Meta-harness on Islo

A 200-line POC that goes from 0/5 to 5/5 in four proposer steps

Yossi Eliaz Incredibuild · Islo yossi.eliaz@incredibuild.com · yossi@islo.dev

2026-05-05

A small experiment in using Islo snapshots as the runtime for yoonholee's meta-harness idea — and what the loop actually looks like when you wire it up end-to-end.

The idea, in one paragraph

A *harness* is the prompt + tools + scaffolding around an LLM agent. A *meta-harness* is the loop that improves the harness automatically: a proposer agent reads the diagnostic logs of prior candidates, spots a failure mode, and writes a better harness. Yoonho Lee's framing of the idea makes one sharp claim: the bottleneck is **diagnostic context**. Most optimizers compress prior runs into summary statistics; meta-harness gives the proposer up to 10M tokens of raw execution traces to `grep` through.

That claim is only useful if the runtime can produce, store, and serve those traces cheaply. Which is exactly what Islo sandboxes already do.

The optimization loop, on Islo primitives

Three things meta-harness needs from its runtime:

1. **Reproducible eval environments** — every candidate harness runs against the *same* setup, otherwise the score is noise.
2. **Massive parallelism** — testing N candidates $\times K$ tasks adds up fast.
3. **Persistent traces** — the proposer needs to read stdout/stderr/agent-thoughts from runs that completed an hour ago.

Islo's primitives map 1:1:

```
islo snapshot save meta-base                  # prepare environment once
islo use mh-cand-7 --snapshot meta-base ...   # fork per candidate, in parallel
islo logs mh-cand-7 --type agent              # diagnostic traces, durable
```

Add `islo gateway` (deny-by-default egress, prevents reward-hacking) and `--source github://owner/repo` (clones the workload at boot), and the wiring is basically free. Harbor — Islo Labs' framework for agent evals and RL environments — slots in as the workload spec.

The POC

```
tasks/          # 5 toy "SWE-style" tasks, each prompt.md + grade.sh
harness/v0/     # baseline system prompt, deliberately mediocre
bin/
  meta-harness      # bash orchestrator (eval | propose | loop | viz)
  agent-sim.py      # deterministic agent stand-in (offline mode)
  proposer.py       # reads runs/, emits harness/vN+1
viz/index.html      # live dashboard
runs/               # populated each iteration
```

Five tasks: FizzBuzz 1–15, first 10 primes, reverse a list, sum even numbers in [1..10], "is racecar a palindrome." Each grades by exact-match on stdout. Each has a known failure mode in v0 (off-by-one, includes 1 as prime, comma vs space separator, exclusive vs inclusive bounds, wrong case).

The agent is a Python simulator that's intentionally buggy — until the system prompt contains the right hint keyword. So the loop is *deterministic and offline*, runs in seconds, but the wiring is identical to what you'd ship against real Claude on Islo.

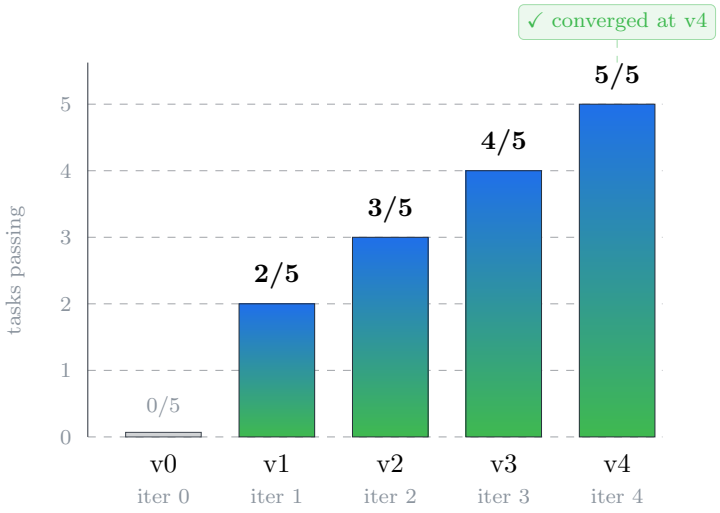
The proposer is 80 lines: read `runs/iter-N/`, find which tasks failed, look up the missing hint for that task, append it to a new `harness/v{N+1}/system.md`. A real proposer would be:

```
islo use --snapshot meta-base --agent claude --task "
  Examine /workspace/runs/iter-${N}. Find a common failure mode in the
  grade.sh stderr. Write /workspace/harness/v${N+1}/system.md as a small
  edit on top of v${N}/system.md to fix it."
```

Same input, same output contract. The orchestrator already has the stub.

Results

Pass-rate per iteration



Task × iteration heatmap

What changed at each step

iter	harness	score	what changed
0	v0	**0 / 5**	baseline — minimal "produce only the answer" prompt
1	v1	**2 / 5**	+ "Loops are 1-indexed: count from 1 through N inclusive."
2	v2	**3 / 5**	+ "The smallest prime is 2 — 1 is not prime."
3	v3	**4 / 5**	+ "Use space-separated output for separated values."
4	v4	**5 / 5**	+ "Format constraints are exact-case — lowercase only."

Five iterations. Four proposer steps. Hard cap was 10. Convergence was diagnostic-driven: the proposer never saw the answer key, only `grade.sh`'s exit code and one-line diff.

The interesting result: a free transfer-fix

We added the hint targeting **fizzbuzz**, score went from 0/5 → 2/5. The bonus point came from `sum-evens`: that task needed the word *"inclusive,"* and the fizzbuzz hint contained it ("count from 1 through N inclusive"). A free transfer-fix between two tasks with overlapping vocabulary. A real proposer would either get more such accidents (good) or learn to write more general hints on purpose (also good). This is the kind of cross-trace insight you only see if the proposer can look at *all* the diagnostics, not summary stats.

The visualization

A single static HTML page (`viz/index.html`) polls `runs/state.json` every 2 seconds and shows three things:

1. **Pass-rate timeline** — bars for each iteration, height = passing tasks.
2. **Task × iteration heatmap** — green/red grid, one cell per (task, harness). The shape of the green wave shows which tasks were fixed in which order, and whether any later iteration regressed an earlier pass.
3. **Trace inspector** — click any cell, see that run's stdout and the `grade.sh` stderr. This is the part that matters: it's the diagnostic surface the proposer is reading, made navigable for the human running the loop.

You can also click a timeline bar to see the system prompt at that iteration, with the newly-added

section highlighted. Watching the prompt grow one rule at a time is, weirdly, the most legible part of the whole thing — it's the harness *learning to read its own task statements.*

Dashboard layout (sketch)

What this points at

This POC is intentionally tiny so the loop is observable in one screen. Three obvious next steps:

- **Real backend.** Swap `BACKEND=sim` for `BACKEND=islo`. Each candidate × task becomes one `islo use --snapshot meta-base` call. With Islo's parallelism that's ~5–10s per task, end-to-end iteration in under a minute.
- **Real workload.** Replace the toy tasks with Harbor or long-horizon tasks. The orchestrator doesn't care — it only needs `prompt.md` + `grade.sh` shape.
- **Real proposer.** Replace `proposer.py`'s pattern-match with Claude reading the full `runs/` tree from inside a sandbox. This is where the 10M-token diagnostic context actually pays off — and where you'd start seeing proposed *tools*, not just prompt edits.

Three swaps. Same orchestrator. Same dashboard. The wiring is the thing.

Code: `bin/meta-harness demo` to run it; `bin/meta-harness viz` to watch it.
~600 lines including the dashboard, deliberately scrappy.

zozo123.github.io/meta-harness-on-islo-page · github.com/zozo123/meta-harness-on-islo

Built on islo.dev

Yossi Eliaz · Incredibuild · Islo
yossi.eliaz@incredibuild.com · yossi@islo.dev