TECHNICAL PAPER · C

REP_CHURN: Replacement-Dominant Churn as a Process Signature of AI-Assisted Development

A deterministic, ML-free churn-shape metric; measurements across seven codebases showing a gradient from hand-maintained (0.57) to agent-built (0.998); its limits; and how to reproduce every figure with git and arithmetic.

First release · 3 July 2026

© 2026 CodeDelta · codedelta.app

ABSTRACT

Conventional churn analysis counts lines *added*, *deleted*, and *changed* between two snapshots of a codebase. This paper examines the *shape* of that churn rather than its volume. We define **REP_CHURN = (ADD + DEL) / CRN**, bounded 0–1: the share of all change activity that was wholesale replacement rather than in-place modification. The hypothesis is that AI-assisted development exhibits a distinctive replacement-dominant signature — generated code is accepted or regenerated, rarely hand-edited — whereas traditional maintenance spends a large fraction of its churn reworking existing statements. Measurements across seven codebases support the hypothesis: repositories with near-zero AI-agent commit attribution cluster at lower REP_CHURN with substantial in-place modification (a hand-maintained JavaScript framework: REP_CHURN 0.57, with 43% of churn modifying existing statements), while an intensively agent-assisted codebase measures REP_CHURN 0.998, with 0.19% of churn modifying existing statements. The metric is deterministic diff arithmetic — no machine learning, no stylistic inference — and every figure is reproducible from public history. We state its failure modes explicitly: bulk file moves, vendoring, and greenfield growth also produce high values, so REP_CHURN indicates a *development process*, not the authorship of any individual change.

01 Definition

Given the standard two-snapshot churn quantities over logical statements (LLOC) — *CHG* (statements modified in place), *DEL* (statements removed), *ADD* (statements introduced), and total churn *CRN* = *CHG* + *DEL* + *ADD* — define:

```
REP_CHURN = (ADD + DEL) / CRN
```

Its complement, `CHG / CRN` (the *rework share*), is often the more intuitive reading: the fraction of all change activity spent editing statements that already existed. REP_CHURN = 1 describes change composed purely of insertions and removals; REP_CHURN = 0 would describe change composed purely of in-place edits. The classification of statements into CHG/DEL/ADD follows the two-pass sequence-alignment method described in Paper A; REP_CHURN adds no new measurement machinery, only a ratio over quantities already computed.

© 2026 CodeDelta · codedelta.app

02 Rationale

Human maintenance of source code is dominated by *editing*: renaming, adjusting conditions, threading a parameter, tightening an expression. Each such act modifies existing statements in place and is classified as CHG. Development assisted by code-generating AI proceeds differently as a matter of workflow mechanics: the tool produces a block; the human accepts, rejects, or asks for regeneration. Unsatisfactory code tends to be *replaced by a new generation* rather than repaired line by line. At the diff level this produces DEL + ADD patterns rather than CHG, regardless of the style or quality of the generated text.

This suggests a process signature that is invisible to stylistic authorship detection but visible to plain churn accounting: **as AI assistance increases, the rework share of churn should collapse.** Because the signature arises from the workflow rather than the text, it cannot be prompt-engineered away, and it does not degrade as generation quality improves — if anything, better generators make replacement cheaper and the signature stronger.

03 Method

Seven codebases were measured. For six public repositories, two snapshots were taken (1 January 2026 and 2 July 2026, the nearest commits preceding each date) and compared with a two-pass LLOC differ (Paper A). The seventh subject is the authors' own agent-assisted commercial codebase, measured over one month of development (1 June – 3 July 2026) using the same engine in git-native mode.

As independent ground truth for the degree of AI involvement, each repository's commit history was scanned for machine-readable AI co-authorship attributions — `Co-Authored-By` trailers and equivalent markers left by named AI coding agents (GitHub Copilot's agent, Claude Code, Cursor, and eight others). Such attribution is voluntary and strippable, so measured rates are **floors, not totals**. Matching is exact-pattern against agent-specific strings; dependency bots and other scripted automation cannot match.

© 2026 CodeDelta · codedelta.app

04 Results

CODEBASE	CHARACTER	AGENT-SIGNED COMMITS	REWORK SHARE (CHG/CRN)	REP_CHURN
Express	JavaScript framework, est. 2010, hand-maintained	0.02%	42.9%	**0.57**
LangChain	AI-infrastructure library	0.3%	10.3%	**0.90**
FastAPI	Python web framework	0.03%	8.6%	**0.91**
Redis	C database, est. 2009	0.1%	6.6%	**0.93**
CrewAI	AI-native agents library	3.9%	2.7%	**0.97**
GitHub CLI *	Go tool; 248 Copilot-co-authored commits in the period	2.2%	0.9%	**0.99 ***
Authors' codebase	agent-assisted commercial product (one month)	65.9%	0.19%	**0.998**

The rework-share column tells the story most plainly. The hand-maintained end of the table spends over forty percent of its churn *editing what exists*; the agent-assisted end spends a fifth of one percent. The gradient tracks the independent attribution ground truth throughout — with one instructive exception, marked *, discussed below. In absolute terms, the agent-assisted subject also produced churn at roughly six times the monthly rate of the fastest-moving public repository in the sample — replacement-dominant development is fast as well as differently shaped.

** The GitHub CLI figure includes a large repository restructure in the measurement window: bulk file moves classify as DEL + ADD and inflate REP_CHURN. This is the metric's principal failure mode, and the reason the value carries an asterisk rather than a conclusion.*

© 2026 CodeDelta · codedelta.app

05 Limits and correct usage

- **Not a per-change verdict.** A high-REP_CHURN period may reflect AI assistance — or vendoring, code generation of the traditional kind, bulk reorganisation, or greenfield growth. The metric characterises a development process over a period; attribution of individual changes requires independent evidence (commit trailers, review records).
- **Sensitive to bulk file operations.** Moves and renames read as DEL + ADD unless rename detection is applied. Where available, rename-aware accounting and a changed-files-only variant should be reported alongside the headline figure.
- **Thresholds are not calibrated.** Seven codebases establish a gradient, not a scale. No specific value should be read as a boundary between assisted and unassisted development.
- **Longitudinal use is the strongest use.** Comparing a codebase with its own history avoids all cross-project confounds: a repository whose rework share collapses across consecutive periods has changed how it is developed, whatever the absolute numbers.

The defensible claim, stated once and precisely: *replacement-dominant churn is characteristic of AI-assisted development; it is not proof of it.*

06 Reproducibility

Every figure above can be reproduced without specialised tooling:

1. Check out two dated snapshots of a repository (`git rev-list -1 --before=<date> HEAD` , then `git worktree add`).
2. Compute CHG/DEL/ADD over logical statements with any differ implementing the method of Paper A (any LLOC-level differ will produce closely comparable ratios).
3. REP_CHURN is one division. The attribution ground truth is one `git log` scan for agent co-author trailers.

The proposed next study is longitudinal: the same repositories sliced year by year, 2019–2026, testing whether the rework share exhibits an inflection as AI coding assistance became widespread. The method above applies unchanged.

© 2026 CodeDelta · codedelta.app

References

1. Paper A (this series), *Automated Source Code Churn Measurement: Logical and Physical Line Differencing in Large Codebases* — paper-churn.html
2. Myers (1986), *An O(ND) Difference Algorithm and Its Variations* — doi.org/10.1007/BF01840446
3. Measured repositories: expressjs/express, langchain-ai/langchain, fastapi/fastapi, redis/redis, crewAIInc/crewAI, cli/cli (snapshots 1 Jan / 2 Jul 2026, public GitHub history).

© 2026 CodeDelta · codedelta.app