ML LABS INTELLIGENCE

What a Good AI Technical Assessment Includes

Many AI assessments produce opinions without enough decision value. This guide shows the minimum outputs a real technical assessment should deliver before larger spend is approved.

TECHNICAL

Author Omar Trejo

Date 2026-03-25

An AI technical assessment should reduce risk, not simply describe it.

That sounds obvious, but many assessments stop at commentary. They summarize architecture, list a few concerns, and call the job complete. The buyer leaves with more language, not more decision quality.

A good assessment does something harder. It converts ambiguity into a recommendation that changes what the team does next. That means the output has to cover more than technical observations. It needs to connect the current system or proposal to delivery risk, economic exposure, and an explicit next move.

The Output Should Change the Decision

The point of assessment is not to prove that someone looked carefully. The point is to decide whether the proposed path should proceed, change, or stop.

That is exactly where **RAND's analysis of AI project failures**, **research on requirements engineering for AI systems**, and **large-scale work on ML engineering practice** converge. AI initiatives fail less from lack of models than from weak translation between intended business value and the technical system that is actually being approved.

The Minimum Outputs a Good Assessment Must Deliver

A real technical assessment should leave the buyer with five outputs.

1. **Current-state read**: what exists now, what assumptions the proposal depends on, and what matters technically.

2. **Risk map**: the main delivery, architecture, data, reliability, or cost risks in plain language.
3. **Decision recommendation**: proceed, re-scope, change approach, or stop.
4. **Commercial framing**: budget range, effort shape, or spend exposure implied by the recommendation.
5. **Next-step path**: what to do immediately after the assessment, not just what is theoretically true.

Output One: Current-State Read

The assessment should start by making the current technical reality legible. That includes the workflow under review, the relevant systems, the proposed architecture, the operating constraints, and the assumptions embedded in the current plan. If the reviewer cannot explain the current state clearly, the rest of the document will drift into opinion.

This is also where weak assessments often go shallow. They repeat the proposal back to the buyer without testing whether the proposal depends on hidden requirements, missing integrations, or fragile data assumptions. **Google's ML engineering guidance** treats those hidden assumptions as a primary source of downstream failure for a reason.

Output Two: Risk Map

A useful risk map does not try to enumerate every conceivable issue. It identifies the few risks most likely to break the decision. These usually sit in one or more of the same buckets: unclear requirements, weak data path, brittle integration surface, poor operating model, or economics that no longer hold once the system is real.

The buyer should be able to read the risk map and understand which risks are acceptable, which need mitigation before approval, and which should kill the proposed path entirely. If the risk section only says "there are tradeoffs," the assessment did not do enough.

Output Three: Decision Recommendation

This is the output most weak assessments avoid. A good assessment does not hide behind neutrality once the evidence is clear. It should recommend one of four paths: proceed, proceed with conditions, re-scope, or stop.

That recommendation is the actual value of the work. Everything else exists to support it. **Research on hidden technical debt in ML systems** is useful here because it shows how easy it is to approve a technically plausible system that quietly creates long-term operating burden. Assessment should catch that before commitment, not document it after.

Output Four: Commercial Framing

A technical assessment that ignores commercial exposure is incomplete. Buyers do not approve architecture in a vacuum. They approve spend, delivery shape, and risk concentration. The assessment should therefore estimate what category of commitment the recommendation implies: bounded build, prerequisite infrastructure work, phased execution, vendor replacement, or halt.

This does not require fake precision. It requires useful precision. A range and a shape are often enough. The buyer needs to know whether the recommendation points to a modest follow-on build, a larger architectural change, or a stop decision that prevents expensive waste.

Output Five: Next-Step Path

The final output is a concrete next-step path. If the recommendation is "re-scope," the buyer should know exactly what needs to be narrowed. If the recommendation is "proceed with conditions," the conditions should be named. If the recommendation is "stop," the team should know what made the path non-viable and what category of alternative should be explored instead.

> A technical assessment is only complete when the buyer knows what to do next, what not to do next, and why.

Where Weak Assessments Fail

Most weak assessments fail in one of three ways. They stay descriptive instead of decisional. They stay technical without translating the commercial consequence. Or they stay cautious to the point that no real recommendation is made. All three failures leave the buyer with more document length and less buying clarity.

That is why a short, opinionated assessment is often more valuable than a longer neutral one. If the work does not alter the next decision, it did not produce enough leverage.

Boundary Condition

Assessment is not the right first move when the team still has not decided what workflow or system to evaluate. If the uncertainty is upstream, the assessment will be forced to invent the decision surface instead of pressure-testing it. That is usually a scoping problem, not an assessment problem.

Assessment is also not the whole answer when the system is already live and the next issue is not architecture choice but ongoing optimization across several active priorities. In that case the output may still be useful, but the team should recognize that execution continuity, not diagnostic clarity alone, is becoming the bigger constraint.

First Steps

1. **Write the decision under review.** If you cannot state the build, vendor, architecture, or workflow decision in one sentence, you are too early for assessment.
2. **Collect the materials that actually matter.** Architecture diagrams, code context, workflow details, incident examples, and cost concerns should be visible before the review starts.
3. **Ask what decision the output must change.** If the answer is unclear, tighten the scope until the assessment can produce a real recommendation.

Practical Solution Pattern

Run technical assessment as a decision instrument, not a documentation exercise. Start by making the current system or proposal legible, then identify the few risks most likely to break the decision, issue a direct recommendation, translate the commercial consequence, and finish with a concrete next-step path. If one of those outputs is missing, the assessment is still too weak.

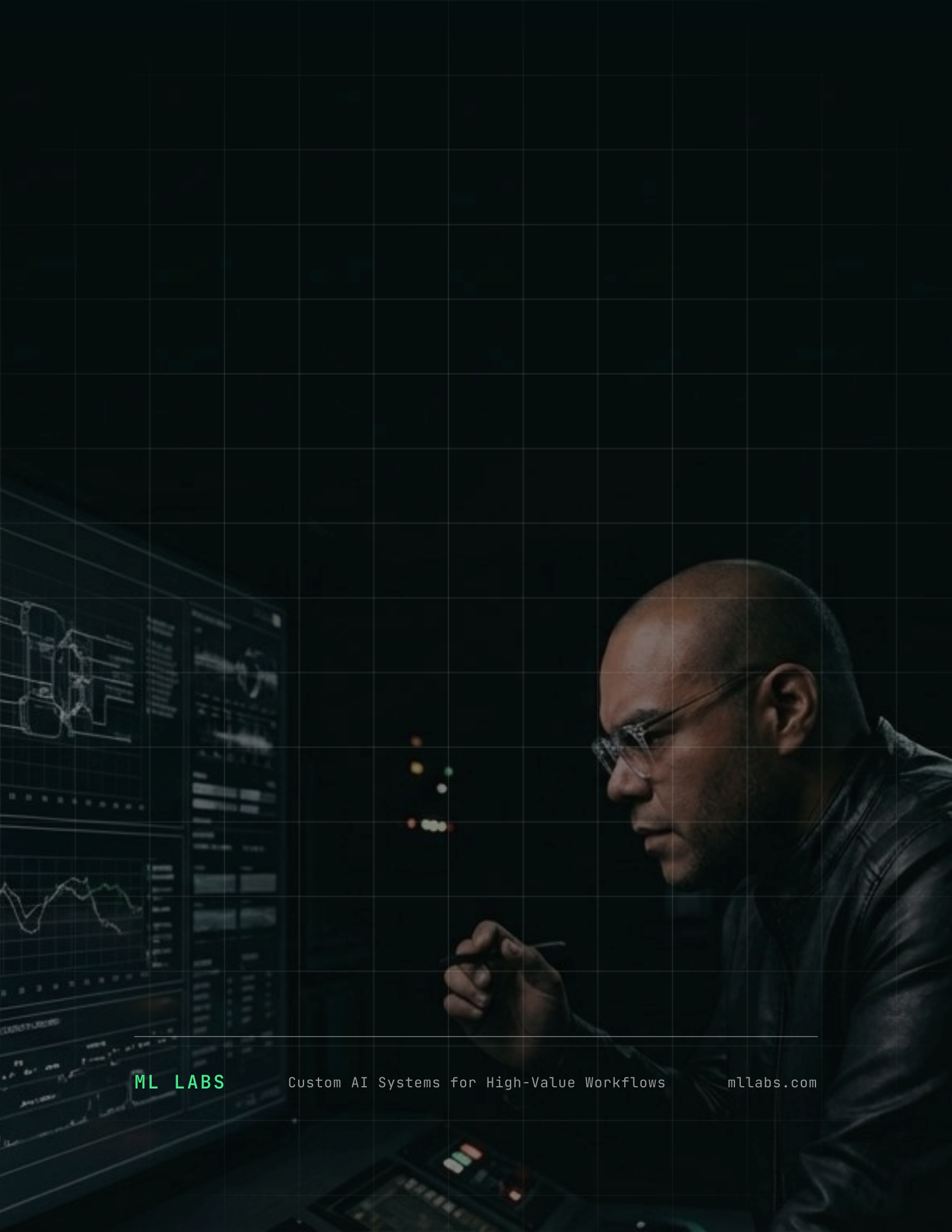
This works because buyers need the technical review to alter commitment quality, not just increase reading volume. The assessment earns its value when it changes whether a build proceeds, how it proceeds, or whether it should stop entirely. If the workflow is already clear and the next uncertainty is technical trust before larger spend, **AI Technical Assessment** is the correct next surface. If the target is still fuzzy, scoping should happen first instead.

References

1. Ahmad, K., Abdelrazek, M., Arora, C., Bano, M., & Grundy, J. **A Systematic Mapping Study on Requirements Engineering for AI-Intensive Systems**. arXiv, 2022.
2. RAND Corporation. **Analysis of AI Project Failures**. *RAND Corporation*, 2024.
3. Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., Nagappan, N., Nushi, B., & Zimmermann, T. **Software Engineering for Machine Learning: A Case Study**. *ICSE*, 2019.
4. Google. **Rules of Machine Learning: Best Practices for ML Engineering**. *Google Developers*, 2024.
5. Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J., & Dennison, D. **Hidden Technical Debt in**

Machine Learning Systems. *NeurIPS*, 2015.

6. HBR Editors. **Keep Your AI Projects on Track**. *Harvard Business Review*, 2023.



ML LABS

Custom AI Systems for High-Value Workflows

mllabs.com