AI BUSINESS HARNESS · RECURSIVE SELF-IMPROVEMENT SYSTEM

AI Business Harness: Recursive Self-Improvement System

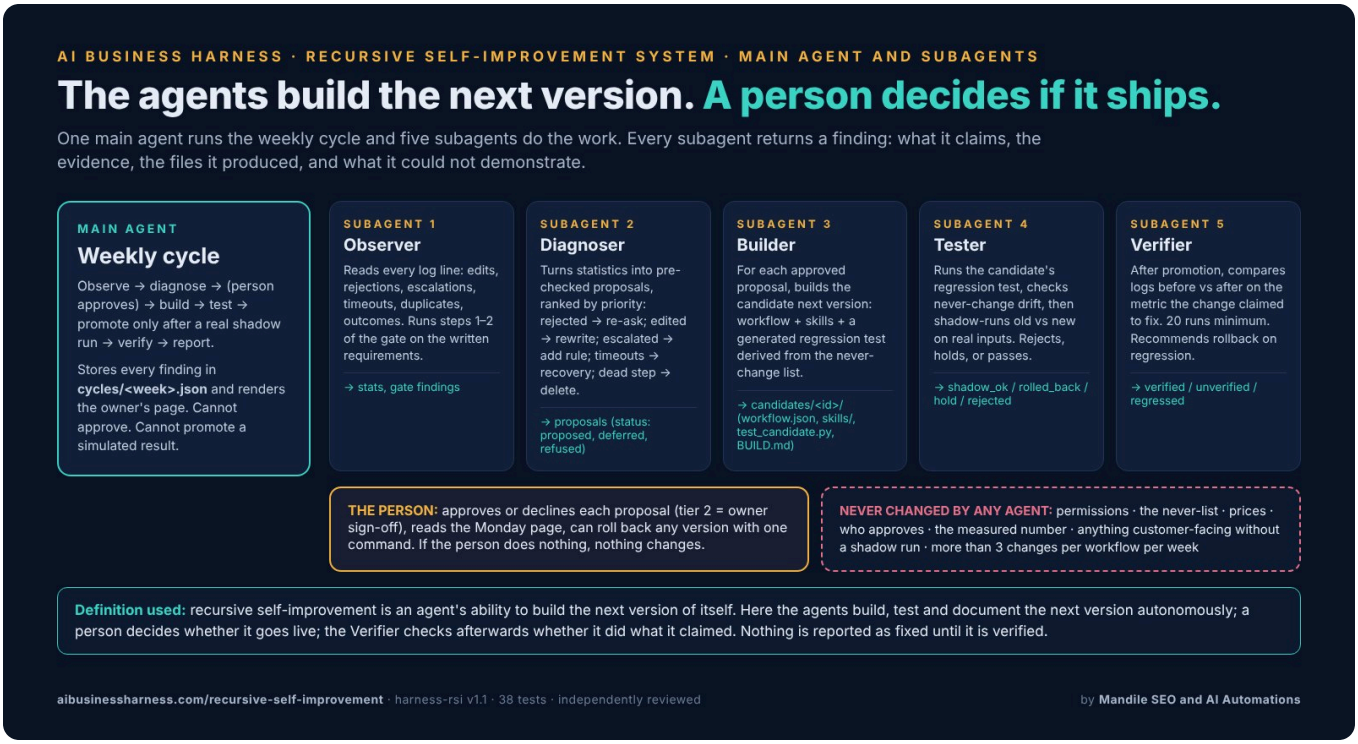
Main agent, five subagents, one person. Architecture, examples, use cases, applications, and what has been demonstrated versus what has not.

Jeremy Mandile · Mandile SEO and AI Automations · November 2026 · harness-rsi v1.1

> **Definition.** Recursive self-improvement (RSI) is an agent's ability to build the next version of itself. In this system the agents build, test and document the next version of a workflow autonomously. A person decides whether it goes live. A Verifier checks afterwards whether the change did what it claimed. Nothing is reported as fixed until it is verified.
> **Rule of this document.** Demonstrated findings come first and are labelled with how they were demonstrated (a test, a recorded run, a file). Anything not demonstrated is labelled a hypothesis. No defect is called fixed unless the repair was verified by a test that failed before and passes after.

1. What "the next version of itself" means here

The thing being improved is a **workflow**: a versioned definition (steps, skill texts, approval map, rules, requirements) that the harness runs for a business. Version N+1 is built by the agents from version N plus the logs of N in use. The agents also improve their own checklist (the gate questions) and their own regression tests, because the Builder generates a new test for every candidate from the never-change list. What the agents cannot build a new version of: their permissions, the never-change list, who approves, or the number they are measured against. Those are frozen outside the loop.


AI BUSINESS HARNESS · RECURSIVE SELF-IMPROVEMENT SYSTEM · MAIN AGENT AND SUBAGENTS
The agents build the next version. A person decides if it ships.
One main agent runs the weekly cycle and five subagents do the work. Every subagent returns a finding: what it claims, the evidence, the files it produced, and what it could not demonstrate.
MAIN AGENT
Weekly cycle
Observe → diagnose → (person approves) → build → test → promote only after a real shadow run → verify → report.
Stores every finding in cycles/<week>.json and renders the owner's page. Cannot approve. Cannot promote a simulated result.
SUBAGENT 1
Observer
Reads every log line: edits, rejections, escalations, timeouts, duplicates, outcomes. Runs steps 1–2 of the gate on the written requirements.
→ stats, gate findings
SUBAGENT 2
Diagnoser
Turns statistics into pre-checked proposals, ranked by priority: rejected → re-ask; edited → rewrite; escalated → add rule; timeouts → recovery; dead step → delete.
→ proposals (status: proposed, deferred, refused)
SUBAGENT 3
Builder
For each approved proposal, builds the candidate next version: workflow + skills + a generated regression test derived from the never-change list.
→ candidates/<id>/ (workflow.json, skills/, test_candidate.py, BUILD.md)
SUBAGENT 4
Tester
Runs the candidate's regression test, checks never-change drift, then shadow-runs old vs new on real inputs. Rejects, holds, or passes.
→ shadow_ok / rolled_back / hold / rejected
SUBAGENT 5
Verifier
After promotion, compares logs before vs after on the metric the change claimed to fix. 20 runs minimum. Recommends rollback on regression.
→ verified / unverified / regressed
THE PERSON: approves or declines each proposal (tier 2 = owner sign-off), reads the Monday page, can roll back any version with one command. If the person does nothing, nothing changes.
NEVER CHANGED BY ANY AGENT: permissions · the never-list · prices · who approves · the measured number · anything customer-facing without a shadow run · more than 3 changes per workflow per week
Definition used: recursive self-improvement is an agent's ability to build the next version of itself. Here the agents build, test and document the next version autonomously; a person decides whether it goes live; the Verifier checks afterwards whether it did what it claimed. Nothing is reported as fixed until it is verified.
aibusinessharness.com/recursive-self-improvement · harness-rsi v1.1 · 38 tests · independently reviewed
by Mandile SEO and AI Automations

2. The agent tree

Agent	Input	What it does	Output	Autonomous?
Main agent	the workflow directory	Runs the weekly cycle in order and stores every finding in `cycles/<week>.json`; renders the owner's page	cycle record, report	Yes, except approval and promotion of anything not shadow-run for real
Observer (subagent 1)	`logs/*.jsonl`	Per-step statistics: edit, rejection, escalation, timeout, duplicate rates; clean streaks; the measured number. Runs gate steps 1–2 on the requirements	`outputs/observe-<week>.json`, gate findings	Yes
Diagnoser (subagent 2)	stats, workflow	Pre-checked, ranked proposals with evidence and expected effect; defers when the weekly budget is spent; waits for new evidence on steps changed this week	`proposals/<id>.json`	Yes (proposing only)
Builder (subagent 3)	approved proposals	Builds the candidate next version: `workflow.json`, `skills/`, a generated regression test, a build note marked NOT TESTED	`candidates/<id>/`	Yes
Tester (subagent 4)	candidates, a shadow runner, real inputs	Runs the regression test; checks never-change drift; shadow-runs old vs new; decides promote / hold / roll back; rejects on any failure	proposal status + test record	Yes
Verifier (subagent 5)	promoted proposals, logs after promotion	Before/after on the side metric the change claimed to fix; needs 20 runs after; marks verified / unverified / regressed; recommends rollback on regression	`verification` on the proposal	Yes
The person	the Monday page	Approves or declines; tier-2 sign-off for loosening an approval; can roll back any version	decisions	—

Every subagent returns the same structure, a **Finding**: `claims` (short sentences), `evidence` (numbers and file paths that back each claim), `produced` (files), and `hypotheses` (what it could not demonstrate, kept separate). The main agent never merges a hypothesis into a claim.

3. One cycle, step by step (recorded)

Recorded on the sample brokerage speed-to-lead workflow (12 steps, 240 logged runs, sample data) with the example shadow runner. Commands and outputs are in Appendix A.

1. `rsi cycle demo` with no approvals: the Observer reports 240 runs and two gate findings (a requirement with no owner; one justified by habit). The Diagnoser writes five proposals, three allowed and two deferred by the 3-per-week budget. Builder, Tester and Verifier report nothing to do. **Workflow stays at v1.** (Demonstrated: `test_cycle_without_a_person_changes_nothing`.)
2. `rsi cycle demo --approve <two ids>`: the person approves the skill rewrite and the routing rule. The Builder writes two candidates, each with `test_candidate.py` and `BUILD.md`. The Tester passes both regression tests, finds no never-change drift, shadow-runs each on 60 real-shaped leads, and passes both. The main agent promotes both (v2, v3). The Verifier marks both **unverified: needs 20 runs after the change; has 0**. (Demonstrated: `test_cycle_with_approvals_builds_tests_and_promotes`.)
3. Later, with 20 runs of logs after the change: `rsi verify demo` marks the rewrite **verified** when the edit rate on those runs is at least 10 points lower than before, and **regressed** with a rollback recommendation when it is not. (Demonstrated: `test_verifier_marks_verified_only_with_evidence_after_the_change`, `test_verifier_flags_regression_and_recommends_rollback`.)
4. If a candidate is tampered with (a permission added by hand), the Tester rejects it: the generated regression test fails and the drift check names the key. (Demonstrated: `test_tester_rejects_a_candidate_that_drifts_a_never_change_item`.)
5. Without a real shadow runner, the Tester's run is labelled SIMULATED, the proposal is held, the main agent does not promote it, and the expected effect is listed under hypotheses, not claims. (Demonstrated: `test_simulated_shadow_is_never_promoted_by_the_main_agent`.)

4. Examples

Example A — a skill that people keep editing. Agents edited the AI's follow-up email in 75% of runs; the logged reasons were "too long", "market commentary", "wrong tone". Diagnoser: REWRITE_SKILL. Builder: a candidate whose skill text is a four-line draft with no commentary (drafted by whichever model the business uses, or the labelled fallback). Tester: regression test passes; shadow run shows the measured number unchanged and the edit rate 70% → 20% on the slice. Promoted. Verifier, two weeks later: verified only if the live edit rate on 20+ runs is at least 10 points lower. If the owner sees "promoted, unverified" on the Monday page, that is the truthful state.

Example B — escalations with one cause. 6% of routing runs escalated, every one a Spanish-speaking lead with no bilingual agent on call. Diagnoser: ADD_RULE (book a callback with the bilingual agent instead of escalating). The rule text is a draft; the owner edits it before approval. Shadow run: escalations for that case 100% → 0%, measured number up. Promoted. Verifier watches the escalation rate on the step after the change.

Example C — a change that fails its own test. Suppose the Builder's candidate for a routing rule somehow also set the follow-up email step to run without approval (customer-facing). The generated test `test_only_the_target_step_changed` fails, so the Tester rejects the candidate before any shadow run. (Demonstrated: `test_generated_regression_test_fails_if_customer_facing_approval_loosened`.)

Example D — the loop improving the business that sells it. The same cycle runs on the sales engine: the scorecard questions and the outreach lines are steps with logs (replies, opt-outs, bookings). A line that is edited before every send is a REWRITE_SKILL proposal; a question whose answer never changes the scorecard is a DELETE_STEP. The never-change list for that workflow includes the outreach rules (real name, address, opt-out, no cold texts), so no agent can propose around them.

5. Use cases by industry

Industry	Workflow under the loop	What the loop typically finds first	What stays with a person
Real estate brokerage	Speed to lead, routing, nurture, contract-to-close	Over-edited follow-ups; one-cause escalations; dead qualifying questions	Offers, pricing, contract questions, fair-housing-adjacent wording
Plumbing / HVAC / electrical	After-hours call handling and booking	Urgency triage questions nobody uses; timeouts on the dispatch calendar write	Quotes over a threshold, warranty claims
Dental / med spa / DPC	Consult booking, intake, reminders, review follow-up	Intake questions answered the same way every time; reminder timing	Anything clinical, procedure pricing
Computer repair / recycling	Drop-off booking, pickup scheduling, "worth fixing?" triage	Triage script edits; duplicate bookings	Any quote; data-destruction certificates
Auto services	Quote request handling, booking, post-service follow-up	Follow-up wording; ad-lead reply timing	Pricing exceptions
Any local business (SEO)	The visibility harness: crawl, schema, entity, freshness, monthly report	Report sections nobody reads; checks that never change a recommendation	Every published change
The agency itself	Scorecard, outreach, booking, proposal	Outreach lines edited before every send; questions that never change the score	Every send, post and price

6. Applications beyond one workflow

- **Fleet of workflows.** One store per workflow; the main agent runs each in turn. The weekly budget is per workflow, so a 10-workflow client sees at most 30 small changes a week, each approved.
- **Model swap.** The only model call is the skill-rewrite draft. Swapping models changes nothing else; the same logs produce the same proposals. A business can run the loop with its platform's model of choice and change later.
- **Audit trail.** `versions/`, `proposals/`, `candidates/`, `cycles/` together answer "what changed, when, on what evidence, who approved, what happened after". That record is what a franchise, a broker or a compliance reviewer asks for.
- **Vendor comparison.** Point the Observer at the logs of a vendor tool (exported edit/escalation records) and the Diagnoser produces the same findings without any power to change the vendor's product: a

scorecard for whether the tool is worth keeping.

- **Onboarding.** Running the gate (`rsi gate`) on a client's written process before building anything is a paid diagnostic in its own right: requirements with no owner and steps that never change an outcome are found before a line of automation is written.

7. What is demonstrated, what is not

Demonstrated (tests and recorded runs in this package): - The cycle changes nothing without a person; simulated shadow runs are never promoted; every version is kept; rollback restores skill text as a new version; the weekly budget holds across proposal types; tier-2 sign-off is required to loosen an approval; the never-change list is enforced at precheck, at the Builder's generated test, and at the Tester's drift check; the Verifier only marks "verified" with 20+ runs after the change and a measurable improvement. 19 tests, 11 guardrail refusals.

Not demonstrated (hypotheses): - That the loop's proposals improve a real client's measured number. The recorded runs use sample logs and an example shadow runner. The first client install is where this is measured, against that client's own baseline. - That the model-drafted rewrites are good. The fallback draft is a labelled template; a real model's output has not been evaluated here. The Tester and Verifier exist precisely because this is not assumed. - The thresholds (50% edit, 5% escalation, 10% timeout, 20 clean runs, 3 changes a week, 10-point verification margin) are starting values, not tuned on client data.

8. Independent review

See Appendix B: an independent reviewer audited v1 for defects with reproductions. Each finding is listed with its status: **fixed and verified** (a regression test that failed before the fix passes after), **fixed, verification pending**, **not reproduced** (hypothesis), or **accepted as a limitation**.

Appendix B: independent review findings and their status

An independent reviewer (a separate agent with no part in writing the code) audited v1.0 and returned ten findings, each with a reproduction test that passed by demonstrating the defect. Every fix below was

verified the same way: the reviewer's demonstration now fails on v1.1, and an inverted regression test (`tests/test_review_fixes.py`) fails on v1.0 and passes on v1.1.

#	Severity	Finding (v1.0)	Fix (v1.1)	Status
1	High	Never-change and tier-2 rules keyed on the proposal's kind and top-level keys; an ADD_RULE carrying `approval: auto` passed as tier 1 and removed a human gate on a customer-facing step. Stored `tier` was trusted at approve.	The gate judges the effect of a change on a copy of the workflow; any loosened approval outside a pure LOOSEN_APPROVAL is refused; tier is recomputed at approve and at promote; promote refuses tier 2 without the owner's sign-off flag.	Fixed, verified (3 tests)
2	High	Weekly budget check-then-spend was not atomic; two concurrent promotes produced 4 changes in a week in 4 of 15 trials. All writes were plain truncating writes.	Promote and rollback run under a per-workflow file lock; every write is temp-file + rename.	Fixed, verified (in-process race and 15 two-process trials, 0 of 15 exceed the budget)
3	Medium	A changed step was re-judged on all pre-change logs as soon as one new record arrived; a declined rewrite came back under a new id when stats shifted.	A changed step is judged only on logs after the change and needs 30 of them; declined, rolled-back and rejected proposals are blocked for the same (kind, step) until 30 new runs, then return with their history noted.	Fixed, verified
4	Medium	Deterministic proposals (same id) were blacklisted forever after a rollback.	Ids are made unique on re-proposal; the decision horizon above governs when they return.	Fixed, verified
5	Medium	One malformed log line disabled every command; `ingest` of a `.json` file was silently dropped; string `seconds` or `value` crashed observe.	Malformed lines are skipped and counted (reported on the owner's page); ingest normalises the extension; numeric fields are coerced, bad values ignored.	Fixed, verified
6	Medium-low	Clean streak sorted run ids as strings (`r10` before `r9`).	Streak follows append order with a stable sort on date.	Fixed, verified
7	Low-medium	DELETE_STEP proposed on a single `changed_outcome: false` record among nulls.	Needs 30 recorded outcomes for the step.	Fixed, verified
8	Low	A shadow result from an older version was promoted onto a newer one; a second rollback re-applied the just-undone change.	Promote requires the shadow run to have been on the live version (the main agent re-shadows automatically); rollback retires undone versions and never restores them by default.	Fixed, verified
9	Low	Revived deferred proposals took budget ahead of new higher-priority ones.	Revived and new proposals are ranked together before the budget is applied.	Fixed, verified

#	Severity	Finding (v1.0)	Fix (v1.1)	Status
10	Low	Rollback with only a baseline raised a bare `max()` error; promote onto a deleted step was labelled deferred; a hand-edited status could promote a simulated result.	Clear error; status follows the refusal reason; promote checks the shadow mode itself.	Fixed, verified

Reviewer items probed and found OK (no change): cross-week budget behaviour, division-by-zero guards, simulated shadow hold, rollback to a non-latest version.

Accepted limitations (not fixed, stated): the budget is per ISO week, so six changes in 48 hours across a Sunday/Monday boundary are possible; rollbacks neither spend nor refund budget; a crash between writing the workflow and its version snapshot is narrowed by atomic writes and the lock but not formally proven impossible (hypothesis from the reviewer, not reproduced).

Appendix A: the recorded main-agent run (v1.1)

```
AI BUSINESS HARNESS - IMPROVEMENT LOOP v1.1 - MAIN AGENT TEST RUN (2026-11-03)

$ rsi init demo --example
example workflow written to /tmp/demo3

$ rsi cycle demo --inputs examples/leads-sample.jsonl     (no approvals yet)
# Improvement cycle 2026-W45 - workflow v1

## observer
- 240 runs observed across 12 steps
- measured number (booked_showing_rate) = 0.296
- gate: DELETE_REQUIREMENT R3 (no one owns it. Step 1: no name, no requirement.)
- gate: CHALLENGE_REQUIREMENT R4 (owner is 'former ops manager', reason is 'it's what we always did'. Step 1:
re-ask the current owner.)

## diagnoser
- rewrite_skill-S11-ae4fb9: REWRITE_SKILL on S11 - edited by a person in 75% of 240 runs [proposed]
- add_rule-S7-47babf: ADD_RULE on S7 - escalated in 6% of 240 runs; top reason: Spanish-speaking lead, no
bilingual agent on call [proposed]
- add_recovery_rule-S9-a2f9cb: ADD_RECOVERY_RULE on S9 - timeouts in 20% of 240 runs [proposed]
- delete_step-S5-00bc4a: DELETE_STEP on S5 - in 240 runs where it was recorded, the step never changed
routing, booking or the outcome [deferred]
- loosen_approval-S12-dff06a: LOOSEN_APPROVAL on S12 - 240 consecutive runs approved untouched [deferred]

## builder
...

$ rsi cycle demo --approve rewrite_skill-S11-ae4fb9,add_rule-S7-47babf --inputs examples/leads-sample.jsonl
## builder
- built candidate v2 for add_rule-S7-47babf (ADD_RULE on S7) with a regression test
- built candidate v2 for rewrite_skill-S11-ae4fb9 (REWRITE_SKILL on S11) with a regression test

## tester
- add_rule-S7-47babf: regression test passed; shadow real 0.367 -> 0.4; verdict PROMOTE (no regression on the
measured number; escalation_rate 1.0 -> 0.0)
- rewrite_skill-S11-ae4fb9: regression test passed; shadow real 0.367 -> 0.367; verdict PROMOTE (no
regression on the measured number; edit_rate 0.7 -> 0.2)

## verifier
- add_rule-S7-47babf: unverified - needs 20 runs after the change; has 0
- rewrite_skill-S11-ae4fb9: unverified - needs 20 runs after the change; has 0

Human decisions this cycle: [('add_rule-S7-47babf', 'approved'), ('rewrite_skill-S11-ae4fb9', 'approved')]
Promoted this cycle: ['add_rule-S7-47babf', 'rewrite_skill-S11-ae4fb9 (re-shadowed on v2)']
```

```
Rule of the report: a change is 'fixed' only when the Verifier marks it verified on logs after the change.
Everything else is 'promoted, unverified'.

(saved to /tmp/demo3/cycles/2026-W45.md)

$ ls demo/candidates/rewrite_skill-S11-ae4fb9
BUILD.md
skills
test_candidate.py
workflow.json

$ python3 -m pytest -q demo/candidates/rewrite_skill-S11-ae4fb9/test_candidate.py
5 passed in 0.01s

$ rsi ingest demo after.jsonl   (20 runs after the change, edit rate 10%)
ingested 20 records into after.jsonl
$ rsi verify demo
add_rule-S7-47babf: unverified - needs 20 runs after the change; has 0
rewrite_skill-S11-ae4fb9: verified - edit_rate 75% -> 10% on 20 runs after the change

$ rsi guardrails demo
GUARDRAILS: ALL PASS

$ python3 -m pytest -q tests
38 passed in 5.39s
```

© 2026 AI Business Harness · A Mandile SEO service · aibusinessharness.com/recursive-self-improvement · Results vary by business; nothing in this document is a guarantee.