

Anomaly detection for drug targets without an agentic framework

Why the attention-against-the-record app is built with one pinned reasoning engine and a scientist who rules, and what a classic agentic build would and would not give. Raminderpal Singh, 24 September 2026. Companion to the preprint [1].

What the app does

For every drug target in a disease, scripts count how many people are working on it. One reasoning engine, fixed at one version, is then asked one question about each target from a fixed list of evidence: **does the public record explain that level of attention?** The engine argues the field's side, then the other side, then judges. It does this three times and all three answers are published. A **finding** is a target where at least two of three runs found no reason in the record: a target nobody works on with no stated reason, or a crowded target the record no longer supports. When a run offers a reason that no rule covers, the run stops there and a scientist writes the rule before the next round. The app recommends nothing; a person decides what a finding is worth.

This is the architecture of [1], section 5, applied one decision earlier than its two example systems: one pinned engine, provenance on every value, replication published as a spread, no verdict on the target, and the human decisions surfaced by running against the sources (requirement M4 in [1]).

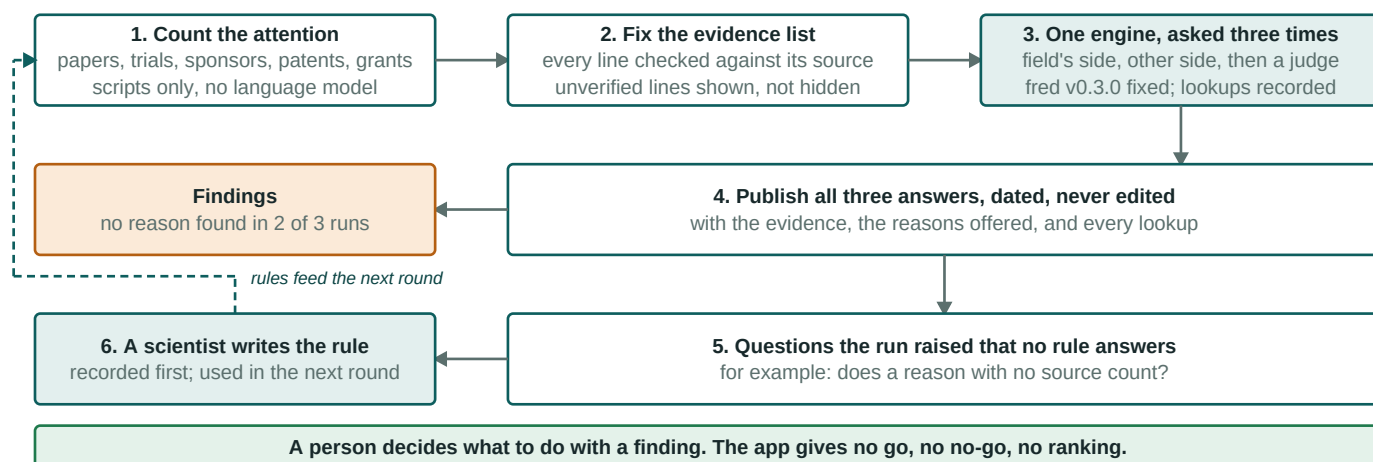


Figure 1. How a round runs. Steps 1, 2 and 4 are scripts. Step 3 is the only place a language model runs, and it is told whether few or many people are on the target, never the number. Step 6 is the loop that makes the method a loop rather than a specification followed by a build ([1], section 4).

The two ways to build it

A classic agentic build is an orchestrator dispatching a researcher, an analyst, a critic and a judge, each with its own prompt, tools and state, converging on an answer. It is the prevailing pattern and it is faster, wider and easier to hand to another engineer. The build here is the opposite shape: fixed inputs, one engine, three repetitions, a written rule for every judgment call, and a person at the end. Table 1 sets the two side by side on what matters for a decision-support tool. The assessments are design judgments against the standard in [1], not measurements.

Table 1. Building the app with one pinned engine against a classic agentic build.

What matters	This app (one pinned engine)	Classic agentic build	Who wins
Which targets are on the list	A script draws it from counted sources. Fixed before the round.	An agent searches and picks. Differs per run.	This app: the list can be checked.
How attention is measured	Counted by script. The engine is told few or many, never the number.	A tool call whose result is passed to the next agent.	This app: the count stays independent of the verdict.
What evidence the engine sees	A fixed list, checked against sources before the run. Unverified lines shown.	A researcher agent decides what to fetch, each run.	This app: no reason found means not in this list. Agentic: it means my search missed it.
How the question is argued	One engine, both sides, then a judge. Same list. Three runs, all published, never merged.	Planner, critic and judge agents pass state and converge on one answer.	This app: disagreement is visible and belongs to one engine. Agentic: one agent's error becomes the next one's fact [2].
What a finding is	Arithmetic: 2 of 3 runs say no reason found.	A judgment the system makes, often self-evaluated.	This app: anyone can recompute it.
A run raises a question no rule covers	It stops. A scientist writes the rule. The next round uses it. Four such rules exist now.	The agent picks a fallback, logged if at all. A harness asks yes or no, only where a gap was expected.	This app: the science decision stays with a person and is written before use (M4).

What matters	This app (one pinned engine)	Classic agentic build	Who wins
Engine version	Pinned. A new version makes a new dated entry.	A config edit; earlier outputs not attributed.	This app: reproducible (R3).
The record	Dated, digested, never edited. Predictions before, scored after, misses kept.	Progress files; context compacted; misses lost.	This app (M2).
Who decides	A person. No go, no no-go, no ranking.	The system converges and recommends.	This app, by requirement (R4).
Speed	Serial. Hours per target, weeks per round.	Parallel fan-out.	Agentic. Not needed: rounds are dated.
Breadth of search	One question, one list. Reasons outside the list are missed.	Wide, open-ended search across sources.	Agentic. But an unrecorded search cannot be checked.
Transfer to another engineer	Hard: rules are prose one person enforces.	Easier: rules are hooks.	Agentic.

The requirement codes are those of [1], section 2: R3 reproducible, R4 undecided, M2 recorded, M4 decided by a scientist as the data surfaces the decision. Reference [2] is the MAST failure taxonomy cited in [1].

What the agentic build cannot produce

A checkable negative. A finding says the record holds no reason. That is only worth anything if the reader knows what was searched. Here the evidence list is fixed and checked outside the engine, and the registry search is recorded even when it comes back empty, following the rule in [1] that a corpus cannot carry the proposition that no paper exists. A researcher agent that chooses its own path returns no reason found, which means its search did not find one and cannot be attributed to a defined search.

Independent parts. A finding here is arithmetic on a count from a script, a list classed outside the engine, and three verdicts from one pinned engine (Figure 2). In a chain of agents the count becomes a tool result passed to a critic, and the critic's output becomes the judge's input. The independence that makes the finding checkable is what the handoffs remove.

A spread instead of a verdict. Three runs are published and never merged, so the spread measures one engine's non-determinism and nothing else. Debate and orchestrator designs converge, and [1] cites evidence that they are sensitive to their agreement settings and do not reliably beat plain sampling. A merged answer is also a verdict on the target, which this app must not emit.

Science decisions made by a person, in writing, before use. Four rules on the app are waiting for a decision because a run offered a reason no rule covered: does an unsourced claim of undruggability count; does a documented reason for leaving a target expire when later evidence answers it; do genetics alone justify a crowd after comparable failures; what to do with the middle band. A coding agent resolves these as it goes and the choice is visible only if logged; a many-agent product resolves them at run time, differently each run; a harness can stop at a gap only if its author anticipated it, and returns yes or no. None of those four questions is answered yes or no.

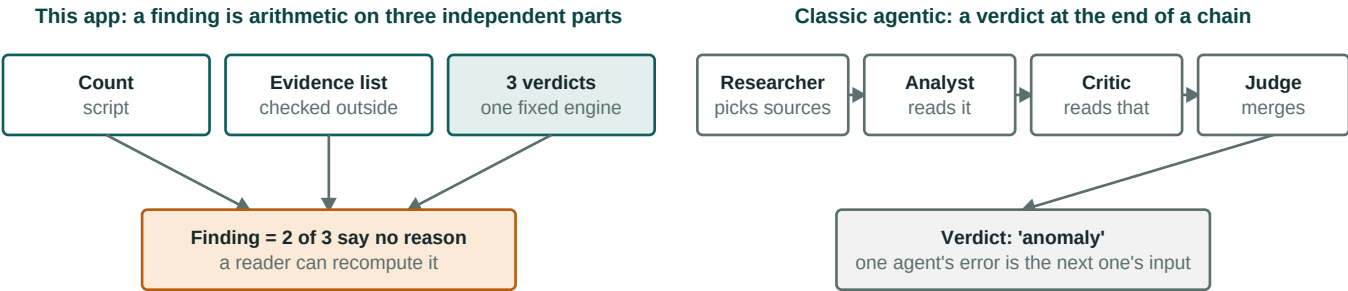


Figure 2. Where the anomaly comes from. Left: three parts produced by different mechanisms, combined by a rule anyone can apply. Right: a chain in which each agent reads the one before it and the last one decides; provenance is reconstructed afterwards ([1], Figure 2 and Table 2).

How the build method is used

Table 2. The four requirements on the method, and where each is visible in the app.

Requirement in [1]	How the app meets it
M1 Accountable: one person can defend every mechanism	Every finding decomposes into a count, a list and three sourced verdicts. Nothing on the page asks a reviewer to trust the model.
M2 Recorded: falsified hypotheses survive	Predictions are written before each round and scored after, and misses are kept. The prototype carries written predictions for ten targets, to be scored on the first real round. Entries are dated, digested, never edited.
M3 Externally verified	Nothing is checked by the engine that produced it. Attention is counted by script, evidence is re-resolved outside the engine, and the page carries a digest of its data. The cold read of the prototype caught two judge runs applying draft rules before they were recorded, and the fix restored the rule that a ruling takes effect only after it is written.
M4 Decided by a scientist as the data surfaces the decision	The Rules tab is the register. The waiting rules are the product's science, argued with a domain collaborator and recorded before the next round runs. This is the loop of [1], section 4, applied to content rather than code.

Where the agentic build wins, and why it does not decide this

Speed, breadth and transfer are real advantages and [1] concedes all three. The app does not need speed because its output is a dated round, not a live feed; the engine serializes and there is nothing to parallelize on one machine, and the design for scale is several pinned instances over disjoint targets under the same rules, none passing state to another. It gives up breadth on purpose, because a reason found by a search nobody recorded cannot be checked by a reviewer. Transfer is the cost that stays: the rules are prose one person enforces. The claim is not that an agent framework cannot produce a page like this one. It is that it cannot produce a finding a reader can verify, or hold the science decisions with a person, and those two properties are the product.

What would make it drift, and should be refused

Letting the engine find targets rather than a script listing them. Letting it choose evidence rather than receiving a fixed list. Merging the three runs, or adding a fourth role that decides. Running continuously rather than in dated rounds. Adopting a waiting rule automatically because runs keep flagging it. Each adds speed or breadth and removes one of the four properties above.

The risk to measure first

The constraint cuts both ways. If the engine often fails to find a reason that is plainly in the evidence list, findings inflate, and a wider agentic search would have found the reason. The first real round should record how often the field's side comes back empty on targets where a reason exists, because that number decides whether the fixed list is a strength or a defect in practice. It is a measurement, not an assumption, and it belongs in the record with the predictions.

References

[1] R. Singh. Decision Support for Pre-clinical Drug Discovery Without Agentic Engineering: One Pinned Reasoning Engine and a Human Who Decides. Preprint, 23 September 2026. <https://raminderpalsingh.com/singh-2026-attribute-never-assert-preprint.pdf>

[2] M. Cemri et al. Why do multi-agent LLM systems fail? NeurIPS 2025 Datasets and Benchmarks Track. arXiv:2503.13657. Cited as [1] in the preprint.