RCE-VC: Law of Version Control

David Charlot, Open Interface Engineering

Research study · draft · Open Interface Engineering · research.openie.dev

Revision `af57f48` (22 November 2026) · Web: research.openie.dev/papers/rce-vc/ · Archival LaTeX: /pdfs/latex/rce-vc.pdf (.tex)

Measurement. Research study, draft, not a journal final. Milestones `M0` to `M2` are verified in public `openIE-dev/rce-vc` at commit `aba99fce` (`49` offline tests passing). Wall-clock times (`69.7` to `73.6 µs`), admission-path shares, and Apple M5 Max per-process `L3 reported_j` (`317.4` to `365.4 µJ` per `100,000`-unit single-core admission) were measured on the reference build. Receipt costs also carry the stated `est_uj` schedule. No hardware power rail or wall meter was attached (`measured_j = None`, `board_metered = false`). No FPGA board result. No fabricated citations.

RCE-VC (a version-control reference, separate from the Reality Code Engine language) admits each change into one canonical repository state or refuses it with a recorded reason, splits Rust and Python files into name-free top-level items, and reads and writes ordinary Git history. Milestones M0, M1 and M2 are implemented at public commit aba99fce (49 offline tests). At 100,000 units on Apple M5 Max, admission takes 69.7 to 73.6 microseconds and 317 to 365 microjoules of per-process kernel rusage energy (reported_j; measured_j is unset).

Contents

Abstract 2

Terms and labels 2

1. What the reference is, and what it is not 2

2. Law of Version Control 3

3. Quantities 3

4. The repository form 4

5. The transition 5

6. Propositions the tests hold 5

7. Measured wall time 6
7.1 Path shares at 400 units . . . 6
7.2 Integration cost against repository size . . . 7
7.3 Measured wall time and on-chip `reported_j` at 100,000 units (commit `aba99fce`) . . . 7

8. Stated cost on receipts 8

9. The Git projection 8

10. Limits of this version 9

11. Next work 9

12. Copyright and license 10

Revision `af57f48`, 22 November 2026. Reference implementation: openIE-dev/rce-vc at `aba99fcec4414a32bff30ab075668350d3fb80bf` (13 November 2026), updating the initial `8bf9c9af` tree with `rcevc-keys` top-level Rust/Python item splitting, the `rcevc-core::assure` boundary (`49` offline tests), and Apple M5 Max `reported_j` admission measurements recorded on 22 November 2026.

Abstract

RCE-VC is a version-control design and a small reference in Rust. Many writers, people or programs, submit changes against one shared state. A named check admits the result, or the change is refused and the reason is kept. Writers can stay on Git. The reference is not the Reality Code Engine language. It uses the same framing: a Form, a total step, a hash-chained log, name-free keys, and a ladder of gears with a model last.

In plain words

1. Any number of people and programs may write at once. A change enters the shared history when a named check accepts the result. If the check does not accept it, the change stays out and the reason is written down. Who wrote it does not decide.
2. Changes that do not touch the same things do not have to wait for one another. When they do overlap, a price written down in advance ranks the results that pass, and a result that was accepted once, including a resolved conflict, is reused. A model is asked only after the cheaper steps fail, and its answer still has to pass the check.
3. The reference reads and writes ordinary Git history and now splits Rust and Python files into name-free top-level items (`rcevc-keys`), so a writer need not leave Git. At `aba99fce` on Apple M5 Max, admitting a change against a 100,000-unit repository took `69.7-73.6 µs` and `317-365 µJ` in operating-system per-process kernel `ri_energy_nj` (`reported_j`). Receipt `est_uj` fields remain a stated integer table (`424,153 µJ` mean on the same workload), and no external hardware rail or wall monitor was attached (`measured_j = None`, `board_metered = false`).

Terms and labels

- SHA: SHA-256, the hash function the reference uses for keys, content digests and the log
- BLAKE3: a cryptographic hash named as the design target for internal keys; the reference does not use it yet
- HAMT: hash array mapped trie, the persistent map that holds repository state
- UTF: UTF-8, the text encoding the Git file mapping accepts
- MSRV: minimum supported Rust version (`1.85` for `rcevc-core` and `rcevc-proj-git`, `1.90` for `rcevc-keys`)
- CI: continuous integration (`cargo test --locked --offline --all-targets` on `rustc 1.100.0`)
- CAS: compare-and-swap reference update (`CasRef` and `GitDir::compare_and_swap_ref`)
- FIPS: Federal Information Processing Standards; the SHA-256 checks use the FIPS 180-4 vectors
- RFC: Request for Comments; the inflate code follows RFC 1950 and RFC 1951
- SATB, CPU, GPU, PID, SMC, SoC: Satiation Benchmark telemetry discipline, central processing unit, graphics processing unit, process identifier, System Management Controller, and system-on-chip
- VC1, VC2, VC3, VC4, VC5: the five propositions the reference tests, named with their tests below
- CONGRA: a corpus named in the project notes for a future replay; the name is not expanded on this page
- README: the overview file shipped in the reference repository

Three labels stay apart on this page across whole-chip (L2) and per-process (L3) telemetry:

- **Measured (wall time and counts)** means a wall-clock time, instruction count or path count from the release build (`aba99fce` on Apple M5 Max, and the README baseline printed at `8bf9c9af`).
- **Reported (`reported_j`: L2 and L3 counters)** means an operating-system or on-chip counter reading on Apple M5 Max (`macOS 27.2.1`): whole-chip `IOReport` via `macmon` (L2, shared-host label `A2`, gated by Falsifier 6 at `<= 5.0%` pre/post idle drift) and per-PID kernel `proc_pid_rusage(RUSAGE_INFO_V6).ri_energy_nj` (L3, label `D`). No external hardware rail or wall-plug power probe was attached (`measured_j = None`, `board_metered = false`).
- **Stated (`est_uj / est_j`)** means the default integer cost table copied onto receipts inside `rcevc-core`. It is an analytical table written in source code, never a counter or wall-meter reading.

1. What the reference is, and what it is not

The project's aim is a history that many humans and agents can write at once, without asking a writer to stop using Git. The framing in the project notes is that branching, diffing and merging into one main line is an old interface,

that compatibility is what keeps that interface in place, and that a deterministic rule should decide wherever a rule can, with a model last.

The Reality Code Engine (RCE) is a programming language studied in its own paper on this site. RCE-VC is not built on that language. It does not parse RCE programs, emit them, or call the RCE checker. What it shares is the framing: a Form for the repository, a total deterministic step called δ, a hash-chained log whose fold is the state, name-free keys, and a ladder of gears in which a model runs last and is not trusted.

At commit `aba99fce` (`13 November 2026`), the public workspace is version 0.1.0 and holds **three** crates:

1. `rcevc-core` is the deterministic core: no clock, no randomness, no I/O, no dependencies, and `#![forbid(unsafe_code)]` (MSRV `1.85`). It now also includes `rcevc-core::assure`, an assurance boundary ensuring that a checked action is minted only from an outcome already in that repository's log and applies only at the head and gate it was checked against, while `CasRef` retries one operation identifier so another writer's identical target is never mistaken for success.
2. `rcevc-keys` splits Rust (`.rs`) and Python (`.py`) source files into top-level items using vendored `tree-sitter 0.27.2`, `tree-sitter-rust 0.24.2`, and `tree-sitter-python 0.25.0` (MSRV `1.90`, built offline from `vendor/` with a C compiler and zero network fetches). Each item's identifier is the core name-free key of a normalized token body — invariant to item renames, whitespace, and comments — while exact original bytes stay in `parts` so `project` concatenates them back losslessly; non-`.rs`/`.py` paths or parse-error trees fall back cleanly to one leaf over the original bytes.
3. `rcevc-proj-git` is the Git projection (`#![forbid(unsafe_code)]`, MSRV `1.85`), carrying its own SHA-1, zlib and Git object code plus `GitDir::compare_and_swap_ref` so the core stays free of dependencies.

CI runs `cargo test --locked --offline --all-targets` and `cargo clippy --locked --offline --all-targets -- -D warnings` on `rustc 1.100.0`, executing **49 offline tests** (`29` in `rcevc-core`, `8` in `rcevc-keys`, and `12` in `rcevc-proj-git`, verified against `git 2.43`). Propositions numbered past VC5 are not stated on this page.

2. Law of Version Control

The reference states it as follows. The same paragraph opens the repository README and the crate documentation. It is quoted here once.

> Version control is the admission of changes to a canonical, name-free repository state. Each change is admitted only when a named checker accepts the post-state under a declared contract, or is refused with a recorded reason; changes whose dependency closures do not meet are admitted without coordination; among admissible integrations, a stated cost ranks the candidates; and every accepted result, including every resolved conflict, is reused.

The rest of this page is what the reference at `aba99fce` implements, what was measured on it, and what it does not yet do. The statement is the standard the reference is built to. It is not a report that existing version-control tools already behave this way.

3. Quantities

A **unit** is a body of text and a list of dependency slots. If the dependency list is empty, the unit is a leaf and the body is literal text, so a dollar sign in a shell script stays a dollar sign. If the list is not empty, a dollar sign followed by an index selects that dependency, and two dollar signs are a literal dollar sign.

The **key** of a unit is a name-free identity. A rename changes which slot a name is bound to. It does not change the key. The reference defines

$$\mathrm{key}(u) = H(\texttt{rcevc/unit/v1},\ \mathrm{body}(u),\ \mathrm{key}(d_0), \ldots, \mathrm{key}(d_{m-1}))$$

where H is SHA-256 through a versioned, domain-separated encoder, and d_i are the dependency units. The **content digest**, which the trie hashes, is a different value: $H(\text{body}, \text{dependency names})$.

A **draft** is an unvalidated map, used for genesis and for anything read from outside. Sealing a draft checks it and produces a **state**. A state is well formed: dependencies resolve, every index in a body is bound, and the dependency graph has no cycle. It holds three persistent maps. The unit map is a Merkle HAMT. The key map and the reverse-dependency index are persistent and are not part of the Merkle root. Cloning a state and reading its root are $O(1)$. Two states are equal when their roots are equal. A diff walks only subtrees whose hashes differ, so its cost follows the difference, not the size of the repository. Dependents whose key changed, but whose own unit did not, are left out of the diff on purpose. The closure, defined below, carries those dependents. Putting them in the diff would turn a shared dependent into a false conflict.

An **edit** changes slots and validates once at the end, over the reverse-dependency closure of the slots touched. States inside an edit may be ill formed. That is intentional, for example a deletion followed by a put.

A **change** carries an opaque writer label, a base state, a level, a list of operations, and a message. The operations are put, delete and rename. The levels are name-only, erratum, meaning change and interface change. The name-only level is checked by the kernel: the multiset of keys must be unchanged. The change identifier covers every field. The message is stated intent, for example a commit subject. Admission does not read it, and it does not depend on the writer label.

For a change c against a state s,

$$\text{writes}(c, s) = \text{diff}(s,\ \text{apply}(s, c)).$$

Reads are the dependencies of puts and the sources of renames. The closure of c in a state and in the post-state is the writes, the dependents before, the dependents after, and the reads. Two changes commute on s when each one's writes miss the other's closure:

$$\text{commutes}(a, b, s) \iff \text{writes}(a) \cap \text{closure}(b) = \emptyset \ \wedge\ \text{writes}(b) \cap \text{closure}(a) = \emptyset.$$

The reference tests that commuting changes give the same root in either order.

The step δ is `submit`. It is total on user input. It returns an acceptance with a receipt, or a refusal with a reason, and it appends one hash-chained log entry either way. The head is the fold of that log. Replay verifies the chain and re-integrates every entry. It uses the recorded resolution for a conflict. It does not call a proposer. It checks each pre-state root, each post-state root, and each refusal reason.

4. The repository form

The README lines the implemented form up as nine parts, plus the `rcevc-keys` item splitter and `rcevc-core::assure` boundary.

Part	Where	What it does
R, name-free keys	`model.rs`, `rcevc-keys`	Keys as above; `rcevc-keys` splits `.rs` and `.py` files into top-level items with lossless byte projection. A rename changes no key.
S, persistent state	`hamt.rs`, `model.rs`	Merkle HAMT. Clone and root in $O(1)$, diff in the size of the difference, canonical trie shape. Keys and the reverse-dependency index update on the edit, not on the whole repository.
Δ, typed changes	`change.rs`	Put, delete and rename against a declared base, with a declared level. Name-only is checked by the kernel.
X, commutation	`change.rs`	Writes are the actual diff. The closure adds dependents before and after, and reads. Disjoint closures commute.
K and G, checker and gate	`gate.rs`	Each checker states a property class, a linking assumption, a trusted base, an authority, a scope and a digest.
δ, the ladder	`repo.rs`, `assure.rs`	Lookup, commutation, structural merge, re-validation, proposer, plus CAS ref assurance (`CasRef`). The gate runs on every path.
D, the log	`repo.rs`	Hash-chained entries. The head is the fold. Replay rebuilds it with no proposer.
U, reuse	`repo.rs`	A resolution is stored under a name-free conflict signature, and only after the gate has accepted it.
C, cost	`repo.rs`	The stated per-rung table (`est_uj`). Receipts carry the sum.

A checker's authority is one of pinned, replay, proved, derived or inferred. Its scope is one of three. **Unit**: re-check units in the delta. **Closure**: the verdict depends on the unit and its transitive dependencies, so the gate also re-checks dependents. **Global**: check the whole state. If the previous head passed and the declared scope is true, a scoped re-check matches a full check. The test `delta_gate_agrees_with_full_gate` records that agreement over 600 trials, with more than 100 refusals. A checker that lies about its scope breaks VC2. Scope is trusted. Nothing in this version enforces it by hiding the rest of the state.

The gate digest covers every checker's statement, and that digest feeds the genesis hash. The trie shape is canonical: a prefix holds nothing, a single leaf, or a branch of two or more keys. Roots do not depend on the order history arrived in. Everything hashed goes through the encoder with a versioned domain tag of the form `rcevc/<thing>/vN`. Changing what a tag covers means bumping the version.

5. The transition

`submit` always takes one of the following paths. The gear names are the ones written on the receipt.

1. **G0, lookup.** The same change identifier against the same head, already refused, returns the stored refusal. The stated cost of that retry is the G0 row alone. The test `g0_identical_retry_costs_one_lookup` records this.
2. **Base.** The base snapshot must exist, or the reason is an unknown base. Applying the operations to that base must succeed, or the reason is an apply error or an ill-formed unit. A name-only claim that does not leave the multiset of keys unchanged is refused.
3. **G1, commutation.** Let the moved set be the diff from the base to the current head. If the change's closure, in the base and in the head, misses that set, the change is applied to the head directly. No merge.
4. **G2, structural merge.** Otherwise the reference merges, slot by slot, over the union of what moved on the head and what the change itself changed. One side unchanged: take the other side. Both sides equal: take either. Any other slot is a conflict.
5. **Conflict.** The signature is $H(\text{key}_{base}, \text{key}_{head}, \text{key}_{theirs})$, which does not use the slot's name. G0 reuse returns a stored resolution for that signature. If none is stored, G4 asks the proposer. If there is no proposer, or it returns nothing, the change is refused with the conflicted slots. During replay the recorded resolution is used, and the proposer is not called.
6. **G3, re-validation.** The merged assignment is built on the head with a well-formedness check. If that check fails, the change is refused as ill formed.
7. **Gate, on every path, including G1.** The candidate and the diff from the current head are handed to the gate. A failing checker refuses the change and names the checker.
8. **Accept.** Proposer resolutions are copied into the reuse store only now. The new head is snapshotted. The receipt records the gears used, the gate evidence, the resolutions, and `est_uj`.

Refusals are logged too. The head does not move.

6. Propositions the tests hold

At `aba99fce`, `cargo test --offline` runs 49 tests (`29` core, `8` keys, `12` git), mapping the propositions below:

Claim	Test named in the repository
VC1, totality, no limbo: every submission is accepted or refused with a reason, and logged	`vc1_every_submission_is_accepted_or_refused_and_logged`
VC2, green by induction: every head is well formed and passes the gate	`vc2_every_head_is_well_formed_and_passes_the_gate` (400 random submissions, stale bases, poisoned changes)
VC3, commutation implies the same result in either order	`vc3_commutation_implies_order_independence` (3,000 random trials, more than 500 commuting pairs) and `vc3_disjoint_writers_integrate_at_g1_in_either_order`
VC4, a semantic conflict is caught when the text does not conflict	`vc4_identical_edits_merge_and_semantic_conflicts_are_caught_at_g3` (an arity change against a new caller)
VC5, a resolved conflict is reused by its name-free signature	`vc5_resolved_conflicts_are_reused_by_name_free_signature`
A model result is a candidate	`model_output_is_a_candidate_never_trusted`
The state is the fold of the log	`replay_reproduces_head_without_any_proposer`

Claim	Test named in the repository
A rename changes no key, and name-only is checked	`renames_change_no_keys_and_name_only_is_verified`
G0 idempotent retries cost one lookup	`g0_identical_retry_costs_one_lookup`
Scoped gate agrees with a full gate	`delta_gate_agrees_with_full_gate` (600 trials, >100 refusals)
Snapshots untouched by later edits; incremental matches from-scratch	`snapshots_share_structure_and_diff_is_exact`, HAMT tests in `hamt.rs`
SHA-256 matches the published vectors	FIPS 180-4 vectors in `hash.rs`

VC5's test accepts a conflict through the proposer, then submits the same conflict under another slot name. The second acceptance takes the G0 lookup gear, does not take G4, does not call the proposer again, and has `est_uj` less than one hundredth of the receipt from the proposer path. That ratio is a comparison of stated receipt costs.

A separate test feeds the proposer a poisoned result. The gate refuses it, the head is unchanged, and the refused resolution is not stored for reuse.

7. Measured wall time

The workload, as the swarm example implements it: every writer in a round submits against that round's starting head, so later submissions in the round are stale. One writer in five edits the hot module (`below(5) == 0`). One submission in fifty carries a forbidden token (`below(50) == 0`), which is the 2% poison rate the README states. The proposer in the example is a deterministic stand-in that joins the two bodies. It is not a learned model. The example stores 10 units in each module. The fixed seed in the example is `0x5eed`.

7.1 Path shares at 400 units

The README's path table is the default swarm: 64 writers, 20 rounds, 40 modules, 400 units, 1,280 submissions. Shares are of all submissions.

Path	Share
Accepted at G1 (commuted, no merge)	57.0%
Accepted at G2, then G3 (structural merge, re-validated)	27.3%
Accepted at G2, then G4, then G3 (conflict, proposer, re-validated)	13.8%
Refused by the gate	1.9%

The four shares sum to 100. Every refusal was a poisoned change, including 6 proposer outputs that inherited the poison. 1,259 of the 1,280 submissions were on a stale base, and 98.1% of those stale submissions were admitted with no writer rebase. The example made 182 proposer calls. Replay from genesis, with no proposer, reproduced the same head.

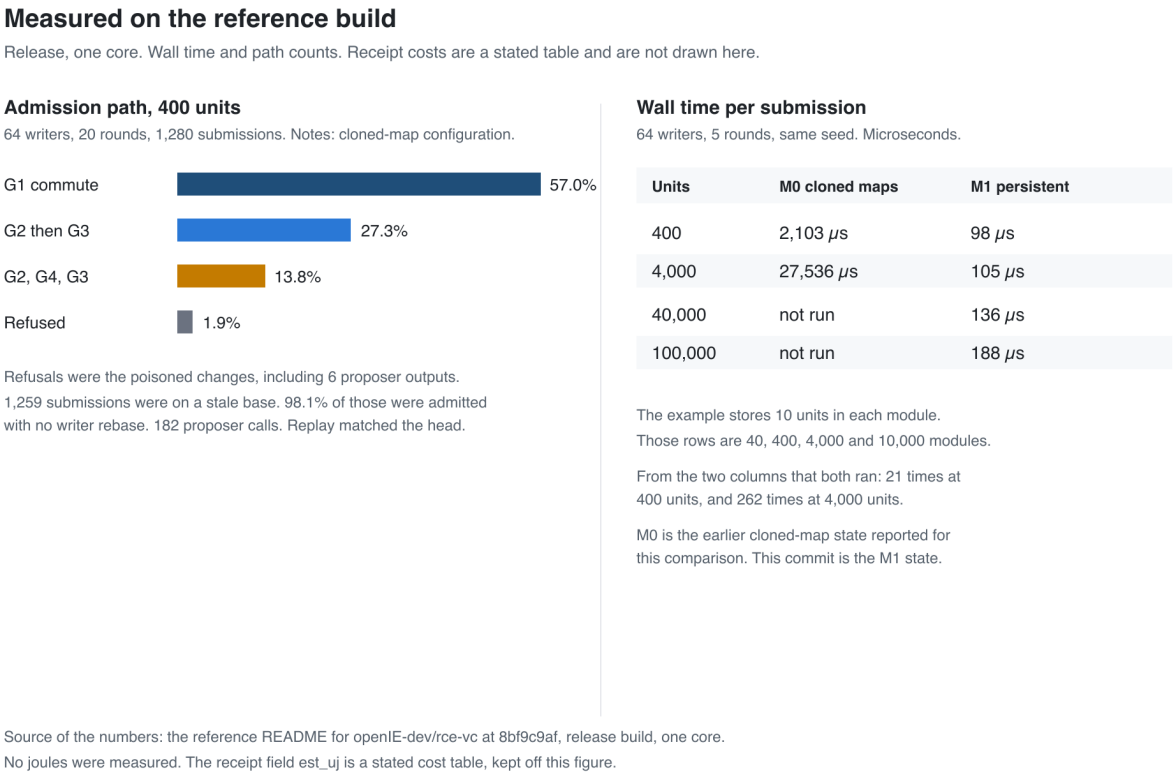

Figure 1. Measured path shares for the 400-unit, 20-round swarm, and measured wall time per submission for the 5-round timing runs on the baseline reference build. Release, one core. The stated receipt costs are not on this figure.

7.2 Integration cost against repository size

The README's timing table is the same shape of workload with 64 writers and 5 rounds, same seed, release build, one core. The cell is measured wall time per submission. M0 is the earlier state that cloned maps. M1 is the persistent Merkle state.

Units	Modules	M0, cloned maps	M1, persistent state (README host)	Ratio printed in the README
400	40	2,103 µs	98 µs	21 times
4,000	400	27,536 µs	105 µs	262 times
40,000	4,000	not run	136 µs	
100,000	10,000	not run	188 µs	

The 100,000-unit row is the example invoked as `swarm -- 64 5 10000`: 64 writers, 5 rounds, 10,000 modules. M0 grew with repository size and M1 stayed near flat, with the remaining rise coming from trie depth and from how much moved inside a round, because snapshots share structure, diff and the gate skip equal subtrees, an edit re-keys only edited units and their dependents, and the gate re-checks the declared scope of the delta.

7.3 Measured wall time and on-chip `reported_j` at 100,000 units (commit `aba99fce`)

On 22 November 2026, `openIE-dev/rce-vc` at commit `aba99fce` (`rustc 1.100.0`, release build) was executed and instrumented on Apple M5 Max (`macOS 27.2.1 arm64`) under the dual-channel SATB `reported_j` harness: whole-chip `IOReport` via `macmon` (L2, label `A2`, Falsifier 6 `<= 5.0%` idle-drift gate) paired with per-PID kernel `proc_pid_rusage(RUSAGE_INFO_V6).ri_energy_nj` (L3, label `D`), at `100,000` units (`10,000` modules × `10` units):

100,000-unit workload (aba99fce, Apple M5 Max)	Admissions / trial	Trials	Median wall time / admission	Median L3 reported_j / admission	Instructions / admission	Stated CostTable est_uj / admission
README swarm (64 writers × 5 rounds)	320 (315 acc, 5 ref)	5	69.69 µs (63.5–84.1 µs)	365.35 µJ (346.4–395.7 µJ)	1,147,137	438,530.4 µJ
Sustained window (64 writers × 500 rounds)	32,000 (31,357 acc, 643 ref)	3	73.56 µs (69.3–75.7 µs)	317.21 µJ (278.9–369.0 µJ)	1,153,976	424,152.7 µJ

- **Genesis and path breakdown**: Building the 100,000-unit Merkle HAMT genesis state took a median 1.244 s once (5 trials). In the 32,000-submission sustained run (2.35 s window), 26,103 submissions (81.6%) admitted at G1Commute, 2,594 (8.1%) at G2Structural > G3Revalidate, 2,660 (8.3%) via G4Proposer with 2,660 conflict signatures promoted for reuse, and 643 (2.0%) poisoned submissions were refused (30,857 of 31,490 stale submissions = 98.0% admitted without writer rebase).
- **L2 vs L3 consistency and Falsifier 6**: On the two sustained trials that cleared Falsifier 6 (2.67% and 1.22% idle drift), whole-chip L2 IOReport net marginal energy was 9.22 J and 4.66 J (288.2 µJ and 145.6 µJ/admission) beside per-PID L3 ri_energy_nj of 11.81 J and 8.92 J (369.0 µJ and 278.9 µJ/admission), while the middle trial (7.28% > 5.0% idle drift from concurrent host jobs) had its L2 net subtraction automatically invalidated by Falsifier 6. Notice that the placeholder CostTable (424–439 mJ/admission) overstates CPU-only admission by roughly three orders of magnitude because its 5,000,000 µJ (5 J) Proposer entry anticipates a neural proposer rather than the example's deterministic stand-in (measured_j = None, board_metered = false).

8. Stated cost on receipts

The default table in CostTable at aba99fce, in microjoules, remains a stated price per row inside repository receipts:

Row	What adds it	Stated µJ
G0	Charged at the start of every attempt. Charged again for each reused conflict signature. A memoised refusal is this row alone.	1
G1	Charged when the attempt reaches the commutation test	10
G2	Charged when the attempt performs the structural merge	100
Checker	Charged once per checker, on every path that reaches the gate, including G1	1,000
Proposer	Charged once per proposer call	5,000,000

G3 is written on the path and has no row of its own. A receipt's est_uj is the sum of the rows that attempt added. Replacing this table with hardware-rail measured_j is milestone M6 and remains open; Section 7.3 provides the first operating-system reported_j calibration (317–365 µJ per 100,000-unit admission with a deterministic stand-in proposer).

9. The Git projection

The projection is milestone M2, in rcevc-proj-git. A leaf unit is the file bytes. A unit with dependencies is written as a file with a short header, which the reference treats as a bijection on UTF-8 files:

```
#rcevc-unit v1
#dep <slot>
#body
<body>
```

One #dep line is written for each dependency.

Out. export writes one commit on main for each accepted entry. The author is the writer label. The subject is the change message, or a line naming the change identifier and the gears if the message is empty. Each accepted commit carries the trailers RCE-VC-Change, RCE-VC-Pre, RCE-VC-Post, RCE-VC-Path, RCE-VC-Evidence, RCE-VC-Resolved, RCE-VC-Est-uJ and RCE-VC-Entry. RCE-VC-Est-uJ is the stated microjoule sum, not a measurement. The ref refs/rcevc/log has one commit per log entry, refusals included, and a refusal commit also carries RCE-VC-Reason. Given the same log and the same time base, export produces the same commit identifiers.

In. `ingest` walks parent commits from a tip until it finds a commit whose `RCE-VC-Post` trailer is a snapshot. That snapshot is the base. The tip tree becomes one change, put and delete, against that base. If the content multiset is unchanged, the level is name-only. The branch is the proposal. The ladder decides. The writer is not asked to rebase.

The README records these checks against git 2.43, in `tests/git_interop.rs` (12 tests in `rcevc-proj-git` at `aba99fce`):

Check	Result recorded in the README
`git fsck --full --strict` on the exported history	clean
Every commit on `main` re-imports to the root in its `RCE-VC-Post` trailer	yes
Two exports of the same log in different directories	identical commit identifiers
Git's trailer parser reads the receipts	yes
Two plain Git commits on a branch whose base has gone stale	ingested as one change, admitted at G1, `main` moves forward
Objects Git itself wrote, including dynamic-Huffman deflate	read
A plain Git commit that contains a forbidden token	refused, recorded on `refs/rcevc/log`, `main` unmoved

Limits of this projection, as the README states them: loose objects and loose refs only; file mode 100644 on the way out, so the executable bit is not yet a unit attribute; symlinks, submodules and non-UTF-8 files are refused with a reason rather than guessed.

The project notes name Mergiraf, a structured merge that uses tree-sitter, as work aimed at parse-shaped files. That is a convergence observation. This page does not score the reference against it, and it does not score the reference against any other version-control system.

10. Limits of this version

At commit `aba99fce`:

1. `rcevc-keys` splits Rust (`.rs`) and Python (`.py`) files into top-level items (with dependencies pointing only at earlier items in the same file), but TypeScript splitting and a 50-package / 1,000-repository round-trip evaluation are not yet done; other file types remain whole-file leaf units.
2. Git storage is loose objects and loose refs. There are no packfiles and no packed refs. Export writes mode 100644 only. Symlinks, submodules and non-UTF-8 files are refused.
3. Each exported commit rebuilds the whole tree.
4. The reverse-dependency sets are cloned when they are updated. A unit with many dependents makes that update large.
5. Key computation is recursive. A dependency chain of more than about 10,000 units could overflow the stack.
6. One full-content root is linear in the number of units. Ingest uses it only to guess the name-only level. The kernel's name-only check walks the diff.
7. Snapshots are kept in memory, with no retention policy and no separate store.
8. `submit` is single-threaded (`CasRef` guards ref swaps, while `submit` admits commuting changes sequentially). Leases are not built.
9. A checker is trusted to tell the truth about its scope (Wasmtime sandboxing M3 is not yet built).
10. On-chip `reported_j` (`317-365 µJ` per `100,000`-unit admission) has been recorded externally via kernel `ri_energy_nj`, while in-repo receipt fields still carry the stated `est_uj` table and external hardware-rail `measured_j` (M6) remains open (`measured_j = None`, `board_metered = false`).

11. Next work

- **Complete M1 (`rcevc-keys` expansion and corpus round trip).** Add the TypeScript grammar alongside Rust and Python, support cross-file item dependencies, and verify byte-identical round trips across at least 50 real crates and packages with independent edits to separate functions of one file commuting at G1.

- **Complete M2 (BLAKE3 internal keys and `gix` packfiles).** Move internal hashing to BLAKE3 (keeping FIPS 180-4 SHA-256 vectors as their own test and bumping domain tags) and move Git object storage to `gix` (`gitoxide`) for packfiles and packed refs, followed by a 1,000-repository history round trip.
- **M3 (`rcevc-gate` on Wasmtime).** Run checkers as fuel-limited, deterministic WebAssembly modules with pinned digests so declared scope is enforced by exposing only the declared slice.
- **M4 (leases and monotone channel).** Concurrent streaming for monotone changes without coordination.
- **M5 (conflict corpora replay).** Replay Merge-Bench, CONGRA and AgenticFlict through the ladder, counting resolutions at G0, G1, G2, G3 and G4.
- **M6 (`rcevc-meter`).** Replace `est_uj` with calibrated `reported_j` and external hardware-rail `measured_j`.

12. Copyright and license

Copyright 2026 Open Interface Engineering, Inc.

The reference implementation in openIE-dev/rce-vc at commit `aba99fcec4414a32bff30ab075668350d3fb80bf` (`13 November 2026`) is under the Apache License, Version 2.0. The license file in that tree states the copyright as Open Interface Engineering, Inc., and the crate manifest sets `license = "Apache-2.0"`. This page is the study text for that reference.