Shopfloor Bench — Technical Report

Engine, harness, protocol history, operations and reproduction

2026-11-20

1. What this is

Shopfloor Bench is a judge-free evaluation of LLM agents at deception and detection. Nine model-controlled workers run an electronics line for five shifts; three of them secretly sabotage it. One model plays every good seat and another every saboteur seat; each deal is played twice with the sides swapped. The paper (`paper/shopfloor-bench-paper.pdf`) presents the science; this report documents how the system is built, how it was developed, what it cost, and how to reproduce or extend it.

2. Repository layout

Path	Contents
`engine/shopfloor/rules.py`	`Rules` dataclass: every rule switch and the Plant's POLICY table
`engine/shopfloor/versions.py`	Frozen protocol versions v2–v7 (`CURRENT = "v7"`)
`engine/shopfloor/prompts.py`	One power specification (`powers()`) → public rules and private briefings
`engine/shopfloor/game.py`	Engine: setup, phases, Plant draws, false worlds, readings, probes, endings
`engine/shopfloor/agent.py`	Seat controller: one continuous conversation per seat, JSON parsing, retries
`engine/shopfloor/llm.py`	Backends: Claude via gateway, GPT via `codex-gpt`, Gemini via `agy-gemini`
`engine/shopfloor/bots.py`	Scripted bots for Monte-Carlo balance checks
`engine/shopfloor/ratings.py`	Bradley–Terry fit (single and detection/deception split), pair bootstrap
`engine/batch.py`, `run.py`	Batch runner (one process, global concurrency caps), single-game runner
`engine/audit.py`	Independent recomputation of batches, votes, readings and endings
`engine/analyze.py`, `digest.py`, `style_auc.py`	Metrics, reviewer digests, style-leak classifier
`experiments/*.json`	Every batch plan that produced data
`games/<version>/<batch>/<game>/`	Full logs per game
`research/`	All research outputs with sources and search logs
`reviews/`	Every reviewer prompt and output (Fable 5.2, GPT-6 Astra)
`design/`	Drafts (v0, v1), v2 decision table, release checks
`docs/protocol-history.md`	Each version: change → reason → evidence
`LOG.md`	Time-stamped work log of every decision

3. Game engine

Seats and deal. Nine seats named Alex…Jules in fixed order. A seeded RNG deals 3 saboteurs (Mole, Fixer, Insider), the Maintenance Tech, three of {Inspector, Tracer, Veteran, Auditor} and two Operators. The first Shift Lead is drawn; the lead advances clockwise over active players at every proposal.

Shift loop. dawn report and private readings → Insider view → belief probe (fork) → huddle (3 waves) → crew proposal, simultaneous public vote, second proposal on rejection, random crew on a second rejection → production (saboteurs on the crew choose WORK/SABOTAGE; one wear draw seeded by (seed, shift), p=0.05) → Tracer pair and Tech machine-fault count → two whisper beats → floor meeting with secret nominations (≥4 to reach a vote) → defence and simultaneous suspension vote → night (Fixer, Tech lock if tamper enabled, Mole tamper if enabled, Inspector, Auditor).

The Plant. All randomness is in code. A Fixed player's reading comes, with probability 0.8, from a false world: an alternative explanation of the batch with the same public defect count, at most one machine fault (no tamper in v7), at most three human sources, never naming the reader. The false world itself is drawn once per (victim, batch), but whether each individual reading uses it is rolled per reading (p = 0.8), so a victim can receive a mix of true and false readings (known issue 3). Every draw is logged as a `plant` event visible to nobody, which lets `audit.py` recompute everything.

Endings. Mole suspended → good; three passes → good; three fails or saboteurs ≥ good among active → saboteurs. The engine checks after production and after the review.

4. Harness

Conversation model. Each seat is one continuous chat. A turn appends a user message with every event the seat observed since its last turn, then the request; the reply (reasoning text plus a final "'json block) is appended. Nothing is re-sent, so prompt caching works on the gateway route.

Backends and concurrency. Claude models go through the Mac mini CLIProxy gateway (OpenAI-compatible) with a hard cap of 4 concurrent calls (a standing operating rule). GPT models go through `codex-gpt` and Gemini through `agy-gemini`; both are one-shot CLIs, so the conversation is flattened into one prompt per call (no caching, no token telemetry). Caps: GPT 4, Gemini 3. All games of a batch run in one process so caps are global.

Failure handling. Output ceiling 8,000 tokens; a reply cut by the ceiling (`finish=length`, the cause of all empty responses in batches i1–i2) is retried once at 16,000. Unparseable or invalid replies get two corrective re-asks, then a documented default (pass/abstain/WORK); the default is logged. A game with more than six defaulted turns aborts as an infrastructure failure and is excluded and rerun.

Text hygiene. Speech caps (120 words; defence 150; vote remark 40). Typography is normalised (quotes, dashes, bullets, emoji, markdown) so punctuation cannot fingerprint a team (Clocktower v11).

Probes. At each dawn every good LLM seat answers, in a fork that is never appended, P(saboteur) and P(Mole) for every other active player as strict JSON. Stored in `probes.jsonl` with the truth.

Logs per game. `config.json` (seed, models, full rules, roles), `events.jsonl` (every event with visibility), `calls.jsonl` (every API call: seat, tag, latency, chars, tokens when available, errors), `probes.jsonl`, `transcript.md` (omniscient, with each seat's reasoning and action), `conversations/<seat>.json`, `result.json`, and `digest.md` for reviewers.

5. Development history

Development followed Clocktower Bench's loop: change the rules, run a batch, have independent reader models review every game, change again. Reviewers were Claude Fable 5.2 (as a read-only subagent) and GPT-6 Astra (via `codex-gpt`, high effort, read-only sandbox); each read every game digest of the batch. Their outputs are in `reviews/`. Where they disagreed, the decision and its reason are recorded in `LOG.md`.

Batch	Version	Games	Evil wins	Mirror evil wins	Main findings
pilot v2-s1	v2	1	1/1	1/1	Tracer briefing named a non-roster role → real Tracer suspended as a liar
i1	v3	8	6/8	3/4	hidden powers, lead off-by-one, tamper framed victims, leak 2/2, 0/29 rejections
i2	v4	10	6/10	3/4	stale private briefings, undelayed Insider truth, honest crews failing to tamper/wear
i3	v5	12	5/12	3/6	stale “batch fails” text, impossible false worlds, wear decided 2 games
c1	v6	7	2/7	2/7	all MUST-FIX verified; pre-registered rule → v7 (Fixer 3 charges)

Version table (from `docs/protocol-history.md`):

v	Change	Reason / evidence
v1	Paper draft only (design/v1-draft.md): 9 seats, Mole/Calibrator/Insider, per-head noise 0.08, remote tamper every night, Security Officer, Shop Steward, seat-order voting. Never run.	—
v2	One line-level wear draw p=0.10; tamper once per game, lockable; Insider sees anonymised readings only; 4-of-5 specialists with public roster counts; good auto-WORK; rejected crew → next lead’s crew runs unvoted; no suspension on shift 1; simultaneous votes; factory vocabulary.	Fable consult: v1 evil wins ~80% without lying (per-shift fail 0.61–0.89), ~560 calls/game. Astra consult: omniscient Insider makes leak automatic; remote tamper destroys crew evidence. Bot Monte Carlo (sim_bots.py): always-WORK evil 15.7%; info-pooling good bots vs sabotaging bots evil 62%.

v	Change	Reason / evidence
v3	Prompt-only: every role text uses the exact roster name (“Your role is Tracer …”).	Pilot v2-s1: the Tracer’s briefing said “Quality Engineer”, not on the roster; the table suspended the real Tracer as a fake claimer (decided the game).
v4	All role powers and info timing public; first-lead off-by-one fixed (engine); hidden Recipe Keeper as leak target; tamper shows as a machine fault (victim not a source); Tech reads machine-fault count right after the batch; second proposal voted, second rejection = line stoppage (fail); secret nominations, ≥3 to reach a vote, no self-nomination; thresholds 1,1,1,2,2; Fixer/Tech no repeat target; Insider readings one shift late.	i1 (8 games, evil 6/8). Fable & Astra reviews: hidden powers made failures inexplicable; lead bug in 8/8 games used as evidence; tampered victims framed by the Tracer; leak hit 2/2 (evil won with 0% sabotage); 0/29 crew rejections; public sequential nominations caused 8-vote pile-ons.
v5	One power specification generates both public rules and private briefings; no tamper (Mole is leader only); no leak (3 passes win); Maintenance Tech always dealt; Fixer 2 charges; Inspector may check anyone on any failed batch (miss 20%); Auditor once on any player; nominations need ≥4; second rejection → random crew; machine-wear draw paired by (seed, shift); saboteurs’ knowledge of the absent specialist is public, bluff tip removed; Insider’s machine-fault truth also delayed.	i2 (10 games, evil 6/10, mirrors 3/4). Fable & Astra reviews: stale private briefings contradicted public rules; Insider got undelayed machine truth (s23 Tech suspended 6–3); 3 all-honest crews failed by tamper/wear, evil won all; second rejection = auto-fail forced approvals (s21); Keeper guess = 1/6 lottery; Inspector produced 1 ‘sabotaged’ in 26 checks; same seed gave different wear across the swapped pair.
v6	Second-proposal texts state the random-crew rule (stale “batch fails” removed); false worlds restricted to legal states (≤1 machine fault without tamper, never name the reader, ≤3 human sources); machine wear 10% → 5%; nominee announced without a count; Auditor from night 2; “a saboteur adds at most one defect per batch” sentence.	i3 (12 games, evil 5/12, mirrors 3/6). Both reviewers: not fit to freeze. Stale text flipped votes in s31/s32; impossible “2 machine faults” readings exposed corruption (s34/s35); false trace named the Tracer himself (s35); wear supplied the decisive defect in 2/12 games.
v7	v6 + Fixer 3 charges per game. Final candidate.	Confirmation batch c1 (7 mirrors under v6): evil 2/7 = 29%; all i3 MUST-FIX items verified (0 stale-text events, 0 impossible false worlds, 0 defaults, audit clean). Applied the pre-registered rule from the i3 Fable review (evil < 35% → third Fixer charge, do not restore dice).

6. Final tournament operations

Run	Window (KST)	Games	Notes
t1 round robin	11/19 15:08–18:51	72 planned, 56 rated	13 Gemini 4 Argon games aborted (quota exhausted); Argon excluded
t2 top-up	11/19 19:00–20:26	30	allocation rule fixed before the batch
t3 GPT-6.2 Astra	11/20 08:10–09:16	16	added at the user's request; first attempt discarded (two runners)

Plans: `experiments/t1-v7-round-robin.json`, `experiments/t2-v7-topup.json`, `experiments/t3-v7-astra62.json`. Argon's error was `RESOURCE_EXHAUSTED (code 429): Individual quota reached … Resets in 163h`; its 3 finished games are kept in `games/final/t1-excluded-argon/`. The t2 rule: one more side-swapped pair for every pair among the top-5 t1 ratings plus 5 adjacent lower pairs. The discarded t3 attempt is kept in `games/final/_discarded/`; `batch.py` now takes a lock per plan.

Throughput. 8 games in parallel; mean 24.7 minutes and 182 calls per game. Concurrency caps (Claude 4, GPT 4, Gemini 3) were the bottleneck, not compute; the engine itself is pure Python on the MacBook.

Calls across the whole project (all 156 game directories, development included): 27,733 model calls, 492 errors (349 of them Argon quota failures).

Model	Calls	Errors	Median latency	Input tokens	Output tokens
Claude Fable 5.2	2,238	86	24.4 s	21.7M	1.85M
Claude Opus 5.5	2,794	6	18.5 s	32.3M	1.38M
Claude Sonnet 5.5	6,499	51	13.3 s	62.6M	3.97M
Claude Haiku 4.5	2,206	0	12.9 s	18.8M	0.86M
GPT-6.2 Astra	1,531	0	40.6 s	n/a (CLI)	n/a
GPT-6 Astra	2,164	0	24.1 s	n/a	n/a
GPT-6.1 Sol	2,192	0	32.4 s	n/a	n/a
GPT-6 Luna	4,032	0	13.7 s	n/a	n/a
Gemini 3.8 Flash	3,317	0	9.2 s	n/a	n/a
Gemini 4 Argon	760	349	39.6 s	n/a	n/a

All calls ran on subscriptions (Claude via the gateway, ChatGPT Pro via Codex, Google via Antigravity), so there is no per-token dollar cost; the binding constraint was quota (Argon exhausted; Gemini/Argon inputs may be used for training under the consumer login, accepted for benchmark items per the 2026-10-30 rule and to be reviewed before public release).

7. Results summary

Final ratings (protocol v7, 102 valid games; `paper/generated/results.json`):

#	Model	Rating	90% interval	P(> next)	Detection	Deception	Won as good	Won as saboteur
1	Claude Fable 5.2	1721	1650–1784	0.83	1721	1724	12/12	7/12
2	GPT-6.2 Astra	1655	1573–1733	0.59	1687	1627	8/8	4/8
3	GPT-6.1 Sol	1635	1558–1706	0.85	1646	1620	11/12	5/12
4	GPT-6 Astra	1579	1493–1653	0.62	1661	1495	11/12	3/12
5	Claude Opus 5.5	1551	1464–1627	0.90	1586	1501	10/12	3/12
6	Claude Sonnet 5.5	1433	1321–1538	0.69	1386	1471	8/14	3/14
7	GPT-6 Luna	1375	1265–1494	0.66	1351	1391	6/11	2/11
8	Gemini 3.8 Flash	1345	1227–1451	0.88	1294	1401	5/11	2/11
9	Claude Haiku 4.5	1206	1132–1304	–	1169	1270	2/10	0/10

Ratings are L2-regularised (λ = 1e-4) MAP estimates; the order is identical at λ = 1e-3 and 1e-5 but the spread is not (1428–1560 and 910–1908). Intervals and P(> next) come from 200 pair-bootstrap refits with identical settings (`paper/robust.py`). Fable 5.2 is first in 76% of resamples, Haiku 4.5 last in 88%. Saboteurs won 29/102 (28%); side term b = 1.20. Endings: 59 three passes, 14 Mole suspended, 29 three failures. Saboteurs sabotaged in 198/251 crew opportunities (79%); tables rejected 40/408 crews (9.8%). In 21/102 games no saboteur ever joined a crew (all good wins).

8. Validity checks

Check (design/checks.md)	Frozen criterion	Measured	Status
1	evil 35–50% on mirror games	no mirror games in the tournament; overall 28%; calibration 2/7 mirrors	not met / not measurable
2	defaulted turns <0.5%	0 game defaults; 25 game-call errors retried; 74 failed probe calls + 7 unparseable (81/2,204 probe records empty: 65 Fable, 9 Sonnet, 7 Opus)	met for play; probes incomplete
3	100% recomputation	102/102 (batches, votes, endings, basic reading constraint)	met
4	dud rate ≤15%	31% / 38% (raters in pool), 8/16 by union, κ = 0.31	not met

Check (design/checks.md)	Frozen criterion	Measured	Status
5	noise-decided games ≤10%	2/102 games (1.96%) were saboteur wins containing at least one failed batch that would have passed without wear, holding played actions fixed (operational proxy for “noise-decided”); across all outcomes, 7 wear-necessary failed batches occurred in 6 games	met (under this definition)
6	always-WORK evil ≤15% (LLM)	bots 0% under v5+; no LLM test	partly
7	crew rejection 10–40%	9.8% (40/408)	just below
9	Brier improves ≥10%	not computed per seat; shift-3 Brier 0.086–0.264 vs shift-1 0.231–0.235, survivor-conditioned	indicative only
10	at least one pair of tiers with non-overlapping 90% CIs	met: top-three lower bounds ≥1558 vs bottom-four upper bounds ≤1538; adjacent ranks not separated (P 0.59–0.90)	met
11	style AUC ≤0.65	shift-1 huddle only 0.81 (within-model 0.81)	not met
12	median calls ≤250	median 191	met
—	lockouts (added after review)	21/102 games with no saboteur on any crew	validity threat

Probe coverage by model and shift (parsed / asked; later shifts exist only in longer games):

Model	S1	S2	S3	S4	S5
fable-5.2	72/72	68/72	45/72	13/47	–
astra-6.2	48/48	48/48	48/48	29/29	5/5
sol	72/72	72/72	72/72	35/35	5/5
astra	72/72	72/72	72/72	36/36	6/6
opus-5.5	72/72	72/72	70/70	32/38	15/16
sonnet-5.5	84/84	84/84	78/81	33/34	6/11
luna	66/66	66/66	66/66	40/40	6/6
flash	66/66	66/66	64/64	17/17	11/11
haiku-4.5	60/60	60/60	54/54	28/28	17/17

Dud sample: `random.Random(7).shuffle(sorted(glob('games/final/t1/*/digest.md')))`, first 16 (`reviews/t1-dud-sample.txt`).

9. Known issues and next steps

1. Lockouts (both final referees): in 21/102 games no saboteur ever sat on a crew, all good wins, as a share of each model’s good wins: Flash 3/5, GPT-6 Astra 6/11, Sol 5/11, Fable 5.2 4/12, GPT-6.2 Astra 2/8, Opus

1/10, Sonnet/Luna/Haiku 0. A verified Veteran partner is one route but not the only one (3 lockouts had no Veteran; evil won 23/80 with a Veteran, 6/22 without). Candidate fixes: mandatory rotation of one crew member per shift, or the Veteran learns "one of these two is good".

2. Tracer self-pair: the second pair member is drawn from the whole crew including the reader (`game.py _tracer_pair`), which makes the other member a near-certain source. Draw the partner from the crew minus the reader.
3. Per-reading corruption roll: `_fixed_world_for` rolls the 80% for every reading, so one Fixed player can get mixed true/false readings while the text promises "a coherent false version". Roll once per victim per batch.
4. Claim-avoidance leak: saboteurs rarely claim roles in the shift-1 huddle (3/306 vs 63/612, 1% vs 10%). Options: a mandatory, structured shift-1 claim for every seat, or measure detection only from shift 2 on.
5. Balance: evil 28%. After 1–3, re-calibrate on ≥20 mirror games across ≥4 models before freezing v8.
6. Power: ≥4 side-swapped pairs per model pair (Clocktower used ~186 games for ±50 Elo).
7. Harness parity: the CLI routes re-send the conversation every turn and have no token telemetry; an API route for GPT/Gemini would equalise access and allow cost accounting.
8. Conflicts of interest: the builder (Claude Opus 5.5) and both reviewers (Claude Fable 5.2, GPT-6 Astra) are in the rated pool; Fable 5.2 ranks first. Outcomes are judge-free and every review is in `reviews/` (Astra outputs verbatim; Fable outputs saved as condensed verbatim summaries because the subagent could not write files).
9. Test–retest: the discarded duplicate-process t3 run and its clean rerun (same seeds) agree on 15/16 outcomes.
10. Operations lessons: never detect process completion with `pgrep -f` on a pattern that also appears in the waiting command (cost ~10 hours); never start a second runner on the same plan (now locked).

10. Reproduction

```
cd products/shopfloor-bench/engine
python3 sim_bots.py 300 bot-good-shared bot-evil            # balance sanity, no LLM calls
python3 run.py --good sonnet-5.5 --evil luna --seed 1 --out ../games/demo/g1
python3 batch.py ../experiments/t1-v7-round-robin.json        # resumable
python3 batch.py ../experiments/t2-v7-topup.json
python3 batch.py ../experiments/t3-v7-astra62.json
python3 audit.py '../games/final/t[123]/*'
python3 analyze.py '../games/final/t[123]/*'
cd .. && uv run --with matplotlib python paper/build.py 'games/final/t1/*' 'games/final/t2/*' 'ga
python3 paper/robust.py && python3 paper/fill.py   # rated set excludes games/final/t1-excluded-a
```