HOW DO ITEST MY AI?

Introduction

Most teams don't test their AI. They *think* they do - they run a few prompts, eyeball the outputs, and call it "good enough." That isn't testing. That's wishful thinking dressed up as validation. That is why an MIT study found that 85b to 95% of AI deployments fail. Real testing means understanding that an LLM is not software; it's a probabilistic system with unstable behaviour, hidden failure modes, and sensitivity to context, sampling, and pressure. If you don't test for those behaviours, you don't know what your model will do in production - you only know what it did once, in a demo.

Proper AI testing means using **real work in real workflows**, measuring drift, mapping the model's behavioural envelope, and validating structure with deterministic checks. It means stress-testing the model under the same conditions it will face when the stakes are real. And it means adopting a behavioural testing framework that exposes the model's weaknesses before your users do.

This is why the **LLM INQUISITOR METHODOLOGY** exists. It is not software, but a full methodology that can be applied in any system utilising AI. It is **free to download and use** from INQUISITOR LABS, and it gives you a structured, repeatable way to evaluate an LLM's stability, compliance, sampling sensitivity, and real-world performance. If you want to know whether your AI actually works - not whether it looked good once - this is where you start.

Section 1. What You're Probably Doing Now - And Why It Produces Shallow Results

1. **You test with cherry-picked prompts.**

   Teams choose clean, simple, well-structured inputs that make the model look good. These prompts don't reflect real work, real messiness, or real user

behaviour. They produce shallow confidence because the model is only being asked to perform under ideal conditions.

2. **You rely on one-off outputs instead of distributions.**

   A single “good” answer means nothing. LLMs are probabilistic systems; the real behaviour is in the *spread* of outputs, not the one sample you happened to like. If you don’t measure variance, you don’t know the model — you know a lucky roll.

3. **You test at one temperature and assume it generalises.**

   Teams run the model at a default temperature and assume that’s how it behaves. It isn’t. Change the sampling settings and the behaviour shifts dramatically. If you don’t explore the behavioural envelope, you’re blind to instability.

4. **You test in isolation, not in workflow.**

   A model that looks fine in a sandbox, collapses when placed inside a real workflow with real constraints, real edge cases, and real user inputs. Testing outside the workflow produces shallow results because it ignores the system the model must survive inside.

5. **You test for correctness, not behaviour.**

   Teams check whether the answer “looks right,” not whether the model is stable, consistent, drift-resistant, or structurally reliable. This is how hallucinations slip through: the model behaves unpredictably, but the one output you saw happened to be fine.

6. **You don’t test for drift over time.**

   LLMs change behaviour across calls, across contexts, and across days. If you only test once, you never see the drift. Shallow testing assumes the model is static; real testing reveals it isn’t.

7. **You don’t test under pressure.**

   Models behave differently when the input is long, ambiguous, contradictory, or messy. They behave differently when the context window is full. They behave differently when asked to maintain structure. If you don’t stress-test, you don’t know the failure modes.

8. **You trust the provider’s defaults.**

   Most teams assume the API’s sampling settings, safety layers, and versioning are stable. They aren’t. Providers silently override parameters, update models, and shift behaviour. Shallow testing never detects this.

9. **You don’t measure compliance or consistency.**

Teams check whether the model answered the question, not whether it followed instructions, maintained format, or respected constraints. This produces shallow confidence because the model's structural failures remain invisible.

10. **You don't use a behavioural testing framework.**

    Without a structured methodology, testing becomes vibes: "It seems fine." That's not engineering. That's gambling. This is exactly why **LLM INQUISITOR METHODOLOGY** exists - to replace shallow, ad-hoc testing with systematic behavioural evaluation.

11. **You test under ideal server conditions.**

    Most teams run their evaluations when the provider's servers are quiet, latency is low, bandwidth is stable, and the model is operating in its best-case state. This produces shallow results because **none of these conditions hold in production**. Real users hit the system during load spikes, degraded nodes, throttled throughput, and safety-layer recalibrations. A model that looks stable under ideal server conditions can fall apart the moment the infrastructure is under stress. If you don't test under realistic load, you're not testing the model - you're testing the provider's marketing environment.

Section 2. Behavioural testing under load: the LLM INQUISITOR METHODOLOGY approach

Behavioural testing is not about checking whether the model can answer a question. It is about observing how the model behaves when the conditions are not ideal, when the workflow is real, and when the load is intentional. This is the core of the LLM INQUISITOR METHODOLOGY: controlled pressure, structured observation, and evidence-driven evaluation.

1. **Test the model inside real workflows.** A model behaves differently when placed inside the actual workflow it must survive. You test it where the real constraints, dependencies, and failure points exist. Anything else is theatre.
2. **Apply controlled load across multiple axes.** Load is not just token length. It includes ambiguity, contradiction, structural demands, context saturation, and multi-step reasoning pressure. Behavioural testing requires applying these loads intentionally and observing how the model responds.
3. **Track behaviour across phases, not prompts.** LLM INQUISITOR uses a phase-structured workflow: initiation, exploration, load, long-form behaviour,

termination. Each phase reveals different failure modes. Testing by prompt hides these patterns.

4. **Capture a full evidentiary trace.**

   Inputs, outputs, transitions, axis changes, collapse signatures, and deviations must be logged. Without evidence, you cannot compare runs, detect drift, or justify decisions.

5. **Use multi-sample inference to map the behavioural envelope.**

   Run the same task at different temperatures and sampling settings. Compare the outputs. This reveals stability, uncertainty, hallucination zones, and collapse points.

6. **Measure collapse signatures explicitly.**

   Drift, contradiction, constraint loss, context collapse, and structural failure are not accidents. They are signatures. INQUISITOR classifies them so you can detect them consistently.

7. **Test under degraded and high-load conditions.**

   Long contexts, ambiguous instructions, conflicting requirements, and heavy structural demands expose weaknesses. Behavioural testing requires pushing the model until it breaks, not stopping when it looks good.

8. **Test across time to detect drift.** Run the same evaluation across days, contexts, and model versions. Behavioural drift is real and often severe. If you do not test over time, you will not see it.

9. **Validate structure with deterministic checks.**

   The model generates text. The system verifies structure. Behavioural testing includes checking whether the model maintains format, constraints, and required fields under load.

10. **Pin the model version and regression test.**

    Providers update models silently. Behavioural testing requires pinning versions and running regression tests to detect unexpected changes.

11. **Test with adversarial but realistic inputs.**

    Not malicious attacks. Realistic edge cases. Messy data. Conflicting instructions. Human-style errors. These reveal how the model behaves when the world is not clean.

12. **Use INQUISITOR scoring to quantify behaviour.**

Stability, constraint integrity, context retention, reasoning coherence, and collapse frequency are measurable. INQUISITOR provides a scoring structure so you can compare runs objectively.

13. **Treat collapse as a transition point, not an automatic stop.**

    When the model loses constraints or context integrity, you mark the point of collapse and change how you observe, but you do not automatically stop. You continue only if there is something new to learn: recovery patterns, re-stabilisation under new prompts, or how different axes of load affect the ability to recover. Behavioural testing includes both collapse and recovery, so you understand not just how the model fails, but whether and how it can be brought back into a stable operating state.

14. **Document the behavioural envelope and acceptable boundaries.**

    You must define what behaviour is acceptable, what is tolerable, and what is a failure. INQUISITOR produces the evidence you need to set these boundaries realistically.

15. **Use the results to shape system design.**

    Behavioural testing is not an academic exercise. The results inform scaffolding, constraints, validation layers, and workflow design. The model's behaviour determines the system, not the other way around.

Conclusion

Testing an AI system is not about validating perfect outputs. It is about revealing the weaknesses, risks, and behavioural instabilities that the model will show once it leaves the lab and enters real workflows. Proper testing means documenting those weaknesses honestly, understanding the conditions that trigger them, and adopting approaches to mitigate them. That can include coding solutions to detect invalid input and output, adding deterministic validation layers, or formally adopting work practices that teach users how much trust to place in the system and when to withhold it.

It also means training users to recognise collapse signatures, instability patterns, and behavioural drift so they can intervene early rather than assuming the model is still operating correctly. This is why the **LLM INQUISITOR METHODOLOGY** exists: to expose the behaviour the model will not show you in a demo, and to give teams a structured way to understand, measure, and mitigate the risks before they become failures in production.

Testing is not reassurance. Testing is exposure. And exposure is what makes the system safer.

Links:

Inquisitor Labs Homepage:

https://assimilatedhuman.github.io/inquisitor-labs/index.html

-document ends-