

Designing a useful evaluation

Define the capability a test measures, preserve its conditions, and report what its results can and cannot establish.

Purpose and inputs

Use an evaluation to answer a specific development question: which constraint fails, where evidence is lost, or whether a changed workflow improves the deliverable. An aggregate score is useful only with the task definition and scoring rules beside it.

Specify the target capability, representative task set, reference artifacts, permitted tools, model configuration, and resource limits. Keep evaluation cases separate from materials used to tune the system under comparison.

Working steps

1. Define observable criteria: Separate final correctness from evidence use, constraint adherence, and recovery after errors. Choose repeatable checks where possible and a review rubric where judgment remains necessary.
2. Run a controlled comparison: Fix inputs, tool access, and budgets before comparing runs. Retain outputs and relevant action traces, and record failed runs alongside successful ones.
3. Inspect the failure patterns: Review errors by task and criterion. Use repeated runs where variability matters, and distinguish scoring disagreements from failures in the submitted work.

Deliverables

- Evaluation plan
- Versioned task set
- Run records
- Results and limitations

Review and acceptance

Deliver enough information to repeat the procedure and interpret its limits. Report coverage, uncertainty, exclusions, and changes to the rubric; do not generalize beyond the tested setting.

This guide describes a project approach. It is not a customer result, a performance benchmark, or a statement of certification.

Related guide: [/methodology/evaluation-design](#) | Browse downloadable demonstrations at [/samples](#).