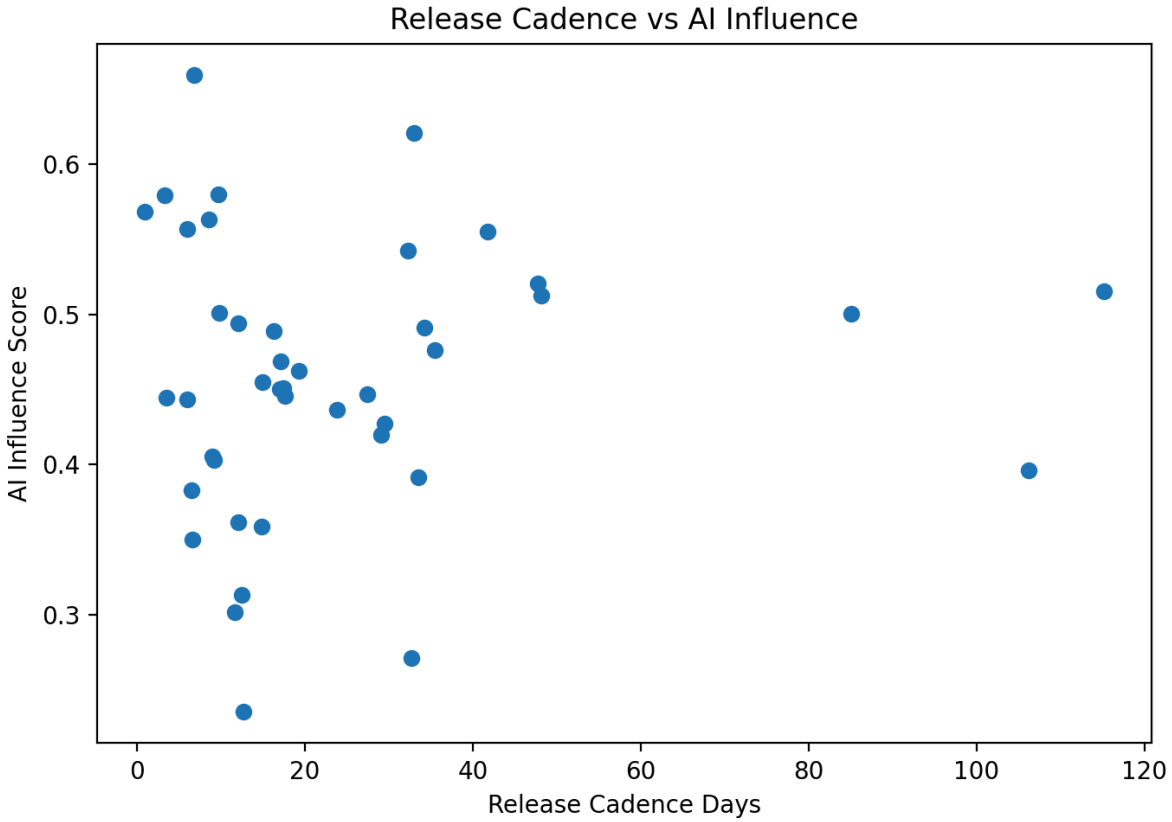

Figure 3. Release cadence versus AI Influence Score across the analyzed repositories. No clear monotonic relationship is visible, suggesting that inferred AI influence does not systematically track release frequency in this dataset.

4.3 Traffic Light Clarifies Signal Reliability, Not Repository "Goodness"

Traffic-light status was used as an interpretive aid, not as the sole trigger for escalation to full-history analysis. Initial visualization of repository groups showed that most repositories fell into the strongest-confidence class, with smaller subsets in moderate or weak-confidence buckets. The important interpretation is not that "green repositories are safe" or "orange repositories are dangerous," but that recent-window readings vary in how strongly they support structural inference.

This framing proved especially important in outlier repositories whose fast-mode results diverged sharply from their full-history profiles. It is subsequently utilized in the Biguá Analyzer pipeline to generate the AI analysis regarding the repository, serving as a data confidence metric.

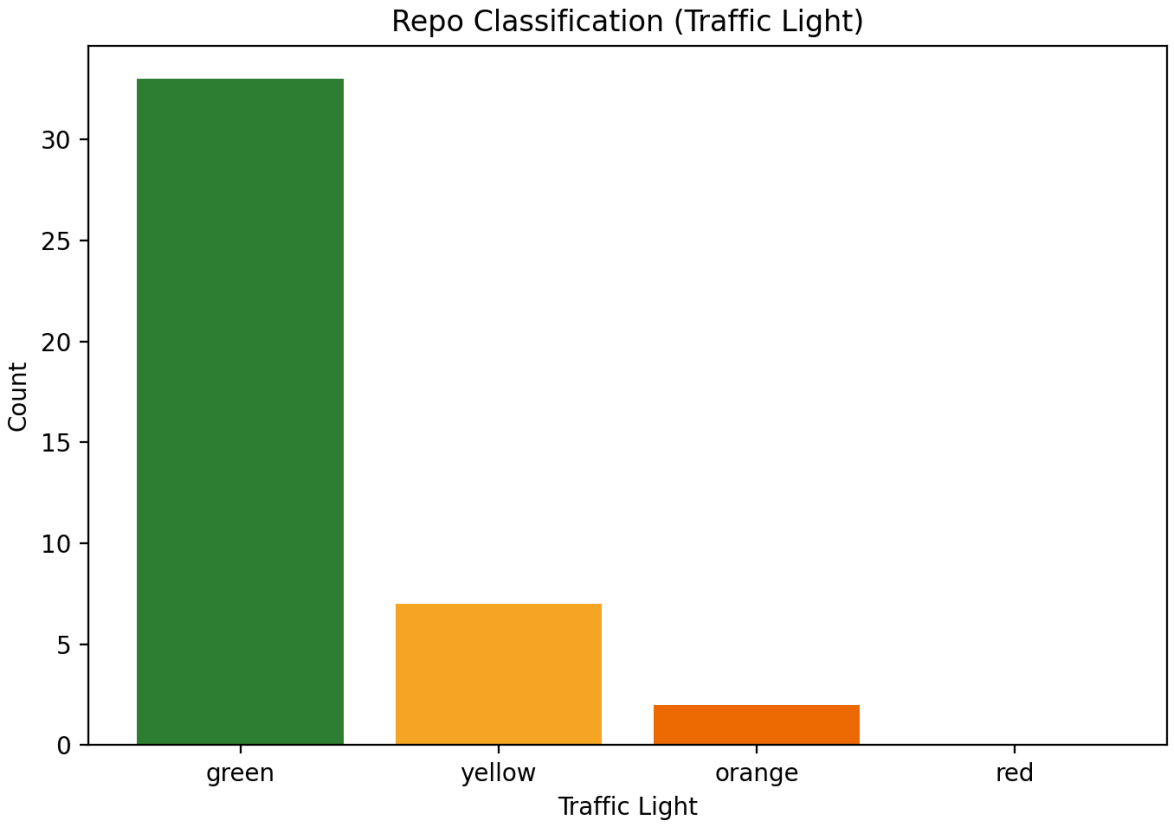

Figure 4. Traffic light distribution across the analyzed repository set. The classes represent signal quality and interpretive confidence for the current analysis, rather than absolute project risk.

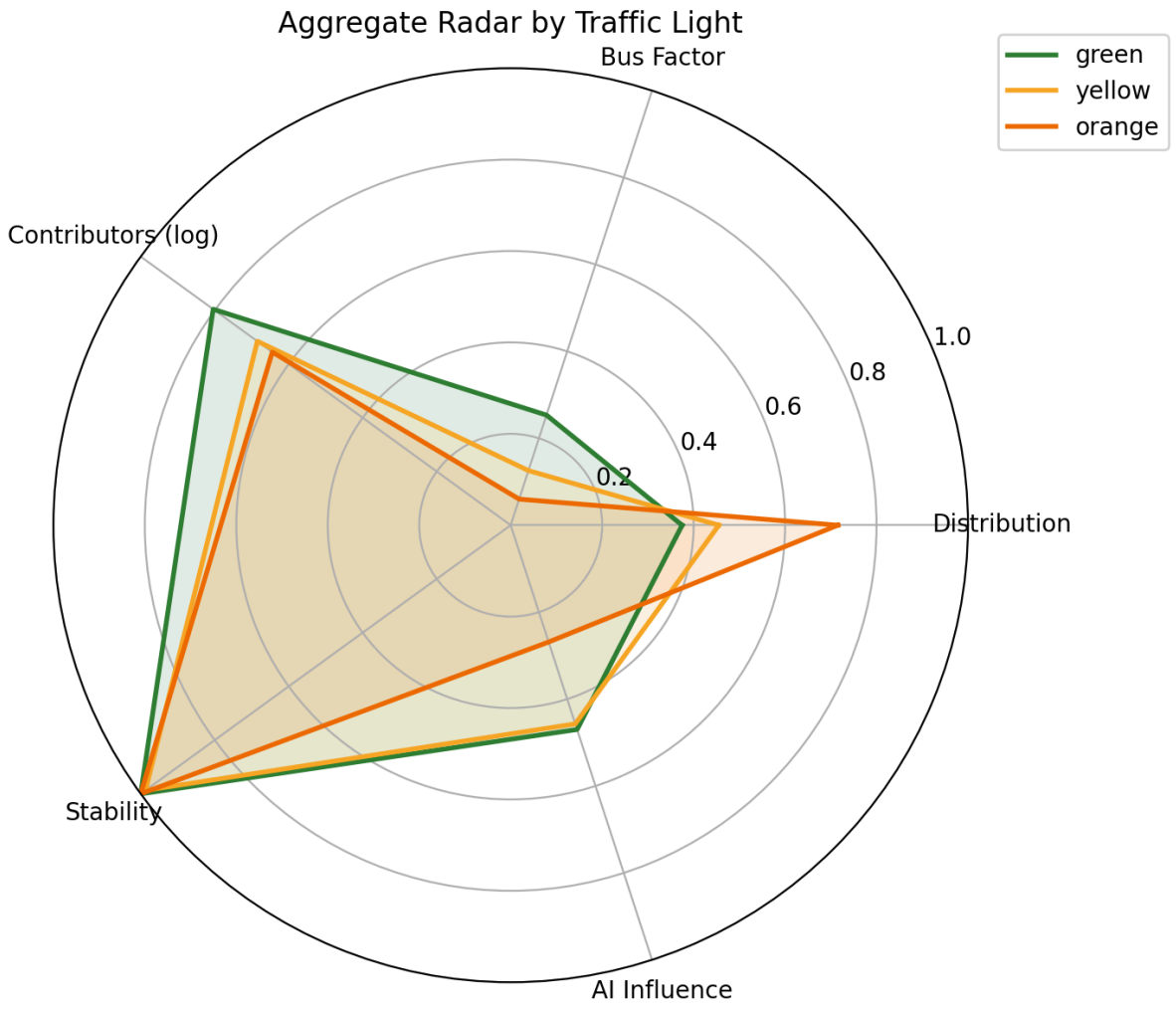

Figure 5. Aggregate radar profiles by traffic light class. The classes show distinct multimetric patterns across distribution, bus factor, contributor scale, stability, and inferred AI influence, indicating that the traffic-light system captures broader differences in signal profile rather than any single metric alone.

4.4 Outlier Repositories Exposed a Key Methodological Boundary

Two repositories—`encode/httpx` and `spf13/cobra`—showed unusually large deviations between fast recent-window analysis and full-history analysis. In both cases, the recent-window view underrepresented historical concentration patterns and produced more “modern” profiles than the full-history mode.

Based on the values we extracted:

For `spf13/cobra`, fast mode suggested a more hybrid and less historically concentrated structure, while full mode restored a strongly human-led interpretation, a much higher Gini coefficient, substantially higher turnover, and a broader historically distributed bus factor.

For `encode/httpx`, the same pattern appeared: fast mode reduced the apparent historical concentration and raised AI influence relative to full mode, while full mode showed a clearly human-led project with stronger concentration and much lower inferred AI influence. The full-history report for `httpx` explicitly shows a highly concentrated pattern, low AI influence, human-led classification, and strong confidence. The full-history report for `cobra` similarly shows a mature, human-driven profile with high turnover, low AI influence, and strong confidence. Another interesting thing to note in cobra full is that historical concentration and broad long-tail participation (High Gini and High Bus factor 75p) can coexist.

These outliers demonstrate an important methodological fact:

> **Fast mode and full mode are not contradictory; they are observing different temporal questions.**

Fast mode is useful for triage. Full mode is better suited for final structural characterization.

Table 2 - Fast versus full analysis for outlier repositories

Revisiting the outliers with full-history analysis restores stronger historical concentration and lower inferred AI influence in both cases

Metric	httpx fast	httpx full	cobra fast	cobra full
Top contributor share	0.3690	**0.4668**	0.0968	**0.1453**
Bus factor 50p	2	**2**	7	**11**
Bus factor 75p	16	**13**	18	**75**
Gini coefficient	0.5232	**0.7974**	0.3715	**0.6441**
Developer turnover	0.0679	**0.9094**	0.0231	**0.9104**
AI influence score	0.4362	**0.1454**	0.3959	**0.0875**
Effective SDLC mode	hybrid	**human**	hybrid	**human**

Note: Turnover here refers to contributor-base churn across the analyzed window, not organizational attrition in the HR sense.

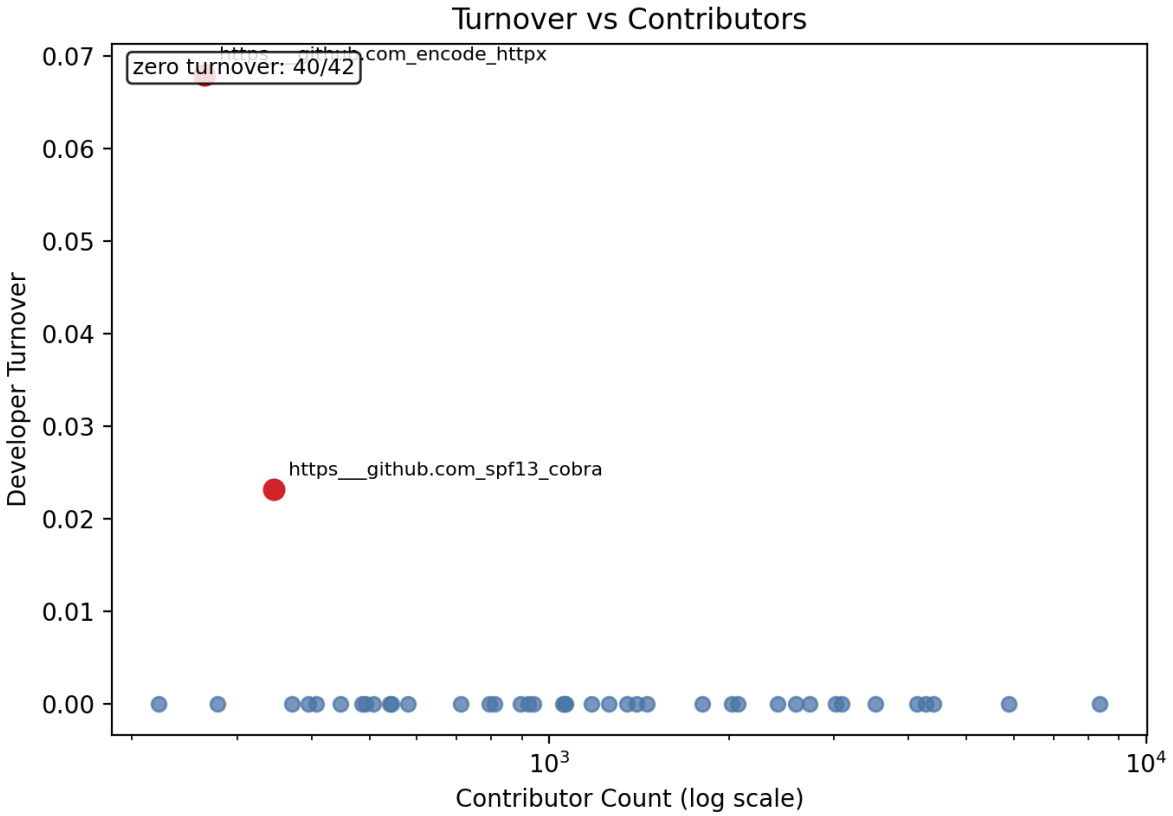

Figure 6. Developer turnover versus contributor count (fast mode). Most repositories cluster near zero turnover, while the highlighted outliers depart sharply from the rest.

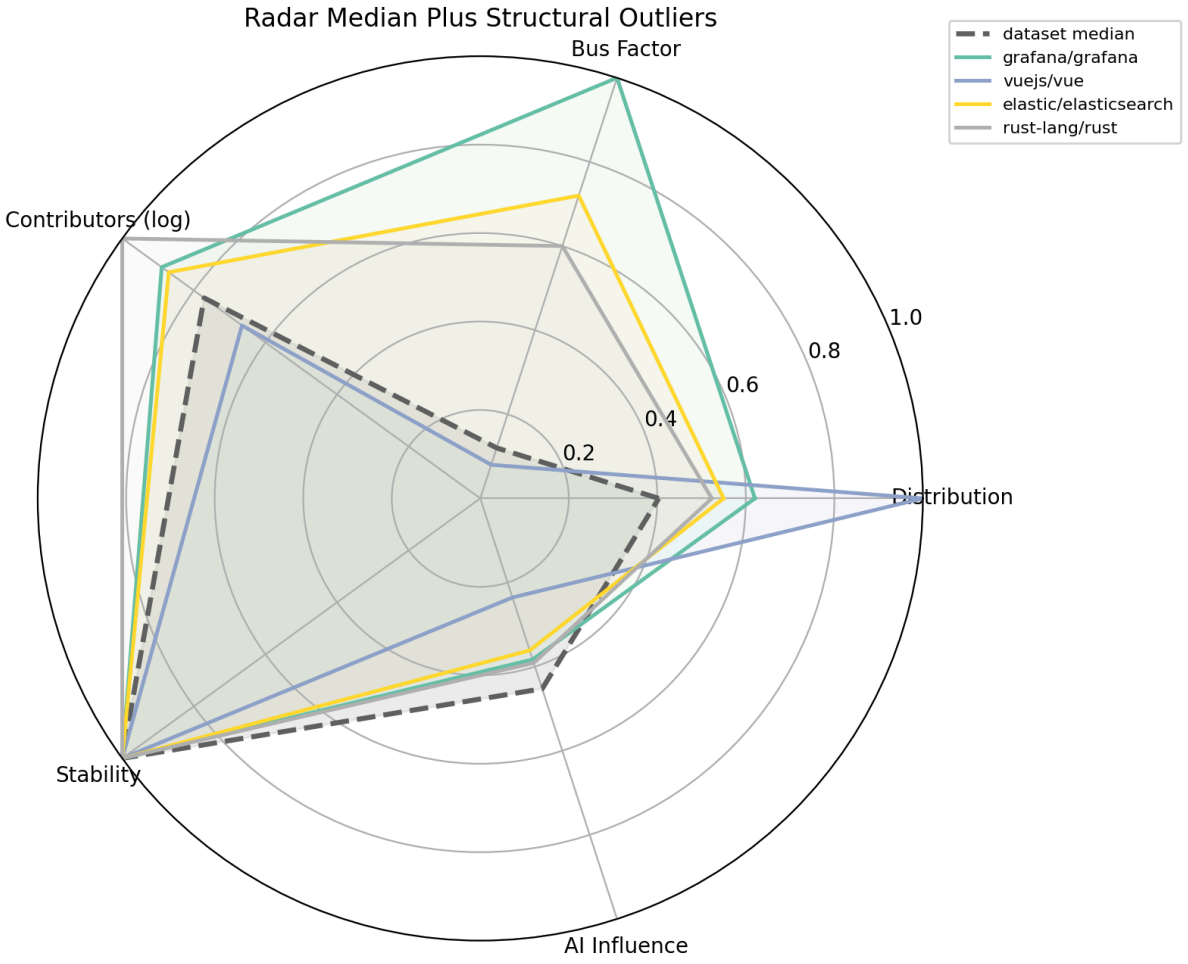

Figure 7. Dataset median plus selected structural outliers. Outlier repositories diverge most strongly on concentration, redundancy, and contributor scale rather than on AI influence.

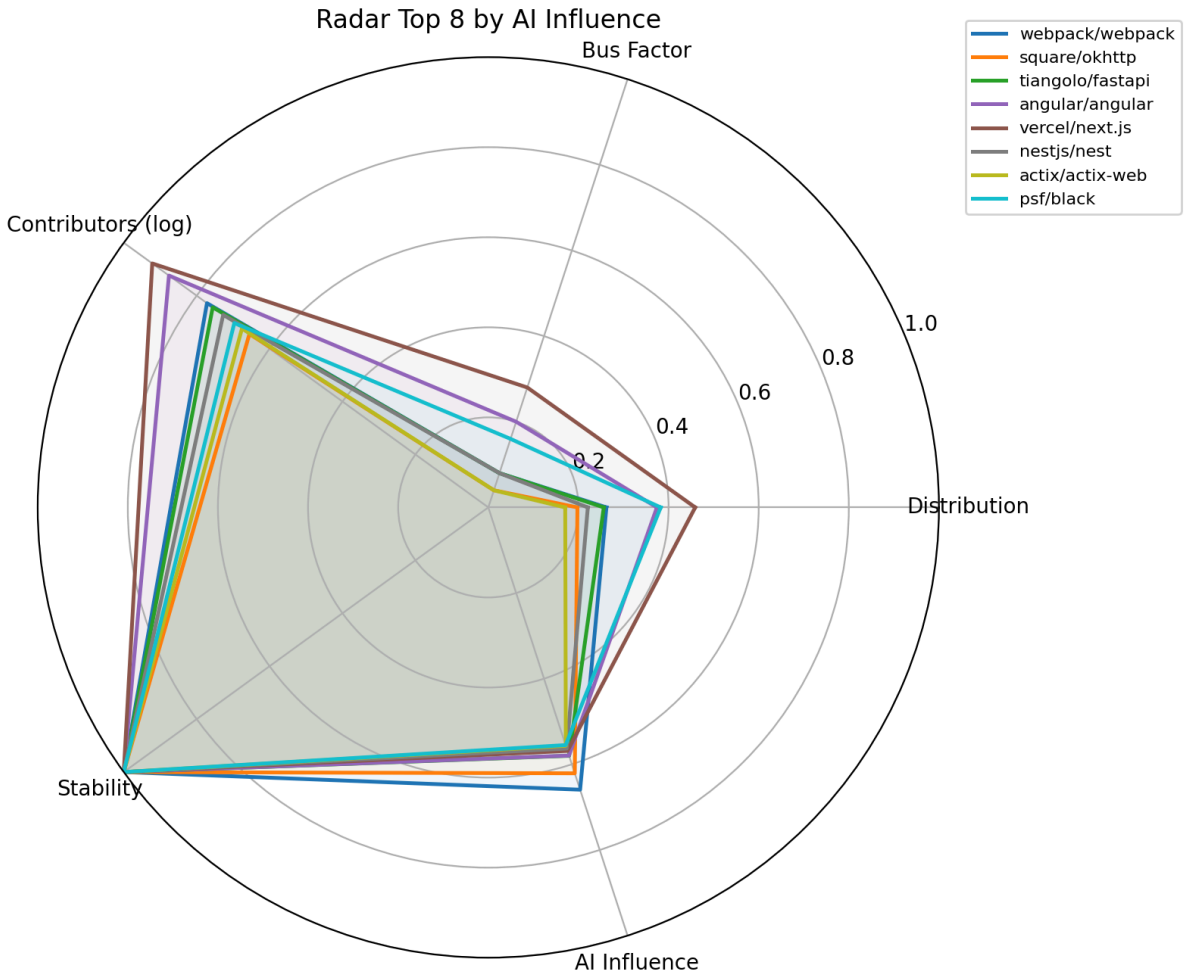

Figure 8. Radar profiles for the eight repositories with the highest inferred AI influence. Consistent with the preceding median-and-outlier view, these repositories cluster in the more stable portion of the sample, with uniformly high stability and greater variation in contributor scale, contribution distribution, and bus factor. This reinforces that higher inferred AI influence does not correspond to a single structural pattern.

4.5 What Changed in the AI Era, Practically Speaking

The data supports a narrower and more precise claim than “AI changed OSS.”

What changed is the **meaning of some observable signals**:

- Low recent turnover is no longer a strong differentiator by itself.

- Short-window stylistic regularity is less informative than it was in pre-AI interpretation.
- Recent activity can look cleaner, more uniform, and more AI-shaped without changing the underlying historical structure of the project.

What did **not** change is more important:

- concentration still signals structural dependency,
- redundancy still signals resilience,
- long-tail historical participation still matters,
- project structure remains more explanatory than AI presence.

5. Discussion

5.1 Reframing Dependency Risk in the AI Era

The core mistake for AppSec teams is treating AI presence as the central OSS risk variable. Our results suggest that this is not the most useful framing. AI can alter local observables, but it does not replace the underlying sociotechnical architecture of a project.

The more defensible conclusion is:

> Risk in OSS remains primarily structural, and AppSec review should reflect that.

5.2 What Common AppSec Heuristics No Longer Mean

The AI era weakens some heuristics that security teams may still rely on when triaging dependencies.

A recent window with low churn and regular style may once have been interpreted as evidence of stability or mature consistency. Today, that same pattern may also reflect AI-assisted regularization rather than durable project resilience.

Similarly, moderate recent concentration may not mean low structural risk in mature repositories if the sampled window excludes the historical long tail that actually defines the project.

5.3 Structural Invariance, Practical Interpretation

The strongest synthesis of the findings is this:

> The AI era did not fundamentally change OSS structure; it changed the interpretability of certain observable metrics that AppSec teams may use during dependency review.

That distinction is central to the paper.

6. Limitations

This study has several limitations.

First, **fast-mode analysis is based on sampled recent commit windows**, which makes it well suited for screening but not always sufficient for final structural inference. To address this, repositories with unusually large deviations or suspicious outlier behavior were rerun in **full-history mode**. This revealed that some mature repositories can appear less concentrated, lower-turnover, and more AI-influenced in recent-window analysis than they do when evaluated across their historical contribution structure.

Second, **AI influence is inferred rather than directly observed**. However, this inference is not ad hoc. It is based on a documented, reproducible, deterministic method composed of multiple sub-signals and guardrails. The limitation is therefore epistemic, not procedural: the method estimates AI-related patterns from repository evidence, but cannot prove per-commit authorship origin.

Third, the framework operates primarily on repository-visible data and therefore does not capture off-platform coordination, private review processes, or other invisible governance mechanisms.

Fourth, turnover remains difficult to interpret without contextual separation of core maintainers and peripheral contributors. High turnover can reflect instability, but it can also reflect healthy episodic participation in popular OSS projects.

Finally, the analyzer's recent-window logic assumes, at least partially, that recent observable behavior is a useful approximation of structural state. The outlier cases show that this assumption is weaker in mature repositories with relatively sparse recent activity compared with their historical scale.

7. Future Work

Several directions follow naturally from this work.

One is to formalize **representativeness guardrails** for mature repositories in fast mode, especially when recent samples are moderate but still historically unrepresentative.

Another is to extend the framework with **multi-window comparisons**, explicitly contrasting recent, medium-term, and full-history structural signatures.

A third direction is to deepen the AI influence model through external validation datasets, richer metadata, and additional safeguards for legacy-heavy repositories.

Finally, integrating these structural signals with **security outcomes**—such as patch latency, exposure handling, vulnerability remediation, or incident patterns—could make the framework directly useful for software supply chain risk assessment.

8. Conclusion

This study argues that the most important effect of the AI era on OSS is not structural disruption, but interpretive distortion in some visible repository signals.

AI-assisted development appears widespread, but it does not replace the classical determinants of OSS resilience. Contribution concentration, contributor redundancy, and historically distributed participation remain the strongest indicators of structural fragility or robustness.

Recent-window analysis is valuable as a screening tool, and the traffic-light system provides a useful confidence model for interpreting those screenings. But mature repositories with sparse recent activity can require full-history analysis before an AppSec team should treat the structural reading as decision-grade.

The central conclusion is therefore:

> AI changes how some repository signals look, but structure still determines what they mean.

Appendix

AI Influence Decomposition and Supplementary Materials

The AI Influence Score includes calibration and interpretability outputs intended to reduce false positives in young and legacy repositories.

Key fields:

- ai_influence_score
- ai_influence_confidence
- ai_influence_rationale
- ai_temporal_adoption_prior
- ai_temporal_anomaly_weight
- ai_dominant_activity_period

Intermediate heuristic outputs:

- ai_h1_textual_markers
- ai_h2_explicit_attribution
- ai_h3_temporal_prior
- ai_h4_burstiness
- ai_h5_style_shift
- ai_h6_large_low_discussion
- ai_h7_output_asymmetry
- ai_h8_tooling_footprint
- ai_h9_generated_text_pattern

Reproducibility and supplementary materials.

Full metric definitions are available in the project documentation in the Biguá Analyzer repository, including the metrics.md reference. The study dataset (all_repos.csv), exported screening results (fast_results2.csv), full-history comparison results for the deviation follow-up set (httpx_cobra_full.csv), per-repository radar visualizations are provided as well the AI analysis for each repo generated by the Biguá analyzer tool as supplementary artifacts in the project Releases (in releases - v0.4.3, v0.5.0 and v0.5.1).

Research dataset

URL, Language

https://github.com/golang/go,go

https://github.com/gin-gonic/gin,go

https://github.com/spf13/cobra,go

https://github.com/hashicorp/terraform,go

https://github.com/kubernetes/kubernetes,go

https://github.com/prometheus/prometheus,go

https://github.com/grafana/grafana,go

https://github.com/etcd-io/etcd,go

https://github.com/helm/helm,go

https://github.com/go-gorm/gorm,go

https://github.com/nodejs/node,js

https://github.com/expressjs/express,js

https://github.com/facebook/react,js

https://github.com/vuejs/vue,js

https://github.com/angular/angular,js

https://github.com/axios/axios,js

https://github.com/lodash/lodash,js

https://github.com/vercel/next.js,js

https://github.com/nestjs/nest,js

https://github.com/webpack/webpack,js

https://github.com/spring-projects/spring-framework,jvm

https://github.com/elastic/elasticsearch,jvm

https://github.com/apache/kafka,jvm

https://github.com/square/okhttp,jvm

https://github.com/google/guava,jvm

https://github.com/netty/netty,jvm

https://github.com/psf/requests,python

https://github.com/pallets/flask,python

https://github.com/django/django,python

https://github.com/numpy/numpy,python

https://github.com/pandas-dev/pandas,python

https://github.com/tiangolo/fastapi,python

https://github.com/scrapy/scrapy,python

https://github.com/pytest-dev/pytest,python
https://github.com/encode/httpx,python
https://github.com/psf/black,python
https://github.com/rust-lang/rust,rust
https://github.com/tokio-rs/tokio,rust
https://github.com/serde-rs/serde,rust
https://github.com/hyperium/hyper,rust
https://github.com/rust-lang/cargo,rust
https://github.com/actix/actix-web,rust

References

Avelino, G., Passos, L., Hora, A., Valente, M. T., and Costa, D. A novel approach for estimating truck factors. *Proceedings of the 24th IEEE/ACM International Conference on Program Comprehension (ICPC)*, 2016.

Corso, V., *et al.* An empirical assessment of four AI-based code assistants. *Proceedings of the 32nd IEEE/ACM International Conference on Program Comprehension (ICPC)*, 2024.

Crowston, K., Howison, J., and Annabi, H. Information systems success in free and open source software development: Theory and measures. *Software Process: Improvement and Practice*, 11(2), 123–148, 2006.

Jansen, S. Measuring the health of open source software ecosystems: Beyond the scope of project health. *Information and Software Technology*, 56(11), 1508–1519, 2014.

Mens, T., *et al.* An overview and catalogue of dependency challenges in open source software package registries. *Proceedings of BENEVOL 2024*, CEUR Workshop Proceedings, 2024.