

The New AI Benchmark Is Cost per Completed Task

The model benchmark tells you what a model can do. The production benchmark tells you what the system can finish—and at what cost.

Mykola Lobanov, Founder & Chief Product and Technology Officer
October 15, 2026 · 2996 words

For the last few years, the AI industry has optimized around model benchmarks:

- Accuracy.
- Context length.
- Tokens per second.
- Reasoning scores.
- Coding benchmarks.
- Cost per million tokens.

Those metrics are useful. They are also incomplete.

This is not a replacement for capability benchmarks. Accuracy, reasoning, coding, latency, and token efficiency still matter. They are leading indicators. Cost per completed task is the application-level metric that tells an engineering or business leader whether those capabilities survive contact with real tools, real data, real users, and real approval processes.

An enterprise does not buy tokens. It buys completed work.

It buys a resolved customer issue, a reviewed pull request, a reconciled account, a verified incident response, or a research brief that a decision-maker can actually use.

That changes the most important question in AI engineering:

How much does it cost to produce one reliable, accepted result?

The answer is not the model's token price. It is the cost per completed task.



Only the work that clears verification and approval counts. Everything the pipeline consumes on the way is part of the price of that one completed task.

A model response is not a completed task

An agent can return an answer and still fail the business process.

It may:

- Select the wrong tool.
- Use stale or incomplete context.
- Retry unnecessarily.
- Produce an artifact that requires major rework.
- Trigger a human review that takes longer than doing the work manually.
- Pass a model-based evaluator but fail a deterministic business rule.
- Take an unsafe action that must be rolled back.

The right unit of measurement is therefore not the response. It is the accepted task.

A task counts as completed only when it satisfies a written contract:

- The required artifact exists.
- The acceptance test passes.
- The output is within the required quality threshold.
- Required approvals have been completed.
- No unresolved rework remains.
- The result arrives within the service-level target.
- The agent has not violated data, security, or policy boundaries.

For a coding agent, that might mean a pull request passes the test suite, meets review standards, and is ready for merge. For an incident-response agent, it might mean the cause is documented, the remediation is verified, and the service is stable. For a support agent, it might mean the customer's issue is resolved without an unnecessary escalation.

“The model generated a plausible answer” is not an acceptance criterion.

The metric

At its simplest:

Cost per completed task = total cost of the workflow ÷ accepted tasks

For a run-level accounting model:

Fully loaded cost per completed task = (model + tools + compute + orchestration + human review + rework) ÷ accepted tasks

The numerator should include more than inference:

- Model input and output tokens.
- Reasoning and retry calls.
- Tool and API usage.
- Search, retrieval, and embedding operations.
- Sandbox, browser, container, and GPU time.
- Orchestration, queues, storage, and observability.
- Human review and approval time.
- Rework, rollback, or failure-recovery effort.

The denominator is the number of tasks that passed the acceptance contract—not the number of requests submitted and not the number of model responses returned.

The metric is valid only after a quality, safety, and service-level gate. A system should not improve its benchmark by producing a cheaper but incomplete artifact, skipping a required approval, or silently lowering the acceptance standard.

Teams should report two versions of the metric:

Metric	What it includes	Use
Marginal cost per completed task	Runtime costs for models, tools, compute, storage, and retries	Compare implementation choices and routing policies
Fully loaded cost per completed task	Marginal cost plus human review, rework, operations, and allocated platform cost	Make business and investment decisions

This distinction prevents two common mistakes. A team can make the API bill look low by hiding review time. Or it can allocate every platform expense into every run and make a useful workflow appear uneconomic. Publish the allocation rules and show both views.

Do not publish one blended number for unrelated task classes. A simple extraction task, a repository migration, and a production incident have different acceptance criteria and cost distributions. Report cost per completed task by cohort, then show the portfolio roll-up separately.

Why this benchmark matters now

Agents are no longer limited to single-turn text generation. They plan, call tools, manipulate files, run code, interact with business systems, and wait for humans.

That makes the workflow itself the economic unit.

Recent production research supports this shift. The Measuring Agents in Production study surveyed 306 practitioners, conducted 20 in-depth case studies across 26 domains, and filtered its analysis to 86 systems in production or pilot phases. Among the production findings, 68% of agents executed at most ten steps before human intervention, 70% relied on prompting off-the-shelf models rather than weight tuning, and 74% depended primarily on human evaluation. Reliability remained the top development challenge. Measuring Agents in Production, version 4

The numbers contain an important lesson: production value is created by controllable systems and human workflows, not by model intelligence alone.

The infrastructure is also becoming easier to measure. OpenAI’s updated Agents SDK makes the harness, sandbox, state, and long-running execution more explicit. It supports controlled sandbox execution and state snapshotting and rehydration, so a failed environment does not necessarily destroy the run. OpenAI Agents SDK

Anthropic describes a similar separation in Managed Agents: the session records what happened, the harness runs the agent loop, and the sandbox performs computation and tool work. Those components can fail, scale, and be replaced independently. Anthropic Managed Agents

The architecture has a direct measurement implication. The control plane owns the task budget and policy; the execution plane emits model, tool, compute, and side-effect events; the evidence plane joins those events to the acceptance result. Without that join, teams can report an API bill but not the cost of the work.

Once the workflow has a durable session and an event trace, the cost of each decision, tool call, retry, approval, and artifact can be attached to one task. That is the foundation for a real production benchmark.



Every abandoned branch was paid for. Token price measures the branches; cost per completed task measures the one that survived.

A simple example: the cheaper model loses

Consider an illustrative coding workflow with 100 repository tasks. The figures below are assumptions, not vendor pricing:

Metric	Fast model	Strong model
Runtime cost per attempt	\$0.40	\$1.60
Acceptance rate	60%	90%
Human review per attempt	6 minutes	2 minutes
Rework and recovery pool	\$120	\$60
Accepted tasks	60	90
Fully loaded cost per accepted task	\$12.67	\$4.67

Assume a reviewer costs \$60 per hour. The fast model has a lower runtime bill, but its failures create more review and rework. The strong model costs four times more to run, yet produces an accepted task at less than half the total cost.

The arithmetic is visible: the fast system costs (\$40 runtime + \$600 review + \$120 rework) ÷ 60 accepted tasks = \$12.67, and the strong system costs (\$160 runtime + \$200 review + \$60 rework) ÷ 90 accepted tasks = \$4.67.

The example is intentionally simple. Real systems also need to account for latency, queueing, tool failures, security reviews, data residency, and the cost of incidents. But the direction is important:

The lowest token cost is not the same as the lowest cost of completed work.

What belongs in a serious benchmark

A production benchmark should measure the complete path from request to accepted outcome.

1. Define a task contract

Write down:

- The input and authorized data sources.
- The tools the agent may use.
- The actions it may take automatically.
- The actions that require approval.
- The acceptance test.
- The time and cost budget.
- The escalation and fallback path.

Without a task contract, teams can quietly redefine success after seeing the output. That makes model comparisons meaningless.

2. Use a fixed task cohort

Benchmark a representative set of tasks rather than a single impressive example.

Include:

- Normal cases.
- Ambiguous cases.
- Large-context cases.
- Tool failures.
- Missing data.
- Adversarial or poisoned inputs.
- Cases that should be escalated to a human.

The same task cohort should run through each model, harness, or deployment pattern. Record the task version, prompt version, tool definitions, model version, and policy configuration.

Keep the operating conditions visible as well: concurrency, cache state, retrieval index version, context limits, timeout, retry budget, and pricing date. Randomize run order where practical and repeat the cohort enough to show a distribution rather than a lucky pass.

Report results by task class and difficulty. A single average can hide the fact that one system is excellent on routine work but unstable on the long-tail cases that consume the most human time.

3. Count all attempts

A failed run is still a cost.

Do not report only the successful traces. Include:

- Aborted runs.
- Timeouts.
- Invalid tool calls.
- Retries.
- Human escalations.
- Rollbacks.
- Tasks rejected by policy.

The benchmark should make failure visible rather than allowing a system to hide it by retrying until something works.

4. Capture structured traces

A useful trace records:

- Task identifier.
- Model and prompt versions.
- Context and retrieval operations.
- Tool calls and tool results.
- State transitions.
- Retry and recovery events.
- Sandbox and compute usage.
- Approval requests and decisions.
- Human review time.
- Final artifact and evaluator result.

The trace does not need to expose private chain-of-thought. It needs to preserve enough operational evidence to explain cost, behavior, and outcome.

At minimum, each attempt should be joinable through a stable task identifier and include: attempt status, model, tool and compute cost, human minutes, acceptance result, failure reason, and recovery outcome. If those fields cannot be reconstructed after the run, the benchmark is not reproducible.

5. Score acceptance separately from quality

One score is not enough.

Cost per completed task should be calculated only for outputs that clear the quality and safety floor. Then report the floor itself: acceptance rate, policy failures, and the reasons tasks were rejected. This prevents a system from looking efficient because it produces low-cost work that humans cannot use.

Report at least:

Metric	Why it matters
Acceptance rate	Shows how often the workflow finishes successfully
Cost per accepted task	Shows the economic unit the business pays for
p50 and p95 completion time	Exposes normal and worst-case user experience
Human minutes per task	Measures review burden and operational leverage
Tool-call error rate	Finds integration and schema problems
Recovery rate	Shows whether the system can survive failure
Escalation rate	Shows where autonomy stops
Unauthorized-action rate	Measures safety and policy performance
p50 and p95 cost per accepted task	Shows normal and long-tail economics
Cost variance	Finds unstable tasks and runaway loops

A model that is cheap on average but has a long tail of expensive failures may be a bad production choice.

The benchmark is a workflow scorecard, not a model leaderboard

Vendor benchmarks usually isolate a capability. Application benchmarks should evaluate the system around the model.

When comparing two agent systems, hold constant:

- The task cohort.
- The business data.
- The tool access.
- The acceptance tests.
- The approval policy.
- The maximum time and retry budget.
- The human-review protocol.

Allow the system to use the model and harness that it is designed for, but record those choices explicitly. A model with a specialized harness should not be compared to a raw API call and presented as a model-only result.

The fair comparison is:

Given the same business task and the same risk boundary, which system produces more accepted work per dollar and per hour of human attention?

This is especially important as organizations combine proprietary models, Chinese and open-weight models, private deployments, and smaller task-specific models. Model diversity makes routing more powerful, but it also makes a common workflow benchmark necessary.

Use cost as the benchmark, not the objective

Cost per completed task is an economic metric, not a definition of usefulness.

An agent can reduce cost by answering less, escalating more often, skipping verification, or producing an artifact that technically passes a weak evaluator. That is why the benchmark needs guardrails:

- A minimum acceptance rate.
- A minimum quality score or deterministic test result.
- A maximum unauthorized-action and policy-failure rate.
- A service-level target.
- A defined business outcome.

Compare cost only among systems that clear those gates. Then ask the business question: how much value does one accepted task create, and how much margin remains after the fully loaded cost? A low-cost task with no operational value is not efficiency.

Cost per completed task changes architecture

Once this metric becomes the target, several design choices become clearer.

Route by task, not by brand

Use a strong model for ambiguous planning, a faster model for extraction and routing, a coding model for repository changes, and a private or open-weight model where data residency or volume justifies the operational cost.

The model gateway should normalize tool calls, structured outputs, context limits, retries, safety responses, and usage accounting. This lets the team compare models on the same task instead of comparing disconnected demos.

Optimize the whole loop

The biggest cost may not be inference. It may be:

- A poorly described tool that causes repeated calls.
- A context window filled with irrelevant documents.

- A sandbox provisioned for a task that only needed an API.
- A human reviewing low-quality output line by line.
- A retry that repeats an irreversible action.

Tool contracts, just-in-time context, caching, deterministic validation, and clear stop conditions can reduce cost without changing the model.

Treat human review as an engineering variable

Human review is not an embarrassment to hide. It is part of the system.

Measure:

- Minutes per review.
- Percentage of runs reviewed.
- Defects found.
- Approval latency.
- Rework after approval.
- Whether reviewers trust the evidence shown to them.

An approval interface that reduces a review from ten minutes to two minutes may create more value than a small reduction in token price.

Make recovery cheap

Agents fail. Production design should make failure bounded and recoverable:

- Persist the session and workflow state.
- Use idempotency keys for external side effects.
- Separate the harness from the sandbox.
- Retry only safe operations.
- Resume from a checkpoint rather than restarting the task.
- Escalate when the expected recovery cost exceeds the task budget.

The recovery path belongs in the benchmark. A system that completes 90% of tasks but loses all progress when a container dies is not equivalent to a system that resumes from the last verified step.

Three useful benchmark examples

Software engineering

Measure cost per accepted pull request, not cost per generated line of code.

Include test-pass rate, reviewer minutes, rework, rollback rate, security findings, and time to merge.

DevOps and cybersecurity

Measure cost per verified incident or remediated finding.

Include time to mitigation, false-positive rate, approval latency, change-failure rate, and whether the agent can reproduce the issue safely before proposing a fix.

Customer operations

Measure cost per resolved case.

Include first-contact resolution, escalation, customer recontact, policy violations, tool errors, and human minutes. A fast answer that causes a second ticket is not a completed task.

These examples are more useful than a general “agent accuracy” number because they connect technical behavior to an operating result.

A practical 30-day rollout

A team does not need a perfect observability platform to start measuring this.

Week 1: Choose one workflow

Select a bounded process with a clear owner, repeated volume, and an acceptance test. Write the task contract and define what counts as a failed, escalated, or completed run.

Week 2: Instrument the path

Capture model calls, tool calls, retries, sandbox time, approvals, human minutes, artifacts, and acceptance outcomes under one task identifier. Start with a small event schema and make the cost fields explicit.

Week 3: Run a controlled comparison

Compare two models, routing policies, or harness configurations on the same task cohort. Keep the tools, data, risk policy, and acceptance tests constant. Run in shadow mode or behind an approval gate if the workflow has side effects.

Week 4: Make one operating decision

Choose the system that gives the best accepted-work economics under the quality and safety gates. The result may be a model change, better tool design, a shorter context, more automation of review, or a decision not to automate the workflow yet.

The goal is not to produce a perfect leaderboard. It is to create a repeatable feedback loop between architecture, operations, and business value.

What model and agent vendors should publish

The next generation of model and agent evaluations should publish more than a headline score.

For each workflow, vendors should report:

- Task definition and acceptance criteria.
- Task cohort, difficulty distribution, and number of repetitions.
- Completion rate.
- Cost per completed task.
- Cost per attempt.
- p50 and p95 latency.
- p50 and p95 cost.
- Human intervention rate and review time.
- Tool-call and recovery behavior.
- Safety and policy failures.
- Model, harness, tool, and infrastructure configuration.
- Cache state, concurrency, price assumptions, and measurement date.

A benchmark that omits human review, failed attempts, or infrastructure cost is not necessarily wrong. It is simply measuring a narrower unit. The problem begins when that narrow unit is presented as the cost of production work.

The new question for engineering leaders

The old question was:

Which model has the highest score?

The better question is:

Which system completes our important tasks reliably, safely, and at a cost the business can defend?

That question changes the role of AI engineering. Teams must own the task contract, the control plane, the execution boundary, the approval path, and the evidence layer. Model selection remains important, but it becomes one decision inside a larger production system.

For StarDevs, this is the practical opportunity: build model-agnostic agent systems for real software, SaaS, cloud, DevOps, and cybersecurity workflows—and measure the completed work, not the demo.

The future benchmark is not intelligence per token, and it is not a single leaderboard number. It is a workflow scorecard that connects capability, reliability, human effort, safety, latency, and unit economics.

It is cost per completed task.



(<https://stardevs.gg/images/blog/cost-per-task-card.png>)

The argument in one page, including the worked comparison.

Research notes

- Measuring Agents in Production, arXiv version 4
- OpenAI: The next evolution of the Agents SDK
- Anthropic: Scaling Managed Agents
- Anthropic: Agents for financial services
- OpenAI Evals API

Published by StarDevs, Inc at <https://stardevs.gg/blog/cost-per-completed-task/>. © 2026 StarDevs, Inc. This PDF is generated from the article at that address; the web version is the one that is kept up to date.