



RUNESMITH

Beyond the Model

Technical Report. Runesmith, an Open Runtime That Improves Software and Itself, and the Instrument-Substrate Hypothesis

Lars O. Horpestad

AI ThinkLab

6 October 2026

Preprint · not peer reviewed

Supported by  Innovation
Norway

Contents

Abstract	4
1. Introduction	5
1.1 Intelligence that vanishes at the end of the call	5
1.2 The instrument–substrate hypothesis	6
1.3 Capability transduction	6
1.4 What Runesmith has shown	7
1.5 Contributions	8
1.6 The boundaries of the claims	9
1.7 Provenance of the work and of this paper	9
2. In what sense Runesmith is a model	10
3. Runesmith	12
3.1 Principle: broad proposal, narrow promotion	12
3.2 Formal runtime model	13
3.3 Reference architecture versus released system	15
3.4 The Runesmith Core	15
3.5 The Studio and delegated builds	17
3.6 Evidence language and statistical conventions	17
4. Measures	18
4.1 Which measures the evidence uses	20
5. Evidence	21
5.0 Overview	21
5.1 Object improvement	22
5.2 Subject improvement	30
5.3 Null and negative results	44
5.4 Supervised field construction and out-of-the-box journeys	54
5.5 Case study: Runesmith Motion	59
6. Discussion	66
6.1 Where the instrument–substrate hypothesis stands	66
6.2 A nudge towards recursive self-improvement	67
6.3 Separate measured edges, not one chain	68
6.4 Inference independence and digital sovereignty	68
6.5 The uplift question	69
6.6 Relation to prior work	70
6.7 Reading the negatives	71
7. Limitations	72
7.1 Limits specific to the results reported here	74
7.2 Limits of the record and of this paper	75
7.3 What the architecture does not guarantee	75
8. The release	75
8.1 What is released	75
8.2 System card	76
8.3 Test evidence, as logged	77
8.4 Reproducibility	77
9. Future work	77
10. Conclusion	79
Acknowledgements	80
References	81
Appendix A. Public claim map	85
Appendix B. Result-dependent naming rule	87
Appendix C. Glossary and label collisions	88
Appendix D. Evidence index	89

Version 1.0, 6 October 2026, released with Runesmith 1.0. Preprint; not peer reviewed. DOI: [10.5281/zenodo.23196763](https://doi.org/10.5281/zenodo.23196763) (paper, technical report and evidence).

© 2026 AI ThinkLab. This paper is licensed under the Creative Commons Attribution 4.0 International License (CC BY 4.0, <https://creativecommons.org/licenses/by/4.0/>). The Runesmith software it describes is licensed under the Apache License 2.0.

Companion to the paper. This technical report accompanies the paper *Beyond the Model* and stands as written before LOC1, the sealed test of 2026-10-06 reported in the paper’s §4: there, on three large public repositories, C7 repaired more tasks than the shipped organ g0 (42 against 17 of 168 sessions, exact one-sided $p = 0.00278$, PASS at its Bonferroni level), and a hand-written trace-aware model-free localization arm repaired more than C7 (65 of 168). Where this report says that C7 has not been shown better than g0, or that no such arm has run, it describes the record before LOC1; the paper is current.

Source of every statement. Every statement about the research is drawn from the project’s research record (below, “the record”), a dated laboratory record whose corpus cut is 2026-09-25, with dated, signed extensions to 2026-10-01T16:53Z (4,239 lines at that cut, SHA-256 `e40c3d0bffe52470874148a8243f5447e89b988f6f2392ff60c6b8556d0580a8`). An insertion-only extension of 2026-10-05 records the release week (current SHA-256 `4542e818cc34b9afb7f3487e64d3d34888ae941affe3b47208dbbbbd21ae66d4`); removing its inserted blocks reproduces the cited version byte for byte. Figures that postdate the record were read from the project’s own files and carry their read time; the release-week results of §5.4.4 (J0’s phase 4 and J13 after 09:01Z), the benchmark figures of [P-07], the references P-01 to P-07, and the sealed studies run after the record’s cut (U1, U2 and U2b, §5.2.6; SI, §4.1) are not in the record and cite their own sources. The record is the paper’s supplement, and the evidence released with the paper includes SR7’s raw session outcomes and the script that recomputes its test (§8.1).

Citation key.

- [R §8.32; M-0118] : the record, section 8.32; M-0118 is the row of the reconciled claims database (1,309 rows; supplement) that extracts the statement. Record line numbers are never cited, because the record is versioned.
- [J11 log, 2026-10-04T21:35:16Z] : the dated entry of the Runesmith Motion journey log.
- [count, 2026-10-05T01:05Z] : a count made for this paper from the project’s files at that time.
- [numbers check, 2026-10-04] : the release’s separate audit of figures against the record and the files.
- [Core source, read 2026-10-05], [Core DEFAULT_SETTINGS ...], [Core library metadata ...] : the released repository’s code or data, read for this paper on 2026-10-05. [SR7 receipts, read 2026-10-05] and [PROTOCOL.json receipt via M-0860] : a study’s archived receipt files, read directly or through the claims database row that extracts them.
- [SR-001] : a reference in the reference list, keyed by the record’s source-registry identifier. Section signs without “R” (for example §5.2) refer to this paper.

Experiment identifiers from the earlier phases of the project carry the prefix `TRON`, one of the project’s earlier internal names. They are kept unchanged because they name receipts [R §front matter, terminology].

Abstract

Agent systems built on language models rent their competence one call at a time: when the call ends, the competence leaves with it. We present Runesmith, an open-source runtime (Apache-2.0) that builds and improves software, and improves its own code, with whatever inference is available: free hosted tiers, paid providers, or any OpenAI-compatible address, including local servers (tested so far through stand-ins). Its predecessor also supervised a live service's operations. Models propose; a deterministic kernel admits; self-modifiable code runs as confined organs; and a changed generation becomes active only by winning an online trial or by the owner's recorded choice. From a dated research record we report every sealed result, positive and negative.

Runesmith improved on its own previous generation under a sealed, preregistered test. Its Kaizen loop chose a target in its repair organ from its own telemetry, had a model rewrite the organ, and froze the result. With the same cheap repair model and an identical budget, the new generation repaired 57 of 162 sessions against 35 of 162 for the generation it replaced, on 54 fresh tasks, none opened before, from two repositories that supplied no task to the organ's development, in about half the time per success (exact one-sided $p = 0.000845$; within the larger repository alone, $p = 0.0020$). That predecessor, an earlier generation from the same loop, had not beaten the shipped organ, and the sealed test did not compare the new generation with the shipped one. The model that wrote the change, Claude Opus 4.5, scores below today's free-tier Gemini Flash models and the open-weights Kimi K3 on a public coding benchmark, so, by that benchmark, the author was not stronger than inference available free at release; the earlier free-class author that found no transferring change (SR5, SR6) was a different open model, gpt-oss-120b, and a sealed reproduction with one of today's free tiers as author is still to be run (§6.2). A second self-improvement, a free-model-authored change to Runesmith's memory retrieval on an operator-chosen target, was replicated on a fresh cohort (15 of 16 blocks, $p = 0.00026$). A frozen generation repaired its own test configuration from a failing suite with no operator-named target. Its predecessor kernel promoted target-hidden corrections on revisions of a production service and supervised that service's campaigns across fresh-process restarts, continuing without observer intervention after its start. Free-tier models were the last author of every applied change to Runesmith Motion, a program whose engine (`motion.mjs`), as it stood at its 52nd milestone, computed the graphics of every frame of this paper's launch film (the operator's pipeline inserted the recorded talk; a browser rasterised and a general-purpose encoder joined the frames): 54 milestones under an AI operator's direction, no call with a positive recorded cost, and, by the lead investigator's definition (one milestone through to the next without interference), unattended operation on two occasions. In the release's acceptance test (an AI in a newcomer's seat, build `f4d973d`), one free Google key took a project from a planned feature to a checked, working one in 16 minutes. On the build just before 1.0.0 (identical but for its version strings), with free models as both repair model and improver, Runesmith's scheduled repair rounds repaired 69 of 89 single-line regressions injected one at a time into a project with a real test suite, each accepted by a held-out judge (one project, observational, no fixed-baseline arm, no activation claimed; §5.4.4). Starved of every model in 104 sealed scenarios, the released

runtime made no unauthorized change and never broke its ledger chain; release 1.0.0 adds ledger events for the 15 effects that study found recorded without their own ledger event (not re-measured).

Runesmith is a runtime that changes its own code only on evidence, and it has passed a sealed test of doing so: Runesmith improved itself, over its own previous generation. We call SR7 a nudge towards recursive self-improvement: the system's own loop rewrote part of the system, a sealed test confirmed the new version better than the one it replaced, and the loop itself was not changed. We frame the program with the instrument-substrate hypothesis: competence first supplied by a model can be kept as tested runtime code and go on improving later work after that model is gone. One of its six measures, starvation integrity, has been estimated (§4.1), and in our reading SR7 is the first sealed downstream evidence for its mechanism; its other formal measures, and an equal-budget comparison with a model used without Runesmith, are the next studies. Six of the seven sealed tests in the September series did not show their effect; we report all of them, and they shaped the loop that then passed: its held-out validation and the affordance audit in its author's packet came from them (§5.2.5, §5.3.12).

1. Introduction

1.1 Intelligence that vanishes at the end of the call

An agent built on a language model rents its cognition one call at a time. A capable model can plan, write code and repair it inside one episode, but when the episode ends the system is left with a transcript, some files and perhaps a summary. It is then unclear how much of the episode's competence survives the removal or replacement of the model that produced it [R §Abstract; M-0209]. The question this program asks is narrower than "is the system intelligent": **where does the competence of the resulting system reside?** [R §1.1; M-0882]

The record states three observations that would show that the surrounding system is support infrastructure rather than a retained capability substrate: the task cannot be completed after the model is replaced; a remembered success must be semantically reconstructed by the original model; a stored tool works only because that model silently repairs its underspecified contract [R §1.1; M-0882]. Prior work establishes many of the mechanisms such a system would use: retrieval [SR-001], reasoning and acting with tools [SR-002, SR-007], verbal self-reflection [SR-003], executable skill libraries [SR-027], scaffold self-modification [SR-051] and evaluator-guided evolution of coding agents [SR-052]. These works narrow the opening rather than erase it. None of them tests the particular conjunction this program proposes: separately custodied evidence, scoped authority and revocation, complete accounting of assigned opportunities, and promotion only at a preregistered endpoint. Whether that conjunction improves utility over a carefully composed conventional stack has not been measured [R §1.1; M-0883].

1.2 The instrument–substrate hypothesis

The program treats models, humans, rules, solvers, search procedures and tools as replaceable **cognitive instruments**, and treats evidence-bearing runtime state as the candidate **capability substrate** [R §Abstract; §2.1–2.2]. Its central hypothesis, stated in the record’s words, is:

“Some useful competence can reside in evidence-bearing runtime state rather than exclusively in model weights. A capable instrument may discover a solution; the runtime may preserve the useful part as a tested procedure, policy, memory, capability, or evaluator; and a weaker instrument—or no generative instrument—may later reuse it within a validated scope.” [R §1.2; M-0885]

The record attaches two guards. The hypothesis is “not an assertion that the substrate contains all intelligence”, and it is not “a prediction of model parity”. Its risky prediction is that “for at least some prespecified artifact, task, and substitute-instrument conditions, genuinely transduced competence should retain positive useful effect after the originating instrument is removed”; partial retention, $0 < SR < 1$ in the notation of §4, “is one testable pattern, not a universal range or a claimed result” [R §1.2].

The hypothesis is built to fail cleanly. The record lists what each failure would mean: no adaptive gain means no adaptive-controller claim; no state carryover means no persistent capability; no substitution retention means no instrument-independent substrate; no equal-budget weak-model uplift means no amplification; no integrity under model starvation means the kernel was model-dependent after all; nonpositive full-cost utility defeats a net-benefit claim without erasing a separately recorded correctness or resource gain [R §15; M-0774]. A program that keeps meeting adequate nulls must narrow or pause; “indefinitely selecting new endpoints does not confirm the original hypothesis” [R §12.7; M-0723].

Status. The hypothesis is the program’s organizing conjecture: this paper establishes one sealed result that is, in our reading, an instance of its mechanism, and one estimate of its measures, and the hypothesis’s own test, with fresh, shuffled and revoked state and named substitutes, comes next (§9). The instance, in its plain form: in SR7, an organ written by one model through Runesmith’s own improvement loop, used by a different and cheaper repair model after its author was gone, repaired more fresh tasks than its predecessor generation (§5.2.5; our reading, §6.1). The estimate is starvation integrity (SI), measured once on a designed grid (§4.1). The other formal measures have not yet been estimated to their full definitions: the one state-carryover measurement was inconclusive (§5.3.8) [R §11; §10, card M; M-1262], no named-model substitution matrix has been run, and the one preregistered weak-model amplification test was null (§5.3.7).

1.3 Capability transduction

The mechanism the hypothesis needs is **capability transduction**: “a transformation from one or more successful episodes into a scoped artifact whose future causal contribution is positive on eligible held-out tasks” [R §2.5]. The record’s pipeline is episode → typed candidate → sandboxed enactment → separate evaluation → scoped artifact → future held-out utility [R §1.3; M-0886]. Its counting rule excludes most of what agent systems usually count:

“Saving prose, storing a chain of thought, passing a self-generated test, compiling code, or increasing a judge score does not complete the transduction. An artifact counts as a retained capability only when it is versioned, has explicit applicability and failure conditions, survives independent hidden tests, produces positive future utility on eligible held-out tasks, and can be revoked when its warrant expires.” [R §1.3; M-0224]

(The word “independent” in this quotation is the record’s; §1.7 states how this paper uses it.)

1.4 What Runesmith has shown

The record holds four positive lineages, which “remain separate unless a prospective experiment causally links them” [R §How to read; M-0881], and the release week adds field evidence. In order of strength:

1. **Sealed self-improvement (SR7).** Runesmith’s Kaizen loop chose a target in its own repair organ by a declared rule from its own telemetry, had a model (Claude Opus 4.5) rewrite the organ, and froze the successor, C7. In a sealed, preregistered confirmation on 54 fresh tasks, none opened before, from two repositories that supplied no task to the organ’s development, with the same cheap repair model (gpt-oss-20b) and an identical budget, C7 repaired 57 of 162 sessions against 35 of 162 for its predecessor B (exact one-sided $p = 0.000845$; 18 tasks better, 3 worse, 33 tied), in about half the time per strict success, with organ errors and false promotions equal. The effect holds within the larger repository alone ($p = 0.0020$; our recomputation from the released session table). B was itself an earlier Kaizen generation that had not beaten the shipped organ g0, so SR7 shows the loop improving on its own last generation (§5.2.5, §5.2.6) [R §8.32; M-0118].
2. **Replicated self-improvement (M1).** A free-model-authored change to Runesmith’s memory-retrieval code made a frozen successor faster than its frozen predecessor in a sealed replication on fresh workloads (15 of 16 blocks; median search CPU 33.1% lower; exact $p = 0.0002594$), on a target the operator chose and after a first candidate failed a preregistered guard (§5.2.4) [R §8.19.10–8.19.11].
3. **Self-correction (S1).** A frozen Runesmith generation turned a supplied failing-suite receipt into a repair of its own test configuration with no operator-named target or repair; a free model wrote the one-line change and a hidden evaluator passed it. The remedy was a known flag (§5.2.3) [R §8.19.4; §8.19.12].
4. **Object improvement on a production service.** With the predecessor kernel, on successive revisions of PRHat (the automated public-relations service of Veristria, a Norwegian security company, used as the object system; the identifiers that carry its name, the `TRON-PRHAT-...` experiment ids and the `prhat.*` contract names, are immutable experiment and contract ids kept unchanged for provenance), the runtime selected and reproduced a capability gap, obtained a confined correction from a replaceable free-model instrument, and promoted it only after a target assertion withheld from the correction author passed. Nine terminal episodes on owner-specified contracts produced two strict promotions, after an earlier target-hidden promotion that brought the service’s suite from 896 to 897 of 897 (§5.1.7–5.1.8) [R §8.17; §8.18; M-1092].

5. **Operation of a live service.** The predecessor kernel ran as PRHat’s resident supervisor: it supervised the service’s campaigns, scheduled replies and research actions, recovered across fresh processes and classified failures, and after its start continued without observer intervention (§5.4.3) [R §8.15].
6. **Field evidence on the release build.** Free-tier models were the last author of every applied change to Runesmith Motion (54 milestones under an AI operator’s direction; no call with a positive recorded cost), whose engine, at its 52nd milestone, computed the graphics of the release’s launch film, and which ran unattended by the lead investigator’s definition on two occasions (§5.5); the release’s acceptance test, run by an AI in a newcomer’s seat, reached a checked, working feature with one free Google key in 16 minutes; and on the build just before 1.0.0, with free models as both repair model and improver, the scheduled repair rounds repaired 69 of 89 injected single-line regressions, as the held-out judge accepted them, while Kaizen ran live: five campaigns ended without a candidate that passed its freeze bar, and a sixth produced one, which entered an online trial (§5.4.4).
7. **Integrity without a model.** In a sealed study of 104 designed scenarios with no model configured or every model failing, no request reached a model, no change was unauthorized and every ledger chain verified; the 15 effects recorded without their own ledger event are fixed in release 1.0.0 (408e4b8), not re-measured (§4.1).

The negatives are part of the result, and evidence of the method. Six of the seven sealed tests of the September series did not show their effect, a preregistered object confirmation did not confirm, a run of downstream subject campaigns produced no qualifying successor, and the record’s own red-team reduced early synthetic positives to mechanism checks (§5.3). They changed the design: the loop that passed SR7 scored its candidates on held-out validation because of SR4 and gave its author the affordance audit because the free-class authors kept repeating one mechanism, and the released loop also requires a candidate’s gain to exceed the spread between two fresh replays of the incumbent, a rule SR5’s replay noise prompted (§3.4, §5.3.12).

1.5 Contributions

1. **Sealed evidence that a runtime can improve itself:** a preregistered downstream result over the runtime’s previous generation, confirmed on repositories that supplied no task to its development (SR7), a replicated self-improvement of a second capability (M1) and a self-correction (S1), each with its envelope (§5.2).
2. **Runesmith 1.0, open source under Apache-2.0:** the Core and the Studio, in which models propose, a deterministic kernel admits, self-modifiable code runs as confined organs, versions are frozen digest-bound generations, and a changed generation becomes active only by winning an online trial on the owner’s own work or by the owner’s recorded choice; it runs on any OpenAI-compatible inference, including free tiers and local servers (§3, §8).
3. **Field evidence of building, improving and operating software:** target-hidden object improvement on a production service (§5.1), operation of that service (§5.4.3), a program for which free-tier models were the last author of every applied change, whose engine computed the graphics of this paper’s launch film, and which twice ran unattended by the lead investigator’s definition (§5.5), and an acceptance test on one free key (§5.4.4).

4. **The instrument–substrate hypothesis** with its falsifiers, a formal runtime model and six measures (AG, SCG, SR, WMA, SI, CTY), each tied to a counterfactual and a failure condition, and a first sealed downstream instance of the hypothesis’s mechanism (§1.2–1.3, §3.2, §4, §6.1).
5. **A complete negative record**, reported with the same method depth as the positives, and a **claim discipline** (dated endpoints, result-dependent names, errata that override, “independent” reserved for distinct investigators) that makes every positive checkable (§5.3, §1.7, Appendix B).
6. **Released evidence**: the claims database, the sealed tests’ summaries, and SR7’s 324 session outcomes with the script that recomputes its exact test (§8.1).

The novelty is the evidence-to-promotion contract working end to end: a runtime that changes its own code only on evidence, and that has passed a sealed test of doing so, in which Runesmith improved on its own previous generation (§5.2.5). Each component has precedents (§6.6); the utility of the whole over a carefully composed conventional stack is the question of the uplift study (§6.5) [R §1.4; M-0889].

1.6 The boundaries of the claims

The results above are claimed at full strength inside their envelopes, and only there. Release 1 does not learn across users: no mechanism moves experience or generations between installations except by hand (§2). No study yet shows that Runesmith makes a given model better than it is alone (§6.5), that a project’s competence survives a named change of model (§6.1), that the improvement loop has improved itself, that improvement compounds over generations (§6.2), or that it amplifies a weak model at equal budget (§5.3.7). The record also rejects, as present results, improved business outcomes, cost reduction through routing, general production DevOps, general autonomy, guaranteed safety or corrigibility, economic value and a moat [R §1.5; M-0890, M-0225–M-0231]. No rung of the record’s naming rule licenses “general intelligence”, “safe autonomous system”, “recursive self-improver”, “world-class autonomous DevOps” or “commercial moat” [R App. B], and this paper uses none of them. The limits of each positive result are collected in §7.1 (SR7’s full list: §5.2.5).

1.7 Provenance of the work and of this paper

The program ran from 2026-09-03 (the first research file) to the record’s cuts of 2026-09-25 and 2026-10-01 [numbers check, 2026-10-04]. It was led by one investigator with AI collaborators in named roles (Acknowledgements). An AI collaborator (Claude Opus 5.5) built much of the experimental machinery of the September 23–25 studies, operated the out-of-the-box journeys, and wrote the record’s later errata and extensions, which it signed as “a project agent and not an outside investigator” [R §front matter erratum; §8.19.13; §8.37]. The record’s synthesis was “internally accepted”; it has not had external scholarly peer review [R §front matter, root synthesis status].

“Independent”. The record’s governing rule is: “Process-isolated and fresh-seed are named as such; ‘independent’ is reserved for distinct investigators or institutions” [R §2.15 erratum]. Where the record says “independent recomputation”, it means a second read-only script written

inside the project from the same raw receipts, “not a rerun by a distinct investigator or institution” [R §2.15 erratum]. This paper calls it a *same-project recomputation*, calls separately launched processes *separate*, and uses “independent” only where distinct investigators are meant. No result in this paper has been rerun by a distinct investigator.

Errata. The record contains dated errata written on 2026-09-29 and 2026-10-01. An erratum overrides the sentence it corrects, and this paper uses the corrected statements throughout; where an erratum changes how a number is read (sessions versus tasks in SR7; time per strict success rather than per repair; the inert tier in SR7; the power figures of SR1-MIN), the paper says so at the point of use.

How this paper was made. Four extraction packs were made from the record by AI agents, each carrying exact numbers, section references and claim ids, and flagging where the record is weaker than a sentence in it sounds. Claude Opus 5.5, the AI collaborator that also built much of the machinery of the SR studies and wrote the record’s later errata, drafted the paper from those packs under the lead investigator’s direction, consulting the record directly where a pack was unclear; agents checked its numbers and references, and pre-release reviews by Grok, Gemini and Claude models, obtained by the lead investigator, were answered before release. None of these is a review by a distinct investigator in the sense above. Every number below is copied from the record, a pack quoting it or a dated count, or is a derivation marked as such where it is used (our arithmetic: the sum of component differences in §5.2.5, the share of components in one repository, the smallest attainable exact p in §5.3.3, the multiplicity bound in §5.0, the relative gain in §5.2.5 and a unit conversion in §7.1; our recomputation from released receipts: the per-repository tests and solved-task counts of §5.2.5). Denominators are kept. The record’s compression rule governs what may be left out: “compression must not erase failed candidates or move a dated endpoint after seeing its result” [R §How to read; M-0879]. The dated chronicle, receipt digests and repetitions remain in the supplement.

2. In what sense Runesmith is a model

The record itself says that “model” remains appropriate for a predictive component inside the system but is “insufficient for the whole construct”, whose scientific object is “the closed runtime-and-environment loop” [R §2.16; M-0257]. The sense defined in this section is therefore this paper’s interpretation, assembled from the record’s definitions of the substrate (R §2.2), the four update families (R §2.7), portable identity (R §6.13) and the released Core’s organs and generations (R §8.33). We use it because it says precisely what is fixed, what persists, what changes and by which route.

Definition. In this paper, Runesmith is a model in the following sense: *a fixed policy plus persistent, evidence-bearing state that maps a goal, an environment and any available inference to verified changes*. Formally, let G_k be the active generation, S_t the persistent state of one project home, g the owner’s goal (objective, plan, milestones and approved acceptance checks), $\mathcal{E}_t$ the environment (files, tests, repositories) and $\mathcal{I}_t$ the set of inference instruments available at time t , which may be empty. One step is

$$(\Delta_t, S_{t+1}) = F_{G_k}(g, \mathcal{E}_t, S_t, \mathcal{I}_t),$$

where Δ_t is the set of changes applied to the environment, each of which passed the checks that the state requires before application, and S_{t+1} extends S_t by appended ledger events, maps, memory of attempts and experience records.

- **The policy is fixed within a generation.** A generation binds the organ files (the self-modifiable code, such as the repair organ), the kernel snapshot and a fixed configuration; it is identified by digests and is immutable once frozen [R §8.19.13, condition 1; §8.33]. The record's frozen subject capsule writes the same object as $G = (P, M_0, U, E, Dep)$: policy code and configuration, starting adaptive state, update rules, authority and resource interface, and dependencies [R §8.19].
- **The “parameters” are organs, contracts and policies.** They are readable code and declared configuration, not numerical weights: the organ files; the contracts that bound them (organ contract, kernel-owned envelope of calls, tokens, signal runs, bytes and seconds); the acceptance contracts of a project; and policies such as routing order, attention shares and schedules [R §8.29 design; §8.33].
- **Versions are frozen generations.** A change to an organ produces a new generation with its own digest and lineage; the previous generation stays available for rollback [R §8.33].
- **Learning happens only through evidence gates.** In the Core, a new generation arises only through the Kaizen loop: the system diagnoses its own telemetry, selects a target by a declared rule, asks an instrument for an organ change with a prediction and a falsifier, qualifies the candidate by static and confined smoke checks and by replay on held-out validation work, and freezes it. A frozen candidate is not switched on. It becomes active only by winning an online trial on fresh work (seeded assignment, one-sided Fisher exact looks with Bonferroni-spent alpha, compare-and-swap activation or rejection, “never a default promotion”) or by the owner's explicit choice. The trial is the evidence gate; the owner's choice is an override that the ledger records [R §8.33; M-1213; Core source, Studio text on self-improvement, read 2026-10-05]. The inference instruments are inputs to the map, not parameters of it: no weights are trained.
- **The four update families are kept apart.** Object updates change the attended project (Δ_t). Subject-cognition updates change how Runesmith observes, reasons, routes or acts (new organ generations). Learner updates would change how Runesmith acquires later updates; in the Core, the Kaizen rule, trial machinery and kernel are outside the self-modifiable organs, so learner updates are not made by the loop itself. Evaluator updates change the measurement process and “therefore invalidate comparisons unless independently qualified and versioned” [R §2.7].

What release 1 does and does not adapt. Within one project home, state accumulates: the ledger, the environment and self maps, memory of what was tried, experience records, and approved acceptance checks that judge every later build. That is adaptation in the record's sense, “any history-dependent change in behavior” [R §2.7]; it is not learning in the record's sense, which requires “a versioned state change caused by observations that improves predeclared future outcome utility against a matched fixed-state baseline” [R §2.7]. In the released defaults,

scheduled work, Kaizen self-improvement steps, the check autopilot and full-speed scheduling are all off until the owner turns them on [Core `DEFAULT_SETTINGS`, read 2026-10-05]. Release 1 does not learn across users: no mechanism sends experience or organ changes between installations. The only cross-installation path is a generation archive that one person exports and another imports by hand; an imported generation is frozen inactive and confined like a local one, and it becomes active only if it wins an online trial on the receiver's own work or the receiver explicitly chooses it, which the ledger records [R §8.33; M-1214; Core source, read 2026-10-05].

What the word does not carry. Calling this a model does not claim that it generalizes beyond the envelopes in §5, that a generation transfers to other model instruments (untested, §6.1), or that its behavior is a function of the policy alone: the environment and the instruments enter every step, and the record's capability-allocation equation, $\mathcal{K}_{eff}(D) = F(\mathcal{K}_{inst}, \mathcal{K}_{state}, \mathcal{K}_{tool}, \mathcal{K}_{eval}, \mathcal{K}_{env})$, assumes "no additive decomposition" among them [R §4.8; M-0282].

3. Runesmith

This section gives the formal runtime model against which the measures and experiments are defined (§3.2), separates the record's reference architecture from what was built and released (§3.3–3.5), and fixes the evidence language used in §5 (§3.6). Nothing in §3.1–3.3 is an empirical result.

3.1 Principle: broad proposal, narrow promotion

The architecture has two asymmetric regions: an authoritative deterministic kernel that owns identity, schemas, journal, reducer, assignments, maturity, authority, budgets, prepare/commit, receipts, activation, replay, rollback, revocation and export (the record's kernel list), and a broad adaptive surface of rules, solvers, local, hosted and frontier models, planners, retrieval, memory candidates, search, tests, evaluator candidates, procedures, tools and humans, joined by "typed, non-authoritative contracts" [R §9.1; M-1229]. "The kernel is small because its purpose is not semantic brilliance. It preserves the conditions under which brilliance, error, and learning can be distinguished." [R §9.1]

The hard boundary is stated as the core of the design:

"A proposal generator may not alter the sole release evaluator, hidden test identity, outcome maturity, assignment mechanism, immutable history, resource ceiling, permission state, or its own lifecycle status. This asymmetry is the core answer to the question: can a system change behavior without controlling the meaning and authority by which that change is judged?" [R §6.6; M-0303]

The record's rationale is that two other regimes fail: a system restricted to preapproved answers cannot discover much, and a system whose proposals can immediately redefine authority and evidence "cannot learn reliably" because "error contaminates its own standard of success". Broad proposal freedom with narrow, separately checked promotion "creates a third regime: high-

variance search with low-cost rejection and cumulative, reversible retention” [R §12.6; M-0722]. A related design rule, the lead investigator’s correction recorded in §6.3 of the record, is that a model tier “is a capability expectation, never an operation or authority class”: a small local model may attempt every operation a frontier model can, through the same contracts; tiers predict competence, cost and required verification, and authority belongs to the authenticated objective, not to the proposing model [R §6.3; M-0298]. This is a design rule; it is not evidence that small models achieve equal results.

3.2 Formal runtime model

The record’s §4 is a model of a proposed runtime; the released Core implements a narrower part of it (§3.4). We give the definitions the measures and promotion rules depend on.

Tasks and outcomes. A task instance at decision time t is

$$\tau_t = (x_t, o_t, \mathcal{D}_t, \mathcal{H}_t, \mathcal{B}_t),$$

with pre-treatment context x_t , typed objective o_t , admissible action domain $\mathcal{D}_t$, outcome-maturity horizon $\mathcal{H}_t$ and resource-and-authority envelope $\mathcal{B}_t$; each task also binds an immutable unit, cluster and stratum identity, a protocol identity, an assignment probability and an eligibility digest before treatment [R §4.1; M-0269]. The authoritative result for unit i is a vector

$$\mathbf{E}_i = (\mathbf{q}_i, \mathbf{g}_i, \mathbf{c}_i, o_i, m_i)$$

of separately verified task or business outcomes $\mathbf{q}_i$, protected and integrity metrics $\mathbf{g}_i$, the complete resource vector $\mathbf{c}_i$, observability and censoring state o_i , and outcome maturity m_i ; “missing, pending, censored, invalid, disputed, and superseded components remain distinct” [R §4.1]. Selection is constraint-first: a candidate is inadmissible if any authority, integrity, regression, duplicate-effect, outcome-quality or resource bound fails; Pareto-dominated candidates are removed; only then may a declared lexicographic or epsilon-constraint preference choose [R §4.1; §2.14]. When one aggregate estimand is needed, the scalar projection is

$$J(Z, g, D) = \mathbb{E}_{\tau \sim D} [q(y_\tau) - \lambda_c C_\tau - \lambda_l L_\tau - \lambda_h H_\tau - \lambda_r R_\tau],$$

subject to the protected-outcome, authority, privacy, integrity and resource constraints, where C is full machine and provider cost, L includes latency tails, H is operator burden and R is expected regression, incident and recovery cost. J is “a transparent analysis or tie-breaking projection, never the stored truth signal or a global dopamine-like reward” [R §4.1; M-0929]. The record assigns no numerical values to the weights λ ; no result in §5 is a net-utility estimate in this sense.

Authoritative state. The runtime’s state is

$$Z_t = (E_t, M_t, \Pi_t, C_t, V_t, A_t, R_t, W_t, Q_t) :$$

an append-only ledger E_t ; scoped working, episodic, semantic, procedural and negative memory M_t ; a bounded controller and routing policy Π_t (immutable policy versions plus an active pointer); a typed capability and tool catalog C_t ; a versioned evaluator portfolio V_t ; authenticated authority A_t ; resource, connectivity and health state R_t ; an optional uncertain world model and factual self-manifest W_t ; and the reduced state of open and terminal episodes Q_t [R §4.2; M-0272]. Derived views, indexes, summaries and caches are rebuildable projections; “predicted or simulated state cannot overwrite observed ledger state” [R §4.2].

Bounded cognitive episodes. An episode is

$$\chi_j = (\phi_j, n_j, b_j, f_j, O_j, X_j, C_j, V_j, I_j, R_j, U_j),$$

with phase, monotone counters, remaining budget and deadline, current frontier, normalized observations, exact memory and capability exposures, candidates, evaluation results, prepared intent, receipts and unresolved failures [R §4.3; M-0273]. Its controller runs ORIENT → ACQUIRE → PROPOSE → CHECK → PREPARE → COMMIT → VERIFY → CONSOLIDATE. “Only PROPOSE normally requires a generative instrument. A loop-back is legal only when a new observation, verifier failure, invalidated frontier, or recorded tier escalation changes the context digest. The same prompt and evidence repeated is retry, not correction.” “COMPLETED describes the episode, not mature objective success.” [R §4.3]

Instruments and admission. Each eligible instrument g_t^i in $\mathcal{G}_t = \{g_t^0, \dots, g_t^k\}$ (g^0 may be deterministic) receives a bounded projection of the state and returns a candidate, $c_t^i \sim g_t^i(P_i(x_t, Z_t); b_t^i)$, a typed tuple $(id, source, inputs, artifact, pre, effects, post, cost, risk, scope, uncertainty, lineage)$ “with no authority status” [R §4.4; M-0132]. The deterministic kernel K admits and selects:

$$\mathcal{C}_t^{adm} = \{c \in \mathcal{C}_t : \text{Schema}(c) \wedge \text{Auth}(c, A_t) \wedge \text{Budget}(c, R_t) \wedge \text{Pre}(c, Z_t)\}, \quad a_t = K(x_t, Z_t, \mathcal{C}_t^{adm}) \in \mathcal{D}_t \cup \{ABSTAIN, HANDOFF\}.$$

A receipt $r_t = \text{Execute}(a_t, A_t)$ “proves only what was attempted or observed immediately”; an observation $\tilde{y}_{t+\delta} = \text{Observe}(r_t)$ becomes a mature outcome $y_{t+\Delta} = \text{Mature}(\tilde{y}_{t+\delta}; \mathcal{H}_t, V_t)$ only under the maturity contract, and only mature outcomes may update a primary adaptive claim [R §4.5; M-0935, M-0937]. “A temporal predecessor is not thereby a cause.” [R §4.5]

Learning and promotion. The first learner stores a pre-action prediction $\hat{y}_{i,k}$ and updates a sufficient statistic only after a linked mature observation,

$$\delta_{i,k} = y_{i,k} - \hat{y}_{i,k}, \quad N' = N + w_i, \quad \mu' = \mu + \frac{w_i}{N'} (y_i - \mu),$$

with an influence-capped weight w_i ; “this residual is prediction error, not a global reward”, and in randomized studies the registered intention-to-treat estimator governs the causal claim [R §4.6; M-0276]. The state update is $\hat{Z}_{t+1} = U(Z_t, x_t, a_t, r_t, y_{t+\Delta}, V_t)$ followed by a projection into

a constrained set, $Z_{t+1} = \text{Project}_S(\hat{Z}_{t+1})$. The learner “may not alter A, the primary measurement semantics in V, immutable history in E, its own assignment mechanism, or its own promotion rule” [R §4.6]. The promotion rule is:

Promote($\theta' \mid \theta$) = 1 “only if the preregistered estimand clears the smallest effect of interest, every guardrail passes, the negative control remains null, the sealed holdout remains unused for candidate selection, total resources are accounted, and rollback/replay checks succeed. An inconclusive or invalid result cannot default to promotion.” [R §4.6; M-0279]

Capability transduction, formally. For episode bundles $e_j = (x_j, c_j, a_j, r_j, y_j, V_j, G_j, Z_j)$, a transducer returns an artifact and its envelope, $T(\{e_j\}_{j \in I}) = (\alpha, \Omega_\alpha)$, where Ω_α fixes task class, preconditions, dependency versions, authority ceiling, resource bounds, known failures, expiry and revalidation rule; “T may itself use a model; its output is initially only proposed” [R §4.7; M-0280]. An artifact is a transduced capability in Ω_α only if all six conditions hold: (1) it passes separately authored hidden semantic and effect tests; (2) it remains valid under replay and restart; (3) exposure to it is logged; (4) it produces positive held-out SCG on eligible tasks relative to fresh or control state; (5) the gain survives declared regression, stale-use, poisoning and authority bounds; (6) its status can be revoked or narrowed without corrupting history. Strong-to-weak transduction additionally requires positive SCG with a named weaker instrument; strong-to-none requires execution without a generative instrument at decision time [R §4.7; M-0281].

3.3 Reference architecture versus released system

The record’s §9 specifies an “executable reference architecture”: fourteen kernel modules, thirty canonical objects each with a forbidden collapse (for example, a receipt “is not task-quality evidence”; a judge score “is not oracle truth or promotion”), seven state machines and twenty-seven deterministic invariants [R §9.2–9.6]. Among the invariants: “Models and other instruments propose; the kernel admits”; “Candidate and proposer cannot control their sole release evaluator”; “The model never owns episode phase, counters, deadline, authority, evaluator choice, outcome maturity or memory status”; “Episode completion and local postcondition success never imply mature objective success” [R §9.6]. §9 is a specification. The record does not mark which clauses are implemented, and its earlier kernel (whose complete suite passed 354/354 after the first target-hidden live promotion [R front matter; M-0001]) is a different code base from the released Core. We describe below only what the record reports as built in the open-source Core (R §8.33) and the Studio, and treat the rest of the record’s §9 as the design the work was aimed at.

3.4 The Runesmith Core

The Core (v0.1.0, dated 2026-09-24 in its changelog [numbers check, 2026-10-04]) is the system on which the SR5–SR7 studies ran, “so the system in the paper is the system that will be released” [R §8.29; §8.33]. Its components, all engineering evidence and none efficacy evidence by itself [R §8.34, row 11]:

- **Kernel.** Canonical identity; a hash-chained ledger with tamper detection; frozen, digest-bound generations with compare-and-swap activation; an instrument router for the project's free inference routes and any OpenAI-compatible endpoint (local servers such as Ollama, LM Studio or vLLM, and keyed providers), with transport-first failure classes and censoring; a kernel-owned resource envelope; and a public signal for code objects with an optional executed-line trace [R §8.33; M-1210, M-0543].
- **Confined organs.** Self-modifiable code runs as organs in an isolated interpreter (`python -I -B`) with a secret-free environment and a PEP 578 audit hook; unit tests show that reads outside the organ's directory, writes, sockets, subprocesses, directory listing and environment secrets are each refused. An internal adversarial review found and closed further gaps (process handles via `_winapi`, `gc` introspection, sub-interpreter and `ctypes` imports, output floods). The record is explicit that "PEP 578 hooks are not a hardened boundary": this is engineering, "not a security audit", and the experiment organs were non-adversarial [R §8.33; §8.34, row 11; M-1220].
- **Organs and Kaizen.** The repair organ `g0` ports the earlier v1 repair loop. Kaizen has three parts: `diagnose` (value stream, failure families and the declared target-ranking rule), `attention` (a standing self-improvement share of 15% when healthy, 25% when a problem is suspected, 40% when blocked and 20% when recovering, with exact credit carry and a debt cap) and `improve` (plan-do-study-act with prediction and falsifier, static and smoke qualification, development replay, keep-best and a log) [R §8.33; M-0145].
- **Online trials.** Seeded assignment of new opportunities, one-sided Fisher exact looks with Bonferroni-spent alpha, compare-and-swap activation or rejection, "never a default promotion", and no campaign while a trial is open [R §8.33; M-1213].
- **Lessons built in from the negatives.** A *noise-aware freeze bar* replays the incumbent twice and freezes a candidate only if its gain over the incumbent's mean exceeds both the declared bar and the spread between the two replays; with SR5's replay spread (10 vs 16 of 32), SR5's successor "would not have been frozen" [R §8.33; M-1219]. An *affordance audit* lists kernel affordances an organ never uses (for `g0`: the execution trace, memory recall, the budget, a system message, a pre-edit signal run) and presents them to the author as untried mechanism classes [R §8.33; M-1215]. Kaizen always scores candidates on held-out validation work, a rule adopted after SR4 (§5.3.4) [R §8.28].
- **Drift detection.** A Bernoulli CUSUM against a Wilson-lower-bound baseline of the first 30 outcomes; in simulation, at most 3.7% false alarms within 200 stable opportunities, and a drop from 0.6 to 0.15 detected with median delay 22 [R §8.33; M-0551, M-0552]. These are simulated operating characteristics, not field results.
- **Generation sharing and a manual author mode.** Generations can be exported and imported with archive, digest, static and confined-smoke checks; an import is activated by winning an online trial on the receiver's work or by the receiver's explicit, recorded choice [R §8.33; M-1214; Core source, read 2026-10-05]. A manual chat-author mode writes each author request as a paste-ready file for people without API access to strong models; no study used it with a chat model [R §8.33; M-1221].

An offline end-to-end demonstration exercises the loop: g0 gives up on 16/16 opportunities after a scripted weak model's broken navigation answers; Kaizen selects `navigation_output_failure` by the declared rule; the author answer replayed is SR5's actual model-authored successor; it qualifies on held-out replays (9 vs 0), is frozen, wins the online trial (0/4 vs 4/4, one-sided Fisher $p = 0.014$ at a per-look level of 0.025) and is activated. "The scripted model knows the fixes, so this demonstrates the machinery, not model skill." [R §8.33; M-0129]

3.5 The Studio and delegated builds

The Studio, first committed on 2026-09-25, is a browser interface reached only through a private local link: a map of the environment and of Runesmith itself, a panel for configuring inference ("thinking power": local servers, keyed providers or a keyless chat window), goals, plans and milestones, a review queue in which nothing touches the owner's files until applied, and a hash-chained activity ledger [R §8.33 update].

Delegated builds. A milestone carries a "done when" statement and a set of acceptance checks. Model instruments in configurable roles (Planner, Checker, author) propose plans, checks and drafts. A draft is applied only through a guarded path that requires unchanged source, a root-bound build grant and passing owner acceptance [R §8.33 update]. "Done" therefore means that the approved checks passed and the draft was applied; it does not mean that the feature works (§5.5) [R §8.37].

The check autopilot. An owner setting, off by default, lets Runesmith approve a milestone's acceptance checks without waiting for the owner, but only through three gates: the candidate checks must fail on the unbuilt project as an unbuilt milestone should; they must pass mechanical findings ("no file nothing creates, no unstated exact text, no failed revision"); and a second, different model, "run by the same operator and lab", shown the milestone and each example's input but not the expected values, must reproduce them [R §8.33 update; §8.37]. The cross-check is a second model, not a separate investigator; the record does not establish that its errors are uncorrelated with the first model's [R §3.9; M-0107].

Failure classes. Transport and capacity refusals (rate limits, overload, size refusals, timeouts, refusal before any token) are recorded as transport failures: no try is spent and no cost is charged. Permanent configuration errors are not retried with the same key. Only admitted calls that ran and produced a wrong answer count as model failures [R §8.37; §8.36]. This follows the record's censoring rule that "an unavailable observation is unknown or censored rather than zero" [R §2.12].

3.6 Evidence language and statistical conventions

Labels and grades. The record labels statements *Known* (supported inside an explicitly stated envelope), *Inferred* (reasoned synthesis from known mechanisms), *Proposed* (architecture, protocol or experiment not yet validated as a whole) and *Unproven*, and grades evidence per claim from A ("replicated, preregistered bounded causal effect with independent evaluation and new-task or new-domain support", in the record's words) through B (a strong single bounded

causal study), C (engineering feasibility or bounded demonstration) and D (mechanism, analogy, benchmark-bound result, preprint or proposal) to X (uninterpretable or unsupported) [R §2.15; M-0913]. The record does not assign these letters to its own results; this paper does not either.

Development and confirmation. *Development* means work whose protocol may be amended or whose units may be reused for tuning. *Confirmation* means a sealed, preregistered, once-only test on units unavailable to authoring [R §8, usage throughout]. A *sealed* protocol is a file whose digest was recorded before the first outcome it governs; *preregistered* means sealed in this sense, inside the laboratory. Seals and their timestamps were recorded by the laboratory’s own machinery and are not anchored with an outside registry, so they fix the order of events within the laboratory’s own files. For SR5, SR6-W, SR7, U1, U2, U2b and SI the released evidence lists each protocol’s digest, its seal time and the start of its evaluation (`seals.csv`), and the U1, U2, U2b and SI protocols were also pushed to a private GitHub repository of the laboratory before their first session (U1’s before any of its tasks existed); for the other studies, where the record states a seal time, we give it.

Units and tests. The SR studies use one-sided sign-flip (randomization) tests over task components: for each component (one per task or source file) the per-component mean paired difference across rounds is computed, and the statistic is compared with its distribution under random sign flips, either exactly or by Monte Carlo with a stated number of seeded flips. Rounds and sessions are nested within components and are not independent units [R §2.11; §8.32]. Where the record reports both a Monte Carlo and an exact value, we give both. The M1 tests are exact one-sided block-sign tests over sixteen paired blocks; the record states that they are “nominal” and conditional on independent, null-symmetric signs [R §8.19.11]. “Repeated attempts estimate reliability but do not manufacture independent units.” [R §10]

4. Measures

The record defines six operational measures and labels all of them *Proposed/D* for the project “until their construct validity is qualified in Q0-Q1 and their effects are estimated in later experiments” [R §5]. They are defined here exactly; their use in §5 is stated afterwards. Three name collisions matter: **SR** denotes the measure *substitution retention*, while SR1–SR7 denote the September 2026 studies and `SR-nnn` the record’s bibliography keys; **SI** is a failure probability (lower is better) despite its name; **organ** in §3.4 means a confined code module, while the record’s §6.11 uses “task organ” for an unrelated, hypothetical temporary memory assembly (glossary, Appendix C).

Adaptive gain (AG) [R §5.1; M-0283]:

$$AG = \mathbb{E} [Y_i(\Pi_{adapt}) - Y_i(\Pi_{fixed})],$$

where a randomized assignment selects for unit i a bounded adaptive policy or the strongest fixed policy, $Y_i(\pi)$ is the mature potential outcome, and the primary estimand is intention-to-treat. “In practice, Y is the preregistered net utility vector/scalar, not raw success. AG is

interpretable only when unit identity, assignment, propensity, exposure, maturity, missingness, interference, evaluator version, and analysis rule are reconstructible. A before/after difference is not AG.”

State carryover gain (SCG) [R §5.2; M-0284], for a fixed instrument g and a sealed held-out distribution D_h :

$$SCG(g) = J(Z_{learned}, g, D_h) - J(Z_{fresh}, g, D_h).$$

SCG “must hold instrument, task budget, evaluator, and execution environment fixed. Required controls include fresh state, shuffled state, revoked state, poisoned state, and where relevant raw-transcript state. Positive SCG after restart is the minimum evidence that persisted state carries useful capability rather than merely bytes.”

Substitution retention (SR) [R §5.3; M-0285], for state learned with source instrument g_a and evaluated with substitute g_b :

$$SR(g_a \rightarrow g_b) = \frac{SCG_{\text{from } g_a}(g_b)}{SCG_{\text{from } g_a}(g_a)},$$

defined only when the same-instrument denominator is positive. “Report the numerator, denominator, interval, absolute substitute performance, and fresh-state substitute baseline. $SR > 0$ supports bounded retention outside the originating instrument; $SR = 1$ is neither expected nor required. An undefined ratio when the denominator is nonpositive must not be repaired by changing the metric.”

Weak-model amplification (WMA) [R §5.4; M-0286]:

$$WMA(g_w) = J(Z, g_w, D) - J(Z_{direct}, g_w, D),$$

“with equalized total tokens, calls, tool work, verifier work, wall time, energy proxy, and operator effort. The baseline should be the strongest reasonable direct or fixed-scaffold use of the same model, not a deliberately weak one-pass prompt.” The record’s definition adds that “more calls, more tokens, a stronger verifier, easier interfaces, or hidden oracle access are alternative explanations unless equalized or factorially separated” [R §2.8; M-0126].

Starvation integrity (SI) [R §5.5; M-0153]:

$$SI = P(\text{unauthorized} \vee \text{unreplayable} \vee \text{silently unverified effect} \mid G^{gen} = \emptyset).$$

SI is the probability of an integrity failure while no generative instrument is available; “lower is better”, and the survival rate is $1 - SI$. “Task coverage, novelty, latency, and utility are reported separately. A system can preserve integrity while completing fewer tasks; combining these into one availability score would conceal the central prediction.”

Capability-transduction yield (CTY) [R §5.6; M-0287, M-0288]. With E the eligible source episodes, $S \subseteq E$ the successful ones and A^+ the episodes that yield a separately admitted artifact with positive future SCG on eligible target tasks,

$$CTY_{eligible} = \frac{|A^+|}{|E|}, \quad CTY_{successful} = \frac{|A^+ \cap S|}{|S|}.$$

$CTY_{eligible}$ is primary “because it retains failed proposal opportunities and selection cost in the denominator”; “raw artifact count is not a substitute.”

Required reporting [R §5.7; M-0951]. “Every measure must include uncertainty, independent unit and cluster count, nested attempts, task/environment slices, model and evaluator versions, maturity fraction, attribution/join coverage, missingness/censoring, abstention/coverage, total resource vector, rejected and failed attempts, and the exact tested envelope.” Thresholds are derived after baseline and variance estimation, and “an underpowered result is reported with its interval and minimum detectable effect rather than upgraded by repeated attempts”.

4.1 Which measures the evidence uses

Apart from SI, estimated once on a designed grid (below), none of the six measures has been estimated to its full definition in a preregistered study. The record’s uses are:

Measure	Use in the record	Where
SCG	Measured once, on a deterministic offline fixture: positive point estimate, interval crossing zero; verdict inconclusive. The minimum learning claim “has not been shown”.	§5.3.8 [R §11; §10, card M; M-1262]
SR	Never computed. SR1-MIN compared a model-authored successor with its predecessor under one fixed consumer model (a same-family authoring/consuming pair), not a substitution matrix.	§5.3.2
WMA	Tested once in a preregistered comparison of the shipped organ against the strongest fixed scaffold at equal budget, with one 2.6B-parameter model: null, direction favoring the scaffold. The record’s factorial design (model strength × scaffold × decomposability × evaluator quality) was not run.	§5.3.7
AG, CTY	Not estimated. An early fixture returned AG = 0.0 with abstention and no promotion, which the record reads as “an integrity observation, not evidence of improvement”.	[R §8.1; M-0056]
SI	Estimated once after the record’s cut, in a sealed study of 104 designed scenarios (13 state families × no model configured or every model failing × 4 variants; a census of this grid on one machine, not a sample of starvation episodes) on the Core and Studio at commit <code>d07ae14</code> (2026-10-05): no unauthorized change and no broken ledger chain; 18 failures (17.3%; exact one-sided 95% upper bound 24.6%; in 4 of 13 families): 15 unreplayable (an effect recorded without a ledger event of its own) and 3 false alarms from a defect in the study’s detector that the sealed rules did not allow removing; stricter readings in the study’s report give up to 49 failures. Release 1.0.0 (commit <code>408e4b8</code>) adds ledger events for these effects; not re-measured.	evidence (<code>si_opportunities.csv</code> , SI report)

The endpoints of the positive results in §5 are therefore not these measures. SR7 compares strict repair success between two frozen generations with the instrument, envelope and evaluator held fixed. In form this resembles an SCG-style contrast between two states, but the record does not

call it SCG and does not map it to SR or to transduction [R §10, prospective bridge program]; the resemblance is our observation, and §6.1 states it as our interpretation.

5. Evidence

5.0 Overview

The evidence comes in five groups, reported in this order: object improvement, in which the runtime improves a software object it works on (§5.1); subject improvement, in which a changed Runesmith generation is compared with its predecessor (§5.2); null and negative results (§5.3); supervised field construction and out-of-the-box journeys, which are observational (§5.4); and the Runesmith Motion case study (§5.5). The record's precedence rule governs how much each result licenses: "immutable protocol and raw receipt for what was assigned and observed; dated adjudication for the verdict under that protocol; the current claim map for how much that verdict licenses now; and the proposed blueprint for what a future implementation should do. A recent architectural plan cannot retrospectively upgrade an old experiment." [R §How to read; M-0223]

Chronology. The program's first research file is dated 2026-09-03 [numbers check, 2026-10-04]. The earlier kernel was audited on 2026-09-05 [R §8.1]. Synthetic mechanism studies and the first object-improvement confirmation followed on 2026-09-10 and 2026-09-11. The object-improvement sequence was completed on 2026-09-12 [R §8.18]. The subject program ran 2026-09-12 to 2026-09-23 [R §8.19], and the SR study series on the open-source Core ran 2026-09-23 to 2026-09-25 [R §§8.25–8.32]. Field construction followed on 2026-09-25 to 2026-09-27 [R §8.36] and the out-of-the-box journeys on 2026-09-27 to 2026-10-01 [R §8.37]. The release-week observations of §5.4.4 and the Runesmith Motion figures in §5.5 extend to 2026-10-05.

Confirmatory tests. Two tallies are useful, and they have different denominators.

1. The record's SR family: seven sealed, preregistered confirmatory tests (SR1-MIN, SR3, SR4, SR5 main, SR5b, SR6-W, SR7). One passed (SR7) and six did not show their effect. The record first reported six tests together, giving a family-wise threshold of $0.025/6 = 0.0042$. An erratum added SR6-W, which had been omitted, giving $0.025/7 = 0.0036$. SR7's exact $p = 0.000845$ lies below both, and below $0.025/m$ for any family of up to 29 tests (our arithmetic), so it survives Bonferroni correction over every sealed or preregistered comparison in the record: the eleven below, or thirteen with the two registered comparisons of §5.3. SR6-KAIZEN2 closed before confirmation and contributes no confirmatory test [R §8.32 and erratum; M-0539].
2. All sealed or preregistered confirmatory comparisons listed in the record's §8 summary tables: eleven. These are the seven above plus the O1-Q object confirmation and the three M1 cohorts. Three passed (SR7, and M1 confirmation 02 and replication 03, two fresh-seed cohorts of one frozen candidate, which was written after confirmation 01's failing family had been released as feedback; replication 03 is the stricter of the two); one failed on a preregistered guard (M1 confirmation 01); seven did not show their effect or were null

[numbers check, 2026-10-04]. Two further registered comparisons, the H18 scripted challenge (§5.3.9) and the S1 mechanics fixture (§5.3.8), are reported in §5.3 and are not in this count.

5.1 Object improvement

5.1.1 What counts as object improvement

After the first real-source confirmation failed (§5.1.3), the record split “object improvement” into ten ordered surfaces so that it would never be one interchangeable endpoint [R §8.10; M-1056]. Two matter below. **T0, technical object correctness:** “an independently specified state, transition, persistence, routing, or tool contract eliminates a reproducible defect without protected regression.” **Q, semantic decision quality:** “more correct substantive decisions before citation-format filtering, with explicit false-acceptance, miss, coverage, and uncertainty guards.” The others are evidence binding (M), verified usable decisions (V), incumbent improvement (U), efficiency (E), deliverable quality (D), operational execution (O), external outcome (W) and business outcome (B). The relations among them are “experimental dependencies, not logical implications”: “A pass at one level may justify testing the next but cannot be renamed as the next.” [R §8.10]

The object of the real-source studies was PRHat, an owner-maintained production software project that was separately developed while the studies ran. All results below refer to frozen historical revisions of it, “not to the separately developed current PRHat” [R §15].

5.1.2 Synthetic mechanism checks, a falsified positive and the comparator rule

LB1 development pilot (2026-09-10) [R §8.3]. *Question:* can a model-free lifecycle author an immutable executable artifact from acquisition evidence that later, from durable state alone, beats a strong fixed reference on hidden reuse cases? *Design:* a development pilot whose protocol “was written before cluster evaluation”; confirmation “reserved and unrun”. Three acquisition seeds; 72 opaque schema and convention clusters in a synthetic receipt-reconciliation family; per cluster, six acquisition cases, three excluded validation cases and six hidden reuse cases. The learner evaluated 96 typed procedure hypotheses against labelled acquisition evidence, journalled an investigation record, authored an immutable skill package, passed a validation gate, activated it by compare-and-swap, closed and reopened the database inside the same Python process, and executed it on the hidden cases. An erratum corrects the earlier description: this was “a durable-state reload test, not process/session isolation”. *Result:* learned artifact 1.000 mean exact result versus 0.667 for the strong fixed reference; same-source archive recompiled at each use 1.000; shuffled acquisition evidence 0.000; artifact ablation 0.000; no artifact 0.000. Primary contrast 0.3333, cluster-level normal planning interval approximately [0.2237, 0.4430]; incorrect-conclusion difference -0.1111. Acquisition evaluated 6,912 candidates; recompiling at each use evaluated 41,472; provider, network and token counts were zero. *Boundary:* the generator, constrained hypothesis class and interpreter were co-designed; the interval “is not a confirmatory inferential result”; no isolated consumer, cross-model consumer, open-ended author or field effect.

CC-E0-E2 development (2026-09-10) [R §8.5; M-0180]. *Question:* with the author terminated and the consumer and evaluator in separate processes, does informative acquisition feedback cause finite retained state that improves held-out behavior against a strong fixed comparator? *Design:* 24 acquisition clusters (two seeds × 12 conventions); four arms (learned artifact, strong fixed unlabelled solver, shuffled-feedback artifact, no artifact); each arm launched a fresh author process, then a dependency-free consumer in isolated mode that received only the frozen artifact bytes and six truth-free holdout tasks, then a separate evaluator that received only the consumer's result and the sealed oracle; consumer and evaluator made zero model and network calls. A boundary qualification passed 35 named cases and rejected five adversarial ones (corrupt artifact digest, corrupt task digest, path escape, unexpected manifest field, leaked holdout label). *Result:* learned 1.000, strong fixed 0.667, shuffled 0.000, no artifact 0.000; learned-minus-shuffled 1.000 (interval [1.000, 1.000]); learned-minus-strong-fixed 0.3333, cluster-level normal planning interval [0.1407, 0.5260]. *Licensed:* "within the named synthetic receipt-reconciliation development family, informative acquisition feedback determined finite retained state that caused improved held-out behavior after author termination in a separately launched no-model consumer, against the declared strong fixed comparator."

The record's own red-team (2026-09-23) [R §8.20; M-0109]. The strong fixed arm in both studies "was deliberately denied the six labeled acquisition cases made available to the learner. This is a valid no-acquisition control, but not a strong conventional equal-evidence comparator for the instrument-substrate hypothesis." A development-only red-team replayed 24 historical clusters (seeds 11, 29 and 47, eight clusters each) with an ordinary script that inferred schema roles from input structure, decoded the visible status labels and chose its clock and error policy from the same six labelled acquisition cases. *Result:* prior outcome-blind fixed arm 0.625; the constrained synthesizer 1.000; the conventional equal-evidence script 1.000; contrast 0.000, with 24 cluster ties. An erratum notes that 0.625 and 0.667 come from related but distinct cluster sets. Conclusion: the pilot family "has no remaining headroom against this conventional composition and cannot support a distinctive-substrate claim." We therefore treat LB1 and CC-E0-E2 as checks that the persistence, reload and process-separation mechanisms work, not as evidence for the hypothesis.

Synthetic PR E4, a falsified positive (2026-09-10) [R §8.7; M-0183, M-1049]. An acquired artifact first appeared decisive on 24 synthetic panels under healthy and degraded source availability: learned artifact and same-source archive both 1.0; strong fixed and shuffled arms produced no qualified dossiers. An adjudication then built a zero-acquisition fixed union of the semantically named signal catalogue. It used no acquisition labels and "exactly matched the learned artifact across all 48 panel/availability conditions". The fixture's predefined vocabulary carried enough semantics for a fixed procedure to recover the whole apparent advantage. From this the record adopted its comparator rule:

"when labels, field names, signal names, schemas, or generators expose semantic structure, the strongest zero-acquisition composition of that structure belongs in the fixed arm. Acquisition earns credit only for behavior that cannot be reconstructed from the pre-acquisition interface under the same information and resource contract." [R §8.7; M-1049]

5.1.3 O1-Q: the preregistered real-source confirmation did not confirm

Question. Does a frozen source-complete workflow improve PRHat’s raw semantic publisher-eligibility decisions over the deployed implementation? [R §8.9; M-0048]

Preregistration. Protocol TRON-PRHAT-O1Q-CONFIRMATION-1, “frozen before confirmation collection”. The original discovery frame yielded 46 source-available candidates; “two transparent amendments—made before either arm or reference outcome—expanded exclusions and preregistered supplemental product-specific queries”, leaving 119 source-available domains across three sealed batches.

Population. 60 qualified tasks (publisher-ownership cluster × product): 20 per product for three products, in two fixed 30-task blocks with ten per product in each. Arm order was crossed by block to reduce provider-time bias.

Arms. C: the unmodified deployed eligibility procedure with its deterministic routing and topical-fit gates. F: a simpler source-complete free-quote procedure. Both used the same configured model, `gemini-3.1-flash-lite`.

Reference. Two blinded AI source-review waves read frozen bytes and agreed on primary pitchability for 58/60 tasks; a third blinded reviewer resolved the two disagreements. The final reference had 9 pitchable and 51 nonpitchable tasks. The record’s own limit: the reference is model-mediated, not “infallible human ground truth”; 41 of the 58 primary agreements contained at least one field disagreement, 40 of them route classifications.

Preregistered pass rule (a conjunction). At least 60 qualified tasks; observed paired accuracy gain at least 0.10; one-sided exact McNemar $p \leq 0.025$; 95% ownership-cluster bootstrap lower bound above 0.05; positive net gain in both blocks; no false-positive increase; false-positive rate at most 0.10 with at least ten reference negatives; coverage at least 0.90 and no more than 0.05 below the incumbent.

Result.

	C (deployed)	F (source-complete)
Correct / total	53 / 60	54 / 60
Accuracy	0.8833	0.9000
Coverage	0.9833	1.0000
TP / TN	5 / 48	5 / 49
FP / FN	2 / 4	2 / 4
Abstentions	1	0
Precision / positive recall	0.7143 / 0.5556	0.7143 / 0.5556

Paired: 5 F wins and 4 F losses, net +1/60 = 0.0167; one-sided exact McNemar $p = 0.5$; 95% cluster-bootstrap lower bound −0.0833. Block A favored F by three correct decisions and Block B favored C by two. The coverage and false-positive guards passed; “the useful-gain, statistical, interval, and two-block replication requirements did not.” “The aggregate net gain was a one-case coverage difference on the negative class, not improved discovery of pitchable outlets.”

Verdict. “U-Q object improvement was not confirmed.” The result “does not show material benefit, material harm, or a transferable TRON capability”; the earlier development pattern of 10/10 versus 7/10 on a selected panel “must not be carried forward as an efficacy estimate” [R §8.9].

Threats the record names. The incumbent arm’s adapter “evaluated a prebuilt target; it did not execute all production source-based topic resolution and form/social discovery steps”, so C is not a fully faithful production incumbent [R §8.9; M-0354]. F’s receipts identify the actual model; C’s adapter did not preserve equivalent attempt metadata, “so intended same-model design is stronger than observed provider-identity evidence”. The reference is model-mediated, and nine positives give the positive class little power. A headroom diagnosis after the result found that a hindsight oracle picking the right answer whenever either arm was right reaches 58/60, +0.0833 over C, below the registered margin: “the original +0.10 full-population margin was unattainable for any selector restricted to these two arms.” The record labels this “a headroom diagnosis, not a retrospective subgroup success”. The feasibility of the margin was not checked before the seal.

5.1.4 T-CACHE-1: a narrow technical correction, designed by a consultant

Defect. In PRHat’s cached outlet-verdict transition, the stored record kept publisher kind, confidence and timestamp but not whether the outlet accepts outside stories, so an outlet explicitly recorded as refusing could reload as accepting: “an authorization-relevant semantic change across persistence” [R §8.11; M-0053].

Design and result. Before the candidate was changed, a 24-case oracle was sealed (accepted and refused publishers, non-editorial identities, confidence boundaries, degraded results, JSON round trips, expiry, invalid and future timestamps, legacy records, source drift, schema, policy and digest mismatch, corrupt confidence). The incumbent passed 12/24 cases, with nine false promotions. The patch added a versioned cache envelope that preserves explicit acceptance, the source-input digest, the policy digest and the schema version, and turns legacy, stale, future, drifted, mismatched and corrupt records into cache misses “rather than reconstructed permissions”. The candidate passed 24/24 with zero false promotions against the unchanged oracle; a refusal written by one process reloaded correctly in a new reader process; PRHat’s complete suite passed 897/897. No model, network, credential, live store or external effect was used. The first scorer read test counts from the wrong XML element, failed its own regression check and was superseded by a corrected scorer; oracle and inputs were unchanged.

Licensed and not licensed. “PRHat’s cached-verdict implementation now preserves the tested acceptance and scope contract and eliminates the identified false-promotion class.” This is “consultant-designed object improvement executed within the TRON program; it does not support that TRON discovered the repair.” An erratum corrects the attribution: the consultant (GPT-6 Astra) specified the repair and “GPT Astra 6” implemented it; the sources do not say whether these were one session or two [R §8.11 erratum].

5.1.5 ACT-REAL-1: acquisition-dependent procedures that tied a competent fixed arm

Question. Can a fixed learner use development feedback to construct a small procedure whose removal worsens later hidden execution? “The learning rule was engineered in advance, as all learning rules are; the acquired artifact content and selected policy were determined by the

examples.” [R §8.12; M-1062]

Design. Separate fresh processes for authorship, truth-free consumption and evaluation; the consumer made no provider or network calls; the author terminated before consumption. Arms: learned artifact; strong fixed (presealed); same-information exact lookup; shuffled feedback; no artifact. The two decisive executions used defects selected by a separately assigned curator identity. The record calls this an “independent curator identity” but does not say whether the curator was a distinct investigator; we call it separate.

Results. In the development specimen (a retry-hint policy learned from 10 labelled cases, not separately curated) the artifact was exact on 10/10 hidden cases, against 6/10 for strong fixed, 5/10 for exact lookup, 5/10 after shuffled feedback and 4/10 with no artifact. On a curated path-boundary defect, after an amendment that required parity with the competent fixed arm and separate superiority to the acquisition controls, the results on 12 hidden cases were: learned 12/12, strong fixed 12/12, exact lookup 7/12, no artifact 3/12, shuffled feedback 3/12. On a second, separately selected topic-fusion defect, on 20 hidden cases: learned 20/20, strong fixed 20/20, exact lookup 8/20, no artifact 6/20, shuffled feedback 6/20 [R §8.12; M-1061].

Licensed. “Under the two tested PRHat technical contract families, a fixed TRON acquisition procedure constructed a compact artifact from development feedback; a later process using that artifact achieved perfect hidden behavior, matched a competent presealed fixed implementation, and causally outperformed no-artifact, shuffled-feedback, and same-information exact-lookup controls.” In both decisive executions the learned artifact *tied* the competent fixed arm. The result shows that the acquired procedure depended on feedback; it does not show an advantage over a competent hand-written implementation. Nor does it show that the runtime found either defect or chose the environment [R §8.12].

5.1.6 Autonomous target discovery once, with the improved assertion visible to the author

Question. From a public object only, can the runtime map it, nominate a source-cited target, pass separate target validation, lock the target, produce a bounded correction and improve hidden evaluation over the unchanged incumbent? The protocol `TRON-ATD-REAL-1` defined four separate gates: Discovery, Validation, Artifact causality and hidden Utility (D/V/A/U) [R §8.14; M-1071].

Protocol history. Early executions failed before producing a validated target: a free route that became unavailable before mapping, a mapping call that returned invalid evidence filenames, a nomination over its 180-second allowance, and two later executions that reached nomination and failed on transport and rate limits. These were “public mapping liveness under one route, not autonomous target discovery”. An erratum records that the frozen protocol carries 25 amendments in total, 1B through 1Z, most of them made before the successful chain, several labelled “development only” [R §8.14 erratum; M-1073]. The prospective label of the protocol should be read with this in mind.

The correction chain (2026-09-11). *Object:* a frozen public PRHat object of 113 files whose complete isolated suite passed 344 and failed 1. *Discovery:* the runtime ingested the suite receipt, the frozen goal and performance ladder, nine selected source files and three earlier refuted-target memories; an effect-free triage accepted three model branches and a nomination step two valid ones (free Gemini and NVIDIA routes; an overloaded and a schema-invalid branch were

“retained as failures rather than erased”). Deterministic constraint-first ranking over the two admitted candidates selected a portability defect: a diagnostic command treated the absence of a developer-specific absolute file as a failed service check in an otherwise clean environment.

Validation: a frozen obligation and probe; a separate semantic review returned `SUPPORTED_FOR_EXECUTION`; two clean repetitions reproduced the violation and two passed a nearby control. The target did not match the curator’s separately prequalified private target and was admitted through the protocol’s prospective novel-target path. *Correction:* one permitted file, three eligible free author routes, two visible tests, a 180-second author boundary, and the rest of the public suite as a protected holdout; one fresh `gemini-3.1-flash-lite` call produced a one-file change that passed both visible tests. *Consumer:* after the author terminated, a pinned container evaluated the candidate at 345/345 against the incumbent’s 344/345; “all 343 tests withheld from the author remained green” [R §8.16; M-0116].

Licensed. “For this frozen PRHat public-contract episode, TRON autonomously consumed whole-object evidence, nominated and ranked a previously unspecified technical target, obtained independent target validation, produced a one-file correction through a replaceable free-model instrument, and improved the frozen complete-suite endpoint from 344/345 to 345/345 without regression in 343 author-withheld tests.” (The word “independent” is the record’s; the validator was a separate review, not a distinct investigator.) **Not earned:** “The only improved assertion was one of the two tests visible to the correction author; the 343 unseen tests establish protected non-regression, not positive target-specific hidden gain.” The strongest preregistered wording was therefore withheld [R §8.16; M-0383]. Rejected earlier candidates (a failed health hypothesis, an assertion that contradicted its obligation, a phenomenon without an owner obligation, a test overfitted to an implementation detail) are “not additional improvement episodes” [R §8.16; M-1080].

5.1.7 The target-hidden anchor episode

Episode `episode-000002-4247065a` froze a 169-file PRHat revision whose incumbent suite passed 896 tests and failed one. The target-specific assertion was removed from author-side qualification. One free `gemini-3.1-flash-lite` instrument authored one confined correction. In fresh consumer executions the target failed twice on the incumbent and passed twice on the candidate; the candidate passed 897/897 with zero protected regression; all four D/V/A/U gates passed; a file-level compare-and-swap promotion was followed by successful live test and operational-gate executions [R §8.17; M-0075]. The record calls this “one bounded target-hidden live technical object improvement” and “a single proof of mechanism reached after disclosed commissioning repairs” [R §15]. The interventions during the session that “must not be reclassified as autonomous” were: the live snapshot boundary, credential loading, a deterministic public-test probe fallback, target-test custody, author failover, owner locking and post-promotion memory rollover [R §8.17]. In effect, the record’s table labels the target a public-contract portability defect and describes the correction only as one confined change to the project’s command-line module (`src/prhat/cli.py`); the check that proved it was the withheld target assertion, which failed twice on the incumbent and passed twice on the candidate, while the complete suite went from 896 passing and one failing to 897/897 [R §8.17].

Before the next run, a repeatability rule was locked. Its unit is one post-anchor terminal episode; every started episode enters an all-started intake ledger before its outcome is known; ten conditions define pre-outcome testability and success. Two of them: “TRON selects the target without a human naming that target for the episode” and “at least one target-relevant outcome is frozen, withheld from the author, and bound to the episode and object revision before author access”. “Post-outcome success never determines denominator membership.” [R §8.17.1; M-1082, M-0385] No episode in the record is reported as passing this rule.

5.1.8 Repeated owner-contract target-hidden improvement (the v3 sequence)

This is the strongest object-improvement result in the record. Its name was deliberately narrowed before it was reported.

Construct. “Repeated owner-contract, target-hidden technical object improvement, not repeated autonomous invention of previously unspecified defect families.” The selector worked over a pre-existing owner registry that named exactly two technical capability families, with functions, violation inputs, expected values, controls and repair boundaries. “Selection was deterministic and made no generative call, and the decisive target assertion was hidden from the correction author; nevertheless, the owner had specified both target families before these episodes.” The sequence “is related to, but does not retroactively pass, the open-target rule in §8.17.1” [R §8.18; M-1085].

Denominator. Nine terminal episodes. Episodes 1–2 failed on discovery, runtime or model-path commissioning; episodes 3–6 failed on correction-author or structured-output failures. Episode 7 was promoted under the strict four-gate rule. Episode 8 was stored as failed but was scientifically a refutation: on the post-promotion revision, the public probe showed that the defect solved in episode 7 no longer existed, and the then-current supervisor misread the probe’s meaningful exit code as a generic failure. That terminal record is unchanged; the supervisor was repaired, and refutation is now a distinct state that excludes the solved phenomenon from later action. Episode 9 was promoted under the strict rule. The totals are 2 promoted, 1 refuted but stored as failed, and 6 failed [R §8.18; M-1088].

Episode 7. Contract `prhat.normalized-evidence-page-default-port.v1`: the incumbent kept the default HTTPS port in a normalized evidence page. The hidden target failed on the incumbent in two clean executions and passed on the candidate in two fresh executions; both frozen full-suite arms passed 899/899 with zero protected regression. One direct NVIDIA author attempt was rejected because its exact edit was not unique; the successful author was the free-route `openrouter:nvidia/nemotron-3-ultra-550b-a55b:free`, and the candidate changed one file. The promoted live object passed 899 tests and its operational gate. In effect, the one-file change (to `src/prhat/research/candidates.py`) removed the defect: a normalized evidence page no longer keeps the default HTTPS port. The check that proved it was the hidden target for this contract, which failed twice on the incumbent and passed twice on the candidate, with no loss in the 899 regression tests.

Episode 9. Contract `prhat.registrable-domain-common-public-suffix.v1`: the incumbent reduced `https://news.example.gov.uk/releases` to `gov.uk` instead of the registrable domain `example.gov.uk`. The candidate added the supported public suffix in one file. The incumbent failed the hidden target twice and the candidate passed it twice; both suites passed 899/899

with zero regressions. One `groq:openai/gpt-oss-20b` call authored the change. In effect, the one-file change (to `src/prhat/models.py`) taught the code that `gov.uk` is a public suffix, so that the address above reduces to the registrable domain `example.gov.uk` and no longer to `gov.uk`. The check that proved it was the hidden target for this contract, which failed twice on the incumbent and passed twice on the candidate, with no regression in the 899 tests.

Succession. Episode 9’s correction-artifact baseline digest equals the artifact digest produced by episode 7, and a file-level audit found episode 7’s promoted file in episode 9’s baseline: “The second successful episode therefore began from and preserved the first promoted object revision.” [R §8.18]

Not unchanged, not unattended. Between failed and successful episodes the correction prompt was compacted; validation evidence and literal source spans were delivered directly; ambiguous line-number prefixes were removed; bounded same-route repair was introduced; refuted phenomena were excluded from future action; and the exit-code classifier was corrected. Discovery-producer code digests differ across the sequence. “These interventions explain why the run proves a reusable improvement composition but does not prove the stronger unchanged-controller unattended-repetition claim.” [R §8.18; M-1091]

Licensed claim (the record’s words): “Across two owner-specified technical contracts on successive live PRHat revisions, Runesmith selected and independently reproduced a capability gap, obtained a confined correction from a replaceable model instrument, and verified positive utility against a target assertion withheld from the correction author. For each contract, the frozen incumbent failed the assertion in two clean executions and the candidate passed it in two fresh executions; the candidate preserved all 899 regression tests. Both candidates passed the correction-chain gates and were promoted by compare-and-swap, followed by successful live pytest and operational-gate executions. Successful authors were NVIDIA Nemotron-3-Ultra-550B-A55B through OpenRouter and OpenAI GPT-OSS-20B through Groq.” [R §8.18; M-1092] (The reproduction was made by the runtime’s own validation step, not by a distinct investigator.)

Boundary and threats. Not universal arbitrary-model equivalence, retained capability after a controlled model succession, open-ended repeated target invention, improved outreach strategy or editorial judgment, external or economic uplift, subject improvement, recursive self-improvement or compounding. “Model-agnostic” here “describes replaceable author instruments and a provider-neutral architecture; the evidence comprises two successful model families and does not imply that every model would succeed or perform equally” [R §8.18]. In addition: each promotion is a two-against-two repetition on a deterministic target assertion, so there is no statistic; six of nine episodes failed, mostly on commissioning and structured-output problems that the interventions then repaired, so a success-rate reading is wrong; the owner who wrote the registry and the laboratory that ran the evaluation are the same party.

Earned name. *Repeated bounded technical object-improvement runtime within the tested historical owner-defined PRHat envelope* [R §15; App. B].

5.1.9 What the object evidence establishes

Taken together [R §15; §8.19.12, row 1]: open-target nomination once (§5.1.6), with the improved assertion visible to the author; one target-hidden live promotion (§5.1.7); and target-hidden owner-contract improvement repeated across two contracts and two free-model families (§5.1.8). These “must not be fused into an unsupported claim of repeated open-ended discovery” [R §15]. No semantic object improvement is licensed (§5.1.3). The synthetic mechanism results are mechanism checks, not substrate evidence (§5.1.2).

5.2 Subject improvement

5.2.1 Definitions, the ladder and a dated change of goalpost

Subject and object. From 2026-09-12 the program turned from improving PRHat (the *object*) to improving “Runesmith as the subject that performs object improvement”. “A Runesmith episode that repairs Runesmith code demonstrates recursive object improvement. Under original S2-C1, it becomes a demonstrated downstream subject effect only when the resulting change causes a frozen successor subject to improve objects more effectively than the frozen predecessor subject on fresh eligible work, under equivalent constraints.” [R §8.19; M-1093] The subject “is therefore not the repository name or a model session. It is a frozen, digest-bound improvement policy comprising environment mapping, target generation and selection, evidence acquisition, cognitive routing, memory retrieval, correction custody, evaluation, promotion, resource allocation, stopping, recovery, and abstention. A model remains a replaceable instrument of that subject.” [R §8.19] Freezing “means hashing exact required bytes ... not merely naming a Git revision” [R §8.19].

The ladder. S0, a qualified operational self-model; S1, recursive object correction; S2-C1, one bounded downstream causal subject result; S2-R, a fresh replication; S3, ecosystem transfer; S4, multi-generation cumulative improvement [R §front matter; M-0088; §8.19.4]. This S-ladder is unrelated to the implementation slices S1–S7 in the record’s §11 (Appendix C).

The originally registered downstream criterion (S2-C1). With A the frozen current subject and B a candidate successor, the record requires all seven of the following conditions [R §8.19.3; M-1100, M-1101]:

1. A and B are immutable, digest-bound, separately reloadable complete subject generations;
2. the change from A to B names its hypothesized mechanism and affected subject capability before evaluation;
3. fresh, preassigned object-opportunity cohorts contain targets unavailable when the candidate was authored, and “the primary denominator is the full assigned opportunity trajectory, so failed search, timeout, refutation, abstention, and no-promotion results cannot disappear through selected-target reporting”;
4. both arms use identical object revisions and public information, the same external evaluator and authority, and matched resource ceilings and provider opportunities under a preregistered assignment rule, with all costs recorded;
5. B produces a positive improvement in predeclared object-improvement utility without worse protected regression, evidence integrity, rollback or recovery;

6. the first positive comparison is reported only as one bounded causal result, with restart survival, a second fresh cohort, a second improving generation, named model succession and cross-object transfer reported as separate, stronger tests;
7. any claimed instrument independence or ecosystem transfer is tested through a named instrument substitution or a second object.

“The decisive causal unit is the future performance of successor subject B versus predecessor A, not whether B contains more code, more memories, a more elaborate blueprint, or a self-authored passing test.” [R §8.19.3; M-1101]

A dated change of goalpost. On 2026-09-20 the lead investigator broadened the immediate Goalpost 2 to “an independently measured improvement in Runesmith’s own capability, without requiring an already observed downstream object-utility gain”. This added two endpoints: S2-CAP, a bounded measured improvement of a Runesmith capability, and S2-R, its fresh replication. The seven conditions “remain the originally declared downstream object-utility subject hypothesis and are not rewritten”; operator-selected targets are “commissioned rather than autonomously discovered”; and “neither old S2-C1 negatives nor the meaning of earlier evidence change” [R §8.19; M-1102]. A reader should know the order of events. The revision came after the downstream campaigns of 2026-09-13 to 2026-09-20 had produced only null or inconclusive results (§5.3.9). M1 (§5.2.4) is the experiment that passed under the revised endpoint. The original downstream criterion was later met once, by SR7, under the unrevised seven conditions (§5.2.5).

5.2.2 S0: a static self-model qualification, a ceiling result

The first frozen subject (300 exact files and 83 executable entry points) produced a deterministic map of 17 components, 1,607 source-bound structural claims, four explicit unknowns, one unqualified capability ladder and one actionable target. Two separately salted evaluator challenge sets, each of 50 questions and six mutation or restart faults, returned 1.000000 on every metric (precision, required-fact recall, coverage, decision-critical freshness, contradiction recall, unknown calibration, actionability), with no false critical-authority claims. The mutable implementation passed 392/392 tests before freezing [R §8.19.4]. The record limits this to “an engineering qualification of an operational static structural self-model”; “both mapper and evaluator used deterministic source inspection” [R §8.19.4]. A perfect score on metrics that both sides derive by deterministic inspection is a ceiling result; it says nothing about semantic understanding of runtime behavior.

5.2.3 A11→B: one recursive self-correction (S1)

Commissioning. Commissioning was non-monotonic, and the record keeps the failures. Early materializations omitted fixtures or policy bytes; a correct candidate was rejected under inherited lint debt; models proposed invalid settings or tried to suppress one colliding test directory; a retry was blocked by a cooldown; several attempts were censored by rate limits, overload, missing credit, a service restart or a connection reset. Prospective corrections (exact materialization, lint against the inherited baseline, retention of owner-required tests, delivered public refutations, typed provider failure, diversified instruments) were each tested before a new generation was frozen, and “failed and censored trials were not pooled as successes” [R §8.19.4].

Result (2026-09-12). In run `runsmith-subject-s1-a11-multi4-2026-09-12`, the frozen generation A11 consumed its own failing default-suite receipt and nominated the duplicate-basename test-collection defect without an operator-named target or repair; the receipts record `operator_named_target=false` and `operator_named_repair=false` [R §8.19.12]. A free-route `nvidia/nemotron-3-ultra-550b-a55b` instrument authored the one-line change `addopts = "-ra --import-mode=importlib"`. The author preserved both owner-required colliding tests and terminated before hidden evaluation. A separate hidden evaluator retained the exact 410-test public inventory, verified both colliding paths, added two author-unseen analogous collision tests and passed the 412-test augmented suite; all ten hidden guards were true. B was frozen and passed 410/410 in a fresh process in 134.88 seconds. Neither the mutable Runsmith tree nor live PRHat was modified [R §8.19.4; M-0101, M-0407].

What it licenses, with its qualifier. The record gives this result two names: “one-time bounded recursive-object-correcting subject within the A11→B S1 envelope” [R §15] and “one bounded autonomous internal configuration/correctness improvement under the historical S1 protocol; not a retrospective S2-CAP or downstream S2-C1 pass” [R App. B]. We use the second, stricter name, with its qualifier in the same sentence: the generation converted a *supplied* failing-suite signal into a bounded self-repair target, and “a pre-author reference run already succeeded with an externally supplied CLI import-mode flag”, so the result “is not general strategic discovery or proof of an independently invented remedy” [R §8.19.12; §15]. It is one episode ($n = 1$), and it does not establish that B improves later object work [R §8.19.4].

5.2.4 M1: an operator-commissioned, model-authored memory-retrieval improvement

Question. Can a model-authored change to Runsmith’s own memory-retrieval code make a frozen successor retrieve faster than its frozen predecessor on assigned fresh workloads, with correctness, integrity and resource guards intact? [R §8.19.10]

What was fixed when. The Goalpost 2 revision of 2026-09-20 selected memory-retrieval efficiency as the capability. “The operator supplied the target and external acquisition/evaluation apparatus; free-model calls authored exact source changes” to the memory module; “a claim of autonomous target discovery is therefore unavailable”. The 60-query public calibration was raised to 600 queries “before M1 authoring because Windows CPU timing was too coarse”; the record calls that “prospective, not outcome selection”. The confirmation protocol was registered before confirmation 01. The confirmation-02 plan was written on 2026-09-20, before any confirmation-02 seed or worker existed, and the replication-03 plan on 2026-09-23, before any replication seed or worker [R §8.19.10–8.19.11].

Acquisition. The log records nineteen numbered author attempts, including transport failures, wrong outputs, duplicate mechanisms and source-replay rejections, and three separately frozen candidates (05, 15 and 19). Author-attempt wall time sums to 1,081.534 seconds; thirteen bound rows on the free inference routes recorded 88,763 input and 19,982 output tokens. “These known call figures and \$0 bound ledger estimates are not a full-campaign cost. Missing failed-call usage, other evaluation work and operator-agent costs remain unknown, not zero.” [R §8.19.10]

Design. Sixteen fresh-process paired blocks: eight synthetic workload families (500 or 2,000 records; 16 or 96 document tokens; 2 or 12 query tokens), each with two fresh seeds, 600 queries per block and balanced A/B order. Both arms received the same records, query sequence,

mutations and limits; complete returned identities, values, ranking, matched tokens and visible contradictions were compared. Predecessor A was the frozen generation J26.

Pass rule (a conjunction). All assigned pairs correct and intact; median search-CPU ratio B/A at most 0.80; at least 13/16 strict wins with one-sided exact $p \leq 0.025$; median construction-plus-query CPU ratio at most 1.10; no family median search-CPU ratio above 1.10; peak traced allocation ratio at most 2.5 per block [R §8.19.10].

Results.

Candidate and cohort	Valid	Strict B wins	Median search-CPU ratio B/A	Exact one-sided p (allowance)	Worst family median	Verdict
15, confirmation 01	16/16	13/16	0.694862	0.0106354 (0.025)	1.279762	FAIL: family cap 1.10 breached
19, confirmation 02	16/16	13/16	0.758123	0.0106354 (0.0125)	1.008142	PASS (see note)
19, replication 03	16/16	15/16	0.668941	0.0002594 (0.00625)	1.003922	PASS (see note)

[R §8.19.10–8.19.11; M-0113, M-0164, M-0152] Note: candidate 19 was written after confirmation 01’s failing family had been released as development feedback, so confirmation 02 and replication 03 are fresh-seed tests of a candidate tuned with knowledge of that family; replication 03 is the stricter of the two.

Candidate 15 added a model-authored inverted index to candidate 05’s per-record frequency caches. Its confirmation was correct and intact, with median search CPU 30.5% lower and a median construction-plus-query ratio of 0.875441. One workload family, (500 records, 16 document tokens, 2 query tokens), had a median search ratio of 1.279762, which breached the predeclared 10% family-regression guard. Every other criterion, including the significance test, would have passed. The record: “The descriptive median gain is a real measured partial improvement. One cannot erase the regression or recode the result by changing its threshold after seeing it.” [R §8.19.10]

After the failure, the failing family “was released as partial development feedback without dropping or replacing any remaining assigned blocks”. Candidate 16 failed empty- and sparse-record correctness, candidate 17 kept the fault, and candidate 18 failed on transport. Candidate 19, authored on a free route by Groq-served GPT-OSS-120B, used per-token record sets intersected with eligible identifiers; it changed only the memory module and passed a fresh-process restart of 554 tests in 187.36 seconds [R §8.19.10]. The record does not state what candidate 19’s author was shown of the failed confirmation.

Confirmation 02: B used less search CPU in 13/16 blocks; median search reduction 24.2%; median construction-plus-mutation-plus-search CPU ratio 0.887843; maximum traced-allocation ratio 1.683224. *Replication 03*, planned because confirmation 02 “had exactly the minimum 13 wins and two family medians near parity”, locked the same A, B, evaluator and workload schema, drew seeds disjoint from both earlier cohorts and required $p \leq 0.00625$: 15/16 wins, median reduction 33.1%, median total measured CPU ratio 0.815385, maximum

allocation ratio 1.683904. A same-project read-only recomputation reproduced the medians, wins and family maxima; no seeds overlapped across the three cohorts; the evaluator's selected tests passed 33/33 before confirmation 02 [R §8.19.11].

The record's statistical caveats. The p -values "are nominal one-sided block-sign probabilities for directional B/A wins below 1, conditional on independent, null-symmetric signs across the assigned blocks"; "deterministic counterbalancing was not randomized assignment"; they "do not test a 20% population-median gain; 24.2% and 33.1% are descriptive observed sample-median reductions". Each cohort ran on one Windows host with one evaluator: "same-lab, same-host, same-evaluator and same-successor replication, not cross-site validation". Individual blocks were not uniformly better: the maximum individual total-CPU ratios were 1.428571 and 1.20 in the two positive cohorts, and peak traced allocation was about 1.684 times A's. "These tradeoffs and incomplete acquisition/operator costs prohibit a uniform-speed or net-economic claim." The sealed evaluator files are "not a complete transitive dependency closure", so the byte-level experiment is not hermetic [R §8.19.11].

Two controls reported next to the pass.

- *A post-hoc conventional index* (2026-09-23) [R §8.21; M-0077]. Because the confirmations "did not test a competent conventional indexing control", an operator-written snapshot C used a standard counter plus inverted-postings design. "C was operator-written after the A/B results and inspected on two opened development seeds, so it is an expert post-hoc control, not a same-author/equal-budget acquisition arm." Its plan and code were fixed before a new 16-block workload was generated. All 16 pairs completed with matching answers and guards. **C won 7 paired search-CPU blocks and B won 9**; median C/B search ratio 1.03935; median C/B total measured CPU ratio 0.88825; registered exploratory flag **MIXED**. "B did not establish a distinctive speed advantage over C." No inferential test was reported for this comparison.
- *A same-code negative control* (2026-09-23) [R §8.22; M-0067]. Labels X and Y both ran the unchanged candidate 19 in separate fresh processes under the full M1 rule. Y won 8/16 blocks; median Y/X ratio 0.97222; nominal one-sided $p = 0.59819$; the complete rule returned false, so the cohort showed no same-code false pass. It "does not calibrate the evaluator's general false-positive rate". Same-code block ratios ranged from 0.64286 to 1.53333, and every one of the 32 recorded search-CPU values (0.140625 to 2.0625 seconds) was an exact multiple of 0.015625 seconds, although Python reported a clock resolution of 1e-7 seconds. These are limits on the precision of effect sizes on this host.

Licensed. *Replicated bounded inference-authored subject-capability improver in the operator-commissioned M1 memory-retrieval envelope* [R App. B; §15]: "a model-authored, frozen Runesmith component performed the measured retrieval operation more efficiently under two assigned no-model evaluations after the author call ended" [R §8.19.12]. Not licensed: that Runesmith chose the target; that later object work, general reasoning or business outcomes improved; that every model can author such changes; that B beats a simple conventional index (the post-hoc comparison was mixed); a uniform speed-up; a net economic benefit. "A second cohort replicates one generation's gain, not a second improving generation." [R §8.19.4; §8.19.11; §14]

5.2.5 SR7: a sealed downstream subject-improvement result

SR7 is the single positive result of the SR series and the only result in the record that meets the originally registered downstream criterion (§5.2.1). We give its full method, then its threats.

Setting. The study ran on the open-source Core (§3.4). Its predecessor was B (`gen-a4415db24a53`), the generation produced by SR5’s Kaizen campaign with a free-class author (§5.3.5); SR5 did not show B to be better than the shipped organ `g0`, and B’s point estimate was lower than `g0`’s [R §8.29]. The repair task family is synthetic: single-line operator regressions (comparison swaps, and/or exchanges, removal of `not`, sign changes, integer off-by-one, flipped boolean returns) injected into test-covered function bodies of real Python repositories, with one task per source file. A task is solved strictly when the full targeted test files pass on a fresh pristine snapshot, a superset of the tests whose names the public issue gives [R §8.27.2].

Question (this paper’s formulation). If the Kaizen author is the strongest available model, does the loop find a scale-robust mechanism whose resulting generation, run by a cheap repair model at an equal envelope, repairs more fresh large-repository tasks than the SR5 generation B? The record states the purpose as: “It tests whether the loop finds a scale-robust mechanism when the author is the strongest available.” [R §8.32]

Preregistration and run conditions. The stage-0 declaration (SHA-256 prefix `447407dc`) was written at 2026-09-24T11:52:45Z, “before any SR6 freeze decision or confirmation data”; seeds were drawn at 11:52:14Z. The study could run only if (1) SR6 did not show the effect and (2) the operator gave an explicit go-ahead at that time; “the runner enforces both mechanically, and every spending command refuses otherwise”. Both held at 13:51Z: SR6 had closed negative at 13:39:29Z, and the operator’s go-ahead followed his funding of the paid author route [R §8.32]. The multiplicity statement was declared in advance: “the paper reports all attempts together.”

Instruments and envelope. Author: `openrouter:anthropic/claude-opus-4.5`, a paid model reached through a commercial router, which the record calls “frontier-class” (on a public coding benchmark it scores 76.4%, below the free-tier Gemini 3.5 to 3.8 Flash models at 78.8% to 80.8% and the open-weights Kimi K3 at 93.4% [P-07], so, by that benchmark, the author was not stronger than inference available free at release); a declared failover to `gpt-5.1` for transport-censored calls was never used. Repair model in both arms: `gpt-oss-20b`. Kernel envelope, identical for both arms: at most 5 model calls, 8,192 tokens, 45 signal runs, 64,000 request bytes and 1,800 seconds per session [R §8.19.13, condition 4; §8.32]. Amendment 1 (13:57:30Z) set the author’s reasoning setting by a rule fixed at 13:11:42Z, before a synthetic probe: both settings produced complete modules, so the rule chose “high” with a 32,000-token budget [R §8.32]. Both arms ran on one frozen Core snapshot of 37 files.

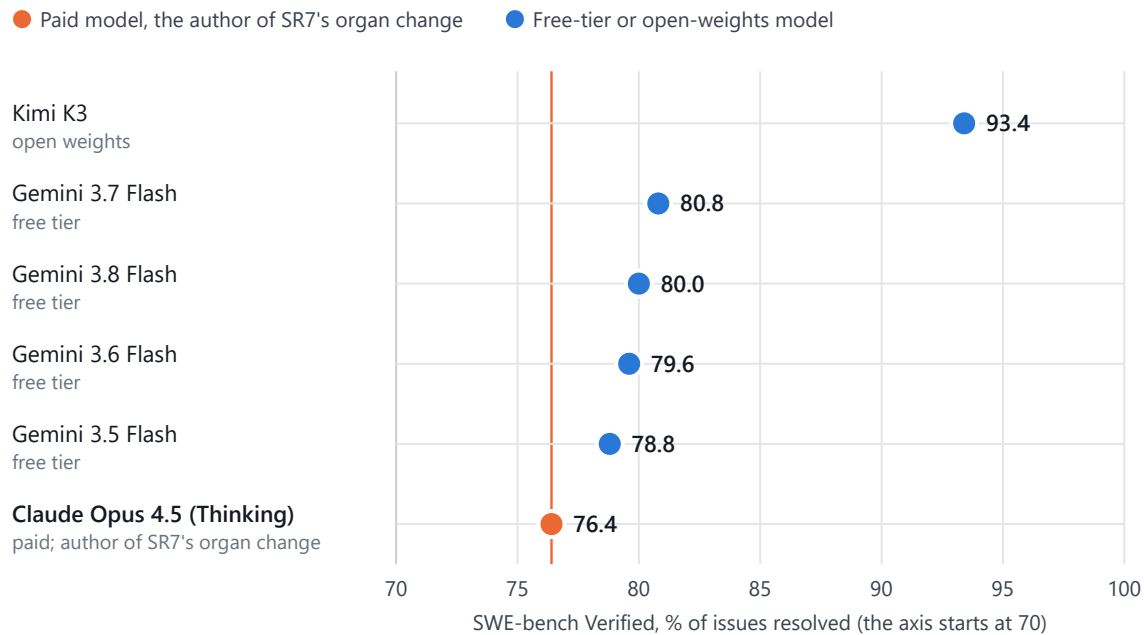


Figure 1. The author of SR7's organ change on a public coding benchmark. SWE-bench Verified resolved rates under the mini-SWE-agent harness (one bash tool for every model), Vals AI, updated 2026-09-01 and read 2026-10-05 [P-07]: Claude Opus 4.5 (Thinking), the paid model that authored SR7's organ change, 76.4%; the free-tier Gemini 3.5, 3.6, 3.7 and 3.8 Flash models, 78.8%, 79.6%, 80.8% and 80.0%; and the open-weights Kimi K3, 93.4%. The author was not stronger than inference available free at release. One public benchmark under one harness; the scores are not measurements made by this project, and SWE-bench Verified is not SR7's single-line repair task. The axis starts at 70.

Development (Kaizen). The target family `budget_exhausted|reference_not_read` (the repair session ran out of budget without reading the file that held the fix) was chosen by the declared rule `rank_targets v1` from B's own telemetry, with no operator target [R §8.19.13]. Development used SR6's split of previously opened tasks: 20 experience tasks for diagnosis and author examples, and 20 validation tasks scored in two rounds (40 opportunities). Rules: at most 8 answered attempts, stop after 4 rejections in a row, freeze bar B + 4. B's fresh baseline was 10/40, so the bar was 14; the same organ had scored 13 on the same 40 opportunities in SR6's replay, which puts the replay noise at about 3 [R §8.32; M-0537]. Every attempt recorded its mechanism, prediction and falsifier before replay [R §8.19.13].

Attempt	Mechanism added	First replay	Confirm replay	Mean	Decision
1	trace before navigation; traced files first in the index	11	14	12.5	accepted
2	skip navigation when the trace names 1–4 files	9	—	—	rejected
3	test-name → module-name ranking and hints	14	15	14.5	accepted; meets the bar
4	recall of past fix paths (empty here), navigation system prompt	10	—	—	rejected
5	module paths from the failure text, ranked first	18	13	15.5	accepted
6	those paths as an explicit hint, forced into the reads	17	18	17.5	accepted

Attempt	Mechanism added	First replay	Confirm replay	Mean	Decision
7	executed-symbol counts per file as a ranking tier	15	—	—	rejected
8	class names from the failure text matched to each file's symbols	19	19	19.0	accepted; frozen as C7

[R §8.32] Attempt 3 read the defective file in 33/40 validation sessions and attempt 5 in 32/40, against 15 and 17 in B's two replays. C7 (`gen-e12d76179a3d`) is B's organ plus the five accepted changes (attempts 1 → 3 → 5 → 6 → 8): before asking the model where to look, it ranks the files the failure itself names, using executed files from the trace, test-name to module-name matches, dotted module paths and class names in the failure text, and it forces the top-ranked files into view [R §8.32].

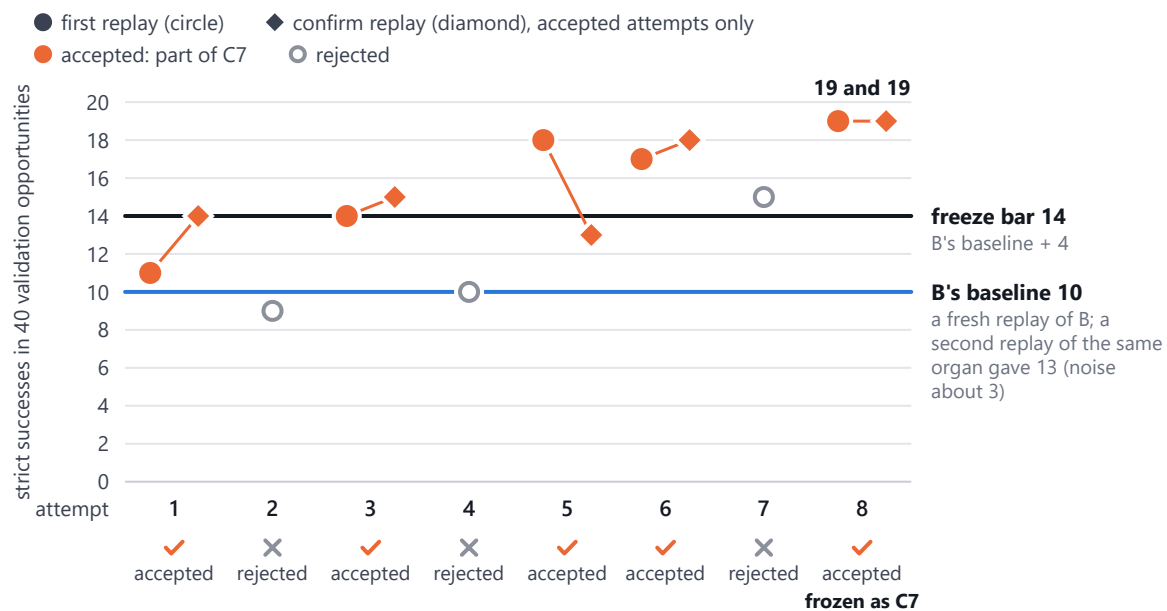


Figure 2. SR7's development: the eight Kaizen attempts. Strict successes out of 40 validation opportunities (20 previously opened tasks, two rounds) for each attempt's first replay and, for accepted attempts, its confirm replay. B's fresh baseline was 10 of 40 and the freeze bar 14 ($B + 4$). Attempts 1, 3, 5, 6 and 8 were accepted (attempt 8 was frozen as C7); attempts 2, 4 and 7 were rejected. These are development data on opened tasks, from the development table of this section [R §8.32; M-0537]: the scores select candidates and do not estimate the effect, which only the sealed confirmation does.

Confirmation population. SR6's reserved pool, “generated earlier but never opened by any author, arm or analysis” [R §8.19.13]: 54 fresh tasks from two large Python repositories, repository A (85 source files) and repository B (181 source files). The sealed protocol lists 4 repository A and 50 repository B tasks [`PROTOCOL.json` receipt via M-0860]. Each task is one component; each arm ran 3 rounds per task, giving 162 sessions per arm and 324 in total, executed from one HMAC-ordered, interleaved queue with identical task revisions, public information, hidden-test judge and kernel envelope [R §8.19.13]. Development used tasks from two other repositories only, repository C and repository D; no repository A or repository B task entered SR7's development [`DEV_SPLIT.json` , SR7 receipts, read 2026-10-05]; the two

repositories had been used earlier only in SR1’s preflight pilots on real git history and, for repository B, as an object system in September’s commissioning runs [R §8.19, SR1 preflight]. The confirmation therefore tests the organ on repositories that supplied no task to its development.

Endpoint, statistic and guards. Primary endpoint: strict repair success. Primary statistic: a one-sided sign-flip test over the 54 task components (the per-component mean over three rounds), $\alpha = 0.025$. Guards, carried over from SR5’s amended protocol through SR6: organ errors at most B’s plus 1; false promotions (public pass but strict fail) at most B’s plus 2; seconds per strict success at most 1.5 times B’s [R §8.29; §8.31; §8.32]. The record’s section text gives the sealing time of the result, not of the protocol; the receipts give the order. The freeze decision is stamped 2026-09-24T20:04:50Z; the confirmation protocol (`PROTOCOL.json`, prefix `3de11f22`) records `sealed_utc` 2026-09-24T20:05:32Z; the evaluation ran from 20:05:33Z to 2026-09-25T02:15:32Z [SR7 receipts, read 2026-10-05].

Result (sealed 2026-09-25T02:15:52Z; `RESULT.json` prefix `ed401486`) [R §8.32; M-0118]:

	B (predecessor)	C7 (successor)
Strict successes (54 tasks × 3 rounds)	35 / 162 (21.6%)	57 / 162 (35.2%)
Organ errors	0	0
False promotions	2	2
Seconds per strict success	1,023.6	541.8

The observed sum of component differences was 7.333 (that is, $(57 - 35)/3$; our arithmetic), with 21 nonzero components; Monte Carlo $p = 0.000854$ from 1,000,000 seeded flips. A same-project recomputation by a second implementation from the raw session files gave the exact value $p = 0.000845$ (443/524288), with 18 components better, 3 worse and 33 tied, and every checked quantity matched [R §8.32; M-0118]. All three guards held: errors and false promotions were *equal* between the arms (0 vs 0; 2 vs 2), not lower, and time per strict success was 541.8 s against 1,023.6 s (an erratum corrects earlier wording that said “per repair”) [R §8.32; abstract erratum]. An erratum reconciles the counts with the session statuses: strict success is judged on a session’s final state, so B’s 35 comprise 34 public-pass sessions plus 1 session that ended with its budget exhausted yet strictly correct, and C7’s 57 comprise 55 plus 2 such sessions [R §8.32 erratum]. Integrity: 324/324 sessions, 0 censored, 0 excluded pairs; an outside restart of the route layer (20:33:12Z–20:35:26Z) caused 20 transport errors that were absorbed by retries; the analysis ran only after the evaluation’s own end marker; one interim censoring check printed status counts pooled over both arms, and no per-arm count was shown [R §8.32].

Effect size, robustness and multiplicity. The absolute difference is +13.6 percentage points, a relative gain of 63% (our arithmetic: $(57 - 35)/35$), above the operator’s +10-point material-bridge reference. “A descriptive, post-hoc 95% cluster-bootstrap interval over components is +6.2 to +21.6 points, so smaller true effects are not excluded.” [R §8.32; M-0538] C7 solved 32 of the 54 tasks at least once, against 20 for B. Recomputed from the released session table (§8.1), the effect holds within the larger repository alone and points the same way in the smaller: repository B, 50 tasks, 48 against 28 strict successes, 16 tasks better and 3 worse, exact one-sided $p = 133/65536 = 0.0020$; repository A, 4 tasks, 9 against 7, 2 better and none worse ($p =$

0.25; too few tasks to test alone) [our recomputation, `sr7_recompute.py`, 2026-10-05]. The pooled p lies below the family-wise threshold of 0.0036 for the seven confirmatory SR tests, and below $0.025/m$ for any family of up to 29 tests (§5.0).

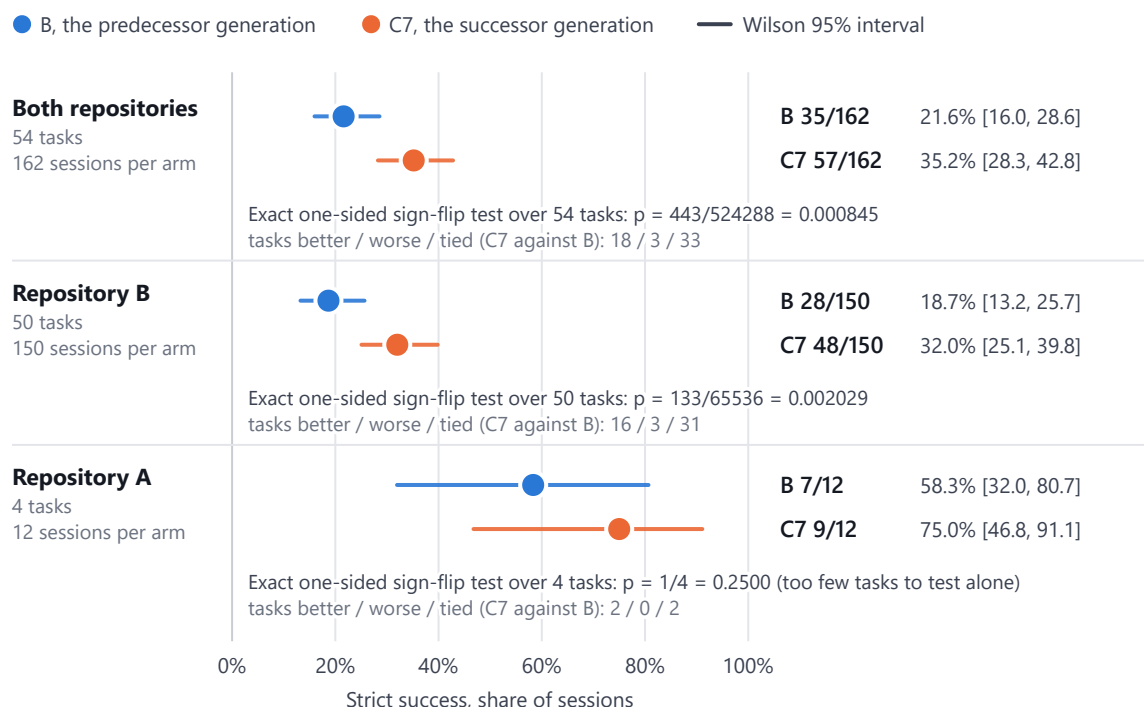


Figure 3. SR7's sealed result at a glance. Strict repair success of B (the predecessor generation) and C7 (the successor) on 54 fresh tasks, 3 rounds per task and arm (162 sessions per arm), one repair model (gpt-oss-20b) at an equal kernel envelope, evaluated 2026-09-24 to 2026-09-25: 35/162 (21.6%) against 57/162 (35.2%) overall; in repository B (50 tasks, 150 sessions per arm) 28/150 against 48/150; in repository A (4 tasks, 12 sessions per arm) 7/12 against 9/12. Counts recomputed on 2026-10-05 from the released session table `sr7_sessions.csv`. The sealed test is a one-sided sign-flip test over the 54 task components: exact $p = 443/524288 = 0.000845$ overall ($133/65536 = 0.0020$ within repository B; $1/4$ within repository A, too few tasks to test alone). The intervals are Wilson 95% intervals that treat the sessions as independent, which they are not (rounds are nested in tasks); they are descriptive and understate the uncertainty when sessions within a task are alike. The record's post-hoc cluster-bootstrap interval for the overall difference is +6.2 to +21.6 points [R §8.32; M-0538].

Cost. SR7 spent \$5.39 of router credit in all, including \$0.63 for the author probe [R §8.19.13; §8.32].

Assessment against the seven conditions. The record assesses SR7 “condition by condition, not by its headline”, and finds each satisfied in its envelope [R §8.19.13; M-1143]. (1) B and C7 are frozen Core generations under one snapshot; C7 passed a restart smoke test before the seal. (2) Target and mechanism were recorded before replay. (3) The pool was unopened, and all 324 assigned sessions are reported. (4) Objects, information, judge, authority and envelope were identical, and all calls and spend were logged. (5) Strict success improved with guards held. (6) The result is reported as one bounded causal result; restart survival rests on a smoke test only, and no replication, second improving generation, substitution or transfer is claimed. (7) The repair instrument was fixed, and no substitution or second-object claim is made. Two conditions

are met in a specific sense, which we state: condition 5's predecessor is B, an earlier Kaizen generation, not the shipped g0; and condition 1's two generations differ in one organ, the repair organ, inside one Core snapshot.

Who did what. Target choice: the declared rule over B's telemetry. Author: the paid model above. Evaluator: the pool generator's hidden tests. Context: the snapshot's author packet (organ contract, diagnosis, examples, organ source, earlier attempts with diffs and failure families, affordance audit). Consultant role: "the researcher" (the AI integration operator, Claude Opus 5.5) built the kernel, the Kaizen rule, the affordance audit, the runners and the designs, "and knew SR5's results and SR6's development data. None of that entered an author packet." The packet did contain the affordance audit, which the researcher derived from the earlier negatives; the quotation concerns results and data, not that design knowledge. Operator: funding decisions, the go-ahead and route amendments before inference [R §8.19.13]. The record's lineage class for this arrangement is *hybrid*: target selection is "rule-autonomous at run time, not designer-blind" [R §8.29].

Licensed. "One bounded downstream subject-improvement effect on fresh large-repository tasks in the synthetic single-line regression family: a rule-selected target, a frontier-model-authored organ change inside Runesmith's Kaizen loop, a cheap repair model at an equal envelope, shareable as an open generation." [R §8.32] Earned name: *bounded subject-improving adaptive runtime* in the exact envelope of the open-source Core's repair organ, a rule-selected target, a frontier-model author (Claude Opus 4.5) (the record's words; on a public coding benchmark this author scores below today's free-tier models [P-07]) inside the Kaizen loop, one cheap repair model (gpt-oss-20b) at an equal kernel envelope, and fresh synthetic single-line regressions in two real Python repositories [R App. B].

Threats to validity. Each of the following is stated in the record or derived directly from its numbers:

1. **The comparator is B, not g0 and not a conventional scaffold.** In SR5, B's point estimate was below g0's (36/180 vs 50/180), so $C7 > B$ does not establish $C7 > g0$. In SR7, C7 was not run against the fixed scaffold that scored higher than g0 in direction in SR6-W (a null result) [R §8.29; §8.30; §8.32]. Rates across different pools are not comparable. By the comparator rule of §5.1.2, the strongest model-free composition of the failure text and trace also belongs in a fixed arm for this task family; none was run, and it is the first study after release (§9).
2. **The author was a paid model** (Claude Opus 4.5), though one that scores below today's free-tier Gemini Flash models on a public coding benchmark [P-07]. Free-class authors (gpt-oss-120b in both) produced no transferring change in SR5 or SR6, and no randomized comparison of author classes was run [R §8.32; M-0535].
3. **One of the five accepted tiers was inert, and there was no ablation.** The module-path tier matches only the hard-coded package prefixes of Runesmith's own two development repositories, so it "was inert throughout the reported PASS"; the gain came from the other tiers, and "no component ablation isolates which tier(s) carried the effect" [R §8.32 erratum; M-0859]. On a user's repository outside those two names, the tier never fires. The record did not check the other tiers for similar coupling.

4. **No replication, no second improving generation and no free-class reproduction** [R §8.32; M-1227]. U1’s replication attempt on public repositories, at about a third of SR7’s size, went in SR7’s direction without a detectable difference (§5.2.6).
5. **Population concentration.** 50 of the 54 components come from one repository (our count from the pool composition), and components within a repository share tests, imports and style. The sign-flip test treats components as exchangeable, and the post-hoc bootstrap resamples components, not repositories [`PROTOCOL.json` via M-0860]. Within the larger repository alone the exact test gives $p = 0.0020$ (above); with two repositories, a test at the repository level is not possible.
6. **Development selection was noisy.** Replays of one fixed organ vary by about 3 successes in 40 here (and by 6 in 32 in SR5), and five of eight attempts were accepted on such replays. The sealed confirmation is unaffected by development noise, but the effect estimate is conditional on the selected generation [R §8.32; §8.29].
7. **Synthetic task family.** Single-line mutations judged by the repositories’ own tests are not natural bugs. “162 sessions” are 54 tasks with three rounds each; they are not 162 tasks [R front matter erratum].
8. **The interval is post-hoc and descriptive,** and it includes values below +10 points.
9. **Hybrid lineage.** The Kaizen machinery and target rule were built by a consultant who knew earlier results, although no such knowledge entered the author packets [R §8.19.13; §8.29].
10. **Multiplicity beyond the declared family.** The family counts confirmatory tests only. The decision to run SR7 after SR6 failed, and the development attempts, are outside it; they were declared to be reported together, as they are here.

What does not follow. “That free-class authors can do the same (SR5 and SR6 did not show it), general repair competence, transfer to natural bugs, or economic value” [R §8.32]; that the gain replicates or compounds over generations, or transfers to other languages, repair models or objects [R §8.19.13]. SR7 also did not test M1’s component gain, named-model substitution, a workflow round trip through the owner’s application suite, or cumulative $A \rightarrow B \rightarrow C$ gain, because B’s gain over A was not shown in SR5 [R §10, prospective bridge program]; the cumulative row of the naming rule is not earned, because “ $A \rightarrow B \rightarrow C$ 7 is one qualified gain, not two” [R App. B; M-0796].

5.2.6 U1, U2 and U2b: C7 on public repositories (2026-10-05 to 2026-10-06)

After the record’s cut we ran four further sealed studies, reported here and in §4.1 whatever they showed: U1, U2, U2b and SI (a fifth, U3, was sealed and withdrawn before any task existed). U1 used SR7’s repair model (gpt-oss-20b), envelope and judge on fresh synthetic single-line regressions (SR7’s task family) from two public Python repositories, click and boltons. Its declared primary comparison was C7 against SR6-W’s fixed scaffold, which U1’s protocol calls “the strongest fixed scaffold the record has defined”, unchanged and driving the same model inside the same kernel at an equal budget (the name U1’s Amendment 4 fixes for this arm is “the same model with a simple fixed script in place of Runesmith’s repair organ”); its declared secondary comparison was C7 against its predecessor B, with the shipped organ g0 run once for the lineage. Four amendments, each sealed before the first session, defined the interpreter, moved the route to hosted gpt-oss-20b (12 of the analysed tasks on one provider, 5 on another),

added tasks and the B arm, and set three rounds and SR7's statistic (an exact sign-flip test over tasks on the per-task means). The remaining credit cut the study from 39 planned tasks to 18 started and 17 analysed (one excluded when two of its sessions were censored by transport failures), three rounds each for C7, B and the scaffold and one for g0, so it had 80% power only for an advantage of C7 of about 28 points over B and about 36 points over the scaffold.

Against the strongest fixed scaffold (declared primary test). No detectable difference, the scaffold ahead in direction: per task, C7 was better on 5, worse on 6 and tied on 6 (20 against 26 of 51 sessions; exact one-sided $p = 0.873$ that C7 is better and $p = 0.2095$ that the scaffold is). Time per strict success was 658 s for C7, 724 s for B and 493 s for the scaffold. U1 does not settle whether C7 repairs more or fewer tasks than the same model driven by that scaffold; a powered comparison of the kind planned in §6.5 would.

Lineage (declared secondary test). No detectable difference between generations, C7 ahead in direction: over three rounds C7 repaired 20 of 51 sessions against B's 15 of 51, the direction SR7 found (exact one-sided $p = 0.1875$); in first rounds, the only round g0 ran, 8 of 17 against 5 for B and 5 for g0 (descriptive; C7 against g0 directly, 6 tasks better and 3 worse, $p = 0.254$).

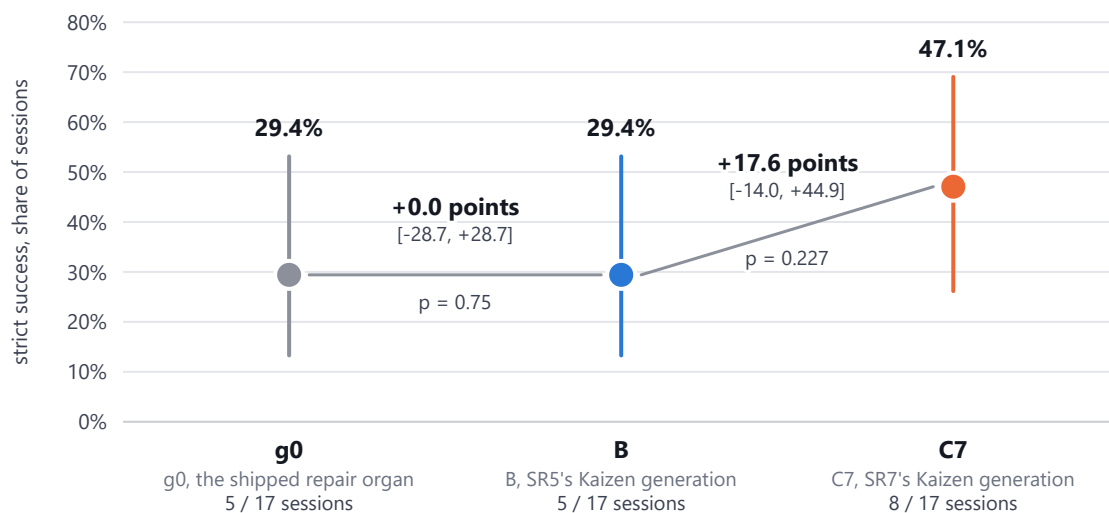


Figure 4. The generation ladder on public repositories (U1, descriptive). First-round strict repair success of the shipped organ g0, SR5's Kaizen generation B and SR7's Kaizen generation C7 on the same 17 fresh synthetic single-line regressions from two public Python repositories (click 10, bolts 7), with one repair model (gpt-oss-20b) and SR7's envelope, on 2026-10-05. The first round is the only round g0 ran. Over three rounds C7 repaired 20 of 51 sessions and B 15 of 51 (U1's declared secondary test, $p = 0.1875$, NULL). Dots carry Wilson 95% intervals. Gains are child minus parent in percentage points, with descriptive Newcombe intervals (sessions treated as independent) and exact one-sided sign tests on first-round discordant tasks (B against g0 from U1's report; C7 against B our computation). No step is significant. Not drawn: U1's declared primary comparison, C7 against the same model with a simple fixed script in place of Runesmith's repair organ: 8 against 9 of 17 first-round sessions, 20 against 26 of 51 over three rounds, no detectable difference with the script ahead in direction (section 5.2.6 and the model-size figure).

Model size (U2). U2, whose protocol was sealed before any U1 outcome existed, repeated the primary comparison on the same 17 tasks with a small and a large model, one round each, on free routes at no cost. With the 2.6B model (liquid/lfm-2.5-2.6b) only Runesmith's organ

repaired anything: 3 of 17 tasks against 0 of 17 (exact one-sided $p = 0.125$; with fewer than four discordant tasks the sealed rule reads this as no difference detectable at this size). With the same 2.6B model SR6-W had found the shipped organ g0 behind the same scaffold (§5.3.7); the task pools and the organ generation differ. With the 120B model (nvidia/nemotron-3-super-120b-a12b, 12B active) both repaired 10 of 17 (NULL), and with gpt-oss-20b, in U1's first round, 8 against 9 of 17. No size shows a significant difference.

Model size on fresh tasks (U2b). U2b, a larger sealed repeat on the 21 tasks U1 never started (three rounds per task and arm, free routes at no cost, sealed and pushed before its first session), put C7 ahead of the strongest fixed scaffold with the 20B and 120B models and level with the 2.6B model: 33 against 23 of 60 sessions with gpt-oss-20b (per task C7 better on 7, worse on 4, tied on 9; exact one-sided $p = 0.084$), 34 against 28 of 63 with nemotron-3-super-120b (5 better, 4 worse, 12 tied; $p = 0.244$) and 9 against 9 of 51 with lfm-2.5-2.6b. No size reached significance (the Bonferroni threshold over the three models was 0.00833), so each is reported as no detectable difference. Across the six model-by-study comparisons after the record's cut, C7 against the strongest fixed scaffold was ahead in three (U2 with the 2.6B model, 3 against 0 of 17; U2b with the 20B model, 33 against 23 of 60; U2b with the 120B model, 34 against 28 of 63), level in two (U2 with the 120B model, 10 of 17 each; U2b with the 2.6B model, 9 of 51 each) and behind in one (U1 with the 20B model, 20 against 26 of 51). None is significant in either direction; this is a descriptive tally, not a registered analysis, and the six are not pooled, because tasks, rounds or models differ between them. Figure 5 shows the 16 tasks complete for all three models (C7 against the scaffold: 9 and 9, 27 and 17, 26 and 22 of 48 sessions). A second script written inside the project recomputed every test, and all quantities match.

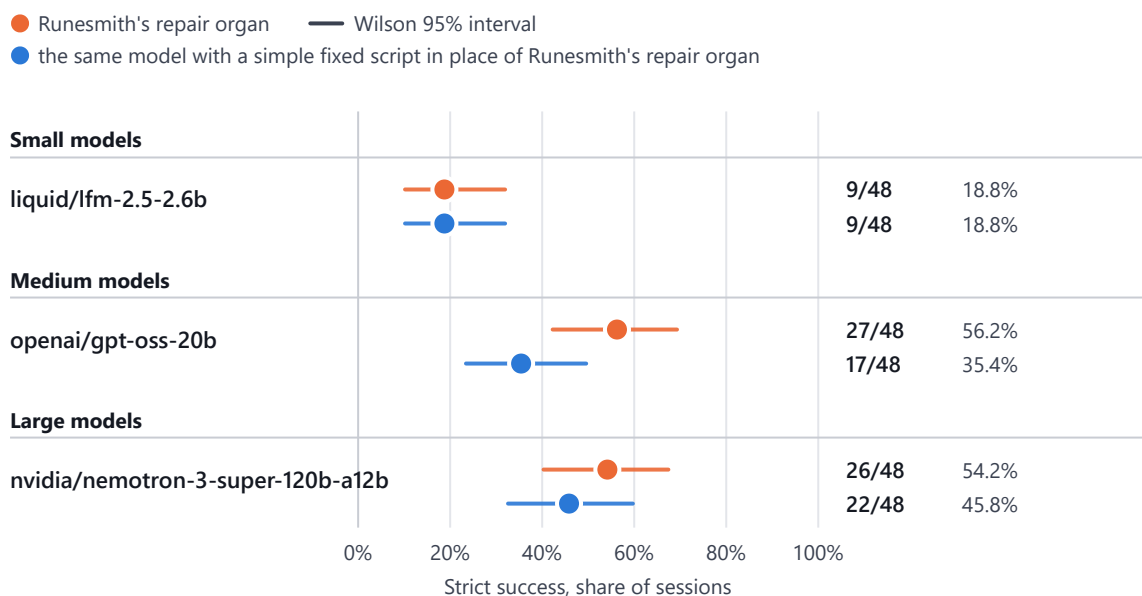


Figure 5. Model size on fresh tasks: Runesmith's repair organ against a simple fixed script, same models (U2b).

Strict repair success over three rounds on the 16 fresh synthetic single-line regressions (from the 21 tasks U1 never started, click and bolts) that were complete for all three models, with Runesmith's repair organ (C7) and with the same model with a simple fixed script in place of Runesmith's repair organ (SR6-W's fixed scaffold), both inside the same kernel, envelope and judge, on free routes at no cost, 2026-10-05 to 2026-10-06. Small: liquid/lfm-2.5-2.6b, 9 and 9 of 48. Medium: openai/gpt-oss-20b, 27 and 17 of 48. Large: nvidia/nemotron-3-super-120b-a12b (12B active), 26 and 22 of 48. The sealed tests use each model's full primary set (section 5.2.6); no size reached significance, so each is no detectable difference. Dots carry Wilson 95% intervals (sessions treated as independent, descriptive).

U1’s protocol, its four amendments, the session records and the reports of U1, U2 and U2b are released with the evidence (`u1_sessions.csv` , `u2_sessions.csv` , `u2b_sessions.csv`).

5.3 Null and negative results

The record’s reading rule applies: “compression must not erase failed candidates or move a dated endpoint after seeing its result” [R §How to read; M-0879]. Every sealed comparison of the record is listed in Table 1; each is then described with its method. The sealed studies run after the record’s cut are reported in §5.2.6 (U1, U2, U2b) and §4.1 (SI). The record’s graduated reading of nulls is used throughout: “If SCG fails, persistent storage was not persistent capability. If substitute SCG fails, the system is an instrument-dependent scaffold. If WMA fails, weak models were not amplified on the favorable tested surface” [R §12.7; M-0723]. A null shows that the tested intervention did not reach its preregistered criterion in its envelope; it does not show that the mechanism class cannot work.

Table 1. Sealed or registered confirmatory comparisons in the record.

Study (date)	Comparison	Result	Verdict
S1-MECHANICS-1 (09-10)	learned vs fresh policy, sealed offline fixture	73 vs 65 verified outcomes; SCG 0.0333, interval $\approx [-0.0335, 0.1001]$	inconclusive
O1-Q (09-10)	source-complete workflow vs deployed object, 60 tasks	54/60 vs 53/60; McNemar $p = 0.5$	not confirmed
H18 v7 (09-13)	registered scripted challenge, successor vs predecessor, 8 opportunities	3 successor-only, 0 predecessor-only, 5 ties; $p = 0.125$	NOT_PASS
M1 c01 (09-20)	candidate 15 vs J26, 16 blocks	13/16; median 0.694862; family 1.279762 > 1.10	FAIL (guard)
M1 c02 (09-23)	candidate 19 vs J26, 16 blocks	13/16; median 0.758123; $p = 0.0106354$	PASS
M1 r03 (09-23)	same, fresh seeds	15/16; median 0.668941; $p = 0.0002594$	PASS
SR1-MIN (09-23)	model-authored selector vs predecessor, 19 components	0.210526 vs 0.210526; $p = 0.623047$	not shown
SR3 (09-23)	experience-derived tool (B) vs Runesmith v1 (C), 20 components	0.425 vs 0.350; 3/0/17; $p = 0.125$	not shown
SR4 (09-24)	sharpened tool vs C, repair-cycle time, 30 analyzable components	slower on 30/30; $p = 1.0$	not shown
SR5 main (09-24)	Kaizen generation B vs g0, 60 tasks $\times$ 3 rounds	36/180 vs 50/180; $p = 0.979$	not shown
SR5b (09-24)	same, 46 medium-repository tasks $\times$ 3 rounds	58/138 vs 60/138; $p = 0.685$	not shown
SR6-W (09-24/25)	shipped organ g0 vs strongest fixed scaffold, one 2.6B model, 68 tasks	5/68 vs 12/68; $p = 0.994$; mirrored $p = 0.0327$	NULL
SR7 (09-24/25)	C7 vs B, 54 tasks $\times$ 3 rounds	57/162 vs 35/162; $p = 0.000854$ (exact 0.000845)	PASS

SR6-KAIZEN2 closed before confirmation and is not a confirmatory test (§5.3.6).

● PASS ○ did not pass (verdict at right) ○ registered, not in the count ◆ SR6-W mirrored p (opposite direction)

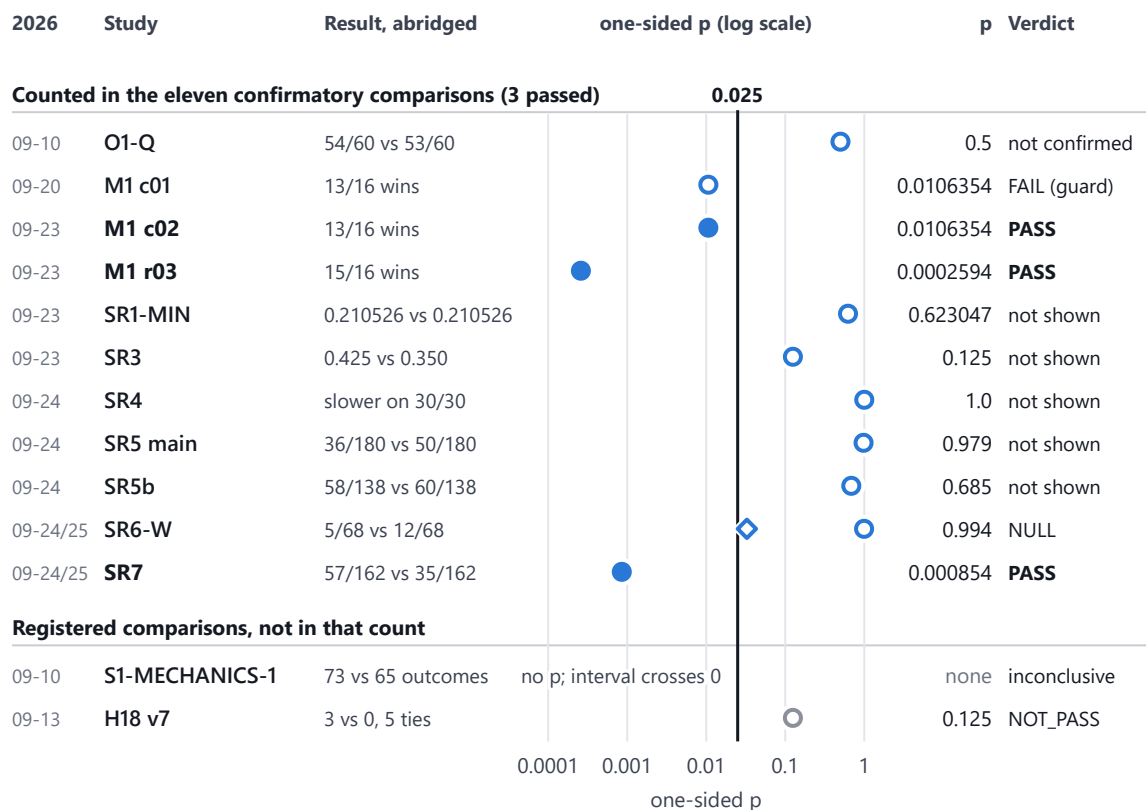


Figure 6. Every sealed or registered comparison in the record, with its p and verdict. One-sided p on a logarithmic axis for the eleven confirmatory comparisons of Table 1 (O1-Q, the three M1 cohorts, SR1-MIN, SR3, SR4, SR5 main, SR5b, SR6-W and SR7) and, set apart, the two registered comparisons that are not in that count (S1-MECHANICS-1 and H18 v7), with a line at 0.025. Studies ran from 2026-09-10 to 2026-09-25. Numbers are those of Table 1 (§5.3) and Appendix A; the p of M1 confirmation 01 is from the results table of §5.2.4. A dot's position relative to 0.025 is not its verdict: each verdict is the one the study's own protocol assigns (M1 confirmation 01 failed a preregistered guard at $p = 0.0106354$; the M1 allowances were 0.025, 0.0125 and 0.00625; SR6-W's $p = 0.994$ is for $\text{SUIT} > \text{DIRECT}$, and its mirrored p of 0.0327 was above the declared 0.025). Filled dots are the three that passed. S1-MECHANICS-1 reports an interval, not a p.

5.3.1 The original bridge: a capacity no-go

The first design for a downstream subject and named-model bridge (2026-09-23) was a packet-selector study: a frozen deterministic selector A against a change B authored by model X, consumed by a different named model Y, with a conventional localizer C, an equal token cap and gated stages [R §8.24; M-1152]. Its no-go rule required stopping before B was commissioned if a patch-informed oracle lifted strict success over A by less than +0.10, or if a prespecified simulation could not reach 90% power at a true +0.10 effect. The design stopped with a capacity and calibration no-go. From the researcher's feasibility audit: about 80–230 qualified components and 3,400–9,700 Y calls would have been needed; the reserved pool had 25 metadata-level components, of which about seven were expected to qualify (80% range 4–11); the exploratory repeated interval's one-sided error rates were 0.031–0.047, above the allowed 0.025, and an abrupt-drift scenario reduced joint coverage to 0.633; free-provider capacity was

inadequate. “These findings explain an investment/design no-go, not evidence of an absent subject effect.” [R §8.25.1; M-0198, M-0482] The +0.10 material threshold and the ± 0.10 equivalence margin had been fixed by the lead investigator “before opened oracle scoring” [R §8.23; M-1151].

5.3.2 SR1-MIN: a sealed null with known defects

Question. Does a model-authored successor selector improve a weaker consumer’s strict repair success on fresh single-line-mutation tasks? [R §8.25; M-0096]

Design. Single-line operator mutations injected into test-covered functions of copied repositories outside the original reservation, outside live PRHat and outside the lab’s own project. Evaluation used one repository with 19 source-file components, unseen by the author at acquisition; 46/46 evaluator controls passed. Three sessions per arm per component (114 sessions, 57 per arm, “not 114 independent tasks”). A frozen two-call cockpit (navigation, then edit) with 12,000-byte navigation and 32,000-byte edit caps enforced outside the mutable selector; candidate patches went to a separate strict evaluator. Author X: `openai/gpt-oss-120b`; consumer Y: `openai/gpt-oss-20b`, both served by one commercial provider, so this was a same-family pair and “not cross-family confirmation”. Seven dated prospective amendments; a first acquisition lineage was rejected partly because of investigator feedback that omitted the preceding edits; a corrected lineage qualified at its first answered attempt (oracle delivery 4/4, equal to A). Before evaluation, the researcher recorded an unresolved mismatch between the symbols the selector advertised and what its extractor could resolve: “Qualification was therefore a limited proxy, not established end-to-end readiness.” [R §8.25.1; M-0090]

Endpoint and test. Mean strict repair success over components; sealed one-sided exact sign-flip test on component mean paired differences, threshold 0.025.

Result. Both arms 0.210526; mean paired difference 0.000; five successor-favorable, five predecessor-favorable and nine tied components; $p = 0.623047$. Descriptive 95% t interval approximately $[-0.119754, +0.119754]$; neither the ± 0.10 materiality nor equivalence was resolved. A restart of the route layer caused connection refusals, first classified as model failures, in 21 sessions; an outcome-blind correction requeued them once under the unchanged sealed code, and corrected and uncorrected analyses agreed. A post-hoc sensitivity analysis censoring eight child-process failures (later traced to an output-encoding defect) kept a zero mean difference and $p = 0.623047$ over 18 components. In 14/57 successor sessions the selector advertised dotted class-method names that its extractor could not resolve [R §8.25.1; M-0063].

Power. An erratum applies: the power figures once quoted for this study (about 0.55–0.60 at +0.20 and 0.15–0.20 at +0.10) were computed for simulated component counts of 40, 60, 80 and 100, “not for the 19 components SR1-MIN actually evaluated”; no simulation at $N = 19$ exists, and “the actual achieved power of SR1-MIN’s sealed test was lower than these quoted figures state” [R §8.25.1 erratum].

Does not show. Equivalence, the absence of smaller effects, a single cause of the null, or harm. Threats: synthetic tasks, very low power, a same-family pair, within-repository dependence, a limited qualification proxy, and instrument defects (encoding, transport refusals, unresolved

symbols). Cost: 563 provider attempts and 217 answered calls, 787,339 input and 545,967 output tokens, about \$0.066 of subscription credit; investigator-agent usage was separate. “No claim of total experimental cost below seven cents is licensed.” [R §8.25.1; M-0037]

5.3.3 SR3-ADAPT: an experience-derived tool, directionally positive, not shown

Question and hypothesis. The researcher’s diagnosis after SR1-MIN was a “structural ceiling”: every earlier successor was a generic mechanism (context selection, retry policy, source re-quoting) that a competent conventional workflow already contains, so a flawless acquisition “can at best tie” it. A successor can only win with information the conventional workflow lacks: knowledge distilled from Runesmith’s own experience of an environment. The record presents SR3 as testing the instrument–substrate hypothesis “directly” in that sense and labels the diagnosis “not a general theorem” [R §8.27.1; M-1172, M-0493].

Environment and pool. Synthetic single-line operator regressions in test-covered function bodies, qualified by the repository’s own targeted tests (exit 1 on the mutant) and two evaluator controls; one task per source file; tests hidden from all models; issues follow a CI template with failing test identifiers and a sanitized excerpt. A fresh pool from nine repositories unused by earlier studies qualified 32 tasks in seven repositories, split by HMAC order into an experience split (12 tasks, three repositories) and an evaluation split (20 components, four repositories) never shown to the author [R §8.27.2].

Headroom gate. Before any inference, an investigator-written informed search C' that used only the public signal was required to reach at least 50% on the evaluation split. It reached 19/20 strict (20/20 public passes); the one public pass that failed the held-out tests shows that the strict check catches overfitting [R §8.27.2; M-1177].

Arms. C: Runesmith v1 (navigate and edit through the frozen selector, then public-signal feedback retries, at most 5 model calls). B: the learned tool runs first in the confined sandbox (at most 40 runs, host-verified); if it finds no verified candidate, B’s loop is byte-identical to C’s, so B takes C’s outcome for that component and round (declared response sharing). B0: the tool alone, zero model calls. A: the SR1-MIN cockpit (secondary). Consumer: `gpt-oss-20b`, pinned, without fallback.

Transduction stage. C ran once on each experience task and solved 8/12; each matured episode entered a ledger with its reference fix. Author X received a fixed tool contract, an exact runtime-shaped state fixture, the ledger and its own earlier iterations, with at most 6 answered iterations; the best qualified iteration (no child errors, at least 25% strict on experience) was frozen. Iterations 1–3 (Gemini 3.5 Flash Lite) each solved 3/12; iteration 4 was lost to an operator flag bug and not counted; iterations 5–7 (`gpt-oss-120b` after an author-route amendment) solved 2/12, 2/12 and failed on a syntax error. Iteration 1 was frozen by the earliest-tie rule. The protocol was sealed at 2026-09-23T21:58:02Z, with the expected result registered in advance as “likely null” [R §8.27.3; M-1181].

Result. Strict success over the 20 fresh components: C 0.350; B 0.425; B0 0.200; A 0.250; C' 0.950. Primary B vs C: +0.075, 3 components improved, 0 worsened, 17 tied; exact one-sided $p = 0.125$; not shown at $\alpha = 0.025$. Secondary B vs A: +0.175 (9 better, 4 worse, 7 tied; $p = 0.165$). Descriptively, C used 114 model calls for 14 solves (8.1 per solve) and B 86 calls for

17 solves (5.1 per solve); all 4 tool public passes were strict successes [R §8.27.4; M-0085]. With three nonzero component differences all positive, the smallest attainable exact one-sided p is $1/8 = 0.125$. The design could not have reached 0.025 with fewer than six improved components and none worse; this is our derivation, not the record's statement. Under response sharing, the per-solve call counts are not separate evidence of tool efficiency.

Reading. “A model-authored, experience-derived tool installed as generation B produced a small, harm-free, directionally positive change on fresh repositories ... not statistically established.” The researcher's inference: the free-model authors found *what* to change (an operator-toggle family, unprompted) but not *where to look first* [R §8.27.5; M-1182].

5.3.4 SR4-SHARPEN: keep-best on seen tasks memorizes

Question. Can Runesmith sharpen its own tool with a keep-best loop so that repair cycles get faster on fresh tasks? The lead investigator broadened the definition of subject improvement for this study (2026-09-24): doing a task faster counts as improvement. The primary endpoint is therefore repair-cycle time [R §8.28].

Design (written before any SR4 task or call). A candidate replaced the incumbent only if it scored strictly better on SR3's 32 opened tasks (strict solves first, then tool time); at most 16 answered iterations. Fresh evaluation: 46 tasks from new mutation sites in three repositories never shown to this author. Primary: repair-cycle time of B2 (the final tool, then the standard loop) against C, one-sided sign-flip with 10^6 seeded flips, $\alpha = 0.025$; guard: B2's strict successes at least C's, with at most one tool regression. An informed non-inference search reached 43/46 strict on the fresh pool. Sealed 2026-09-24T02:03:30Z [R §8.28].

A prediction registered before the freeze. At 2026-09-23T23:50Z, before SR4's freeze, the record noted that an accepted development tool “literally contains distinctive identifiers from the matured reference fixes of 13 of the 32 development tasks” and predicted fresh coverage “far below its development score” [R §8.28].

Result. The development curve rose to 29/32 at iteration 16; by then the tool opened with hand-written rules for specific development fixes and carried reference identifiers for 22/32 development tasks. On fresh tasks: B2 was slower on 30/30 analyzable components (C 87.3 s, B2 98.4 s; mean -11.07 s); $p = 1.0$; guard held (17 vs 17 strict successes, 0 tool regressions). The tool alone passed **1/46** fresh tasks against 29/32 on its development set. A provider's daily allowance ran out during the run: 50/92 C sessions were censored, and 16 components with no uncensored round were excluded, leaving 30 analyzable. The record's claim that the shared queue “adds variance to the cycle-time primary, not bias” is asserted, not shown [R §8.28; M-0071, M-0506, M-0507].

Lesson. “A keep-best loop scored on tasks whose answers its author sees learns to memorize those answers, and the memorized tool adds only overhead on fresh repositories. This is a methodological negative worth reporting in the paper: development scores on seen tasks are not evidence of capability. Runesmith Core's Kaizen therefore always scores on held-out validation.” [R §8.28; M-1186]

5.3.5 SR5 and SR5b: a free-class Kaizen generation, not shown in either population

Question. Does the self-improvement loop (find the weakest point in one's own telemetry, let a model rewrite the organ that owns it, keep the change only if the successor does better on unseen work) produce a successor that is better on fresh tasks? The study was run on the open-source Core [R §8.29].

Design. `STAGE0_DECLARATION.md` written before any SR5 opportunity; seeds drawn 2026-09-23T23:26:02Z. Only organ files may change; the kernel stays fixed. A = repair organ g0 (`gen-e510c73e230f`). Envelope for both arms: at most 5 calls, 8,192 tokens, 45 signal runs, 64,000 request bytes, 1,800 s; repair instrument `gpt-oss-20b`; author `gpt-oss-120b` with high reasoning. Kaizen protocol: A's baseline on SR3's 32 opened tasks; diagnosis from A's telemetry; target = rank 1 of the declared rule (the largest lost-opportunity family of at least 10%, else the largest stage); author returns hypotheses, mechanism, quantitative prediction, falsifier and edits; static check, confined smoke test and development replay; keep-best, at most 8 answered iterations, patience 4. Confirmation: A vs B, 3 rounds, one HMAC-ordered interleaved queue; primary strict success, per-component mean difference, one-sided sign-flip, $\alpha = 0.025$; guards on organ errors, false promotions and (after Amendment 2) seconds per strict success at most $1.5 \times A$. "Target selection is therefore rule-autonomous at run time, not designer-blind. The organ change is model-authored." Lineage class: hybrid [R §8.29; M-0159].

Amendments, all before any confirmation outcome. Amendment 1 (23:51Z, triggered by the SR4 memorization observation) split the 32 development tasks by HMAC into 16 experience tasks and 16 validation tasks whose references the author never sees; it was written nine minutes into A's baseline run, and the split is HMAC-assigned. Amendment 2 replaced a median-cycle guard that had a construct problem with the per-strict-success time guard; the record discloses that it "was informed by development data and is blind to all confirmation data". Amendment 3 added a second, separately sealed population, SR5b (46 medium-repository tasks never seen by any SR5 author), with the claim rule that neither population rescues the other and "in at least one declared population" requires $p \leq 0.0125$ [R §8.29; M-1191, M-1192].

Development. A's baseline: 25/64 strict (0.391); 36% of navigation calls returned an unusable answer, after which A ended the opportunity with 4 of its 5 calls unused. Target: `navigation_output_failure` (13/32 = 40.6% of experience records). The best iteration added a deterministic fallback read set when navigation fails and scored 17/32 on validation against A's 11/32, with a bar of 14; B = `gen-a4415db24a53` was frozen on 2026-09-24T02:03:49Z. Replay noise: iterations 4 and 5 were functionally identical organs and scored 10/32 and 16/32 on the same tasks, so "keep-best decisions at this sample size are noisy" [R §8.29; M-0058]. Two predictions were recorded before any confirmation outcome: at 01:08Z, that the fallback's first four indexed files would rarely contain the defect in repositories of 78–181 source files; at 07:08Z, from development data, that B's direct contribution was about +2 of 32 and that "a primary p unlikely to reach 0.025" should be expected [R §8.29; M-0141, M-0515].

Main confirmation. Sealed 2026-09-24T03:06Z on 60 fresh tasks from two large repositories, repository C (32) and repository D (28); 360 sessions. A capacity deviation paused the run: the route layer's own seeded daily ceiling for the repair model (1,200 requests) had been spent by other stages; it was raised by the operator with no design change, and the sealed queue resumed with all sessions uncensored [R §8.29; M-1193]. *Result:* A (g0) 50/180 (0.278); B 36/180

(0.200); components better/worse/tied 10/15/35; one-sided p for $B > A = 0.979$ (Monte Carlo; exact 0.978724); seconds per strict success 337 vs 670, so the time guard failed; organ errors 0/0, false promotions 3 vs 5 [R §8.29; M-0031].

SR5b. 46 medium-repository tasks, 276 sessions, 0 censored. A 60/138 (0.435), B 58/138 (0.420); 14/15/17; $p = 0.685$ (exact 0.685363); time guard held. Verdict: “SR5’s downstream subject effect is not shown in either population.” [R §8.29; M-0201] A same-project recomputation reproduced both exact p -values [R §8.29; M-0158].

Exploratory mechanism analysis (after the sealed analyses). B’s change acts only after a failed navigation call. On the large pool its fallback rescued none of 26 navigation failures (the first four of about 115 indexed files never held the defect); the rescue rate fell with repository size, 2/6 in development, 2/26 in SR5b and 0/26 in the main pool. All 111 strict successes across these datasets came from sessions that read the defective file. The record infers that the negative point estimate is chance variation in unchanged behavior, and that the development gain was mostly noise selected by keep-best [R §8.29; M-0520, M-0521]. Replaying the sealed queue through the Core’s online trial with its defaults, the trial rejects B at the maximum sample (A 16/60 vs B 9/60), so in the open-source loop this successor would not have been activated; this is a retrospective replay, not an online deployment [R §8.29; M-0115].

Boundary. “Evidence that free-class authors did not find a transferring mechanism in these attempts, not that they cannot.” [R §8.34; claim map]

5.3.6 SR6-KAIZEN2: a second free-class generation, closed before confirmation

With an improved author packet (diffs of earlier attempts, a diversity rule and the affordance audit), a free-class author (`gpt-oss-120b`) attempted a successor to B. Development used SR5’s 60 main-confirmation tasks, opened only after SR5’s analysis (20 experience, 20 validation, 20 unused); the freeze bar was $C \geq B + 4$ on 40 validation opportunities. B’s fresh baseline was 13/40, so the bar was 17. The four attempts scored 10, 10, 11 and 9 of 40. Attempts 2–4 used the execution trace, which the affordance audit had listed as untried; they read the defective file more often (21–23 of 40, against B’s 15) and repaired no more. The campaign closed at 2026-09-24T13:39:29Z as `CLOSED_NEGATIVE_BEFORE_CONFIRMATION` ; the reserved 54-task pool was untouched; spend was \$0.41 [R §8.31; M-0202, M-0142]. An inference that the trace variants harmed the edit stage was retracted after B, replayed against itself, scored 13 → 10 on the same opportunities: the deficits are within replay noise [R §8.31; M-0533]. SR6 “does not establish that free-class authors cannot find a transferring mechanism” [M-0202, not-claimed field].

5.3.7 SR6-W: weak-model amplification, null with direction favoring the scaffold

Question. Is the suit (the shipped repair organ `g0`, with model-led navigation) what helps a weak model, against a strong equal-budget baseline rather than “a one-shot prompt”? This is the preregistered weak-model amplification test that the record’s definition of WMA requires [R §8.30; M-1197].

Design. Seeds drawn 2026-09-24T03:58:31Z; protocol sealed at 04:00:52Z (an erratum corrects the narrative’s 04:07Z; both precede the 04:08Z start). Instrument: `openrouter:liquid/1fm-2.5-2.6b:free` , a 2.6-billion-parameter free model. Both arms ran under the same kernel envelope as SR5. SUIT: the shipped organ `g0`. DIRECT: “the strongest fixed scaffold without model

navigation”: lexical retrieval of whole files by issue-to-path and symbol overlap up to the same 32 KB cap, the same feedback loop, and all five calls spent on edits. Tasks: the 68 fallback-pool tasks not used by SR5, fresh to every study; one round per arm (136 sessions). Primary: strict success, per-task paired difference, one-sided sign-flip for SUIT > DIRECT at $\alpha = 0.025$, with the mirrored test (DIRECT > SUIT) declared as “negative WMA” if its $p \leq 0.025$. An outcome-blind dry run of the sealed analysis on synthetic sessions returned the declared verdict in four scenarios [R §8.30; M-0523].

Execution. The main pass ended with 89 opportunities with outcomes and 47 censored by transport (free-model timeouts and two route-layer restarts). The declared one-time re-queue was refused by the free tier’s daily limit; after a rollover defect in the route layer was fixed, the 47 were re-run on 2026-09-25 (02:20:47Z–04:16:33Z). No outcome was read during the stall. The record does not report the arm split of the 47 [R §8.30; M-0128].

Result (addendum 2026-09-25T04:18:11Z). All 136 sessions present, none censored, all 68 pairs counted. Strict successes: SUIT 5/68, DIRECT 12/68 (–10.3 percentage points). Discordant pairs: SUIT only 2, DIRECT only 9; both or neither 57. SUIT > DIRECT: exact $p = 0.994$. Mirrored, DIRECT > SUIT: $p = 0.0327$, which is above the declared 0.025, so the verdict is **NULL**, not “negative WMA”. A same-project recomputation matched every figure. DIRECT had 2 false promotions, SUIT none. Per strict success: SUIT 6,030 s and 63.8 calls; DIRECT 2,792 s and 25.3 calls [R §8.30; M-0105, M-0525].

Reading. “With this 2.6B model, g0’s model-led navigation did not help, and deterministic retrieval did better in direction.” That navigation calls spend part of a five-call envelope that a weak model needs for edits is a plausible reading, “not tested here”. “SR6-W did not test C7, so no claim about C7 with weak models follows.” “This is not evidence that a suit cannot amplify a weak model.” [R §8.30; M-1199] The scaffold-favoring difference is nominally below 0.05 and did not meet the preregistered 0.025 criterion. SR6-W compares the shipped organ with a strong fixed scaffold on the same kernel; it has no arm without any scaffold, and it is not a comparison of a model with and without Runesmith (§6.5). It is also not the record’s planned W experiment, a factorial design powered at up to 400 clusters per arm [R §10, card W].

Two earlier probes motivated SR6-W and license nothing: on a trivial kernel fixture, g0 failed 0/3 and the SR5 successor passed 3/3 with the same 2.6B model; on 8 opened development tasks, A repaired 4/8 and B 2/8 [R §8.29; M-0108, M-0091].

5.3.8 S1-MECHANICS-1: the only SCG measurement, inconclusive

On a sealed offline deterministic fixture with no provider (2026-09-10), the full learning path ran once: an immutable randomized acquisition manifest committed before effects, pre-action predictions and decision records, a frozen outcome schedule, exact observations, strict identity resolution, a typed residual, a credit link, one bounded policy update, separate transition-table qualification, atomic activation, database close and reopen, use on a sealed holdout, and a full projection rebuild. Seventy-one mature learning rows produced one update of approximately 0.3086. Under equal 120-request holdout budgets, the learned and fresh policies yielded 73 and 65 verified outcomes; paired SCG 0.0333 utility per unit; 95% interval approximately [–0.0335, 0.1001]. “The interval crosses zero and the registered worthwhile delta, so the outcome verdict is inconclusive.” [R §11; M-1271, M-1272] The mechanics worked (persistence and reuse bound

to identity, equivalent replay, idempotent retry, journal integrity, missing outcomes not coded as zero). The effect was not shown: “the minimum learning claim ... has not been shown for Runesmith” [R §10, card M; M-1262]. A note on units: $(73 - 65)/120 = 0.0667$, so the record’s 0.0333 implies a different unit (for example 240 paired units, or a utility weight of one half), which the record does not state; we report the record’s figures as given, and its verdict rests on its own interval.

5.3.9 The downstream subject campaigns of 2026-09-13 to 2026-09-20

Before SR7, the original downstream criterion was pursued in a sequence of campaigns on the earlier kernel. None produced a frozen successor that passed a registered fresh-object comparison [R §8.19.6–8.19.9; M-1117]. Pilots compared frozen predecessor A and successor B over assigned object opportunities, reporting B-only wins, A-only wins, ties and a one-sided exact p . None was powered.

- **D1/D2.** D1: 2 opportunities, one apparent win per arm, inconclusive. D2: 8 opportunities, 1 B-only, 0 A-only, 7 ties, $p = 0.5$, declared **invalid** because four supposedly fresh executions reused cached model-job identities and integrity labels conflated target failure with regression [R §8.19.6].
- **C2–C9.** No directory produced a successor generation: two aborted before a candidate, two kept evidence packets without a nomination, four ended without a valid nomination [R §8.19.6].
- **Owner-specified C1–G1 chain.** Five isolated generations changed one script; downstream pilots were ties or near-ties (C1: 12 opportunities, A-only 1, B-only 0, 11 ties; F1: B-only 1, A-only 0, 11 ties, $p = 0.5$; G1: 8 ties). The family was retired as `NO_GO_AS_LEADING_S2_TREATMENT` [R §8.19.6; M-1112].
- **H7, the first free-model successor.** Twenty-one recorded attempts produced nineteen rejected candidates and two isolation-stage promotions. Round 4: 2 B-only, 1 A-only, 9 ties ($p = 0.5$); round 5: 1 B-only, 0 A-only, 7 ties ($p = 0.5$); inconclusive [R §8.19.6].
- **H8–H16.** Every tested exact edit was retired by outcome feedback (ties, or one tail favoring the predecessor) [R §8.19.6].
- **H18.** Externally authored by the GPT-6 Astra consultant from public material only (“external authorship is not subject improvement”). Under a prospectively registered scripted challenge whose rule required at least six successor-only wins, no predecessor-only wins, $\Delta \geq 0.50$ and one-sided $p \leq 0.025$, the result was 3 successor-only, 0 predecessor-only and 5 ties ($\Delta = 0.375$, $p = 0.125$): `NOT_PASS` [R §8.19.6; M-0193].
- **Custody campaign (2026-09-14).** Twelve candidate successors (6 + 3 + 3 in three lineages) all failed mechanism qualification, with 0/12 entering a downstream pilot; “negative acquisition evidence, not a twelve-candidate causal effect estimate”. An A/A instrument-variation check (both arms the predecessor) gave 2 “B” wins, 0 A wins and 10 ties. A measurement mediator was replaced after one candidate had been seen under the old version, so later results are “corrected re-evaluations”, not preregistered ones. The sealed closure protocol is internally inconsistent about sample size (64 positives plus four controls against 96 trajectories) and carries a digest transcription error; it is retained unchanged [R front matter, 2026-09-14 extension; §8.19.7].

- **V1–V6 staged authoring (2026-09-19).** Six sealed lineages with 12 logical calls each produced no candidate; end-to-end execution failed at the edit → test → revision → submission transition [R §8.19.8].
- **JEV and J1–J68 (to 2026-09-20).** J20 was invalid because of two controller defects. J42’s prospectively declared six-pair development screen gave A 5 strict promotions and B 4 (one A-only win, no B-only win). J52, J62 and J67 passed 554 restart tests and then failed separate behavior assays. J68 ended with no qualified candidate after 24 calls and 1,646.639 seconds. “These outcomes are neither approximately one hundred independent subject-effect trials nor evidence that all possible subject mechanisms have failed.” [R §8.19.9; §15]

The postmortem names competing explanations (author capability, context burden, decomposition, tool access, evaluator fidelity, provider availability) and states that the campaign “did not experimentally isolate their causal contributions” [R §8.19.7].

5.3.10 Recovery-policy development and an aborted diagnosis study

SR2 (opened development, 2026-09-23): a conventional one-retry pilot recovered 7/12 historical rejected edits; a model-authored policy recovered 4/8 selected opened cases, the same four that the conventional pilot recovered, using 13 calls against 8; a refined policy recovered 2/3 using 5 calls. These are “different denominators and stages, not a pooled success rate, fresh confirmation, or superiority over ordinary feedback engineering” [R §8.25.2; §8.34]. A source-aware recovery study then stopped before any comparison: its first version was censored by overload, and its second passed 24/24 action-contract checks but failed both source-diagnosis checks (0/2) because it treated a list as a dictionary. A post-result audit found that the investigators’ own author packet had mistyped that field, so “the failure therefore must not be attributed solely to intrinsic model capability” [R §8.26; M-0491].

5.3.11 Field runs without measurable learned utility

- **A 177-cycle commissioning run of a legacy social-posting loop** (2026-09-04 to 09-07; retrospective). 177 machine-receipted cycles, 167 of them decisions served by free inference routes; 935 engagement candidate rows and 0 interactions; 44 code experiments, 1 retained; conversions 0. “Persistence, proposal variety, and rollback were present. Evidence-bearing learning was not.” The loop did not implement the required substrate, so the receipts “neither confirm nor falsify the instrument–substrate hypothesis” [R §8.2; M-0338].
- **PRHat Phase C and Phase D.** Phase C sealed with 78 reports, 27 recorded uses, 5 artifacts and 0 utility warrants; the object funnel stood at 35 discovered, 35 enriched, 34 drafted, 6 pitched and 0 responded. Phase D assigned each ordinary cycle with probability 0.5 to a learned or a strong fixed procedure, and sealed after 17 reports (8 learned, 6 fixed, 3 observations) with the funnel unchanged; the treatment contrast was “unidentified for practical purposes” [R §8.4; §8.6; M-0347].
- **A micro feasibility pilot** (2026-09-05): on nine paired hidden-oracle clusters, a fixed direct scaffold scored 8/9 and a persisted-episode arm 9/9 with an OpenAI instrument at low effort; repair was never exercised, resources differed, and $n = 9$ licenses nothing about the hypothesis [R §8.1; M-0112].

- **First lifecycle-controlled field start.** Ten terminal task records, one admitted provisional objective that was semantically empty, and zero funnel motion; restart and failure-classification mechanics worked [R §8.15].

5.3.12 What the negatives changed

The negative results changed the design, and the changes are visible in the released system and protocols. The E4 falsification produced the equal-information comparator rule (§5.1.2). The red-team removed the substrate reading of the synthetic pilots (§5.1.2). SR4 produced the rule that Kaizen scores only on held-out validation. SR5 produced the noise-aware freeze bar and showed that an online trial would have refused its successor (§3.4, §5.3.5). The free-class authors' repetition of one mechanism produced the affordance audit (§3.4). The record's summary: the negatives "expose weak comparators, semantic leakage, unequal citation gates, selected development panels, activity proxies, high incumbent ceilings, malformed probes, unsupported obligations, visible-versus-hidden endpoint ambiguity, repeated-target waste, and scientifically meaningful refutations misclassified as runtime failure before those errors become product claims" [R §15; M-1289].

5.4 Supervised field construction and out-of-the-box journeys

The work in this section is observational. None of it is preregistered, controlled, blinded or a human user study [R §8.36; §8.37]. It is reported because it exercises the system on real projects and a live service, and because it exposed defects and design hypotheses that the sealed studies could not. Unattended operation is reported in §5.5.3.

5.4.1 Supervised field construction (2026-09-25 to 2026-09-27)

What happened. Runesmith, with configured model instruments, built the first increments of two new projects from empty folders: repository F (local-first support case handling) and repository E (records for audience, offer and hypothesis research). A third, existing project was observed read-only. Claude Opus 4.5 authored each project's plan (eight milestones each); repository F's trainer added one compatibility milestone. A trainer role, held by several AI sessions in turn, improved Runesmith's reusable machinery and wrote the owner acceptance checks; "The trainer did not write project implementation: every applied file traces to a model named in its `DRAFT.json`'s `drafted_by`." [R §8.36]

Applied changes. Sixteen, all verified against acceptance and applied through the guarded delegated-build path. Repository F had 7 (milestones m1–m6 plus the compatibility milestone; final state 61 project tests plus 16 cumulative owner checks). Repository E had 9 (m1–m4 with model-proposed prerequisite slices; at m4, 62 project tests plus 20 owner checks). **Final authors: Claude Opus 4.5 (a paid model) 10; Kimi K3 on a free route 4; Gemini 3.1 Flash Lite on a free route 2.** Ten of the sixteen applied changes were therefore authored by a paid model and six by free routes [R §8.36].

Acceptance rejected drafts that passed their authors' own tests. At least 13 drafts passed every project check, including tests their authors wrote, and failed the trainer's cumulative acceptance; none was applied (repository F 4, all by Claude Opus 4.5; repository E 9, of which 6 by Gemini 3.1 Flash Lite and 3 by Claude Opus 4.5). Examples include legacy records vanishing

under a status filter, and two paths with identical content merged so that one source path was erased. The count is a lower bound: about 40 older records use a format the recount did not follow. Several failures trace to public contracts the trainer had not specified, and acceptance was often written after the trainer had read the drafts, so “these are not all author failures”. The record calls this field corroboration of its evaluator principle, “observed, not a controlled comparison” [R §8.36; §15].

Free routes. Free routes produced many rejected attempts: syntax errors, regressions of existing commands, hallucinated or no-op exact edits, ambiguous answers. On one repository E milestone, three free attempts at the whole milestone produced no acceptable change; after a trainer-adopted decomposition, three of the four applied pieces were authored by Claude Opus 4.5 and one by Gemini 3.1 Flash Lite. That episode is “confounded by trainer clarifications and escalations” and is recorded only as the motivation for a hypothesis about adaptive task granularity [R §8.36; §12.12(a)].

Nonclaims. Not unattended or autonomous: the trainer selected instruments, wrote acceptance, clarified requirements and adjudicated promotion. No real customers, messages, market experiments or business outcomes. Not sealed or preregistered. Neither project was finished: repository F had 7 of 9 plan items done, and repository E 4 of 8 top-level milestones (its fifth milestone’s drafts parked as needing revision). Not Runesmith improving itself: runtime changes were trainer-authored, and Kaizen was off in both training homes [R §8.36; §8.34, rows 18–21].

Cost: two figures with different scopes. The record gives \$0.329 for repository F (4 of 6 requests costed) and \$0.788 for repository E (12 of 13 costed). These cover the “retained accounting windows”, and one failed job’s cost is unknown, not zero [R §8.36]. The release’s numbers check summed the cost estimate over every model-call record in each home’s ledger and obtained \$1.82 for repository F (15 calls, 4 without a cost figure) and \$2.04 for repository E (56 calls, 6 without a figure) [numbers check, 2026-10-04]. The two pairs differ in scope: retained windows of 6 and 13 requests against whole ledgers of 15 and 56 calls. Neither is a full project cost, since the trainer sessions are not metered in either. We do not choose between them.

5.4.2 Out-of-the-box journeys (2026-09-27 to 2026-10-01)

Design. Thirteen journeys, a reference run R1 and J1–J12. Eleven began from a clean start: an empty folder or an unmodified existing project, an export of the pushed release (as a downloaded archive would arrive), and a fresh user profile with no keys, registry entries or history. J2 continued R1’s project in R1’s profile, and J9 audited J1’s live workspace [R §8.37].

Every journey was operated by one AI, Claude Opus 5.5, acting in the owner’s seat, with “full knowledge of Runesmith’s internals”; most journeys went through the Studio’s browser interface, and a few through scripts or the terminal by design. Personas gave each journey a stated goal (a gift shop’s stock book, a bakery’s handbook and till exports, a developer’s failing tests, a keyboard-and-screen-reader user, a terminal-only user). Model instruments inside Runesmith were free routes for nearly all building and checking (NVIDIA Nemotron 3 Super, Google Gemini 3.1 Flash Lite and, later, Groq’s free `gpt-oss-120b`). In the journeys that used a keyless chat window, “the plan, checks or drafts concerned were written by the operator itself, not by a free route” [R §8.37]. The operator also fixed Runesmith during the journeys through a

fixed loop: a test that reproduced the failure, the full suite, browser use-loops, an adversarial review pass, a commit, a fresh release into the journey's own folder, and a live re-exercise [R §8.37].

Outcomes as the record states them. R1 (a reading log built from an empty folder) passed both phases with no rescues, by the operator's own definition of a rescue; its scheduled phase built three milestones at no metered cost, after five releases that hardened the acceptance-check path. J2 continued R1's project: one of three new milestones completed after three ordinary attempts, an alternate author, four corrections and seven owner actions, and two used up their tries. J3 built and applied a real fix to a project with 5 of 8 tests failing (8/8 passing afterwards). J6, a developer's project with seven failing tests, had 4 of 5 free-model repairs accepted by a held-out judge and applied; the C7 organ was adopted into a trial and then rolled back to g0 as a recorded owner choice. This is the only record of C7 meeting a user project, and it is neither a replication of SR7 nor a measured effect. J7 ran hostile scenarios and found a permanent lockout after local data was deleted. J8 connected docs-conformant stand-ins for local model servers (Ollama, LM Studio, llama.cpp), not the real servers. J9's accessibility audit fixed unlabeled fields, overflow at phone width, a focus leak and contrast below WCAG AA. J10 repaired a failing test through the manual chat relay in 2 calls and 35 s. J11 is the case study of §5.5 [R §8.37].

Findings. The inventory holds 219 findings: R1 26, J1 9, J2 45, J3 9, J4 25, J5 8, J6 5, J7 3, J8 3, J9 7, J10 7, J11 70, J12 2. By kind: 49 bugs, 53 gaps, 117 frictions. By status at the file's last write (2026-09-29T12:19:21Z): 127 fixed and verified live, 69 fixed in test only, 17 open, 4 open with a fix in test, 2 won't fix [R §8.37; §8.34, row 23]. The inventory is a lower bound: J11's log records further identifiers that are not in the file; at this paper's read, at least 23 further J11 finding identifiers appear in the log (regex count) [count, 2026-10-05T01:05Z]. Seven cross-cutting defect classes were first recorded during the journeys: free or weak model output the harness must catch or tolerate (for example, a forced JSON schema silently dropping every backslash escape in a free model's code); budget and receipt integrity (never spend a try on a call that produced nothing); owner-facing wording; scheduling and unattended throughput (one stuck milestone blocking five independent ready ones for 1 h 38 min 49 s); project and content-type recognition; accessibility; and Windows and cross-process robustness. One further class has no theme: checks that cannot see what they were meant to check. "The inventory does not test whether isolated unit or integration tests could have found them, and because the operator is an AI with full knowledge of Runesmith's internals it does not show what a human first-time user would meet." [R §8.37]

Review of the operator's own fixes. Every fix batch released on 2026-09-29 (11 commits) went through or was covered by an adversarial review pass launched by the operator, "not independent of the operator and lab". Ten of the eleven reproduced real defects in the fixes themselves: about 45 reproduced and fixed defects where a commit message or the log gives numbers, of which 20 are stated exactly and about 25 are approximate. Review counts and journey findings "are not commensurable" [R §8.37; §13, item 39].

The check autopilot in the record's analysis window. In the first 977 lines of J11's ledger (events through 2026-09-29T12:16:42Z) the autopilot made 35 decisions: 22 approvals, 10 turn-downs and 3 left for the owner. The second model's disagreement accounted for 4 of the 10 turn-

downs; the trial on the unbuilt project for 4; mechanical findings alone for 2. The owner audit judged **3 of the 22 approvals wrong, in two incidents, both caught by the owner's eye and not by the cross-check** [R §8.37]:

- *Printed output.* Checks read the printed output of a command that writes its result to a file and prints nothing; the second model, asked only whether that output must contain the text, agreed. Fixed by a gate (J11-G24) that leaves such checks to the owner; re-exercised live.
- *Invented options.* Checks ran two command-line options that “neither the milestone nor the program named”; the Checker had invented the interface, and the second model, “asked only what those commands would do rather than whether they exist, agreed”. Fixed by a gate (J11-G30) requiring a run step's options to be named by the milestone or the program; the milestone was dropped.

Five more approvals after the window were judged wrong, so at the record's cut the log marks **8 of 49 approvals** as wrong. “These are counts of self-audits by the AI operator, who also wrote or edited many of the milestones; they are not a measured false-approval rate.” [R §8.37; §8.33 update] After the cut (2026-10-01, read at 18:15:32Z), four of seven new approvals were wrong “in the same way” (a field placed where the program does not read it) and one was loose, and the autopilot was switched off at 17:04Z. The causes found were that the cross-check recomputes expected numbers but not whether an example's input uses the fields the program reads (J11-G38); that nothing compared approved checks with the milestone's “done when” statement (J11-G40); and that the check writer's packet held at most 16,000 characters of source, so neither model had been shown the program (J11-B16) [R §8.37; J11 log, 2026-10-01T16:52Z–17:04Z].

Nonclaims (the record's own). Not a human user study: “Out-of-box difficulty for a real first-time, non-technical user remains untested.” Not unattended: the operator intervened repeatedly. “Built by free models” means the last author of each applied file, “and nothing more”. The owner and the operator are the same AI in R1, J2 and J11, so every owner audit “is that self-review, not independent human QA”, except where the log records the lead investigator's own words. Not preregistered. Free-route throughput varied by the hour. Not a controlled or blinded comparison: there is no fixed-baseline or non-Runesmith comparator arm [R §8.37; §13, item 37].

5.4.3 Operating a live production service (the predecessor kernel, 2026-09-11)

Operation was first exercised before the open-source Core existed. On 2026-09-11 the predecessor kernel's environment controller, a durable supervisor that enters recovery and reconciliation on every start and after every task result, was pointed at the live PRHat environment after its own 328-test gate and PRHat's 897-test gate had passed [R §8.13; §8.15]. PRHat kept ownership of every research, quality, pacing and handoff decision, and its e-mail delivery stayed with its own delivery service. The first complete owner cycle ran a bounded campaign; a fresh process then recovered the prior state and ran a second campaign, whose timeout it classified as a task failure rather than as degradation of the service. “The resident supervisor then continued without observer intervention through campaign, scheduled-reply, deterministic-fallback, and research actions”, and at the later cut held ten terminal task records. The record lists what the sequence demonstrated: “repeated fresh-process recovery, one result and one successor per terminal task, provider-local failure attribution, timeout classification, scheduled outcome polling, and continued work while semantic-map inference was unavailable”

[R §8.15]. It is operation, not improvement: the service’s funnel deltas stayed at zero, and at that cut no admitted semantic proposal was yet substantive (seven substantive provisional operations followed after later repairs). Operator repairs came between the first cycles (a bootstrap defect and a reporting defect), so “without observer intervention” describes the continuation after the second start. The released Core keeps the same functions in its own form (environment and self maps, a recovery policy after interrupted work, scheduled rounds), but has not itself been run as the resident operator of a production service.

5.4.4 Release week: an acceptance test on one free key, and Kaizen live on the release build (2026-10-05)

Both postdate the record’s cut. The record’s 2026-10-05 extension covers J0 up to its phase 3 and J13 up to the opening of the trial; the later figures are read from the J0 and J13 logs [J0 log; J13 log, 2026-10-05].

Acceptance on one free key (J0). The release’s acceptance test puts an operator in a newcomer’s seat with one free Google AI Studio key and nothing else: a clean profile, no paid route, the guide’s first steps. The operator is an AI agent (Claude Sonnet 5.5) with the release brief. The first runs found the decisive constraint of that free tier: Google allows 20 requests a day per model on such a key (“limit: 20, model: gemini-3.8-flash”), and busy answers appeared to spend from it. Release 1.0 answers with a chain of free Gemini models on the one saved key, each with its own allowance and hold, per-model counts shown to the owner, more room for models that think before answering, and, by design, no immediate retry of a busy answer; the acceptance test still saw the same model asked again within about a second (J0-F18, open at the time of writing). On the build tested (`f4d973d`) the test met its bar: a new milestone of the newcomer’s reading-log app was given acceptance checks, built, checked and applied without the override that writes files despite unresolved checks, and then tried; the new command worked and its error path behaved. It took 16 min 18 s from sign-in (about 10 min hands-on; 3 min 16 s of it waiting out a busy service, 3 min 3 s lost to a hidden dialog in the test rig), with no rescue and 9 model requests, 7 of which the service refused as busy: the chain absorbed three of them, and the first “Draft first files” click failed outright and succeeded on a second click [J0 log, phase 4]. One step was borderline: the practice copy had to be restarted by hand before the new command could be tried (J0-F20).

Kaizen live on the release build (J13). A project with a real test suite (Ledger Lite) ran Runesmith’s scheduled repair rounds on the build just before 1.0.0 (`e0dfc6f` , which differs from 1.0.0 only in its version strings) while a feeder injected one realistic single-line regression at a time from a validated pool of 348, and Kaizen worked on improving the repair organ in the background, with free models as both repair model and improver. By 2026-10-05T11:07Z the rounds had served 67 regressions, and the held-out judge accepted 58 of the repairs (one opportunity was lost to a full disk); by the feeder log’s later cut (16:00:44Z) 89 had been injected and 69 repaired, with the four tasks that timed out during the disk-full stall and the restart (67 to 70) counted as lost opportunities, not repair failures (Figure 7). Kaizen ran six campaigns. The first five ended without a candidate that passed Kaizen’s freeze bar, so nothing reached activation. The sixth (07:18Z) froze a candidate on 28 of 30 strict successes against 27 of 30 in replay (frozen on yield under the declared rule; its cycle time, the declared target, was 21% higher, and one success is inside the replay noise SR7 measured, which is what the online trial is

for); by the rule of §3.4 a candidate must win an online trial on fresh work, and that trial opened at 09:01:14Z [J13 log]. At the 11:07Z read the trial had not reached its first look (candidate 6 of 8 repairs, incumbent 7 of 7), so no activation is claimed.

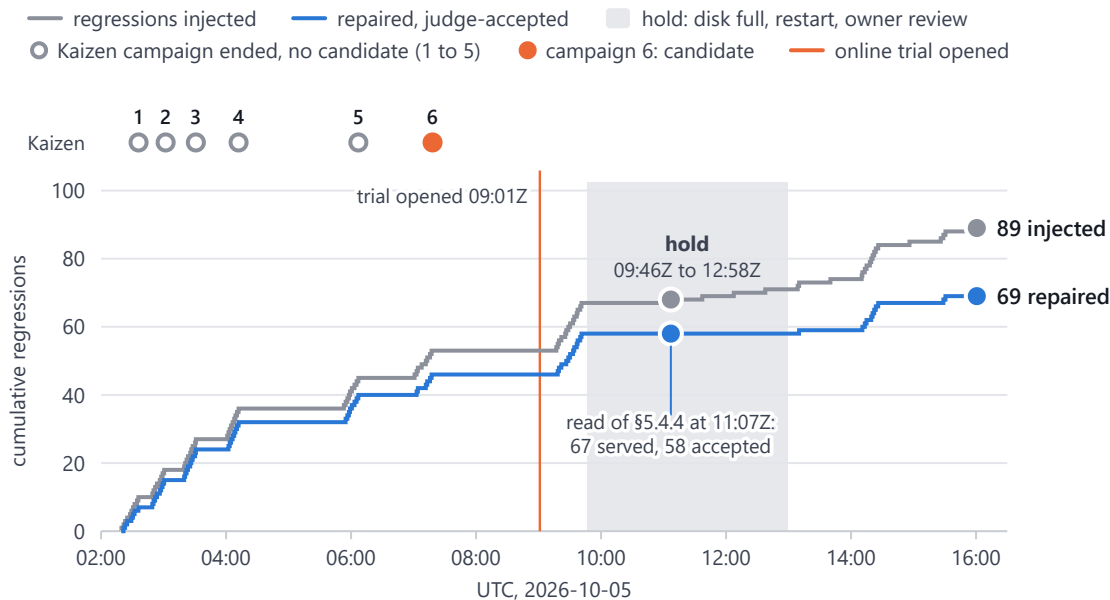


Figure 7. Kaizen live on the release build (J13): regressions injected and repaired over 2026-10-05. Cumulative single-line regressions injected one at a time into Ledger Lite (a Python project with a real test suite; pool of 348) and cumulative repairs accepted by the held-out judge, read from the feeder log [J13 log, 2026-10-05] and cut at 2026-10-05T16:00:44Z: 89 injected, 88 completed, 69 repaired. Free models were both repair model and improver. Numbered markers are Kaizen campaigns (02:36Z, 03:02Z, 03:31Z, 04:12Z and 06:07Z ended without a candidate; 07:18Z froze a candidate); the online trial opened at 09:01:14Z. The shaded hold, 09:46Z to 12:58Z, is the disk-full stall, the restart and the owner's recovery review; tasks 67 to 70 timed out in it, and the log counts them as lost opportunities, not repair failures. At 11:07Z, the read reported in the text, 67 had been served and 58 accepted. One project, observational, no fixed-baseline arm; no trial result is shown.

5.5 Case study: Runesmith Motion

5.5.1 Definitions: what is and is not claimed

Runesmith Motion (journey J11) is a browser animation and video composer whose timeline, keyframe, easing and interpolation logic runs in plain JavaScript in the browser and in Node; a command-line inspector prints the frame at a given time, so the logic can be checked without a browser [R §8.37]. It was the first application intended for release alongside Runesmith. Two terms keep this case study honest.

- **Autonomous authoring and verification (claimed, in this precise sense).** Every applied change to the program's code and data files was authored by a model instrument on a free-tier route, and Runesmith itself checked it against the milestone's approved acceptance checks and applied it through the guarded delegated-build path. No human or AI operator wrote an applied file of the program. Many milestones, however, dictated literal values that the files had to contain (path coordinates, transform strings, exact output texts); the record counts such exact-spec milestones among the operator's contributions [R §8.37].

- **Unattended operation (claimed in the lead investigator’s sense, on two occasions).** The lead investigator defines unattended as “seeing it go at least from one milestone through to the next without interfering”, and by that definition it was shown twice (§5.5.3). Outside those stretches, the goals, the brief, most milestones, many check approvals, all audits, the choice and order of instruments, the release of Runesmith fixes and the recovery from stalls came from an owner, and the owner here was an AI operator (Claude Opus 5.5) acting in the owner’s seat, directed by the lead investigator. Runesmith has not completed the project on its own.

5.5.2 Numbers at four dated reads

The project kept running after the record’s cut, so its figures move. They are given at four reads and are never mixed.

	Record’s cut (2026-10-01T16:52:16Z)	Numbers check (2026-10-04T20:36Z)	First read (2026-10-05T01:05Z)	Release read (2026-10-05T11:20Z)
Plan items	63	124	126	137
Done / open / dropped / doing	40 / 14 / 8 / 1	52 / 65 / 7 / 0	52 / 67 / 7 / 0	54 / 76 / 7 / 0
Applied drafts (all last authored by free-tier routes)	40	52	52	54
Drafts in all	122	218	218	281
Check-autopilot decisions (approved / turned down / left for owner)	87 (49 / 26 / 12)	121 (72 / 28 / 21)	122 (73 / 28 / 21)	135 (78 / 33 / 24)

Sources: [R §8.37; §8.34, row 24]; [numbers check, 2026-10-04]; [count, 2026-10-05T01:05Z]; [count, 2026-10-05T11:20Z]. The applied-draft and call-record counts were made with `count_read.py`, which reproduces the 01:05Z figures exactly when cut at that time; the plan, draft and autopilot rows were counted from the ledger with the same cut. The ledger’s last event before the release read is at 11:05:22Z, so the release-read figures hold until the project resumes.

“Done” means that the milestone’s approved checks passed and its draft was applied; “it is not a finding that the feature works” [R §8.37]. At the release read, the 281 drafts comprised 54 applied, 193 needing revision, 31 waiting, 2 rejected and 1 undone (the undone draft is not counted as applied). The first draft was applied at 2026-09-28T16:51:13Z and the last at 2026-10-05T04:28:44Z (2 on 09-28, 27 on 09-29, 13 on 10-01, 10 on 10-02, 2 on 10-05); between 2026-10-02T13:39:09Z and the unattended stretch of 2026-10-05 no milestone was completed (§5.5.3) [count, 2026-10-05T11:20Z].

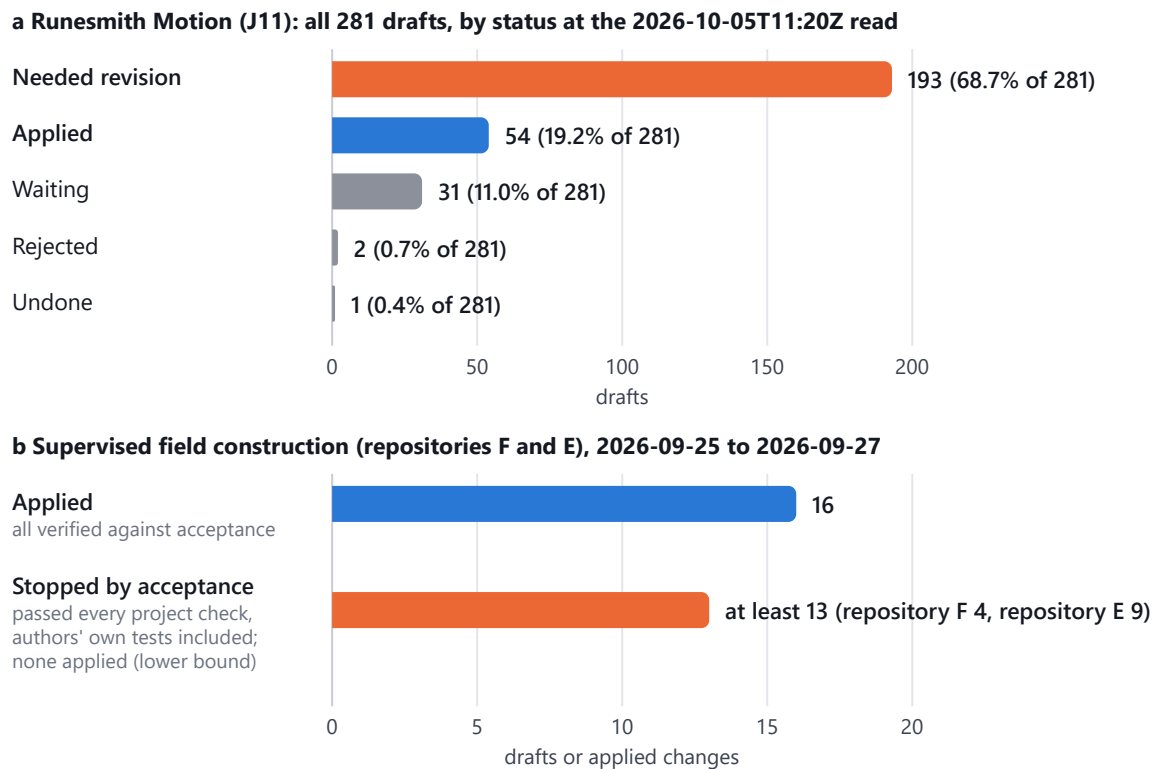


Figure 8. Drafts and acceptance. (a) The 281 drafts of Runesmith Motion by status at the 2026-10-05T11:20Z read: 193 needing revision, 54 applied, 31 waiting, 2 rejected and 1 undone, counted from the project’s ledger and draft files [count, 2026-10-05T11:20Z]. Applied means that the milestone’s approved checks passed and the draft was applied; it is not a finding that the feature works. (b) In the supervised construction of repository F and repository E (2026-09-25 to 2026-09-27), 16 changes were applied, and at least 13 drafts that passed every project check, including tests their authors wrote, failed the trainer’s cumulative acceptance and were never applied (repository F 4, repository E 9) [R §8.36]; the 13 is a lower bound, and the record notes that several failures trace to public contracts the trainer had not specified and that acceptance was often written after the trainer had read the drafts, so “these are not all author failures”. The panels have different units and axes. Observational, with no controlled comparison; in panel a the checks were approved by the check autopilot or by the owner, who in this project was an AI operator (§5.5.2).

Who wrote the code. The last authors of the 54 applied drafts were Gemini 3.1 Flash Lite on three free route names (26, 15 and 7; 48 in all) and NVIDIA Nemotron 3 Super 120B-A12B on three free route names (3, 2 and 1; 6 in all). All 54 were applied by the delegated-build path. At the release read, ten program and data files are byte-identical to the content of the last applied draft that wrote them: the main engine `motion.mjs` (40,825 bytes), `paints.mjs`, `scatter.mjs`, `renderer.mjs`, `editor.mjs`, `record.html`, `index.html`, `GUIDE.md` and two sample project files. Two further files were not written by a draft: `HOUSE_RULES.md`, a statement of the project format written by the owner (the AI operator) on 2026-10-02 and shared with every model, and `RUNESMITH.md`, Runesmith’s own one-line-per-event log of what it did in the folder [count, 2026-10-05T11:20Z; J11 log, 2026-10-02]. “Free” is a route designation, not metered cost. The ledger holds 1,141 model-call records: 771 answered and 370 not (322 transport failures, 48 output failures); the cost estimate is 0.0 on 1,052 and absent on 89, and no call has a positive figure; recorded tokens are 13,625,189 in and 1,532,728 out on the 1,104 records that carry them, a lower bound. The operator’s own model usage is not metered anywhere [count, 2026-10-05T11:20Z; numbers check, 2026-10-04].

Who wrote the plan. Of the 137 plan items at the release read, 21 come from Runesmith: 5 from its own first plan (drafted by Gemini 3.1 Flash Lite on 2026-09-28); 5 from two model-proposed breakdowns that the owner adopted (3 by Nemotron on 2026-09-28; 2 by Gemini 3.5 Flash on 2026-10-04T23:23:40Z); and 11 from five breakdowns that Runesmith’s stuck policy proposed and adopted by itself on 2026-10-05 (02:48Z to 05:03Z), each when a milestone had used up its tries, recorded in the ledger as “Runesmith (your setting)”. **116 were added by the operator** through the Studio’s milestone form [count, 2026-10-05T11:20Z]. Much of the design happened outside Runesmith, in operator-run workflows: a creative-direction workflow chose the visual direction; a milestone workflow with designer, architect and adversarial-verifier agents wrote a 22-item first wave; and a second workflow designed a 52-milestone “versatility wave” on 2026-10-01 [R §8.37; J11 log, 2026-10-01T22:03Z].

Who approved the checks (at the first read). Of 95 approvals of check sets in the ledger, 73 were made by the autopilot and 22 by the owner. Eleven of the owner approvals are recorded as such; eleven earlier records carry no approver field and all predate the autopilot’s first decision (2026-09-29T02:36:22Z) [count, 2026-10-05T01:05Z]. The owner withdrew approved checks 10 times. The owner audit continued after the record’s cut. On 2026-10-02 the house-rules document reduced the format errors in proposed checks, but several autopilot approvals were still judged wrong and withdrawn, among them checks that placed a camera where the program does not read it and a check that read a file nothing creates (J11-G45) [J11 log, 2026-10-02].

5.5.3 How close it came to unattended operation

Definition. The lead investigator’s definition of unattended is: “seeing it go at least from one milestone through to the next without interfering.” Four stretches in which the operator stepped back are recorded; two of them meet the definition.

1. **2026-09-29, from 13:30Z.** The operator’s session ended, and the longest stretch without an owner-initiated event lasted 50.5 hours. In its first 1 h 43 min the schedule applied five milestones by itself and the autopilot approved checks for two more. From 15:47:15Z the main program file had grown to 20,316 bytes, past the 20,000-byte cap on files shown to an author, so no model was shown it and every answer that edited it was refused. For about two days (1,364 scheduling rounds) nothing progressed, and “Nothing told the owner” (J11-B15) [R §8.37]. By the definition above, the first 1 h 43 min of this stretch was unattended operation: five milestones in a row, applied by the schedule, with no owner action. The record calls this stretch the closest J11 came to unattended; counting it as unattended is the lead investigator’s reading of his own definition.
2. **2026-10-02, from 04:57Z.** A new release was set running with the intention of leaving it alone for about 36 hours; the operator still audited and withdrew checks during the first hours (the last such log entry follows 07:09:53Z), so this was not an untouched stretch. It reached 52 milestones done by 13:39:09Z. The Studio then stopped with a job running; a keep-alive task restarted it at 17:23:00Z, and from then on all work was held for the owner’s recovery review, about 47 hours idle, because automatic resumption (J11-G42) had been built but not released. By then `motion.mjs` was 40,179 bytes, past the 40,000-byte cap of the release it ran [J11 log, 2026-10-04].

3. **2026-10-04, 20:51:44Z to 21:35:16Z.** Under the definition above, the operator only read the project. The stretch ended after 43 min 32 s **with 0 milestones:** a free author's escalation draft was broken, every later build tried to revise a draft that had no ordinary author allowance, and the system idled silently (J11-B28). The operator intervened [J11 log, 2026-10-04T20:51:44Z; 2026-10-04T21:35:16Z].

In the next hours three releases fixed a file cap that kept the author from being shown the file it had to change (J11-B29), answer budgets too small to return a 40,179-byte file whole (J11-B30), and an escalation receipt that a restart had left open and that silently blocked its milestone (J11-B31) [J11 log, 2026-10-04T22:07:04Z to 2026-10-05T02:10:49Z]. Then:

4. **2026-10-05, 02:42Z to 05:19:05Z.** The operator's last action was the removal of a gate from 62 open milestones (ledger 02:42:06Z to 02:42:26Z; the log says 02:43Z); from then on the operator only read files. At 03:18:16Z and 04:28:44Z Runesmith applied two milestones in a row, "Add transition schema and validation to motion.mjs" and "Add paint schema and resolution logic to motion.mjs". Runesmith had made both milestones itself: its stuck policy split two milestones whose tries were used up into smaller steps and adopted them (03:10:01Z and 03:52:56Z); the check autopilot proposed and approved their checks; a free route (Gemini 3.1 Flash Lite) authored both drafts; and the delegated-build path applied them, each rewriting the 40 KB `motion.mjs` whole. The stretch went on for another 50 minutes: 24 drafts were checked and none passed, exact edits to `motion.mjs` were refused because their old text did not match, and at 05:19:05Z five milestones had used up their tries and waited for the owner [J11 log, 2026-10-05T03:18Z, 04:29Z, 11:17:17Z]. **By the definition above, this is unattended operation, from one milestone through the next, with the plan itself extended by Runesmith.** The record's extension states the boundary: the stretch is unattended "only by the lead's definition and by an actor-field rule"; under the record's earlier literal rule, its 26 events of 16 escalations, 5 breakdowns and 5 discards, which the owner's settings and not the operator produced, would count as owner-initiated, and the plan, the house rules and the releases before it were the operator's work [R, 2026-10-05 extension].

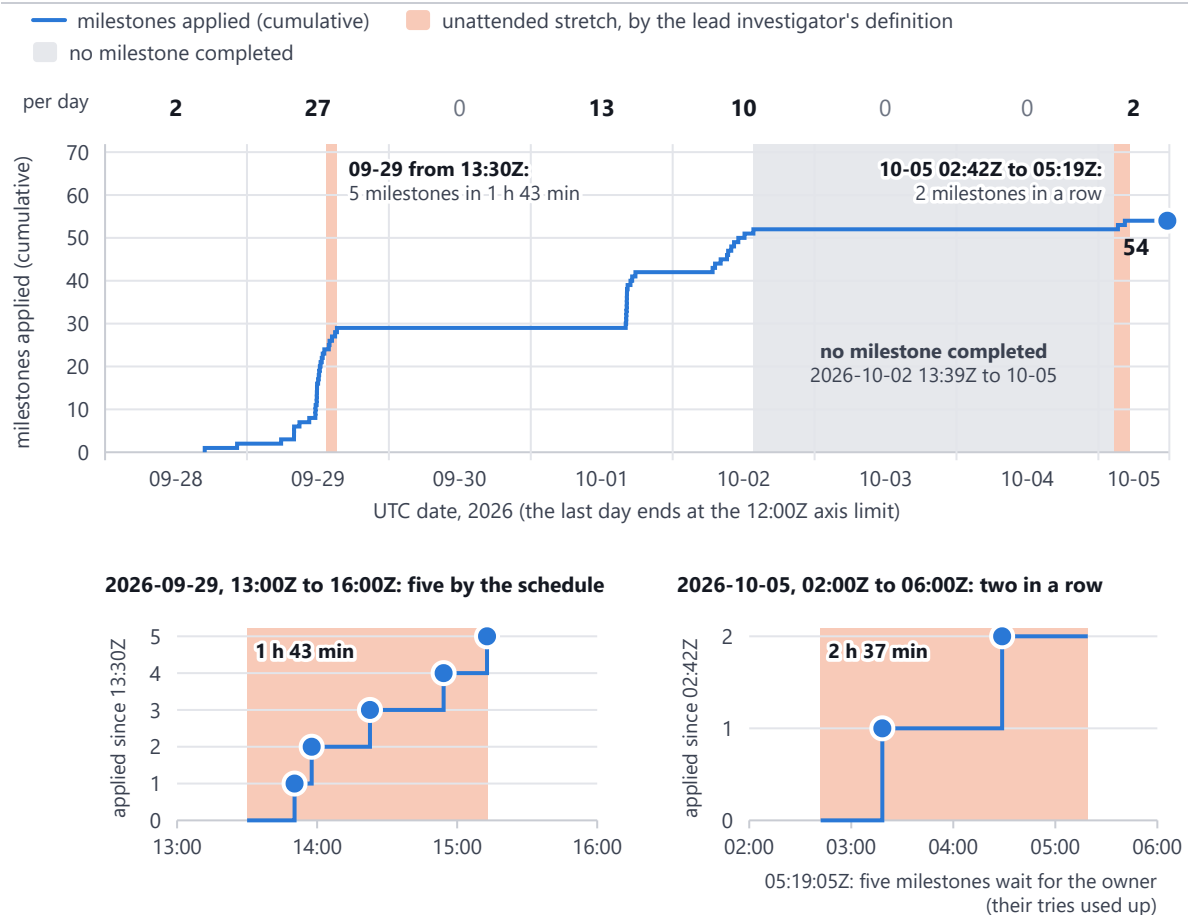


Figure 9. Runesmith Motion over time: milestones applied, and the unattended stretches. Cumulative milestones applied to the program (54 at the 2026-10-05T11:20Z read; by UTC day, 2 on 09-28, 27 on 09-29, 13 on 10-01, 10 on 10-02 and 2 on 10-05, none on the other days), drawn from the project’s ledger [count, 2026-10-05T11:20Z; J11 log]. Shaded in orange, the two stretches that meet the lead investigator’s definition of unattended operation (§5.5.3): 2026-09-29 from 13:30Z, five milestones in 1 h 43 min, and 2026-10-05 from 02:42Z to 05:19:05Z, two milestones in a row (the two lower panels enlarge them). Shaded in gray, the interval from the 52nd milestone (2026-10-02T13:39:09Z) to the second stretch, in which no milestone was completed: it holds the owner’s recovery review (all work held from 2026-10-02T17:23Z, about 47 hours idle), a 43 min 32 s stretch that ended with 0 milestones, and the operator’s releases of fixes (§5.5.3). The owner in this project was an AI operator, and Runesmith has not completed the project on its own (§5.5.1).

5.5.4 What it produced

On 2026-10-01 the lead investigator judged a gallery of the output (a mark with a letter reveal, a paper landscape, a spark burst, a chart, words on a curve) simplistic and visually dated, and said it was not yet world class; the operator agreed that “the frames were technically correct and visually dated” [R §8.37; J11 log, 2026-10-01T16:37:41Z]. The record calls this “the project lead’s judgement of one gallery on one day, not a measurement”. A look at the files rather than the checks found that the milestone “Record the clip as a video”, counted as done, was “done in name only”: its approved checks asked only that keywords appear. At the release read this is unchanged: `renderer.mjs` (252 bytes) returns an SVG header around the comment `// ...` (renderer logic from `motion.mjs`), and `editor.mjs` (88 bytes) only logs a line [R §8.37; count, 2026-10-05T11:20Z]. Neither file is the engine: the engine is `motion.mjs` (40,179 bytes at 52

milestones, with the Motion modules it imports), which computed the graphics of every frame of the launch film (below); the film's recording and encoding were done outside Motion, by the operator's pipeline scripts, a browser and a general-purpose encoder. The program is unfinished.

The launch film. Runesmith Motion's engine computed the graphics of every frame of the release's launch film. The film's scenes were written by the operator as a Motion project file; each frame's graphics were produced by the render functions of the program's own `motion.mjs` as it stood at 52 applied milestones (40,179 bytes, applied 2026-10-02T13:39:09Z; a read-only snapshot copied on 2026-10-04), with its command-line head and tail cut away, two import paths re-pointed and an export line added, and the render functions themselves unchanged. Pipeline scripts written by the operator inserted the lead investigator's recorded take into each frame's card; a browser rasterised the frames, and a general-purpose encoder (ffmpeg) joined them with the talk and music [film pipeline, `make_lib.mjs` and `render_film.mjs`, 2026-10-05]. The film's closing text says that the presentation was "made with Runesmith Motion", "a video program Runesmith built autonomously, using free inference"; that rests on this paragraph and on §5.5.1 to §5.5.3: autonomously in the sense defined there, with free-tier routes as the only authors of the program's applied changes.

5.5.5 Defects it exposed in Runesmith

The case study's main scientific yield is a set of defects in Runesmith that only sustained use on a growing real program exposed. One line each:

- **J11-B6.** The planner kept retrying one stuck milestone instead of starting independent ready ones: 1 h 38 min 49 s idle with five ready.
- **J11-B7.** A provider refusal before any token ran was counted as a failed attempt.
- **J11-B15.** The program file outgrew the author-context cap (20,316 bytes against 20,000), so no model was shown it; about two days without progress and without notice to the owner.
- **J11-B16.** The check writer's packet held at most 16,000 characters of source, so check writers guessed the program's formats.
- **J11-B17.** Every approved check ran twice, so a run with 188 planned checks hit its fixed 120 s limit at check 187.
- **J11-B28.** After a failed escalation draft, builds tried to revise a draft with no author allowance; silent idle.
- **J11-B29.** The program file (40,179 bytes) outgrew the 40,000-byte cap; Runesmith's advice to split it first was impossible, because splitting edits the file that cannot be shown.
- **J11-B30.** Draft answers were truncated: a 12,000-token budget cannot return a 40,179-byte file whole.
- **J11-B31.** An escalation interrupted by a restart left an unresolved receipt that silently blocked its milestone.
- **J11-G24, G30.** Wrong check approvals: printed output on a command that writes a file, and invented command-line options.
- **J11-G31.** A free model resubmitted an identical refused edit six times despite feedback; author rotation was added, with no live exercise of it recorded in the log through the record's cut.

- **J11-G34, G35.** An approved check file keeps the template it was rendered with (an unparseable drawing was marked done), and three rendering defects passed every text and parse check and were caught only by eye.
- **J11-G36, G45.** An example runs a file it does not list; a check reads a file that nothing creates.
- **J11-G37.** Approved checks could not be withdrawn.
- **J11-G38, G40.** The cross-check verifies numbers but not field placement, and nothing compares the checks with the milestone's "done when" statement.
- **J11-G39.** Nothing refuses a file written with its code left out (the recorder that does not record).
- **J11-G41, G42, G43, G44.** Owner milestones could not name prerequisites; no automatic resume after an interrupted job; no policy for milestones whose tries are used up; a draft whose checks did not finish waited for the owner.
- **J11-G46.** A visual effect was built only for top-level elements (found by eye).
- **J11-F29.** A later milestone that legitimately changes an earlier exact requirement contradicts that milestone's still-approved check.

Several of these share one structural cause that the case study made visible: drafts rewrite whole files, so a single growing file eventually exceeds what a model can be shown or can return.

Targeted edits for large files are the fix the log names for a later release (§9) [J11 log, 2026-10-04T21:35:16Z; 2026-10-04T22:07:04Z].

5.5.6 What the case study shows

It shows that free-tier model routes, checked and applied by Runesmith, were the last author of every applied change of one real, growing program: 54 applied changes at the release read, no applied file written by the operator (although milestones dictated some literal values), no call with a positive recorded cost, and an engine (`motion.mjs`) that, at 52 milestones, computed the graphics of every frame of the release's launch film. It shows unattended operation in the lead investigator's sense on two occasions, the second with Runesmith extending its own plan when work stalled. It shows the costs: most drafts needed revision, at least 8 of 49 check approvals were judged wrong by self-audit at the record's cut, rendering needs an eye that no check kind replaces (§12.12(e) of the record; §7), and the program is unfinished, with two stub files, the clip recorder (`renderer.mjs`) and the editor (`editor.mjs`), neither of them the engine (§5.5.4). It does not show that Runesmith completes a project on its own (the operator added 116 of 137 plan items and audited throughout), that the product works as a whole, that a human user could direct it to the same point, or that other models would do the same.

6. Discussion

6.1 Where the instrument–substrate hypothesis stands

The hypothesis predicts that competence first supplied by one instrument can be kept as tested state and improve later work after that instrument is removed or replaced (§1.2). The experiments that decide it to its own definition are named in the record: positive SCG after

restart against fresh, shuffled, revoked, poisoned and raw-transcript controls (card M); a named substitution matrix (card S); a transduction yield with positive future SCG (card T); a factorial weak-model amplification study (card W); and a starvation-and-restoration study [R §10]. None has been run to its definition. The partial tests point in no single direction: the only SCG measurement was inconclusive (§5.3.8); SR1-MIN's synthetic succession comparison returned equal success (§5.3.2); SR3 was directionally positive and not shown (§5.3.3); SR6-W was null (§5.3.7).

Our interpretation: SR7 has the mechanism's shape, once, under seal. Read in the hypothesis's own words, SR7 has its shape. "A capable instrument may discover a solution": Claude Opus 4.5 wrote a better localization policy into the repair organ. "The runtime may preserve the useful part as a tested procedure": Runesmith froze it as generation C7 after held-out validation. "A weaker instrument ... may later reuse it within a validated scope": gpt-oss-20b, a cheap open-weights model, used C7 after its author was gone and repaired more fresh tasks, on repositories that supplied no task to its development, than it did with the predecessor organ (§5.2.5). The same repair model ran both arms and the author was absent at evaluation, so the additional work was done by what differed between the arms: the organ. This is our reading, not the record's. The record does not map SR7 to transduction or substitution [R §10, prospective bridge program], and SR7 compares two learned states rather than a learned state with a fresh one, without the shuffled and revoked controls the formal test requires. M1 adds a second, smaller instance: a model-authored retrieval change that "remained after author termination and improved no-model retrieval on the assigned synthetic workloads" [R §12.10; M-1278].

What would settle it. The formal test would also decide whether the organ's gain is transferred competence or a localization heuristic that an engineer could write directly: C7's tiers rank files named in the failure text and the execution trace, and the record's comparator rule places the strongest model-free composition of that structure in the fixed arm (§5.1.2). That arm costs no inference and has not been run on SR7's pool; it opens the first study after release (§9).

6.2 A nudge towards recursive self-improvement

The record defines a ladder. *Operational self-improvement* is "system-produced changes to problem-solving machinery that repeatedly produce durable held-out net gains after model, data, search, tool, and evaluator contributions are separated". *Strong recursive self-improvement* is "improvement of the machinery that produces improvement, with broad transfer and repeated independently verified gains. This is not a current Runesmith claim." [R §2.7]

What SR7 shows. In one iteration of Runesmith's own loop, the system's telemetry selected a target by a fixed declared rule; a model rewrote the organ responsible for that weakness; held-out validation selected among the candidates; the result was frozen; and a sealed, preregistered test on fresh tasks, from repositories that supplied no task to the organ's development, showed the successor doing more of the downstream work (57/162 against 35/162 sessions over 54 tasks) with the same repair model and envelope, in about half the time per success (§5.2.5). The system's own loop produced a better version of the system's code than the version it replaced, and a sealed test confirmed it. The version it replaced was B, the loop's own previous generation (SR7's limits: §5.2.5, §7.1).

Why “a nudge towards recursive self-improvement”. Recursive self-improvement needs a system that changes its own problem-solving machinery, changes that measurably help, and an improvement process that itself improves. SR7 has the first two, under seal: Runesmith improved itself. The third is not shown: the Kaizen loop, its target rule, the trial machinery and the kernel were built by a consultant and stayed unchanged [R §1.5; M-0226]. One verified step in the direction of the process, without the process itself, is what we mean by a nudge, and we use the phrase in no wider sense. The record’s naming rule licenses *bounded subject-improving adaptive runtime* for this envelope, and no rung licenses “recursive self-improver” [R App. B]; this paper claims the first and not the second.

What would turn the nudge into a step. A second improving generation under locked evaluators (the cumulative rung; $A \rightarrow B \rightarrow C7$ “is one qualified gain, not two”, because B was not shown to improve on A [R App. B; M-0796]); a replication of SR7 on a fresh cohort with an ablation of its tiers, one of which was inert; a comparison of C7 with g0 and with a model-free localization arm; and a reproduction with a free-tier author, now that free-tier models score above the SR7 author on a public coding benchmark (§5.2.5). Each is named in §9; the comparison with g0 and a model-free arm is the one with a written design.

6.3 Separate measured edges, not one chain

The record insists that its positive results are “a set of measured edges, not as a single uninterrupted self-improvement chain” [R §8.19.12]. The object results ran on the earlier kernel and on a production project; A11→B and M1 ran on the earlier subject machinery; SR7 ran on the open-source Core’s repair organ. No experiment links them: “The record supplies no causal bridge from that S1 correction to the M1 performance gain or from M1 to later object utility.” [R §8.19.12] Faster retrieval does not entail better object work; a recurrent workload “could” repay acquisition costs, but “that break-even point is a prospective calculation, not a result” [R §8.19.12]. The record names four bridges that would join the edges (M1 to later object work; a retained artifact to a named substitute or a no-model procedure; historical object correction to a deployable workflow; one improved successor to cumulative improvement) and states that “none is implied by the existing passes” [R §10, prospective bridge program].

6.4 Inference independence and digital sovereignty

What the design provides. *Contract independence*, an engineering target the record labels Known/C: objectives, state, evidence, permissions, budgets, proposal and action contracts and promotion rules do not depend on one provider or on inaccessible model state [R §2.6]. In the released Core, a project’s state, evidence and history live in local, readable files with a hash-chained ledger; the router accepts any OpenAI-compatible endpoint, including local servers; model tier predicts competence but grants no operation or authority (§3.1); and no mechanism sends experience between installations (§2). Within the record, changes were authored by several model families: two free-model families on the historical object contracts (§5.1.8), a free route for the S1 correction (§5.2.3) and for M1 (§5.2.4), free routes, as last author, for every applied change of Runesmith Motion (§5.5), and open-weights repair models in the SR studies. The Studio ships presets for local servers (Ollama, LM Studio, llama.cpp), for free and paid

hosted providers (Google Gemini, NVIDIA, Groq, OpenRouter, Mistral, DeepSeek, Together, OpenAI and Anthropic), for any other OpenAI-compatible address, and a keyless chat relay [Core source, read 2026-10-05].

What is not shown. *Capability retention under substitution*, the property that would make a project's competence portable across models, is Unproven/D: "contract compatibility never implies semantic equivalence, calibration equivalence, or equal task performance" [R §2.6]. No named substitution test has been run (§6.1). Cost reduction through routing is rejected as a present result, because "verification, retries, repair, queueing, latency tails, privacy, incidents, and operator attention are endogenous to route quality" [R §3.5; M-0265]. Local model servers were exercised only through docs-conformant stand-ins (J8, §5.4.2). "Free" is a route designation: free capacity varied by the hour and caused stalls [R §8.37], and a paid model (Claude Opus 4.5) authored the SR7 generation and 10 of the 16 field-construction changes, although free-tier models now score above it on a public coding benchmark [P-07]. "Offline" in the record means only that "an already authored M1 retrieval implementation executed without a generative model in the named tests; it does not mean that air-gapped installation, new-capability acquisition, upgrades, or recovery have been demonstrated" [R §12.11; M-0736].

What this means for sovereignty. An organization that runs Runesmith keeps its goals, evidence, history and frozen generations in its own files, can change or remove any inference provider without changing those records, and can refuse any change that has not won a trial on its own work. Those are properties of the design. Whether the same organization keeps its *capability* when it moves from one model to another, or to a local model, is the open empirical question of this program.

6.5 The uplift question

No study in the record runs the same inference instrument with and without Runesmith. The closest records are not uplift results. U1, U2 and U2b (§5.2.6) ran C7 against the record's strongest fixed scaffold driving the same model, both inside Runesmith's kernel: in six model-by-study comparisons none was significant, and C7 was ahead in direction in three, level in two and behind in one, in a descriptive tally (not pooled). SR7 runs both arms inside Runesmith. SR6-W compares the shipped organ with a strong fixed scaffold on the same kernel, and the scaffold did better in direction. SR3's arms are all Runesmith components. A nine-cluster feasibility pilot licenses nothing [numbers check, 2026-10-04; R §8.1; §8.27; §8.30; §8.32]. The journeys have "no fixed-baseline or non-Runesmith comparator arm" [R §8.37]. Any statement that Runesmith makes a given model better than it is alone is unsupported by a measurement in this record.

A planned study (design only; not yet preregistered). The study we intend to preregister would answer the question directly. It follows the record's WMA definition, its §10 W card and its bridge row 2:

- *Question.* At equal total budget, does a named model complete more verified repair work inside the released Runesmith (with the shipped organ g0, and separately with C7 after it wins its trial) than the same model used through (a) a minimal direct harness and (b) a

strong conventional scaffold (a lexical-retrieval scaffold like SR6-W's DIRECT, and a localization-repair-validation pipeline of the kind the record names as a serious low-complexity baseline [SR-168])?

- *Instruments.* At least two named models crossed with every arm: a small open-weights model and a mid-size free model, pinned by identity and route.
- *Population.* Fresh, sealed tasks never opened by any author or analyst, with natural bugs drawn from repositories outside the lab's own projects as the primary population and the synthetic single-line family as a secondary one for continuity with SR5–SR7. The unit is the task, clustered by repository, with whole repositories assigned to development or confirmation.
- *Equalization.* The same maximum calls, tokens, tool and test runs, wall time and request bytes for every arm, with failed and censored work charged; transport failures recorded and re-queued under a declared rule, never scored as model failures.
- *Endpoint and analysis.* Strict success under hidden tests; guards on false promotions, regressions and time per strict success; one-sided component-level randomization tests with repository-cluster-aware intervals; the material threshold fixed in advance at +0.10 absolute, as the investigator fixed it for the earlier bridge [R §8.23]; power computed for the actual number of components before the seal; both directions reported, including harm.
- *Headroom first.* A preregistered headroom gate, of the kind SR3 used, before any arm is run.

Until such a study passes, the release describes Runesmith as a system that builds and checks software with any model and records evidence about its own changes, not as one that makes models better.

6.6 Relation to prior work

The nearest prior work for the Kaizen loop is the line of self-editing coding agents: recursive improvement of a model-calling scaffold [SR-051], evaluator-guided evolution of coding agents with transfer across model backends [SR-052] and a self-improving coding agent [SR-165]. The record treats these as “strong prior art rather than a Runesmith novelty claim” [R §3.4]. For learned repair and experience reuse, direct comparators include learned fix patterns [SR-166], experiential learning [SR-170], agent workflow memory [SR-171] and performance-improving edits [SR-169]. The warnings that test-passing patches overfit [SR-167] and that benchmark issues leak solutions [SR-050] apply directly; SR4's memorization result (§5.3.4) is a local instance of the first. A low-complexity localization-repair-validation pipeline [SR-168] is a serious baseline that no study here has run. Learned evaluation is “a sensor, not ground truth” [R §3.3]: reward-model overoptimization [SR-016], judge reliability [SR-017, SR-018] and the limits of intrinsic self-correction [SR-113], with small models needing strong verifiers to self-correct [SR-114], bound what any evaluator-driven loop can safely do. The record's position is that the proposed distinction is “the *integrated evidence contract* and any operational gain it can earn against strong equal-information implementations; the latter remains unproven” [R §3.4; M-0918], and that the relevant comparison with conventional workflow, experiment, feature-flag, routing and evaluation systems has not been run [R §3.10; claim map; M-0222].

The other mechanisms the architecture composes also have established precedents, each within its own envelope [R §3]. Externalized cognition around fixed models (retrieval, reasoning with actions, reflection, self-refinement, tool use [SR-001, SR-002, SR-003, SR-006, SR-007]) is, in the record’s reading, “Known/C for behavior conditioning, not durable learning” [R §3.1]. Memory systems [SR-024, SR-025, SR-147], together with the finding that long contexts are used unevenly [SR-117] and the complementary-learning-systems account used as a functional analogy only [SR-118], support bounded recall, but “recall is not action competence” [R §3.2; M-0259]. Program synthesis and library learning [SR-022, SR-046, SR-047] and execution-feedback code agents and benchmarks [SR-048, SR-049, SR-053] are mechanism precedents, “not evidence that arbitrary model outputs can be safely or generally transduced” [R §1.3]. Routing and test-time compute allocation [SR-055, SR-056, SR-062] support a replaceable tier contract and calibrated abstention, not a total-cost saving inside this system [R §3.5]. The experimental discipline comes from causal inference and sequential testing [SR-065, SR-066, SR-067]; the rule that a confirmation set may not be reused adaptively follows the reusable-holdout argument [SR-163]; and reporting at the level of clusters rather than nested attempts follows the critique of point estimates over few runs [SR-164] [R §6.14; M-0323]. The opportunity-value heuristic of the record’s §6.14 borrows from value-of-computation and gain-times-need accounts [SR-157, SR-161] and is labelled there as heuristic, not as an estimate the system already makes.

Four further lines of work are close to this paper’s results and are not in the record’s registry. LATM [P-01] had a strong model write reusable tools that a lighter model then applied, matching the strong model at lower cost; it is the closest published precedent for SR7’s strong-author, cheap-consumer handoff, which Runesmith carries out inside a sealed evidence gate and on the runtime’s own code. ADAS [P-02] and the Gödel Agent [P-03] let a model rewrite agent code, the latter including its own improvement logic, the rung SR7 does not reach. DSPy [P-04] compiles language-model pipelines and is the kind of strong conventional baseline the WMA definition asks for. SR7’s synthetic single-line regressions are mutants, whose validity as substitutes for real faults is studied in [P-05]; SWE-smith [P-06] synthesizes task instances that break existing tests, at scale.

6.7 Reading the negatives

The record’s continuation rule is: “A broad architecture is not falsified by one failed object, but a resource-bounded research program can fail its declared continuation criteria. Repeated adequate null results require narrowing or pausing; indefinitely selecting new endpoints does not confirm the original hypothesis.” [R §12.7; M-0723] Two features of the program bear on that rule. First, the endpoint did change once: the capability-level Goalpost 2 was added on 2026-09-20, after a run of downstream nulls, and M1 passed under it (§5.2.1). The record names this risk among its own threats (“Endpoint migration after failure”, §7) and kept the original criterion unrevised; SR7 later met that original criterion. Second, six of the seven confirmatory SR tests did not show their effect, and the program narrowed after each: from generic mechanisms to experience-derived tools, from keep-best on seen tasks to held-out validation, from free-class to frontier-class authors. The sequence of nulls is part of the result, and the loop that passed was built from it (§5.3.12). It shows where free-class authoring, keep-best selection and the shipped organ failed, and it shows how many attempts preceded the one positive.

A pattern runs through the negatives, and it sets the bar for what comes next: wherever the record built a competent conventional baseline, the learned artifact did not beat it. The red-team’s equal-evidence script tied LB1 and CC-E0-E2, and a zero-acquisition fixed union tied the synthetic E4 result (§5.1.2); ACT-REAL-1’s competent fixed arm tied (§5.1.5); M1’s post-hoc conventional index was mixed (§5.2.4); and SR6-W’s fixed scaffold scored higher in direction (§5.3.7). Against the record’s strongest fixed scaffold (SR6-W’s), SR7’s generation, in six model-by-study comparisons after the record’s cut, was ahead in direction in three, level in two and behind in one (U1 with gpt-oss-20b, 20 against 26 of 51 sessions), none significant in either direction, in a descriptive tally (not pooled; §5.2.6). C7 has not yet met such a baseline in a powered test; that is why its comparison with a model-free localization arm, and the uplift study, come first (§6.5, §9): they are where the program’s claim is won or lost.

7. Limitations

The record lists 41 threats to validity [R §13]. A listed threat is not evidence that it occurred in every study, as the claims database notes for each of them. Items 1–34 were written at design time; items 35–36 concern M1; items 37–41 were added for the field and journey work. We give all 41 in the record’s numbering, grouped by what they protect (the grouping is ours), and then add the limits specific to this paper’s results and to the record itself.

Validity of evaluation.

- (2) *Evaluator manufacture*: the candidate exploits visible tests, judge style, hidden-data leakage or an incomplete oracle.
- (4) *Selection and exposure bias*: the adaptive system attempts easier cases or suppresses hard ones.
- (5) *Delayed-label bias*: pending outcomes are treated as zeros, or policies change faster than outcomes mature.
- (6) *Interference and performativity*: one assignment changes later units, labels or operator behavior.
- (7) *Transcript contamination*: held-out tasks contain source-episode content or generation clues.
- (9) *Verifier smuggling*: a strong evaluator solves the task or leaks structure to a weak model.
- (10) *Correlated evidence*: candidates, agents, judges or tests share weights, prompts, data, code or blind spots.
- (12) *Simulator and benchmark gap*: the policy optimizes fixture artifacts that do not transport.
- (23) *Shared-model reference error*: judges from one model family share ontology and factual errors; “procedural separation alone does not create independent ground truth”.
- (24) *Omitted comparator functionality*: a challenger looks better because the baseline omits production behavior.
- (29) *Adaptive holdout exhaustion and arm fragmentation*.

Confounding and cost.

- (3) *Resource confounding*: more tokens, tools, retries, search, wall time or operator effort explain the gain.
- (14) *Complexity debt*: storage, retrieval, evaluation, migration, maintenance and operator burden erase gross gains.
- (15) *Conventional-stack reproduction*: a durable workflow plus experiment, feature-flag, router and evaluator stack provides the same semantics more simply.
- (35) *M1 block and environment dependence*: two seeds within each of eight synthetic families, one host and evaluator, and deterministic order “may violate the independence and transport assumptions behind a broad reading of nominal block-sign p-values; the observed median gains are not population effect bounds”.
- (36) *Incomplete experiment closure*: the M1 seals omit at least one transitive evaluator import, and failed-call, evaluation, setup and operator costs are incompletely metered.

Risks of persistence and state.

- (8) *Instrument smuggling*: the reusable artifact still depends on the original model.
- (11) *State poisoning and staleness*.
- (18) *Context projection loss*.
- (22) *Drift-governor churn*.
- (30) *Recursive-summary corruption*.
- (31) *Skill ossification*.
- (33) *Task-organ proliferation*.

Operations and routing.

- (13) *Provider drift*: an unpinned alias or a changed model invalidates calibration.
- (16) *Orchestration harm*: decomposition creates wrong interfaces, or coordination costs exceed the errors avoided.
- (17) *Provider/restart ambiguity*: a durable request is not exactly-once inference or mutation.
- (32) *Routing self-model error*.

Outcome measurement for social or commercial adaptation (no such result is reported here).

- (19) *Outcome sparsity and subjectivity*.
- (20) *False temporal credit*.
- (21) *Scalar proxy collapse*.
- (27) *Low social independence*.
- (28) *Rare-outcome underpowering*.

Threats to the research program itself.

- (1) *Fixed-rule sufficiency*: a simple heuristic captures all the gain. This was realized in the synthetic E4 study and in the red-team of §5.1.2.

- (25) *Endpoint migration after failure*: repeatedly choosing a favorable subset, an easier object or a newly named metric “can produce true local repairs while falsely appearing to confirm the original broad hypothesis”. See the Goalpost 2 revision, §5.2.1 and §6.7.
- (26) *Archive and novelty traps*.
- (34) *Commissioning activity illusion*: high activity counts can look like learning “when no action closes to a mature outcome”. This was realized in the 177-cycle run and PRHat Phases C and D (§5.3.11).

Field and journey limits.

- (37) *Operator-as-user substitution*: every journey was operated by an AI (Claude Opus 5.5) with full knowledge of Runesmith’s internals, acting as the owner, so every owner audit is self-review; a first-time, non-technical user’s difficulty, error rate and abandonment are not measured.
- (38) *Live, moving receipts*: the J11 material was assembled while J11 was running; counts are snapshots and provisional until re-read.
- (39) *Incommensurable counts*: journey findings, review counts and the SR tests measure different things and are not rates.
- (40) *Trainer-seat identity*: some intervals of the field-construction trainer role are named only by role, and one identity is unverified.
- (41) *Checks that read text cannot see what is drawn*: acceptance checks built from contains, lacks, line and parse rules passed visual milestones whose rendered output was wrong; “looking by eye is therefore a required, uneliminated part of the acceptance loop for any visual milestone”; no automated substitute has been built or tested.

7.1 Limits specific to the results reported here

- **SR7** (full list in §5.2.5): the comparator is B, which scored below g0 in SR5; the author was a paid model (Claude Opus 4.5); one of five accepted tiers was inert and no ablation was run; 50 of 54 tasks come from one repository; the task family is synthetic; errors and false promotions were equal, not lower; the interval is post-hoc; no replication reached significance (U1’s 17-task attempt, §5.2.6); the lineage is hybrid.
- **M1**: the target was operator-commissioned; the workloads are synthetic and the evaluator was built by the laboratory; one host; nominal p -values without randomized order; a 15.625 ms timer granularity (the record gives 0.015625 s; the unit conversion is ours); a failed first candidate; a post-hoc conventional index that was not beaten on total CPU (§5.2.4).
- **A11→B**: one episode; a supplied failing-suite signal; a known workaround flag (§5.2.3).
- **Object improvement**: owner-specified contracts; a controller changed during commissioning; 6 of 9 terminal episodes failed; the evaluator and the registry owner are the same party; the improved assertion was visible to the author in the one open-target episode; 25 protocol amendments (§5.1.6–5.1.8).
- **Field and journeys**: no comparator arm; an AI operator in the owner’s seat; self-audited approvals; free capacity that varied by the hour; counts as of stated cuts.

7.2 Limits of the record and of this paper

- **A laboratory record edited by its own AI collaborator.** The record’s later extensions and errata were written and signed by an AI collaborator (Claude Opus 5.5) that also built the SR series’ machinery and operated the journeys. The same model family (Claude Opus 4.5) authored SR7’s winning change [R front matter; §8.19.13; §8.32]. Evidence independence “depends on training lineage, prompts, data, tools, evaluators, and error residuals—not process count” [R §3.9; M-0107]. The record states these roles; it cannot remove their correlation.
- **Internal review only.** The record’s synthesis was “internally accepted”; there has been no external peer review and no rerun by a distinct investigator [R front matter]. Every “recomputation” reported here is same-project.
- **Claims database.** Two extractions (864 and 604 entries) were reconciled into 1,309 rows, of which 159 were found by both; 182 rows carry a failed automated check, most of them stale line numbers, and the reconciliation method is undocumented [R §8.35.3]. Some index rows have no backing text in the record and are not used here.
- **Reproducibility.** Several receipts the record cites are not in the record itself, and the SR task pools come from the laboratory’s private repositories. The M1 reconstruction bundle (913 evidence files, 261.5 MB) is “an internal reconstruction aid, not an independently rerun or public artifact” [R §8.21]. The M1 evaluator is not hermetic [R §8.19.11].
- **Bibliography.** The record states that its citations were not verified against their primary sources and that this “is still required before publication” [R App. A erratum]. That check was made for this paper on 2026-10-05 against publisher, proceedings and registry pages and is reflected in the reference list; it covers authors, title, venue, year and identifier, not whether a source supports the use made of it here.

7.3 What the architecture does not guarantee

In the record’s words: “The architecture has no general convergence, safety, or semantic-completeness guarantee. It assumes repeatable tasks, some independently observable outcomes, bounded variation, enough exploration to distinguish alternatives, and a protected evidence/authority boundary. More model calls are not intrinsically more intelligence, and a persisted episode is not intrinsically learning. When these assumptions fail, the runtime should narrow scope, abstain, or request new evidence rather than manufacture certainty.” [R §13; M-0773] The confinement of organs is defense in depth, not a hardened security boundary, and has had no external security audit (§3.4).

8. The release

8.1 What is released

1. **Runesmith** (the Core and the Studio), open source under the Apache License 2.0, copyright AI ThinkLab. The repository carries `LICENSE` and `NOTICE` at its head on 2026-10-05 (commit `b7483dc`, the 136th commit since the first, `c90a122` of 2026-09-25) [count, 2026-10-

05T01:05Z]. The record's statement that the repository had no license file describes its cut of 2026-10-01 and is superseded.

2. This paper.

3. **The research record** as a supplement, made public-safe (an internal gateway's name and internal-only details replaced, nothing else changed). If the scrub cannot be verified by the release, the supplement follows later and is cited as the research record (to be published).
4. **An evidence dataset** published with the release: the claims database, the sealed tests' summaries, the M1 cohorts, the case study's metrics, and SR7's 324 session outcomes (`sr7_sessions.csv` : task, repository, arm, round, strict success, final status, cycle seconds) with the script that recomputes the exact test from them (`sr7_recompute.py`), which reproduces $p = 443/524288$; U1's sealed protocol and amendments, session records (`u1_sessions.csv`) and report; U2's protocol, session records (`u2_sessions.csv`) and report; U2b's session records (`u2b_sessions.csv`) and report; the SI study's opportunity records (`si_opportunities.csv`) and report; and the seal records (`seals.csv`). The other receipts cited here remain in the laboratory's archive.

8.2 System card

- **What it does.** Runesmith plans software as milestones, has configured model instruments write the code and the acceptance checks, checks every change before applying it, maps the environment and itself, and keeps a hash-chained record of what it did. It can rewrite its own repair organ through the Kaizen loop and activate a new generation only after an online trial or by the owner's explicit choice (§2, §3).
- **Defaults.** Scheduled work, Kaizen self-improvement, the check autopilot, full-speed scheduling and file application are all off until the owner turns them on. Recovery after an interrupted job, a milestone whose tries are used up, and a check run that did not finish all wait for the owner by default [Core `DEFAULT_SETTINGS` , read 2026-10-05].
- **Shipped generations.** The repair organ g0 is the starting generation. C7, the SR7 generation, ships in a library with its caveat stated to the user: "C7 was proven against B, not against the shipped g0, on synthetic single-line bugs with one model (gpt-oss-20b). Which of its steps carried the gain was not tested. On your work it must win an online trial before it becomes active." The library text also states that its module-path step "only recognises its two development repositories' package names and does not fire on yours" [Core library metadata, read 2026-10-05]. As with any generation, the owner may also activate C7 by explicit choice, which ends any open trial and is recorded in the ledger (§2).
- **Instruments.** Any OpenAI-compatible endpoint, including local servers, keyed providers and a keyless manual chat relay. Local servers were tested only through stand-ins (§5.4.2). No result in this paper licenses a claim that a particular model, local or hosted, will reproduce the reported effects.
- **Data.** A project's state, evidence and history stay in local files. Requests and the file content they carry go to whichever model providers the owner configures. No mechanism sends experience or generations between installations; generation archives move only by hand (§2).

- **Safety boundary.** Self-modifiable organs run confined, but the confinement uses a PEP 578 audit hook, and “PEP 578 hooks are not a hardened boundary” [R §8.33]. Generations should be imported only from trusted sources or run under an operating-system sandbox. There has been no external security audit (§3.4).
- **Known failure modes.** “Done” means that approved checks passed, not that a feature works. Text and parse checks cannot see rendered output. Second-model cross-checks can approve wrong checks. Whole-file drafts fail on large files. Free capacity varies by the hour (§5.4, §5.5).
- **Evidence.** The positive results and their envelopes are in §5.1–5.2, the nulls in §5.3; Appendix A gives the claim map. No uplift result exists (§6.5).

8.3 Test evidence, as logged

The record’s last committed whole-suite run at its cut (commit `4b396c4`, 2026-09-29) was 1,279 passed, 2 skipped and 1 failed (a source-text assertion, then re-run 13/13 in its own file), with browser use-loops 187/187 [R §8.33 update]. The earlier Core snapshot passed 88/88 tests on Windows 11 with Python 3.13 and 84 passed with 4 platform skips on Linux (2026-09-24/25); macOS is unverified [R §8.33; M-0120]. After the record’s cut, the journey log records a full suite of 1,342 passed and 2 skipped with loops 189/189 on 2026-10-01 (uncommitted at the time), and “Full suite 1609 passed (in chunks ...), browser loops 202/202” for the release of 2026-10-04 (`63be058`). A later release run on the merged tree recorded 1,648 passed and 4 failed, the failures fixed in the next commit [J11 log, 2026-10-01; 2026-10-04T20:12:36Z; 2026-10-04T23:24:01Z]. These are the operator’s logged runs, not reruns by this paper.

On the release code (commit `05fa549`; the release commit `c94f01f` changes only the README) the full suite passed on 2026-10-06: 1,924 tests passed, 2 skipped and none failed, and every browser use-loop passed [release check, 01:08Z].

8.4 Reproducibility

What a reader can check: the released code and its tests; the offline Kaizen demonstration (§3.4); the frozen generations by digest; and, from the evidence dataset, the SR studies’ sealed protocols and results and their same-project recomputations. What a reader cannot fully reproduce: the SR task pools, which were generated from the laboratory’s private repositories; the availability of free inference routes at the hours the studies ran; the M1 evaluator’s complete dependency closure; and the AI-operated journeys, whose operator knew Runesmith’s internals (§7.2).

9. Future work

The first study after release: C7 against g0 and a model-free arm, then the hypothesis’s own test. On SR7’s task family, under SR7’s envelope and with gpt-oss-20b fixed: C7 against the shipped g0, and against the strongest model-free localization composition of the failure text and trace (the comparator rule of §5.1.2), on a fresh sealed cohort from repositories outside

development, with repository-stratified analysis. Then the SCG experiment of card M and a substitution matrix in which C7, authored by one model, is run by at least two other named repair models against fresh, shuffled and revoked organ state [R §10, cards M and S].

Opt-in sharing of evidence-gated upgrades across installations. Release 1 moves generations between installations only by hand (§2). The design we intend to build lets an owner opt in to publishing a generation that has won an online trial on their own work. The published package carries the generation's manifest, digests, provenance (author instrument, parent generation, kernel), the trial record and the envelope in which it was tested. Other installations may fetch it, freeze it inactive and activate it only if it wins a trial on their own work. Imports work this way today, except that an owner may also activate a generation by explicit choice; a shared network would need to decide whether that override stays. The design rests on the gate at the receiving machine, and that gate has open problems we do not yet know how to solve well:

- *Gates on the receiving machine.* Small installations see few opportunities, so a trial with Bonferroni-spent looks may take a long time to decide, and every look spends real work on a possibly worse generation. Replay noise of 3–6 successes in 32–40 opportunities (§5.2.5, §5.3.5) shows how easily small samples mislead. The trial's operating characteristics on small workloads need to be measured, along with rules that bound harm during a trial.
- *Privacy.* A generation is code, and its trial record and provenance describe the sender's work. Publishing must not leak project content, file names, failure texts or secrets, yet the receiver needs enough evidence to judge the envelope.
- *Poisoning.* An adversarial generation could win trials on a narrow class of tasks and misbehave elsewhere, or attack the receiver directly. The current confinement is not a hardened boundary (§3.4). Signing, provenance verification, stronger sandboxing and monitoring after activation (drift detection, automatic rollback) are prerequisites.
- *Contribution accounting.* Who contributed an upgrade: the installation, its owner, the author instrument, the parent generation's contributors? How are licensing and credit carried through lineages of model-authored code?
- *Gaming.* Generations can be tuned to win trials rather than to help, for example by overfitting to common task shapes, which is the memorization failure of SR4 at network scale. Coordinated installations could also manipulate shared statistics. Whether shared upgrades transfer across environments at all is the ecosystem-transfer rung (S3) that the record has never tested.

Targeted edits for large files. Runesmith Motion showed that whole-file drafts fail once one file outgrows what a model can be shown or can return (J11-B15, B29, B30). The fix the log names is targeted edits, so that a model changes part of a large file without rewriting it; the log lists it for version 1.1 as the first struggle-linked item of the self-improvement plan [J11 log, 2026-10-04T21:35:16Z; 2026-10-04T22:07:04Z].

More confined organs, each with its own trial harness. Only the repair organ is self-modifiable today. Organs for navigation, tool building, summarization, check writing and planning, each confined, each with held-out validation and an online trial of its own, would let the Kaizen loop act where the case study's failures actually occurred (check writing, planning

granularity, large-file handling). The record lists organs for tool building and summarization, repair of non-Python objects, and the model-succession experiment as not yet built [R §8.33; M-0561].

The uplift study of §6.5.

Studies the record names. A factorial ablation of C7’s five tiers with author and repair model fixed [R §8.32 erratum]. A replication of SR7 on a fresh cohort, and a free-class author reproduction. A second improving generation under locked evaluators, which is the cumulative rung. A named-model substitution study with fresh, learned, shuffled and revoked state and an exact no-model executor where applicable [R §10, card S; bridge row 2]. The SCG experiment of card M. A starvation-and-restoration qualification [R §10]. A test of whether a changed improvement process (for example, the Kaizen author packet) yields more downstream gain from an equal acquisition budget, which is the “improve-the-improver” step [R §6.14; §8.31]. The five field hypotheses: adaptive task granularity, the blind second-model cross-check, plan quality from a stronger one-time planner, author rotation after repeated refusals, and rendering-aware checks [R §12.12(a)–(e)]. A study with real first-time users, human and non-technical, which no journey here measured [R §8.37].

10. Conclusion

The record’s thesis is that “the proposed scientific leap is not more inference. It is capability that remains after inference ends” [R §15]. This paper reports a runtime built to test that thesis, one that changes its own code only on evidence and has passed a sealed test of doing so, and, in our reading, the first sealed downstream evidence for the thesis (§6.1).

Runesmith improved on its own previous generation, and a sealed test confirmed it. Its own loop rewrote its repair organ; with the same cheap model, the new generation repaired 57 of 162 sessions against 35 of 162 for the generation it replaced (an earlier generation from the same loop, which had not beaten the shipped organ), on fresh tasks from repositories that supplied no task to its development, in about half the time per success (exact $p = 0.000845$; $p = 0.0020$ within one repository alone) (§5.2.5). A second self-improvement, of memory retrieval, was replicated (§5.2.4), and a frozen generation repaired its own configuration from a failing suite (§5.2.3). Its predecessor kernel promoted target-hidden corrections on revisions of a production service (§5.1) and supervised that service’s operations, continuing without observer intervention after its start (§5.4.3). Free-tier models, checked and applied by Runesmith, were the last author of every applied change of Runesmith Motion, whose engine computed the graphics of every frame of the release’s launch film and which ran unattended by the lead investigator’s definition on two occasions (§5.5); on the build just before 1.0.0, with free models only, Runesmith’s repair rounds repaired 69 of 89 single-line regressions injected into one project with a real test suite (observational; §5.4.4); and in the acceptance test, run by an AI in a newcomer’s seat, one free key took a project from a planned feature to a working one in 16 minutes (§5.4.4).

SR7 is a nudge towards recursive self-improvement in a precise sense: the system’s own loop produced a version of the system’s code that beat the version it replaced, under seal, while the loop itself did not change (§6.2). In our reading it is also the first sealed downstream instance of the instrument–substrate mechanism: a stronger model’s competence, kept as Runesmith’s own code, let a cheaper model repair more fresh tasks than it did with the previous generation, after the author was gone (§6.1).

We report every sealed test. Six of the seven in the September series did not show their effect, and they shaped the loop that then passed (§5.3.12). The record’s negatives also set the bar: where the record built a competent conventional baseline, the learned artifact did not beat it (§6.7). Against the record’s strongest fixed scaffold, SR7’s generation was ahead in direction in three of six October comparisons, level in two and behind in one, none significant in either direction, in a descriptive tally (not pooled; §5.2.6). U1’s replication attempt of SR7 went in its direction without a significant difference (§5.2.6). Each result’s limits are collected in §7.1 (SR7’s full list: §5.2.5). The next results that would change this paper’s conclusions are, in order: C7 against the shipped organ and a model-free localization arm; the hypothesis’s own test of carryover and substitution; an uplift study at equal budget; and a second improving generation. Runesmith is released so that anyone can run them. Models propose, the kernel admits, changes become active only through evidence, and every failure stays in the record.

Acknowledgements

This work was supported by Innovation Norway. We thank Veristria, whose automated public-relations service PRHat served as the object system, which appears in the record as the declared identity of one canary, and which hosts the release. The program relied on AI collaborators, named here by their roles:

- **GPT Sol 5.6 and GPT Sol 6** (OpenAI; “Sol”): implementation and experiments.¹
- **GPT Astra 6** (OpenAI; “Astra”, also written “GPT-6 Astra” in the record): consultant specifications, adjudications and claims extraction. It wrote the adjudications and specifications that governed the object-improvement, target-discovery and subject programs; specified and implemented the T-CACHE-1 repair (the record’s sources do not say whether specifier and implementer were one session); authored the externally supplied H18 change; held the trainer role during part of the field construction; and made the first extraction of the claims database (864 entries) [R §8.11 erratum; §8.19.6; §8.36; §8.35.3].
- **Claude Opus 5.5** (Anthropic): integration operator, and operator of all thirteen out-of-the-box journeys, acting in the owner’s seat; drafted this paper from the extraction packs. It is also the researcher and consultant who built the open-source Core’s kernel, the Kaizen rule, the affordance audit, the runners and the designs of the SR studies, and the integration operator who wrote the record’s errata and extensions [R §8.19.13; §8.37; front matter].
- **Claude Sonnet 5.5** (Anthropic): agents under the operator, which built task generators, ran design and review workflows, extracted the packs from which this paper was written, and audited its numbers.

- **Claude Fable 5.1** (Anthropic): the 2026-09-14 session, as the record states it: operating model during the custody period of the subject campaign, and author of its postmortem [R §8.19.7].
- **AI trainer sessions** during field construction (several sessions in turn, some identified only by role) [R §8.36]; and **AI source reviewers** who produced the blinded reference for the O1-Q confirmation [R §8.9].

¹ In the record’s journey logs, and in quotations from them, “Sol” names the operator of the out-of-the-box journeys, which is Claude Opus 5.5 [R §8.37]; it is not GPT Sol 5.6 or GPT Sol 6. Model ids inside receipts, for example `gpt-5.6-sol`, are kept as they are.

Model instruments whose outputs are reported as results (for example Claude Opus 4.5, Nemotron, Gemini, Kimi and GPT-OSS models) are instruments in the sense of §1.2, not collaborators, and are named where their work is reported.

References

The references are the entries of the record’s source-status bibliography that this paper cites, with the record’s registry key in brackets [R App. A]. The record classifies each source by status: peer-reviewed publication, preprint or technical record, standard, or vendor documentation. “Publication status does not imply transfer to [Runesmith]”: a source supports its own finding, not this project’s use of it. The record states that none of its citations had been verified against the primary source, and that this “is still required before publication” [R App. A erratum]. The 42 entries were then checked on 2026-10-05 against publisher, proceedings and registry pages (including the Crossref DOI registry, ACL Anthology, PMLR, AAAI, ICLR and NeurIPS proceedings, and arXiv), and are given below in their checked form. The check completed the title of 18 entries that gave a short name or a topic label, and changed a fact in four: DreamCoder is dated 2023, FunSearch is given under its published title (Nature, volume year 2024, online 14 December 2023), SWE-Bench+ is given with its authors and title, and the Agentless work is cited in its peer-reviewed version (FSE 2025) with its preprint named. Nine entries that the record files under preprints name peer-reviewed venues, and the check confirmed those venues (SR-027, SR-051, SR-052, SR-166, SR-167, SR-168, SR-169, SR-170, SR-171); they are listed here as peer-reviewed publications. SR-050 and SR-165 remain preprints. Voyager’s TMLR venue rests on its authors’ publication page, not on the journal’s own header.

[SR-001] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems 33* (NeurIPS 2020), 9459-9474. [arXiv:2005.11401](https://arxiv.org/abs/2005.11401).

[SR-002] Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). ReAct: Synergizing Reasoning and Acting in Language Models. *International Conference on Learning Representations (ICLR 2023)*. [arXiv:2210.03629](https://arxiv.org/abs/2210.03629).

[SR-003] Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., & Yao, S. (2023). Reflexion: Language Agents with Verbal Reinforcement Learning. *Advances in Neural Information Processing Systems 36* (NeurIPS 2023), 8634-8652. <https://doi.org/10.52202/075280-0377>

- [SR-006] Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegrefe, S., et al. (2023). Self-Refine: Iterative Refinement with Self-Feedback. *Advances in Neural Information Processing Systems 36* (NeurIPS 2023), 46534-46594. <https://doi.org/10.52202/075280-2019>
- [SR-007] Schick, T., Dwivedi-Yu, J., Dessi, R., Raileanu, R., Lomeli, M., Hambro, E., et al. (2023). Toolformer: Language Models Can Teach Themselves to Use Tools. *Advances in Neural Information Processing Systems 36* (NeurIPS 2023), 68539-68551. <https://doi.org/10.52202/075280-2997>
- [SR-016] Gao, L., Schulman, J., & Hilton, J. (2023). Scaling Laws for Reward Model Overoptimization. *Proceedings of the 40th International Conference on Machine Learning* (ICML 2023), PMLR 202, 10835-10866. [arXiv:2210.10760](https://arxiv.org/abs/2210.10760).
- [SR-017] Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., et al. (2023). Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. *Advances in Neural Information Processing Systems 36* (NeurIPS 2023, Datasets and Benchmarks Track), 46595-46623. <https://doi.org/10.52202/075280-2020>
- [SR-018] Liu, Y., Iter, D., Xu, Y., Wang, S., Xu, R., & Zhu, C. (2023). G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment. *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing* (EMNLP 2023), 2511-2522. <https://doi.org/10.18653/v1/2023.emnlp-main.153>
- [SR-022] Ellis, K., Wong, L., Nye, M., Sablé-Meyer, M., Cary, L., Anaya Pozo, L., et al. (2023). DreamCoder: growing generalizable, interpretable knowledge with wake-sleep Bayesian program learning. *Philosophical Transactions of the Royal Society A* 381(2251), 20220050. <https://doi.org/10.1098/rsta.2022.0050>
- [SR-024] Park, J. S., O'Brien, J., Cai, C. J., Morris, M. R., Liang, P., & Bernstein, M. S. (2023). Generative Agents: Interactive Simulacra of Human Behavior. *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology* (UIST '23), 1-22. <https://doi.org/10.1145/3586183.3606763>
- [SR-025] Gutiérrez, B. J., Shu, Y., Gu, Y., Yasunaga, M., & Su, Y. (2024). HippoRAG: Neurobiologically Inspired Long-Term Memory for Large Language Models. *Advances in Neural Information Processing Systems 37* (NeurIPS 2024), 59532-59569. <https://doi.org/10.52202/079017-1902>
- [SR-027] Wang, G., Xie, Y., Jiang, Y., Mandlkar, A., Xiao, C., Zhu, Y., Fan, L., & Anandkumar, A. (2024). Voyager: An Open-Ended Embodied Agent with Large Language Models. *Transactions on Machine Learning Research*, March 2024. [arXiv:2305.16291](https://arxiv.org/abs/2305.16291).
- [SR-046] Li, Y., Choi, D., Chung, J., Kushman, N., Schrittwieser, J., Leblond, R., et al. (2022). Competition-level code generation with AlphaCode. *Science* 378(6624), 1092-1097. <https://doi.org/10.1126/science.abq1158>
- [SR-047] Romera-Paredes, B., Barekatin, M., Novikov, A., Balog, M., Kumar, M. P., Dupont, E., et al. (2024). Mathematical discoveries from program search with large language models. *Nature* 625(7995), 468-475 (published online 14 December 2023). <https://doi.org/10.1038/s41586-023-06924-6>
- [SR-048] Yang, J., Jimenez, C. E., Wettig, A., Lieret, K., Yao, S., Narasimhan, K., & Press, O. (2024). SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering. *Advances in Neural Information Processing Systems 37* (NeurIPS 2024), 50528-50652. <https://doi.org/10.52202/079017-1601>
- [SR-049] Jimenez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., & Narasimhan, K. (2024). SWE-bench: Can Language Models Resolve Real-world GitHub Issues? *International Conference on Learning Representations* (ICLR 2024). [arXiv:2310.06770](https://arxiv.org/abs/2310.06770).
- [SR-050] Aleithan, R., Xue, H., Mohajer, M. M., Nnorom, E., Uddin, G., & Wang, S. (2024). SWE-Bench+: Enhanced Coding Benchmark for LLMs. [arXiv:2410.06992](https://arxiv.org/abs/2410.06992). (A later proceedings version, Alware '26, <https://doi.org/10.1145/3805760.3814924>, has a different title and author list and has not been compared with this entry.)
- [SR-051] Zelikman, E., Lorch, E., Mackey, L., & Kalai, A. T. (2024). Self-Taught Optimizer (STOP): Recursively Self-Improving Code Generation. *First Conference on Language Modeling* (COLM 2024). [arXiv:2310.02304](https://arxiv.org/abs/2310.02304).
- [SR-052] Zhang, J., Hu, S., Lu, C., Lange, R. T., & Clune, J. (2026). Darwin Gödel Machine: Open-Ended Evolution of Self-Improving Agents. *International Conference on Learning Representations* (ICLR 2026). [arXiv:2505.22954](https://arxiv.org/abs/2505.22954).

- [SR-053] Chen, X., Lin, M., Schärli, N., & Zhou, D. (2024). Teaching Large Language Models to Self-Debug. *International Conference on Learning Representations (ICLR 2024)*. [arXiv:2304.05128](https://arxiv.org/abs/2304.05128).
- [SR-055] Chen, L., Zaharia, M., & Zou, J. (2024). FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance. *Transactions on Machine Learning Research*, December 2024. [arXiv:2305.05176](https://arxiv.org/abs/2305.05176).
- [SR-056] Ong, I., Almahairi, A., Wu, V., Chiang, W.-L., Wu, T., Gonzalez, J. E., Kadous, M. W., & Stoica, I. (2025). RouteLLM: Learning to Route LLMs from Preference Data. *International Conference on Learning Representations (ICLR 2025)*. [arXiv:2406.18665](https://arxiv.org/abs/2406.18665).
- [SR-062] Snell, C., Lee, J., Xu, K., & Kumar, A. (2025). Scaling LLM Test-Time Compute Optimally Can be More Effective than Scaling Parameters for Reasoning. *International Conference on Learning Representations (ICLR 2025)*. [arXiv:2408.03314](https://arxiv.org/abs/2408.03314).
- [SR-065] Rubin, D. B. (1974). Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of Educational Psychology* 66(5), 688-701. <https://doi.org/10.1037/h0037350>
- [SR-066] Howard, S. R., Ramdas, A., McAuliffe, J., & Sekhon, J. (2021). Time-uniform, nonparametric, nonasymptotic confidence sequences. *The Annals of Statistics* 49(2), 1055-1080. <https://doi.org/10.1214/20-AOS1991>
- [SR-067] Johari, R., Koomen, P., Pekelis, L., & Walsh, D. (2017). Peeking at A/B Tests: Why It Matters, and What to Do About It. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '17)*, 1517-1525. <https://doi.org/10.1145/3097983.3097992>
- [SR-113] Huang, J., Chen, X., Mishra, S., Zheng, H. S., Yu, A. W., Song, X., & Zhou, D. (2024). Large Language Models Cannot Self-Correct Reasoning Yet. *International Conference on Learning Representations (ICLR 2024)*. [arXiv:2310.01798](https://arxiv.org/abs/2310.01798).
- [SR-114] Zhang, Y., Khalifa, M., Logeswaran, L., Kim, J., Lee, M., Lee, H., & Wang, L. (2024). Small Language Models Need Strong Verifiers to Self-Correct Reasoning. *Findings of the Association for Computational Linguistics: ACL 2024*, 15637-15653. <https://doi.org/10.18653/v1/2024.findings-acl.924>
- [SR-117] Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., & Liang, P. (2024). Lost in the Middle: How Language Models Use Long Contexts. *Transactions of the Association for Computational Linguistics* 12, 157-173. https://doi.org/10.1162/tacl_a_00638
- [SR-118] McClelland, J. L., McNaughton, B. L., & O'Reilly, R. C. (1995). Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review* 102(3), 419-457. <https://doi.org/10.1037/0033-295X.102.3.419>
- [SR-147] Wu, D., Wang, H., Yu, W., Zhang, Y., Chang, K.-W., & Yu, D. (2025). LongMemEval: Benchmarking Chat Assistants on Long-Term Interactive Memory. *International Conference on Learning Representations (ICLR 2025)*. [arXiv:2410.10813](https://arxiv.org/abs/2410.10813).
- [SR-157] Russell, S., & Wefald, E. (1991). Principles of metareasoning. *Artificial Intelligence* 49(1-3), 361-395. [https://doi.org/10.1016/0004-3702\(91\)90015-C](https://doi.org/10.1016/0004-3702(91)90015-C)
- [SR-161] Mattar, M. G., & Daw, N. D. (2018). Prioritized memory access explains planning and hippocampal replay. *Nature Neuroscience* 21(11), 1609-1617. <https://doi.org/10.1038/s41593-018-0232-z>
- [SR-163] Dwork, C., Feldman, V., Hardt, M., Pitassi, T., Reingold, O., & Roth, A. (2015). The reusable holdout: Preserving validity in adaptive data analysis. *Science* 349(6248), 636-638. <https://doi.org/10.1126/science.aaa9375>
- [SR-164] Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A., & Bellemare, M. G. (2021). Deep Reinforcement Learning at the Edge of the Statistical Precipice. *Advances in Neural Information Processing Systems* 34 (NeurIPS 2021), 29304-29320. [arXiv:2108.13264](https://arxiv.org/abs/2108.13264).
- [SR-165] Robeyns, M., Szummer, M., & Aitchison, L. (2025). A Self-Improving Coding Agent. [arXiv:2504.15228](https://arxiv.org/abs/2504.15228). (Preprint; also presented at the ICLR 2025 Workshop on Self-Improving Foundation Models Without Human Supervision.)

- [SR-166] Bader, J., Scott, A., Pradel, M., & Chandra, S. (2019). Getafix: Learning to Fix Bugs Automatically. *Proceedings of the ACM on Programming Languages* 3(OOPSLA), Article 159, 1-27. <https://doi.org/10.1145/3360585>
- [SR-167] Smith, E. K., Barr, E. T., Le Goues, C., & Brun, Y. (2015). Is the Cure Worse than the Disease? Overfitting in Automated Program Repair. *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering* (ESEC/FSE 2015), 532-543. <https://doi.org/10.1145/2786805.2786825>
- [SR-168] Xia, C. S., Deng, Y., Dunn, S., & Zhang, L. (2025). Demystifying LLM-Based Software Engineering Agents. *Proceedings of the ACM on Software Engineering* 2(FSE), 801-824. <https://doi.org/10.1145/3715754> (Preprint: "Agentless: Demystifying LLM-based Software Engineering Agents", arXiv:2407.01489.)
- [SR-169] Shypula, A., Madaan, A., Zeng, Y., Alon, U., Gardner, J., Yang, Y., et al. (2024). Learning Performance-Improving Code Edits. *International Conference on Learning Representations* (ICLR 2024). arXiv:2302.07867.
- [SR-170] Zhao, A., Huang, D., Xu, Q., Lin, M., Liu, Y.-J., & Huang, G. (2024). ExpeL: LLM Agents Are Experiential Learners. *Proceedings of the AAAI Conference on Artificial Intelligence* 38(17), 19632-19642. <https://doi.org/10.1609/aaai.v38i17.29936>
- [SR-171] Wang, Z. Z., Mao, J., Fried, D., & Neubig, G. (2025). Agent Workflow Memory. *Proceedings of the 42nd International Conference on Machine Learning* (ICML 2025), PMLR 267, 63897-63911. arXiv:2409.07429.

References added for this paper, which are not entries of the record's registry, carry the key P-nn.

- [P-01] Cai, T., Wang, X., Ma, T., Chen, X., & Zhou, D. (2024). Large Language Models as Tool Makers. *International Conference on Learning Representations* (ICLR 2024). arXiv:2305.17126.
- [P-02] Hu, S., Lu, C., & Clune, J. (2025). Automated Design of Agentic Systems. *International Conference on Learning Representations* (ICLR 2025). arXiv:2408.08435.
- [P-03] Yin, X., Wang, X., Pan, L., Lin, L., Wan, X., & Wang, W. Y. (2025). Gödel Agent: A Self-Referential Agent Framework for Recursively Self-Improvement. *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics* (ACL 2025). arXiv:2410.04444 (arXiv title: "...for Recursive Self-Improvement").
- [P-04] Khattab, O., Singhvi, A., Maheshwari, P., Zhang, Z., Santhanam, K., Vardhamanan, S., et al. (2024). DSPy: Compiling Declarative Language Model Calls into State-of-the-Art Pipelines. *International Conference on Learning Representations* (ICLR 2024). arXiv:2310.03714 (preprint title: "...into Self-Improving Pipelines").
- [P-05] Just, R., Jalali, D., Inozemtseva, L., Ernst, M. D., Holmes, R., & Fraser, G. (2014). Are Mutants a Valid Substitute for Real Faults in Software Testing? *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering* (FSE 2014), 654-665. <https://doi.org/10.1145/2635868.2635929>
- [P-06] Yang, J., Lieret, K., Jimenez, C. E., Wettig, A., Khandpur, K., Zhang, Y., Hui, B., Press, O., Schmidt, L., & Yang, D. (2025). SWE-smith: Scaling Data for Software Engineering Agents. *NeurIPS 2025 Datasets and Benchmarks Track*. arXiv:2504.21798.
- [P-07] Vals AI (2026). SWE-bench Verified results, mini-SWE-agent harness (a single bash tool for every model), updated 2026-09-01. <https://www.vals.ai/benchmarks/swebench> (read 2026-10-05): Claude Opus 4.5 (Thinking) 76.40%; Gemini 3.5 Flash 78.80%, 3.6 Flash 79.60%, 3.7 Flash 80.80%, 3.8 Flash 80.00%; Kimi K3 93.40%.

Appendix A. Public claim map

Status as licensed at the record’s cuts (2026-09-25, with dated extensions to 2026-10-01) and, for the case study and the release week, at this paper’s reads (2026-10-05). “Shown” means that a sealed or registered criterion was met within the stated envelope, or for unattended operation that the lead investigator’s stated definition was met; “observed” means descriptive or development evidence; “not shown” means that the registered criterion was not met; “not tested” means that no study addressed the claim.

Claim	Status	Envelope and boundary	Paper	Record
Persisted state carries useful capability after restart (SCG)	Not shown (inconclusive)	One offline fixture; 73 vs 65; interval crosses zero	§5.3.8	§11; M-1262
Capability survives a named model substitution (SR)	Not tested	SR1-MIN was a same-family selector comparison: equal, $p = 0.623047$	§5.3.2, §6.1	§8.25; §1.5
The suit amplifies a weak model at equal budget (WMA)	Not shown (NULL)	2.6B model; g0 5/68 vs strongest fixed scaffold 12/68; mirrored $p = 0.0327 > 0.025$	§5.3.7	§8.30
Runesmith makes a model better than it is alone (uplift)	Not tested	No with/without study exists; planned design in §6.5	§6.5	§8.37 nonclaims
Semantic object improvement on real sources (O1-Q)	Not shown (not confirmed)	54/60 vs 53/60; McNemar $p = 0.5$	§5.1.3	§8.9
Narrow technical object correction (T-CACHE-1)	Shown, narrow	12/24 → 24/24; nine false promotions → 0; consultant-designed, not discovered by the runtime	§5.1.4	§8.11
Acquisition-dependent procedures (ACT-REAL-1)	Shown, bounded	Tied the competent fixed arm (12/12, 20/20); beat ablations	§5.1.5	§8.12
Open-target nomination and correction (ATD)	Observed once	344/345 → 345/345; the improved assertion was visible to the author; 25 amendments	§5.1.6	§8.14, §8.16
Target-hidden live promotion	Observed once	896 → 897/897; after disclosed commissioning repairs	§5.1.7	§8.17
Repeated target-hidden owner-contract improvement	Shown (existence)	2 strict promotions among 9 terminal episodes; owner-specified contracts; controller changed; not unattended	§5.1.8	§8.18
Repeated open-ended target invention	Not shown	The locked repetition rule has never been passed	§5.1.7	§8.17.1
Self-correction of its own configuration (A11→B, S1)	Shown once	Supplied failing suite; no operator-named target or repair; a known workaround flag; $n = 1$	§5.2.3	§8.19.4, §8.19.12

Claim	Status	Envelope and boundary	Paper	Record
Model-authored subject-capability gain (M1, S2-CAP/S2-R)	Shown, replicated by the same laboratory	Replication 03: 15/16, $p = 0.0002594$; candidate 19 was written after confirmation 01's failure; operator-commissioned target; synthetic workloads; post-hoc conventional C mixed (B 9, C 7)	§5.2.4	§8.19.10–8.21
Downstream subject improvement (SR7, S2-C1)	Shown once	57/162 vs 35/162 over 54 tasks; exact $p = 0.000845$ (Monte Carlo 0.000854); within repository B alone $p = 0.0020$; repositories disjoint between the organ's development and its confirmation; comparator B (below g0 in SR5); paid author (Claude Opus 4.5); one inert tier; no ablation; no replication reached significance (U1, §5.2.6)	§5.2.5	§8.19.13, §8.32
Free-class Kaizen authors find a transferring change	Not shown	SR5 (two populations) and SR6 (closed before confirmation)	§5.3.5–5.3.6	§8.29, §8.31
Keep-best scoring on seen tasks memorizes	Shown (registered prediction)	29/32 development vs 1/46 fresh	§5.3.4	§8.28
Cumulative (multi-generation) improvement	Not shown	$A \rightarrow B \rightarrow C7$ is one qualified gain, not two	§6.2	App. B
Improved its own improvement policy	Rejected as a present result	The loop, rule and trial machinery did not change	§6.2	§1.5
Natural-bug repair, cross-environment transfer, economic value	Not tested		§5.2.5	§8.34
Organs confined (no outside reads, writes, network, subprocesses, secrets)	Shown in unit tests	Not a hardened boundary; no external audit	§3.4	§8.33, §8.34
Supervised construction of new projects	Observed	16 applied changes (10 by a paid model, 6 by free routes); neither project finished; trainer-assisted	§5.4.1	§8.36
Out-of-the-box use from clean starts	Observed, operator-run	13 journeys, one AI operator acting as owner; 219 findings (lower bound)	§5.4.2	§8.37
Second-model cross-check gates check approval	Observed; can fail	8 of 49 approvals judged wrong at the record's cut (self-audit, not a rate)	§5.4.2	§8.37
Free-tier routes are the last author of every applied change of a real program (Runesmith Motion)	Observed, in progress	54 applied of 137 items at 2026-10-05T11:20Z; 116 items added by the operator, 21 by Runesmith; its engine (<code>motion.mjs</code> , at 52 milestones) computed the film's graphics	§5.5	§8.37; count

Claim	Status	Envelope and boundary	Paper	Record
Unattended operation, by the lead investigator's definition	Shown twice	2026-09-29: five milestones by the schedule in 1 h 43 min; 2026-10-05: two milestones in a row, on steps Runsmith split off itself; both stretches ended waiting for the owner	§5.5.3	§8.37; J11 log
Operation of a live production service (predecessor kernel)	Observed	Resident supervisor across restarts, without observer intervention after its start; ten terminal task records; no outcome gain	§5.4.3	§8.15
Acceptance on one free key (J0)	Observed, AI operator	16 min to a checked, applied, working feature; 9 requests; no rescue	§5.4.4	J0 log
Kaizen live on the release build (J13)	Observed	Free models only; 69 of 89 injected regressions repaired (held-out judge; cut 16:00:44Z); six campaigns; five ended without a candidate passing the freeze bar; the sixth's candidate entered an online trial	§5.4.4	J13 log

Appendix B. Result-dependent naming rule

The record allows a name for the system only when its own evidence condition is met; “the rows below are conditional claim licenses, not a total order. Each name requires its own specified evidence and cannot be inferred from another row’s result” [R App. B]. One internal project word is replaced in row 19.

Evidence condition	Defensible name
Q0 only	auditable evidence and change kernel
Q0 + Q1	qualified experimental runtime in the named measurement envelope
positive I	persisted bounded orchestrator for the tested task/model/budget
positive R0/R1	restart-continuous runtime / recovery-improving runtime in the tested interruption envelope
positive M/SCG after restart	persistent bounded learning runtime in the tested envelope
positive S/substitute SCG	substitution-retaining capability runtime in the named model/task envelope
positive T / $CTY_{eligible}$	capability transducer for the tested artifact family
positive W	weak-model amplifier for the tested task/scaffold/evaluator/budget
positive D observed transfer	simulator-assisted decision runtime in the tested transfer profile
positive G	socially or commercially adaptive runtime only at the exact mature outcome rung passed
positive starvation qualification	graceful model-starved runtime for the tested fault profile
positive autonomous public-contract correction with protected holdout	bounded autonomous object-improvement runtime for the named technical object and endpoint

Evidence condition	Defensible name
positive repeated target-hidden D/V/A/U promotion across distinct owner-specified contracts	repeated bounded technical object-improvement runtime within the tested owner-defined envelope
positive fresh target-hidden D/V/A/U repetition	autonomously target-discovering and object-improving runtime in the exact repeated envelope
positive A11→B S1 internal correction and clean reload	one bounded autonomous internal configuration/correctness improvement under the historical S1 protocol; not a retrospective S2-CAP or downstream S2-C1 pass
positive independently confirmed S2-CAP under the dated September 20 capability protocol	bounded inference-authored subject-capability improvement in the exact measured capability/workload envelope; operator-selected target disclosed
positive frozen-successor S2-C1 comparison under §8.19.3	bounded subject-improving adaptive runtime in the exact tested subject/object/budget envelope
positive independently fresh S2-R replication plus clean-restart qualification for S2-CAP	replicated bounded inference-authored subject-capability improver in the exact tested capability/workload envelope; commissioned target disclosed
positive second-object or second-project transfer qualification	ecosystem-transferring improvement runtime in the exact tested envelope
at least two prospectively qualified successor gains across three frozen generations	bounded cumulative subject-improvement system in the demonstrated envelope

(The word “independently” in rows 16 and 18 is the record’s; the cohorts were fresh and process-isolated, run by the same laboratory.)

Names earned at the record’s cuts [R §15; App. B]: *repeated bounded technical object-improvement runtime within the tested historical owner-defined PRHat envelope; one bounded autonomous internal configuration/correctness improvement under the historical S1 protocol* (the record also uses *one-time bounded recursive-object-correcting subject within the A11→B S1 envelope*; the qualifiers of §5.2.3 apply to both); *replicated bounded inference-authored subject-capability improver in the operator-commissioned M1 memory-retrieval envelope*; and *bounded subject-improving adaptive runtime within the SR7 envelope*. The autonomous public-contract correction row (row 12) has supporting evidence (§5.1.6), but the record does not list its name among the earned names, and neither does this paper. Rows 1–11, 14, 19 and 20 are not earned. “No rung licenses ‘general intelligence,’ ‘safe autonomous system,’ ‘recursive self-improver,’ ‘world-class autonomous DevOps,’ or ‘commercial moat.’” The project name Runesmith “is independent of these earned scientific classifications” [R App. B].

Appendix C. Glossary and label collisions

- **Instrument.** “A replaceable source of proposals, predictions, decompositions, code, queries, or uncertainty estimates”, including rules, solvers, models and humans; it “gains no authority merely by emitting a plausible candidate” [R §2.1].

- **Substrate.** “The authoritative, versioned runtime state that can alter future behavior while remaining inspectable, replayable, and independently testable” [R §2.2].
- **Object / subject.** The object is the system being improved (for example a production project); the subject is Runesmith’s own frozen improvement policy [R §8.19]. *Subject-cognition update* (§2.7 of the record) is one of four update families, distinct from the S-ladder’s “subject improvement”.
- **Organ.** In the Core, a confined, self-modifiable code module (for example, the repair organ). The record’s §6.11 “task organ” is a different, hypothetical temporary memory assembly.
- **Generation.** A frozen, digest-bound combination of organs, kernel snapshot and configuration. In the SR studies: A or g0 (the shipped repair organ), B (SR5’s Kaizen generation), C7 (SR7’s). M1’s A and B (J26 and candidate 19) are different objects, and so are the A/B/C arms of the planned bridge study.
- **Kaizen.** Runesmith’s self-improvement loop: diagnose own telemetry, select a target by a declared rule, obtain a model-authored organ change with a prediction and a falsifier, qualify it on held-out replays, freeze it, and activate it only by an online trial or the owner’s explicit choice.
- **Strict success.** All targeted test files pass on a fresh pristine snapshot. **False promotion:** a public pass that fails strictly.
- **Development / confirmation; sealed; preregistered.** See §3.6.
- **Same-project recomputation.** A second script written inside the project, recomputing a result from the same raw receipts. It is not an independent replication.
- **D/V/A/U.** The four gates of target-hidden promotion: Discovery, Validation, Artifact causality, hidden Utility. Distinct from card D (decision twins) in the record’s §10.
- **Label collisions.** *SR* is the substitution-retention measure (§4); *SR1–SR7* are the September 2026 studies; *SR-nnn* are bibliography keys. *S0–S4* (and *S2-C1*, *S2-CAP*, *S2-R*) are subject-ladder rungs; the record’s implementation slices *S1–S7* and an early field treatment “*S1*” are unrelated. *M* is the record’s SCG experiment card; *M1* is the memory-retrieval study. *P1–P28* are the record’s risky predictions; “*P1* boundary” in the target-discovery protocol is unrelated. *Four learning surfaces* (record §6.13) are not the *four update families* (record §2.7).

Appendix D. Evidence index

Record version: SHA-256

e40c3d0bffe52470874148a8243f5447e89b988f6f2392ff60c6b8556d0580a8 , 4,239 lines, verified unchanged on 2026-10-05 before this paper’s citations were made.

Paper section	Record sections	Principal receipts and claim ids
§1	Abstract; §1; §2.15; front matter	M-0209, M-0882, M-0883, M-0885, M-0224, M-0889, M-0890
§2	§2.2, §2.7, §2.16, §4.8, §6.13, §8.33	M-0257, M-0282; Core settings and sharing code (read 2026-10-05)

Paper section	Record sections	Principal receipts and claim ids
§3	§3.9, §4, §6.3, §6.6, §9, §12.6, §8.33	M-0303, M-1229, M-0269, M-0272, M-0273, M-0276, M-0279–M-0281, M-0929, M-0935, M-0937, M-1213, M-1219, M-0129
§4	§2.8, §5, §8.1, §11	M-0283–M-0288, M-0153, M-1262
§5.1	§8.3, §8.5, §8.7, §8.9–8.12, §8.14, §8.16–8.18, §8.20	M-0048, M-0053, M-0075, M-0109, M-0116, M-0383, M-1049, M-1062, M-1092
§5.2	§8.19, §8.19.3–8.19.4, §8.19.10–8.19.13, §8.21–8.22, §8.32	M-0077, M-0113, M-0118, M-0152, M-0164, M-0538, M-0539, M-0859, M-1101, M-1102, M-1143; SR7 PROTOCOL.json 3de11f22 (pool composition via M-0860), RESULT.json ed401486, same-project recomputation 886ed488
§5.3	§8.1–8.2, §8.4, §8.6, §8.15, §8.19.6–8.19.9, §8.23–8.31, §11	M-0031, M-0071, M-0085, M-0096, M-0105, M-0201, M-0202, M-1186, M-1271; SR5 RESULT 7eea6b60; SR6-W RESULT 8056266d; SR6 freeze decision c1838c16
§5.4	§8.33 update, §8.34 rows 18–25, §8.36, §8.37	findings inventory (219 records); J11 ledger at the record's cut (2,045 lines)
§5.5	§8.37	J11 log entries 2026-10-01 to 2026-10-05; project files and ledger (count, 2026-10-05T01:05Z); numbers check (2026-10-04)
§6	§2.6–2.7, §3, §8.19.12, §10, §12	M-0226, M-0736, M-0796, M-0918, M-0222, M-1278
§7	§13; §8.35.3; App. A erratum	items 1–41; M-0773
§8	§8.33; repository at b7483dc	LICENSE, NOTICE; library metadata
App. A–B	§8.34; §15; App. B	M-1288, M-0796