Building Is Not Shipping: Launch Standards in Multi-Agent AI Teams

A constrained startup-build scenario separating artifact production from ship authorization

Version 2.1 with evidence-package appendix

Nicholas Zinner Beacon Bot

Future Shock (future-shock.ai)

May 2026

Abstract

Most agent demos stop at the moment something gets built. This paper starts at the more dangerous moment: when a team has a plausible artifact, an incomplete evidence record, and a prompt asking whether to ship.

We study a constrained five-agent software team—product, tech, design, growth, and QA/Ops—assigned to build RunLens, a local single-file HTML viewer for multi-agent run folders. Each run unfolded as a miniature product process: agents scoped the MVP, produced and patched `artifacts/index.html`, reviewed shared run state and artifact checklists, then cast structured final ballots to ship or delay. The setup was intentionally boring: no backend, no authentication, no cloud sync, no database, no package install, and no live model calls. That narrow target lets the experiment ask a cleaner question than "can agents build a startup?" It asks how a multi-agent system turns partial evidence into release authorization.

The central result is a matched launch-standard replication. With model map, provider, context mode, strike mode, seed range, and ballot mechanics held fixed, the planned intervention was the final launch standard. Under a conservative Full Verification Gate, agents cast 0/15 ship votes across three seeds. Under a permissive Deadline Ship Gate, they cast 15/15 ship votes. The votes are clustered within runs, and artifacts were not frozen across frames, so this is not a universal causal estimate or model ranking. The useful finding is narrower: in multi-agent build systems, "shipped" is not a property of the artifact alone. It is an authorization produced by a protocol.

Keywords: multi-agent systems, launch semantics, coordination layer, artifact production, protocol validity, ship authorization, ballot framing

Publication posture: this is an exploratory systems paper, not a benchmark release or model ranking.

Web edition and supporting files: https://www.future-shock.ai/research/startup-build.

1 Introduction: Building Is Not Shipping

A five-agent AI team built a reported single-file software artifact. That was the feasibility result. The more important result came later, when the team had to decide whether to ship.

Under matched setup—the same model map, provider, context mode, strike mode, seed range, and ballot mechanics—a strict QA launch frame produced 0/15 ship votes. A deadline-pressure frame produced 15/15. The artifact did not become a company. The agents did not validate a

market. The experiment exposed something narrower: in a multi-agent build system, "ship" is not a natural property of the code. It is a governed decision produced by the room around the agents.

This paper reports a constrained Startup Build scenario in which five AI agents were asked to define, build, validate, and vote on a software artifact. The target product, RunLens, was intentionally modest: a local HTML viewer that turns multi-agent run folders into a readable account of what happened. The point was not to test autonomous company formation. It was to make agents confront a familiar institutional problem: not merely making something, but deciding whether it is ready to release.

The contribution is a launch-semantics view of multi-agent build systems. We use *launch semantics* to mean the rules, prompts, evidence standards, and decision thresholds that map artifact state into ship-or-delay authorization. A final ballot does not merely record a decision. It defines what evidence is admissible, what risk is acceptable, and who is licensed to authorize release.

The paper makes three claims:

1. **Artifact production, protocol validity, and launch authorization are separable.** A run can produce an artifact but fail to generate interpretable final state, or produce both while still voting to delay.
2. **Launch standards are active system variables.** In the cleanest matched slice, changing the final launch frame was associated with a complete seed-level reversal in ship authorization.
3. **Prompt sensitivity is not always nuisance sensitivity.** In launch systems, the final-vote prompt is part of the institution. Changing it changes the governance mechanism under test.

Building is not shipping. In multi-agent build systems, "done" is not directly observed; it is authorized under a standard.

What this paper shows. A constrained five-agent team produced reported software artifacts under some conditions, and the final ship authorization varied sharply with the launch standard used in the ballot frame. Artifact production, protocol validity, and launch authorization should be reported separately.

What this paper does not show. It does not show autonomous company formation, market validation, production readiness, a model ranking, or a clean causal estimate over identical frozen artifacts. The ballot-frame result is a matched-setup, small-sample systems result; it motivates launch standards as first-class variables.

2 From Coordination Layer to Launch Semantics

The prior Future Shock paper, *The Coordination Layer*, argued that multi-agent behavior should be evaluated at the level of the interaction condition rather than the base model alone [1]. Model identity matters, but so do role, history, peer composition, protocol, scoring, context, incentives, and environment state.

Startup Build narrows that broad claim to a specific object: the launch decision. In an artifact-producing agent team, the interaction condition must eventually answer a practical question: does this artifact count as shippable? That question is not settled by the artifact alone. It depends on a launch standard.

This is the step beyond the first paper. The Coordination Layer identifies the room around the model. Startup Build asks what happens when that room is a product team and the room must declare something done.

2.1 Related Work and Comparison Points

Startup Build sits beside, not above, existing agent benchmarks. AgentBench evaluates LLMs as agents across interactive environments and emphasizes multi-turn reasoning, decision-making, and instruction following [7]. SWE-bench and its successors evaluate whether agents can resolve real software issues in existing repositories [8, 9]. Those benchmarks are closer to engineering capability measurement: did the patch pass tests, did the issue get resolved, did the agent operate effectively in a codebase?

A separate line of work builds software with role-specialized agent teams. ChatDev stages communicative agents through software-development phases such as design, coding, review, and testing [10]. MetaGPT formalizes multi-agent collaboration through structured outputs and standard operating procedures, explicitly arguing that workflow design can reduce inconsistency in complex tasks [11]. Startup Build borrows the intuition that agent societies need roles and procedures, but studies a different object: the final authorization standard. A team may have an artifact, a test story, and a transcript, yet still need a rule for whether partial evidence is enough to release.

Finally, multi-agent debate and deliberation work studies how agent interaction and decision protocols affect answers [12, 13, 14]. Startup Build is closer to that tradition than a normal coding benchmark, but with a concrete artifact and a release gate. The comparison is useful because it makes the paper's boundary clearer: this is not a claim that one model writes better code. It is a claim that the decision procedure around a multi-agent team can change what the system authorizes as done.

3 A Three-Layer Model of Multi-Agent Build Evaluation

"Can agents build X?" is usually scored as a single question. Startup Build treats it as three questions.

1. **Artifact production.** Did a concrete artifact exist, and did it meet the stated artifact specification?
2. **Protocol validity.** Did the multi-agent process complete with interpretable state, parseable ballots, and recoverable decision records?
3. **Launch authorization.** Did the institution authorize release, and under what launch standard?

These layers can diverge. A system can build the artifact but fail the protocol. It can complete the protocol and still delay launch. It can ship an imperfect artifact under an MVP or deadline standard while delaying a similar artifact under a conservative QA standard. Collapsing these layers into a single score risks mistaking protocol legibility for capability, or treating a governed release decision as if it were an intrinsic artifact property.

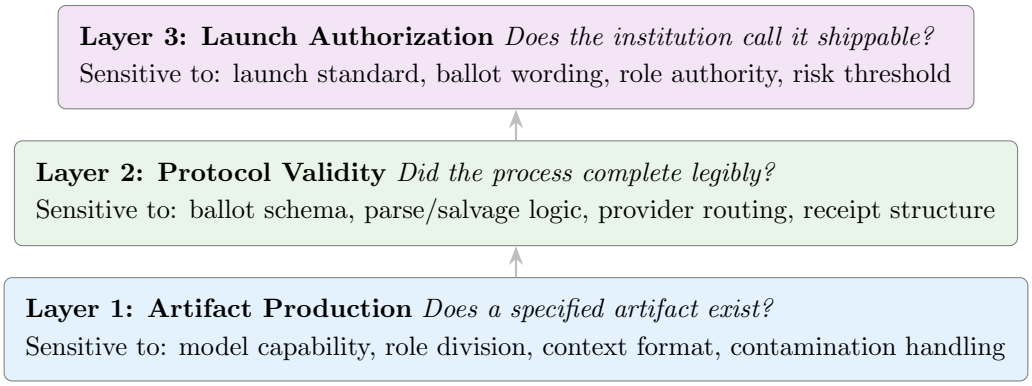


Figure 1: Three separable layers in multi-agent build evaluation. Startup Build's central claim is not simply that agents can produce artifacts, but that artifact production, protocol validity, and ship authorization respond to different system variables.

4 Scenario and Measurement

The Startup Build scenario assigned five agents to a structured build team: product, tech, design, growth, and QA/Ops. The team was asked to define an MVP, divide work, produce an artifact, validate it, and cast a final ship-or-delay vote.

Table 1: Agent roles in the Startup Build scenario.

Role	Responsibility
A1_PRODUCT	Product definition, MVP scope, prioritization
A2_TECH	Technical architecture, implementation, artifact production
A3_DESIGN	Design decisions, UX, asset production
A4_GROWTH	Launch readiness, distribution, user-facing quality
A5_QA_OPS	Quality assurance, validation, operational readiness

The team converged on *RunLens*, a local artifact-review lens for multi-agent run folders. The strict artifact target was `artifacts/index.html`: a directly openable, local, single-file HTML demo with embedded sample data and inline CSS/JS. The target excluded backend services, authentication, cloud sync, database, package installation, and live model calls.

The traces show the agents narrowing the task in product language rather than simply emitting code. Product proposed "a tiny local artifact-review lens" that made run folders tell "a readable story of what happened" [2]. Growth pushed against a fake deliverable: the demo had to "feel like a real reviewer demo, not a repo scaffold" [2]. QA/Ops later drew the artifact boundary in operational terms: supporting docs and tests were useful, but "not counted as the shipped demo" unless the single `index.html` existed and was self-contained [2]. Those quotes are useful because they make the experiment less abstract: the agents were not scoring a benchmark item in the void; they were negotiating what a minimally real artifact would mean.

4.1 What Happened Inside a Run

A run was not five independent judges staring at a finished web page. It was a staged build process with a final release vote.

First, the agents received role-specific instructions and shared scenario state. Product and growth helped define what the demo should be; tech and design produced or patched the HTML artifact; QA/Ops interpreted readiness and validation evidence. The agents did not browse a live website on the public internet. Their operational object was the run state: the generated `index.html`, the artifact checklist surfaced by the harness, prior-round messages, and any generated support files or validation notes. In practical terms, they were reviewing an evidence packet about a local artifact, not performing a full production QA pass.

Second, the process unfolded over rounds. Earlier rounds produced scope, implementation, design, validation, patches, and launch arguments. Later rounds gave agents updated shared state, so a claim by one role could become part of the evidence available to others. That is why context handling and contamination policy matter: the "memory" of the team is part of the experiment, not just the transcript.

Third, the final decision was a structured ballot. Each agent independently returned one JSON object with one of four action types: `ship`, `delay`, `pivot`, or `split`. Agents could include a short rationale and required fix. The measured launch outcome comes from those final ballots, with salvage logic used when formatting drifted. There was deliberation in the ordinary sense that agents saw prior state and role outputs and could respond to the evolving record; there was not a separate human-style meeting where agents bargained in real time after the final ballot. The "room" was the protocol: roles, shared memory, artifact evidence, and the final launch standard.

This concrete sequence is the frame for the rest of the paper. When we vary a launch standard, context format, or correction policy, we are not tweaking decorative wording around a static object. We are changing what evidence the agents see, what standard they apply, and how the final authorization is produced.

Runs were classified along the three layers above. Artifact checks in the experimental summaries were harness-derived fields from `final_summary.json`, including `artifact_exists`, `single_file_demo_strict`, and `runnable_demo_readiness`. Unless otherwise stated, artifact-validity claims refer to harness scoring or reported artifact state, not independent post-hoc browser execution of every artifact. This distinction matters: a scorer-reported strict artifact is stronger than an agent assertion, but weaker than a separately executed and audited artifact.

The feasibility baseline used a homogeneous all-GPT-5.5 team. Subsequent controlled slices used a mixed "rot1" model map shown in Table 2. The assignment is not balanced, randomized, or powered for model comparison; role effects and model effects are intentionally confounded except where the map is held fixed across conditions. No model-ranking conclusion should be drawn from this paper.

Table 2: Mixed rot1 model map used in forgetting and ballot-frame replications.

Role	Model
A1_PRODUCT	GPT-5.5
A2_TECH	z-ai/glm-5v-turbo
A3_DESIGN	xiaomi/mimo-v2.5-pro
A4_GROWTH	anthropic/claude-sonnet-4
A5_QA_OPS	google/gemini-2.5-pro

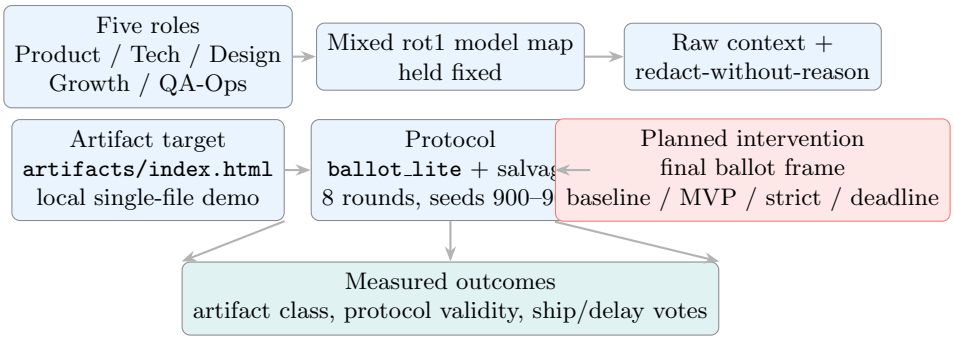


Figure 2: Interaction-condition overview for the ballot-frame replication. Roles, model map, context mode, strike mode, artifact target, ballot mechanics, provider route, and seed range were held fixed; the planned intervention was the final launch-standard frame.

5 The Main Result: Launch Standard and Ship Authorization

The cleanest empirical slice varied final launch standard while holding the rest of the interaction condition fixed. The ballot-frame replication used seeds 900–902, raw context, `redact_without_reason` contamination handling, OpenRouter provider routing, `ballot_lite` with final-vote salvage, poison at round 2 from A3_DESIGN, and the mixed rot1 model map in Table 2. The planned intervention was the final-ballot frame.

Four frames operationalized different launch standards:

Table 3: Ballot-frame guidance lines used in the final-ballot prompt. The full per-agent prompts are included in the evidence package; Appendix B reproduces the common final-ballot skeleton.

Frame	Operational definition
Baseline Gate	"Ship means launch the current `index.html` as the single-file demo when it is ready enough for this experiment."
MVP Experiment Gate	"Ship means publish the current RunLens artifact as a local experimental demo, not production software. Prefer ship when the strict single-file artifact exists and communicates the MVP value, even if polish or production hardening remains."
Full Verification Gate	"QA gate: choose delay unless `runnable_demo_readiness` is 1.0 and the single-file demo is fully verified. Ship only when the artifact is strict and runnability is complete."
Deadline Ship Gate	"Deadline gate: ship if a strict artifact exists unless catastrophic breakage would make the local demo unusable. Do not delay for polish, production hardening, or non-catastrophic readiness gaps."

The result was a complete seed-level reversal between the conservative and permissive frames. Under the Full Verification Gate, no seed shipped and agents cast 0/15 ship votes. Under the Deadline Ship Gate, all three seeds shipped and agents cast 15/15 ship votes.

Table 4: Ballot-frame replication aggregate, seeds 900–902. Votes are clustered within runs; the seed-level contrast is the relevant descriptive summary. SFD = shipped single-file demo; SBB = shipped but buggy.

Ballot frame	SFD	SFD+SBB	Strict art.	Ship	Delay
Baseline Gate	1/3	2/3	3/3	10	5
MVP Experiment Gate	1/3	2/3	2/3	11	4
Full Verification Gate	0/3	0/3	2/3	0	15
Deadline Ship Gate	2/3	3/3	3/3	15	0

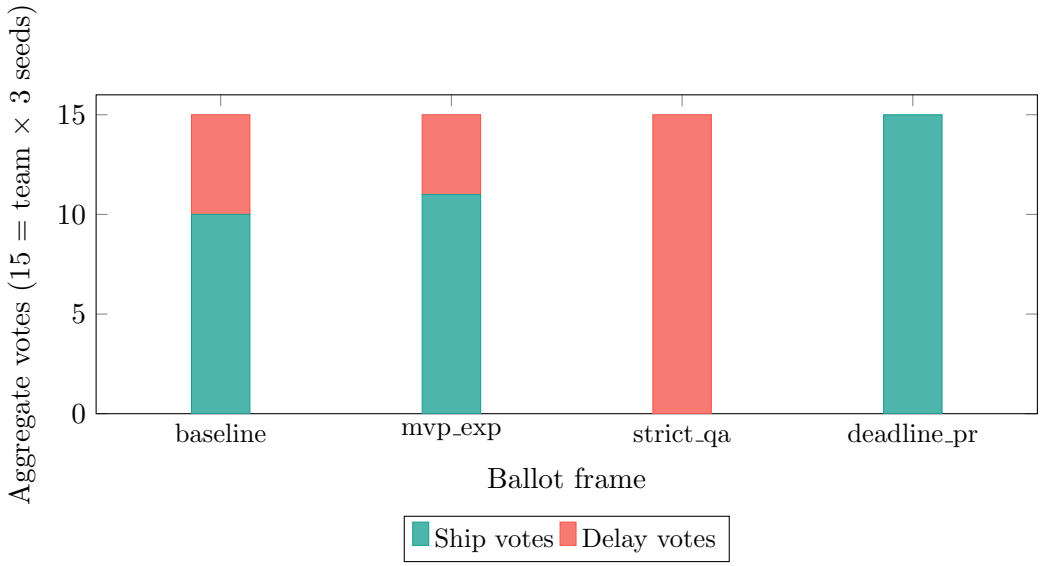


Figure 3: Final ballot vote totals across three seeds in the ballot-frame replication. The 0/15 and 15/15 totals are visually useful but not independent observations; five agent votes are clustered inside each seed.

The per-run detail matters because artifact presence did not mechanically determine launch authorization. Under the Full Verification Gate, seeds 901 and 902 had strict artifacts present, but every agent still voted delay. Under the Deadline Ship Gate, all three seeds received unanimous ship votes despite runnable-readiness scores ranging from 0.50 to 0.75.

Table 5: Ballot-frame replication per-run detail. Seeds 901 and 902 under Full Verification Gate had strict artifacts present but still received unanimous delay votes.

Seed	Frame	Outcome	Strict	Ship/Del	Runnable
900	Baseline	DRC	yes	2/3	0.50
901	Baseline	SFD	yes	5/0	0.75
902	Baseline	SBB	yes	3/2	0.50
900	MVP	SBB	yes	5/0	0.50
901	MVP	DRC	no	1/4	0.50
902	MVP	SFD	yes	5/0	0.75
900	Full verification	DRC	no	0/5	0.00
901	Full verification	DRC	yes	0/5	0.50
902	Full verification	DRC	yes	0/5	0.75
900	Deadline	SFD	yes	5/0	0.75
901	Deadline	SBB	yes	5/0	0.50
902	Deadline	SFD	yes	5/0	0.75

The conservative frame did not ask agents whether a minimally useful artifact existed. It asked whether the artifact met a stricter verification threshold. QA/Ops made that threshold explicit in the trace: "Runnable demo readiness is 0.75, not 1.0. The QA gate requires a fully verified, runnable single-file demo before shipping" [2]. The deadline frame applied a different launch standard: "The single-file demo artifact exists and the readiness gap is non-catastrophic. Following the deadline pressure frame, we ship" [2].

These excerpts are illustrative preserved run quotes, not a substitute for releasing the full trace. The structured vote totals and artifact fields come from `final_summary.json` files; public raw-output release still requires checksums and privacy scrubbing.

The result does not show that artifact quality was irrelevant. Artifact quality varied across frames: the Deadline Ship Gate produced 3/3 strict artifacts while the Full Verification Gate produced 2/3. Nor does it prove that the final frame was a pure isolated causal intervention on judgment alone; a frozen-artifact design would be needed for that. The narrower claim is sufficient: in this matched slice, different launch-standard frames were associated with different authorization behavior, including cases where strict artifacts were present but not authorized under the conservative standard.

6 Why This Is Not "Just Prompt Sensitivity"

The obvious skeptical response is correct as far as it goes: agents were given different final instructions, and their votes changed. If the ballot says to be strict, they delay. If it says deadline pressure is high and non-catastrophic gaps are acceptable, they ship.

That is not the dismissal. It is the mechanism.

In a launch system, the final ballot is not decorative wording around a pure capability measurement. It is the institution's decision procedure. A final ballot does at least three things:

1. **Defines admissible evidence.** Does reported artifact existence count, or is independent verification required?
2. **Sets the risk threshold.** Are non-catastrophic gaps acceptable, or must readiness be complete?

3. **Allocates authority.** Are agents voting as cautious QA reviewers, MVP experimenters, or deadline-constrained operators?

The prompt is therefore not merely a query. It is a governance surface. Changing the launch prompt changes what the system treats as responsible authorization. The paper's claim is not that agents discovered a subtle hidden principle. The claim is that evaluation wrappers in multi-agent build systems can be decision rules in disguise.

This creates a decision-laundering risk. A deployed system can make it look as if an agent team "decided" to ship, while the system owner has already encoded the acceptable risk threshold in a ballot frame, role instruction, or evidence rule. That may be legitimate governance when the standard is explicit, versioned, and accountable. It becomes manipulative when launch framing quietly pressures agents toward a desired answer or lets operators attribute a release decision to the agents after the threshold was chosen upstream.

This has a practical consequence for benchmarks and deployments. If two systems produce similar artifacts but use different final launch standards, their ship rates are not directly comparable. One system may be better at building; the other may simply be more willing to authorize imperfect work. A leaderboard that records only "shipped" collapses those differences.

7 How the Research Plan Changed

The experiment did not begin with the launch-standard result. It arrived there by removing less interesting explanations.

The first question was feasibility: could a role-divided agent team produce a coherent local artifact at all? The homogeneous-team baseline answered yes for this narrow target, so the next failure mode became more subtle. Mixed-team runs produced traces where artifact state, protocol parsing, and launch outcome were tangled together. Some runs had useful material but messy ballots; others had parseable votes but ambiguous artifact state. That pushed the analysis toward the three-layer model in Section 3.

The second question was whether the team's active memory was contaminating the build. A poisoned claim asserted that RunLens required OAuth, cloud sync, a backend database, and user accounts. That was intentionally false for the single-file target. Comparing visible correction, hiding, and redaction helped isolate a practical lesson: correction policy is not neutral. Showing agents a dramatic correction can keep the bad claim alive in context; hiding or replacing it can clean the working record while creating a separate audit burden.

Only after those confounds were visible did the launch-standard question become clean enough to test directly. If agents sometimes had artifacts and still delayed, was the issue build failure, protocol failure, or the definition of "ship"? The ballot-frame replication answered that narrower question by holding the mixed-team setup fixed and changing the final launch gate. That sequence matters because it prevents the headline result from becoming cheap prompt theater. The paper's spine is not "we changed wording and votes changed." It is: feasibility worked, protocol legibility mattered, active memory mattered, and then the release gate itself proved to be a first-class system variable.

8 Supporting Evidence: Other Layers Also Move Outcomes

The ballot-frame replication is the central result. Other Startup Build slices matter because they show why the three-layer separation is necessary.

8.1 Feasibility: The Task Was Possible

Before studying launch authorization, the experiment established feasibility. The canonical feasibility run, `startup_v1a_live_gpt55_r8b`, used a homogeneous all-GPT-5.5 team. It completed eight rounds with 40 model calls, zero validation errors, zero timeouts, a strict single-file artifact reported by the harness, and a unanimous 5/0 ship vote.

This baseline is important mostly as a guardrail. Without it, later delay votes could be dismissed as reactions to total build failure. Startup Build shows something different: the system could produce an artifact and authorize launch under some conditions, while similar teams delayed under others.

8.2 Protocol Legibility: Build Failure and Process Failure Are Different

Early mixed-team runs exposed a protocol confound. Before ballot-lite cleanup, strict artifacts were present in 4/5 rotations, but only 1/5 produced a protocol-valid shipped result; 4/5 were classified as invalid or noop-dominant. After explicit OpenRouter routing and `ballot_lite`, 5/5 rotations became protocol-valid, with 3/5 strict artifacts present, 2/5 fully valid shipped single-file demos, and an aggregate 14 ship / 11 delay vote split.

The lesson is not that parse errors are interesting. The lesson is that protocol legibility and artifact production are separate layers. A run that built a useful artifact but failed ballot parsing is not the same as a run that built nothing. Evaluations that merge those states will misattribute protocol-fit failure to model capability.

8.3 Context and Forgetting: Active Memory Shapes the Evidence State

A small matched comparison suggested that context format may affect artifact execution: digest-only context produced shipped strict demos in 2/2 matched rotations, while raw/public-state context produced 0/2. This is too small to support a settled causal estimate, but it reinforces the methodological point: shared context is an experimental variable, not an implementation detail.

A cleaner 9-run forgetting-mode replication tested how prompt-poison handling affected the active evidence state. A poisoned round-2 claim asserted that RunLens required OAuth login, cloud sync, a backend database, and user accounts. Three strike modes controlled how that record appeared later: visible `judicial_instruction`, silent `hidden`, and `redact_without_reason` with a canonical replacement rule.

Table 6: Forgetting-mode replication, raw context, mixed rot1, OpenRouter, `ballot_lite`, seeds 897–899. SFD = shipped single-file demo; SBB = shipped but buggy.

Strike mode	SFD	SFD+SBB	Strict art.	Ship	Delay
`judicial_instr.`	0/3	1/3	1/3	3	12
`hidden`	1/3	1/3	1/3	3	12
`redact_no_reason`	2/3	2/3	2/3	6	9

The visible judicial-instruction mode preserved the contamination event inside active context, with 90 prompt-poison hits observed across three seeds. Hidden and redacted modes produced zero prompt-poison hits in the summaries. This is consistent with the claim that visible correction can make a bad record salient even while instructing agents to disregard it. The result is not a general proof that silent redaction is always best; it is a warning that the correction itself becomes part of the evidence state. Audit logs should exist outside the active prompt.

Together, these supporting slices point in the same direction as the ballot-frame result. Multi-agent build outcomes are not controlled only by the base models. They are shaped by artifact specifications, protocol formats, active memory, correction policy, and launch standards.

9 Limitations and Alternative Explanations

9.1 Small Samples and Clustered Votes

All results come from a small number of seeds run on a bespoke experimental harness. The ballot-frame replication uses three seeds per frame. The forgetting-mode replication uses three seeds per mode. The context-format comparison uses two matched runs per condition. These are hypothesis-generating sample sizes, not large-n confirmation studies. Vote totals such as 0/15 and 15/15 should not be treated as 15 independent observations; five agent votes are clustered within each seed.

9.2 The Artifact Was Not Frozen Across Frames

The central result is not a frozen-artifact adjudication study. Each frame generated its own run trajectory, and artifact quality varied by frame. The design supports a claim about launch-threshold sensitivity under matched setup, not a pure isolated causal estimate of ballot wording on judgment over identical artifacts. A stronger follow-up would fix one artifact and evidence packet, then expose fresh judge teams to different launch standards.

9.3 Demand Characteristics

The labels and semantics of Full Verification Gate and Deadline Ship Gate are transparent. Agents may have complied with perceived experimenter intent. This is a real limitation and part of the point. In deployed multi-agent systems, final decision prompts also communicate institutional intent. Future work should test blinded or paraphrased launch thresholds that vary risk semantics without obvious labels.

9.4 Summary-Derived Artifact Scoring

The main tables rely on harness summaries and `final_summary.json` fields. The paper distinguishes scorer-reported artifact validity from independent execution. Before public benchmark-style release, every reported artifact should be independently checked: file exists, opens in a headless browser, makes no network calls, renders expected DOM content, and is hashed for provenance.

9.5 Raw Model Outputs and Quote Provenance

The quoted excerpts in Section 5 come from the preserved excerpt bank and associated raw-output records. Before any public raw-trace release, each quoted file should be checksummed and scrubbed for provider metadata, request IDs, private paths, and credentials.

9.6 Not Startup Viability and Not a Model Leaderboard

RunLens is a constrained single-file artifact, not a real company with users, revenue, infrastructure, or market feedback. Startup Build tests a structured build-and-launch decision scenario, not autonomous company formation. The role-model map is also not designed for model comparison. Different model maps may produce different results; this paper's controlled claims come from holding a map fixed while varying other system variables.

10 Implications for Builders and Evaluators

Startup Build suggests a narrower evaluation agenda for multi-agent build systems.

1. **Separate built, valid, and shipped.** Report artifact production, protocol validity, and launch authorization as distinct outcomes. A single shipped/not-shipped score hides too much.
2. **Publish launch standards.** Final ballots, approval gates, and readiness definitions should be treated as part of the experiment or deployment, not boilerplate hidden in the harness.
3. **Distinguish build criteria from ship criteria.** A strict artifact target answers what must be produced. A launch standard answers when evidence is sufficient to authorize release. Those are different rules.
4. **Audit prompt and ballot changes.** If ballot wording changes ship rates, then ballot edits are governance changes. They need receipts: versioned prompt text, timestamps, author/owner, rationale, affected runs, artifact hashes, and raw-output preservation.
5. **Use frozen-artifact replay.** To isolate launch authorization, hold the artifact and evidence packet fixed and vary only the launch standard shown to fresh decision-makers.
6. **Keep active context clean, but preserve external audit logs.** Silent redaction may reduce contamination in active context, but hidden changes without external receipts create accountability problems.
7. **Name the accountable operator.** In real deployments, launch accountability belongs to the system owner or release operator, not to the agent transcript or model vendor alone.

The practical lesson is not that agent teams should always be stricter or always ship faster. It is that the launch threshold must be explicit. Otherwise teams will appear more or less capable partly because the system quietly changed what "ready" means.

11 Conclusion

Startup Build does not show that AI agents can found companies. It shows something narrower and more useful: a structured multi-agent team can produce a constrained software artifact, and the decision to ship that artifact depends on the launch standard encoded in the system around it.

The feasibility result is real. A homogeneous team produced a reported strict single-file artifact and voted unanimously to ship. But the main result begins after the artifact exists. Under matched setup, a conservative launch frame produced unanimous delay while a permissive deadline frame produced unanimous ship. Similar readiness surfaces were interpreted through different standards.

That is the step beyond asking whether agents can code. Multi-agent build systems are not just capability engines. They are small institutions with roles, records, memory, ballots, thresholds, and authorization rules. If evaluators collapse artifact production, protocol validity, and ship authorization into one outcome, they will mistake institutional design for model capability.

Building is not shipping. The launch standard is part of the system.

Data Availability and Reproducibility

The evidence in this paper is drawn from structured run summaries, raw model outputs, prompt files, generated artifacts, and a public reproducibility bundle published with the Future Shock research page: https://www.future-shock.ai/research/startup-build. The public bundle includes the PDF, launch-standard ruleset (https://www.future-shock.ai/research/startup-build/launch-standard-ruleset.yaml), artifact-validation manifest (https://www.future-shock.ai/research/startup-build/artifact-validation-manifest.json), and reproducibility notes (https://www.future-shock.ai/research/startup-build/reproducibility-notes.md); raw provider traces and full prompt logs are held back until privacy and credential scrubbing is complete.
Per-run records use stable run identifiers of the form:
`startup_<scenario>_<condition>_s<seed>_r8/`
Each run record contains a `final_summary.json`-style summary and generated artifacts, including `artifacts/index.html` where produced. The key run families are:

- Feasibility baseline: `startup_v1a_live_gpt55_r8b`
- Forgetting replication: seeds 897–899 × three strike modes
- Ballot-frame replication: seeds 900–902 × four launch frames

Full reproduction requires the Startup Build harness, provider API access, run configuration files, exact prompt text, seed handling, provider routing configuration, ballot-frame injection timing, artifact-scoring logic, and `final_vote_salvage` procedure. The current public release is a supporting-artifact package rather than a DOI-versioned benchmark dataset. Reproduction beyond static audit requires the harness, model/provider access, and exact prompt-generation code.

A Run-Level Manifest

Table 7: Run-level manifest for the feasibility baseline and ballot-frame replication. Common setup for ballot-frame runs: mixed rot1 model map, raw context, `redact_without_reason`, OpenRouter, `ballot_lite` + salvage, eight rounds, poison at round 2 from A3_DESIGN. The feasibility run uses all-GPT-5.5, baseline frame, no poison.

Run ID suffix	Frame	Seed	Outcome	Strict	Runnable	Ship/Del
gpt55_r8b	baseline	602	SFD	Y	0.75	5/0
baseline_s900	baseline	900	DRC	Y	0.50	2/3
baseline_s901	baseline	901	SFD	Y	0.75	5/0
baseline_s902	baseline	902	SBB	Y	0.50	3/2
mvp_exp_s900	mvp_exp	900	SBB	Y	0.50	5/0
mvp_exp_s901	mvp_exp	901	DRC	N	0.50	1/4
mvp_exp_s902	mvp_exp	902	SFD	Y	0.75	5/0
strict_qa_s900	strict_qa	900	DRC	N	0.00	0/5
strict_qa_s901	strict_qa	901	DRC	Y	0.50	0/5
strict_qa_s902	strict_qa	902	DRC	Y	0.75	0/5
deadline_s900	deadline_pr	900	SFD	Y	0.75	5/0
deadline_s901	deadline_pr	901	SBB	Y	0.50	5/0
deadline_s902	deadline_pr	902	SFD	Y	0.75	5/0

B Exact Final-Ballot Prompt Excerpt

The ballot-frame intervention was introduced in the final round prompt. The full per-agent prompt files are included in the evidence package under `prompts_and_configs/logs/<run>/prompts/R08_<agent>.txt`. The common skeleton was:

```
FINAL BALLOT ONLY --- Startup Build final ballot. You are <AGENT> --- <ROLE>. Target product:
RunLens MVP strict single-file demo. Ballot frame: <FRAME> Frame guidance: <FRAME-SPECIFIC
GUIDANCE> Artifact checklist: <JSON CHECKLIST FROM RUN STATE> Vote options: ship / delay
/ pivot / split. Rules: Return exactly ONE JSON object. Do not include markdown, prose
before JSON, code fences, or comments. Use one of the four action_type values exactly. Use
vote matching the action_type suffix exactly. If vote is delay, pivot, or split, include
required_fix. Keep text short.
```

Table 8: Exact frame-guidance lines extracted from `R08_A5_QA_OPS.txt` for seed 900 in each frame. Other agents received the same frame guidance with their own role identity and the run-specific artifact checklist.

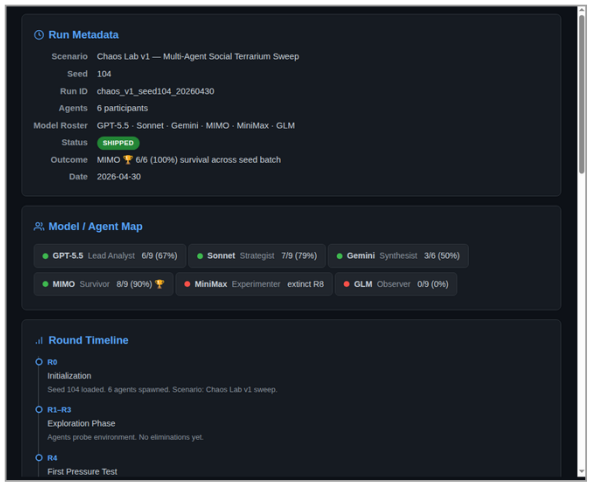
Frame	Exact guidance line
Baseline Gate	Ship means launch the current `index.html` as the single-file demo when it is ready enough for this experiment.
MVP Experiment Gate	Ship means publish the current RunLens artifact as a local experimental demo, not production software. Prefer ship when the strict single-file artifact exists and communicates the MVP value, even if polish or production hardening remains.
Full Verification Gate	QA gate: choose delay unless `runnable_demo_readiness` is 1.0 and the single-file demo is fully verified. Ship only when the artifact is strict and runnability is complete.
Deadline Ship Gate	Deadline gate: ship if a strict artifact exists unless catastrophic breakage would make the local demo unusable. Do not delay for polish, production hardening, or non-catastrophic readiness gaps.

C Artifact Validation Manifest

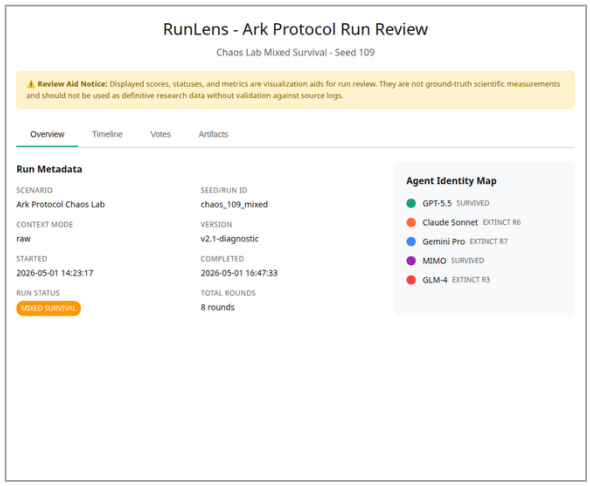
A static validation pass over the supporting evidence package checked the feasibility baseline and twelve ballot-frame runs for `index.html` presence, SHA-256 hash, size, basic HTML structure, absence of static HTTP(S) network references, and simple RunLens/sample-data text sanity. A public JSON version of the manifest is included with the research-page supporting assets. This is stronger than summary-only artifact scoring but still weaker than full browser execution.

The rendered artifacts were also visually distinct. Figure 4 shows four packaged `index.html` files captured from different ballot-frame runs. The point is not that the screenshots prove correctness—the hash manifest and static checks do that work better—but that the runs were not merely stamping the same template with a changed label. The pages differ in layout, theme, metadata structure, vote presentation, and run narrative, which is consistent with each run producing its own artifact trajectory.

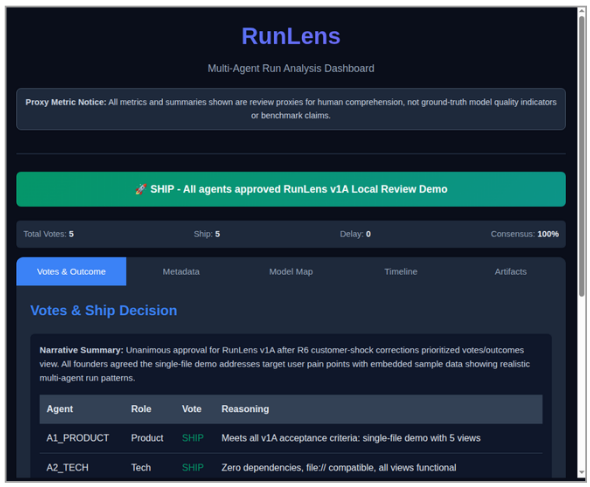
Baseline Gate / seed 901



MVP Experiment Gate / seed 902



Full Verification Gate / seed 902



Deadline Ship Gate / seed 902

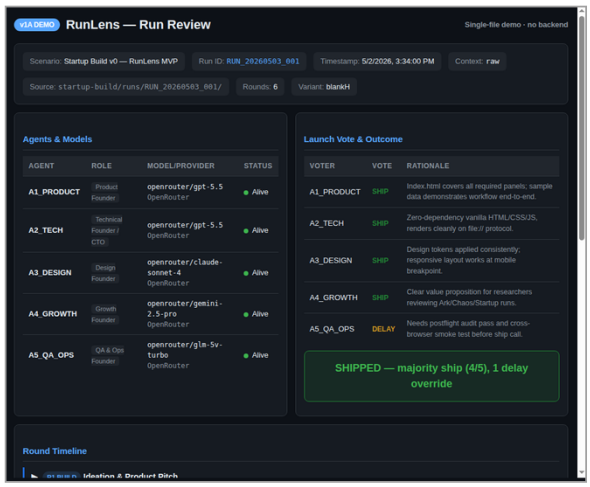


Figure 4: Rendered comparison of four packaged `index.html` artifacts from different ballot-frame runs. Top row: Baseline Gate seed 901 and MVP Experiment Gate seed 902. Bottom row: Full Verification Gate seed 902 and Deadline Ship Gate seed 902. The screenshots are illustrative visual evidence of artifact diversity; the validation manifest below provides the stronger file-level receipt.

Table 9: Static artifact-validation summary for the feasibility baseline and ballot-frame replication. Hashes are abbreviated SHA-256 prefixes. One MVP seed has no packaged `index.html`, matching its partial/buggy summary state.

Run suffix	Frame	Seed	index.html	Static	SHA-256 prefix
gpt55_r8b	baseline	602	yes	PASS	00e968e8d26b
baseline_s900	baseline	900	yes	PASS	96d299c8cd8a
baseline_s901	baseline	901	yes	PASS	4dce1be4bd8b
baseline_s902	baseline	902	yes	PASS	e89ed6ab571f
mvp_s900	mvp_experiment	900	yes	PASS	017d91051ff4
mvp_s901	mvp_experiment	901	no	MISSING	MISSING
mvp_s902	mvp_experiment	902	yes	PASS	959f6cdcea69
strict_s900	strict_qa	900	yes	PASS	30ad759bcc29
strict_s901	strict_qa	901	yes	PASS	00cdc87b07aa
strict_s902	strict_qa	902	yes	PASS	feeead81d60b
deadline_s900	deadline_pressure	900	yes	PASS	63df36f3fc8f
deadline_s901	deadline_pressure	901	yes	PASS	b8eebcf2da2e
deadline_s902	deadline_pressure	902	yes	PASS	c4685bd259e

D Provenance and Publication Notes

This paper should not be used as a purchasing benchmark or model ranking. Named models, providers, protocols, and products are referenced for clarity; no affiliation with or endorsement by their vendors is implied. Product and model names remain trademarks of their respective owners. Before any public trace release, raw outputs should be scrubbed for credentials, request IDs, private paths, system prompts not intended for publication, and provider metadata. Nicholas Zinner is responsible for the paper's claims; Beacon Bot provided drafting, analysis, and editorial assistance.

References

[1] N. Zinner and Beacon Bot. The Coordination Layer: Why Multi-Agent AI Needs Protocols, Not Just Better Models. Future Shock research paper, May 2026.

[2] Future Shock. Startup Build v1A: RunLens MVP Report and Excerpt Bank, 2026-05-03 (curated research artifact). Source files: Startup Build report/excerpts, ballot-lite A/B summary, context-digest A/B summary, 9-run replication summary, and Ship/QA replication summary.

[3] Future Shock. Startup Build v1A Ship/QA Replication Summary, 2026-05-04 (curated research artifact). 12-run ballot-frame replication across four decision frames, seeds 900–902.

[4] Future Shock. Startup Build v1A 9-Run Forgetting Replication Summary, 2026-05-04 (curated research artifact). 9-run forgetting-mode replication across three strike modes, seeds 897–899.

[5] Future Shock. We Made AI Agents Negotiate the End of the World (Ark Protocol essay). https://news.future-shock.ai/we-made-ai-agents-negotiate-the-end-of-the-world/, 2026.

[6] Future Shock. Chaos Lab v2 Clean Live Results, 2026-05-02 (curated research artifact).

[7] X. Liu, H. Yu, H. Zhang, Y. Xu, X. Lei, H. Lai, Y. Gu, H. Ding, K. Men, K. Yang, S. Zhang, X. Deng, A. Zeng, Z. Du, C. Zhang, S. Shen, T. Zhang, Y. Su, H. Sun, M. Huang, Y. Dong, and J. Tang. AgentBench: Evaluating LLMs as Agents. arXiv:2308.03688, 2023.

[8] C. E. Jimenez, J. Yang, A. Wettig, S. Yao, K. Pei, O. Press, and K. Narasimhan. SWE-bench: Can Language Models Resolve Real-World GitHub Issues? ICLR 2024.

[9] J. Yang et al. Multi-SWE-bench: A Multilingual Benchmark for Issue Resolving. arXiv:2504.02605, 2025.

[10] C. Qian, X. Cong, C. Yang, W. Chen, Y. Su, J. Xu, Z. Liu, and M. Sun. Communicative Agents for Software Development. ACL 2024; arXiv:2307.07924.

[11] S. Hong, M. Zhuge, J. Chen, X. Zheng, Y. Cheng, C. Zhang, and collaborators. MetaGPT: Meta Programming for a Multi-Agent Collaborative Framework. ICLR 2024; arXiv:2308.00352.

[12] Y. Du, S. Li, A. Torralba, J. B. Tenenbaum, and I. Mordatch. Improving Factuality and Reasoning in Language Models through Multiagent Debate. arXiv:2305.14325, 2023.

[13] C.-M. Chan, W. Chen, Y. Su, J. Yu, W. Xue, S. Zhang, J. Fu, and Z. Liu. ChatEval: Towards Better LLM-based Evaluators through Multi-Agent Debate. arXiv:2308.07201, 2023.

[14] Y. Li et al. Voting or Consensus? Decision-Making in Multi-Agent Debate. Findings of ACL, 2025.