

OpenAI Models in Practice

Speed, Task Fit, and Mixed-Model Workflows

Jasem Al Salem

RSB-AI-002 · Version 1.0 · 3 October 2026

Abstract

Choosing a language model for useful work requires more than comparing how quickly text appears. Reasoning time, tool execution, coordination, verification, and failed attempts all influence the time and cost of producing an acceptable result. This report combines a targeted review of current official OpenAI documentation with an inspection of RSB's codex-agent-tree workflow. It examines GPT-6 Astra, GPT-6.1 Sol, and GPT-6 Luna, explains practical speed tradeoffs, and develops a framework for assigning different models to different parts of a task. The inspected RSB design assigns planning and execution primarily to Sol, reserves Astra for consequential independent review, and retains a single accountable integrator. This is a documented configuration, not evidence of a measured performance advantage. We distinguish model diversity, parallel execution, and independent review as separate interventions, derive conditions under which mixing models could help, and propose a controlled evaluation. The central recommendation is to optimize time and cost per accepted result subject to a quality threshold, rather than optimize token speed or agent count in isolation.

Scope and contribution

This is an engineering research report and prospective evaluation protocol. It is not a benchmark leaderboard, systematic literature review, or claim of statistical superiority. The focus is text, research, coding, and tool-using work; specialized voice, image, embedding, and open-weight models require separate evaluations. References [1-10] are official documentation retrieved on 3 October 2026. References [11-12] describe RSB's inspected source and historical operational record. No live comparative model experiment was performed for this report.

The contribution is a practical connection between three questions: which model fits a task, which part of the workflow determines its speed, and when combining models is worth the added coordination. RSB's prior work supplies a concrete architecture and operational lessons; the mathematical examples and proposed tests make its performance hypothesis falsifiable.

Keywords: language models; inference latency; model routing; multi-agent systems; verification; human acceptance.

1. Current models and task fit

The current OpenAI guide positions Astra for the most demanding work, Sol for a balance of capability, speed, and cost, and Luna for focused, high-volume workloads [1-4]. These are vendor descriptions, not RSB measurements. There is no universal best choice independent of workload, settings, and the definition of success.

Model	Documented speed positioning	Practical starting use
GPT-6 Astra	Capability-first; Fast and Ultrafast service options. No universal task latency reported here.	Difficult reasoning, ambiguous requirements, conflicting evidence, consequential review.
GPT-6.1 Sol	Balanced speed, cost, and intelligence.	Substantial coding, research synthesis, coordinated deliverables, iterative professional work.
GPT-6 Luna	Fastest and most cost-effective of the three in the current guide.	Constrained extraction, classification, narrow edits, repeated tasks with clear checks.

Table 1. Vendor positioning and RSB starting recommendations. These rows are not a measured tokens-per-second ranking [1-5].

Reasoning effort is a second selection axis

Model choice and reasoning effort should be evaluated together. The retrieved API pages list low, medium, high, xhigh, and max for Astra and Sol. Luna additionally supports none. Sol does not support none or minimal [2-4]. Product interfaces can expose different labels; a Codex setting should not be copied blindly into an API request.

For a bounded extraction task, begin with Luna at low effort and a schema-based validator. For substantial implementation, Sol at medium is a reasonable evaluation baseline. For difficult analysis or consequential review, compare Astra at medium and xhigh. These starting points follow the official selection guidance; raising effort is not a guarantee of correctness, and lowering it is not a guaranteed latency improvement on the complete task [5].

Match the model to the failure cost

Use a lower-cost model when the assignment is narrow, evidence is supplied, and errors are easy to detect. Escalate when ambiguity, missing context, cross-file dependencies, or difficult judgment dominate. A cheap response that causes repeated repair may cost more than one strong response. Conversely, a strong model cannot compensate for unavailable evidence or an incorrect acceptance criterion.

Existing GPT-5.x and GPT-6 Sol deployments remain useful comparators. Their presence in the catalog does not establish equivalence to GPT-6.1 Sol. Preserve exact model identifiers in logs and re-evaluate before migration. This paper deliberately concentrates on the current three-model decision rather than assigning unsupported speed numbers to the full historical catalog.

2. What speed actually means

At least four measurements are needed. **Time to first visible output** measures responsiveness. **Output throughput** measures emitted text tokens per second over a specified streaming interval. **Response latency** ends when a model response finishes. **Time to acceptance** ends only when the full deliverable meets its checks. A fast first token can coexist with a slow or incorrect result.

For a sequential workflow, a useful accounting identity is:

$T(\text{accept}) = T(\text{queue/network}) + T(\text{model}) + T(\text{tools}) + T(\text{coordination}) + T(\text{verification}) + T(\text{rework})$.

These terms must be instrumented without double counting. In concurrent work, elapsed time follows the dependency graph's critical path; adding every worker's duration would measure accumulated work instead. Report human waiting separately from active system time, while retaining both in the user-experienced total. Hidden reasoning is not visible output and should not be reconstructed from streaming text.

Service tiers change the comparison

OpenAI documents Fast mode with an up-to-2.5-times speed claim and specifically describes the improvement for GPT-5.6 Sol. This is an upper-bound vendor claim, not a universal end-to-end multiplier for every model. Astra also has Ultrafast, described as the fastest API service tier. The retrieved Ultrafast guide does not establish a fixed tokens-per-second rate or a guaranteed multiplier for this paper's tasks [6-7].

Fast mode can fall back to Standard under ramp-rate conditions; record the returned service tier. Astra Fast mode has no latency SLA in the retrieved documentation. Fast mode is unavailable with EU data residency for the three focal models, while Ultrafast supports US residency and global processing. Verify availability for the actual account and endpoint before designing around either option [6-7]. Rate limits expressed as tokens per minute are capacity limits, not the output speed of a single response.

Improve the work path before adding agents

The official latency guide recommends shorter outputs, fewer unnecessary requests, parallel independent work, and deterministic code where generation is unnecessary [8]. Apply these changes against a quality check: shortening an explanation is useful only if it preserves the required content. Streaming improves perceived responsiveness but does not prove earlier acceptance.

Caching can reduce repeated prompt processing, but long context is not automatically free or fast. Current models have cache-write charges, and cache reuse depends on eligible matching prefixes [9]. Measure actual cached usage per model; do not assume a cache transfers between models. Persistent connections can also matter for tool-heavy Ultrafast workflows [7]. These optimizations should be evaluated separately from model substitution.

3. What RSB has established

3.1 A concrete mixed-model design

We inspected the local codex-agent-tree checkout at revision 74fcb0ee81bb3e587ac7e536ba336e5d148fa64b. Its documented preference is GPT-6.1 Sol at high effort for the root, Sol at medium for explorer, worker, and researcher roles, and GPT-6 Astra at xhigh for consequential independent review [11]. The root owns planning, integration, verification, and the final response. Independent assignments may run concurrently; dependent assignments remain ordered.

The design contains safeguards that make contributions inspectable: assign bounded tasks, keep one writer per shared file or resource, serialize shared Git changes, and verify worker reports against the integrated result. The configuration is a preference subject to host capabilities and explicit user choices. It does not establish that the root model changes automatically, that every task uses every role, or that a preferred reviewer actually ran.

3.2 An operational lesson about instructions

A historical September 7 audit recorded that two implicitly active instruction packages prescribed model routing and additional task-graph procedures while the configured root remained Astra. The packages were removed from active discovery with backups [12]. This retrospective record supports a narrow lesson: the behavior attributed to a model may partly arise from surrounding instructions. It does not show that removing those instructions improved runtime performance. The historical event was not rerun for this report.

3.3 Findings and limits

Supported observation	Interpretation	Not established
RSB documented distinct planning, execution, and review roles.	Model allocation can be tied to responsibility.	The allocation is optimal.
Shared resources have explicit ownership rules.	Coordination risks are addressed in the design.	Fewer conflicts or defects in measured use.
Worker reports require integrated verification.	Completion is an evidence decision.	Higher acceptance rate than one model.
Instruction packages affected intended orchestration.	Evaluate the whole operating setup.	A causal speed improvement after removal.

Table 2. Evidence-bounded interpretation of the inspected source and retrospective record [11-12].

The repository explicitly states that its evaluation scenarios are not benchmark results. No controlled mixed-versus-single-model dataset, measured latency distribution, or cost-per-accepted-task comparison was identified in the material inspected for this report. RSB's result is therefore a documented engineering approach and a testable hypothesis. Faster completion, lower cost, and stronger review remain potential benefits to evaluate.

4. How mixed models can help

In this report, a mixed-model workflow uses more than one model identity across a task or task population. It does not mean merging model weights or using a mixture-of-experts architecture inside a model. Multiple agents using the same model are multi-agent but not mixed-model. Changing reasoning effort alone is configuration diversity, not model diversity.

Four useful patterns

Routing: assign each incoming task to a model using explicit task features and evaluated rules. A repetitive, verifiable request may go to Luna; an ambiguous synthesis may go directly to Astra. Bad routing can lose quality before any reviewer sees the result, so test difficult cases and false acceptance separately.

Cascade: a lower-cost model attempts a task first; failed checks or uncertainty trigger escalation. This can save cost when most initial attempts pass and escalation is rare. A model's confidence statement alone is not a reliable gate. Prefer executable checks, evidence requirements, and a calibrated evaluator.

Specialist team: an integrator assigns independent, bounded assignments to workers. One model may inspect source material while another implements an already-defined interface. Parallelism helps only where the dependency structure permits overlap. Passing complete conversation histories to every worker can increase cost and confusion; provide the necessary task contract and evidence instead.

Generator and reviewer: one model produces an artifact; another checks it against the brief, source evidence, and tests. This may expose overlooked defects, but distinct model names do not guarantee independent errors. Review should seek counterexamples and inspect the artifact rather than merely endorse the generator's summary.

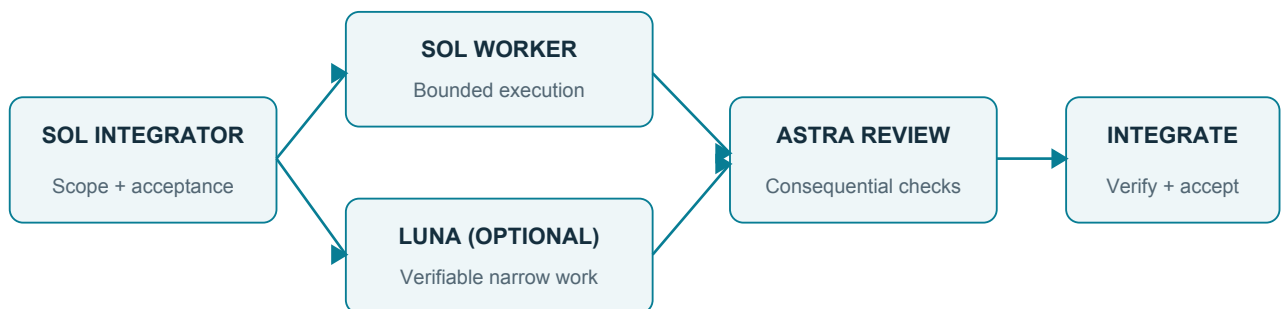


Figure 1. Proposed selective allocation. The observed RSB profile uses Sol workers and Astra review; the optional Luna branch is an extension proposed here. Model roles are not evidence of measured performance.

The common mechanism is selective use of capability: spend reasoning where it can change the outcome, and use lighter execution where the task can be bounded and verified. Keep one accountable integrator. If every worker requires a lengthy briefing, repeated correction, and a complete re-analysis by the reviewer, mixing models may be slower and more expensive than one capable model working directly.

5. When the extra coordination pays off

A simple cascade model

Let $C(s)$ be the expected cost of a lower-cost-model attempt, $C(v)$ its validation cost, $C(h)$ the expected cost of a direct stronger-model attempt, and p the fraction escalated after validation. Under a simplified model in which escalation costs the same as a direct stronger attempt:

$$E[C(\text{mixed})] = C(s) + C(v) + p C(h).$$

The mixed route is cheaper than direct strong-model execution when $C(s) + C(v) < (1 - p) C(h)$. For serial elapsed time, the analogous inequality uses measured times, but acceptance quality must be compared independently. Escalated cases may be harder than average and require context repair; direct strong-model work may also need review; include comparable verification in both costs. False acceptance can make a weak route appear economical while silently degrading quality.

Illustrative arithmetic, not experimental data: suppose a small attempt takes 4 seconds, validation 2 seconds, and a strong attempt 16 seconds. At 25% escalation, expected serial time is $4 + 2 + 0.25 \times 16 = 10$ seconds. At 75%, it is 18 seconds. The break-even escalation fraction is 62.5%. These invented durations explain the dependency; they are not OpenAI performance estimates.

A simple parallel model

For independent workers with durations $t(1) \dots t(k)$, a single parallel phase takes approximately the longest worker duration plus dispatch and integration overhead. Dependencies, capacity limits, and repeated review add further time. Under ideal parallelization, if fraction s of baseline time remains serial, the maximum speedup is:

$$\text{Speedup}(k) \leq 1 / [s + (1 - s)/k].$$

For an illustrative 40% serial fraction and four equal workers, the ideal ceiling is about 1.82 times before overhead. Adding workers cannot accelerate serial planning, a blocked dependency, or one shared integration step indefinitely. This is a structural limit, not a measured property of these models.

Review has value and cost

Independent review may increase elapsed time while reducing costly defects. That can still be beneficial. Evaluate escaped defects, unnecessary rework, reviewer false positives, and human intervention alongside latency. The objective is a constrained tradeoff: reduce time or cost while meeting a predefined quality floor. A mixture that wins only by weakening acceptance is not a successful optimization.

6. A practical operating guide

Work	Starting configuration to evaluate	Required evidence
Repeated field extraction	Luna low; deterministic schema and field checks.	Accuracy on difficult and missing-value cases.
Small localized code change	One Sol agent; medium effort.	Relevant checks and inspection of the actual diff.
Broad implementation	Sol integrator with bounded workers where independent.	Integrated tests, ownership boundaries, complete artifact.
Conflicting research evidence	Sol evidence collection; compare Astra synthesis or review.	Source-to-claim traceability and unresolved disagreements.
Consequential change	Capable implementer plus independent Astra review.	Counterexamples, relevant tests, explicit acceptance.

Table 3. Proposed starting configurations, not benchmark winners. Apply exact user model requirements and product availability [5,11].

Give each role an inspectable contract

A useful assignment states the desired artifact, permitted sources or files, known constraints, acceptance checks, and the evidence to return. For example: inspect the supplied functions, identify a reproducible failure, and return file locations with a minimal reproducer. An implementation assignment additionally identifies ownership boundaries. A reviewer receives the original brief and artifact, not just a persuasive worker report.

Escalate for a reason

Good triggers include a failing check, contradictory sources, missing assumptions that change the answer, repeated tool failure, or a change crossing a consequential boundary. Avoid automatically escalating every minor uncertainty, and avoid indefinite retry loops. Record the reason, preserve useful evidence, and cap attempts using a policy fixed before evaluation.

Keep successful simple tasks simple

For a typo correction or deterministic transformation, direct editing or code may be the fastest complete solution. A planner, worker, reviewer, and synthesizer can add latency without adding information. For broad work, increase parallelism only while independent assignments remain available. One writer and one integrator are ownership rules, not requirements to serialize unrelated research.

Tune one dimension at a time

Establish an acceptable baseline, then change model, reasoning effort, service tier, context size, or parallelism separately. Otherwise, a gain cannot be attributed to mixing models. Log substitutions and actual execution. A configured Astra reviewer is not proof of Astra review; a green isolated test is not proof that integration passes. These distinctions are central to making model-assisted work accountable.

7. Proposed controlled evaluation

This protocol has not been executed. Its primary controlled contrast tests model diversity while holding the team structure fixed. The team-versus-single comparison combines delegation, parallelism, and extra review; it cannot attribute their effects separately. Use frozen tasks, disposable workspaces, matched tool access, and identical acceptance criteria [10-11]. Evaluate at least a strong single-model baseline as well as the workhorse baseline so that the mixture is not compared only against a weak alternative.

Condition	Configuration	Primary question
A	Single Sol; no delegation.	Workhorse baseline.
B	Single Astra; no delegation.	Capability-first baseline.
C	Sol workers and Sol reviewer; bounded parallelism.	Does the same-model team help?
D	Same topology as C; Astra replaces reviewer.	What does model diversity add?
E	Luna attempt, validator, Sol escalation; serial.	Does a gated cascade pay off?

Table 4. Proposed ablation conditions. Compare C and D with matched roles, prompts, concurrency, and review access. Compare effort levels separately.

Begin with a pilot of 12 tasks spanning extraction, localized coding, broader implementation, and evidence synthesis, with three independent runs per condition: 180 task runs. These counts are proposed, not completed, and are insufficient for broad generalization or a stable tail-latency claim. Use the pilot to estimate variance and determine the larger sample needed for a predefined minimum practical improvement.

Randomize condition order within task and time blocks. Reset source state between runs. Separate cold-cache and warm-cache strata; fix concurrency limits and service tiers. Freeze task selection before comparing results. Keep tuning tasks separate from held-out evaluation tasks. Where a model lacks a feature, disclose the mismatch instead of silently substituting a different setup.

Record request and first-visible-output timestamps, completion and acceptance times, exact model and effort, requested and returned service tier, cache usage, token usage, tool durations, retry and escalation counts, total billed cost, and human intervention. Preserve artifact hashes, test outputs, and failure reasons. Record missing measurements as missing, never zero.

Use deterministic checks where possible and reviewers blinded to configuration for judgment-heavy artifacts. Define a task-level pass rubric in advance. Report acceptance rate, defects escaping review, cost per accepted task including spending on failures, median latency, and the full distribution. Retain timeouts and failures. Record time-to-terminal-outcome separately; acceptance time is unobserved for failures and right-censored at a fixed evaluation deadline for unfinished runs. Report the fraction accepted by that deadline alongside latency conditional on success. Use paired task comparisons with uncertainty estimated at the task level, rather than treating repeated runs as independent tasks. Tail percentiles need an adequately larger sample.

8. Limitations and conclusion

Official product documentation describes intended use and service behavior; it is not an independent benchmark. The inspected RSB repository is an instruction design, not a model runtime or a dataset of controlled outcomes. The historical instruction audit is a retrospective record. None of these sources establishes a numerical performance gain from RSB's mixture, and this report does not make one.

The current findings support a narrower conclusion: useful model selection depends on the task contract, the cost of errors, and the complete path to acceptance. RSB has documented a way to allocate work and retain accountability. Mixed models may improve resource allocation, parallel throughput, and review coverage when the assignments are separable and checks are reliable. They may underperform when routing is poor, errors are correlated, context is repeatedly rebuilt, or integration dominates.

The next empirical question is whether the same accepted deliverable can be produced more quickly or cheaply with the mixture than with a strong single-model baseline. Until that comparison is run, the appropriate claim is **a practical, testable workflow with bounded evidence**, rather than demonstrated acceleration.

Authorship, reproducibility, and status

Prepared for RSB AI Research Lab under Jasem Al Salem, with AI-assisted research, drafting, and document production. Human approval of the final interpretation is not asserted. RSB developed the workflow being examined; this is a first-party engineering report. It is a local research package, not a peer-reviewed publication or DOI deposit.

The companion package includes the editable manuscript, document builder, retrieved official source snapshots, local design-source copies, a provenance manifest, and document QA records. No API benchmark calls were made. The arithmetic examples are explicitly hypothetical. Model availability and behavior can change after the retrieval date; reproduce with exact versions and settings rather than assuming future equivalence.

References

- [1] OpenAI. Using GPT-6. Retrieved 3 October 2026. <https://developers.openai.com/api/docs/guides/latest-model>
- [2] OpenAI. GPT-6 Astra model documentation. Retrieved 3 October 2026. <https://developers.openai.com/api/docs/models/gpt-6-astra>
- [3] OpenAI. GPT-6.1 Sol model documentation. Retrieved 3 October 2026. <https://developers.openai.com/api/docs/models/gpt-6.1-sol>
- [4] OpenAI. GPT-6 Luna model documentation. Retrieved 3 October 2026. <https://developers.openai.com/api/docs/models/gpt-6-luna>
- [5] OpenAI. Model selection. Retrieved 3 October 2026. <https://developers.openai.com/api/docs/guides/model-selection>

References and source provenance

- [6] OpenAI. Fast mode. Retrieved 3 October 2026. <https://developers.openai.com/api/docs/guides/fast-mode>
- [7] OpenAI. Ultrafast mode. Retrieved 3 October 2026. <https://developers.openai.com/api/docs/guides/ultrafast-mode>
- [8] OpenAI. Latency optimization. Retrieved 3 October 2026. <https://developers.openai.com/api/docs/guides/latency-optimization>
- [9] OpenAI. Prompt caching. Retrieved 3 October 2026. <https://developers.openai.com/api/docs/guides/prompt-caching>
- [10] OpenAI. Evaluation best practices. Retrieved 3 October 2026. <https://developers.openai.com/api/docs/guides/evaluation-best-practices>
- [11] RSB-HOLDING. codex-agent-tree: README, skill instructions, and evaluation protocol. Local checkout inspected at revision 74fcb0ee81bb3e587ac7e536ba336e5d148fa64b; copies retained in the evidence package. <https://github.com/RSB-HOLDING/codex-agent-tree/tree/74fcb0ee81bb3e587ac7e536ba336e5d148fa64b>
- [12] RSB operational record. Audit and removal of implicitly active orchestration skills, 7 September 2026. Retrospective internal session summary; not publicly archived. Session identifier 01a07ade-7356-7e10-b32a-1d0212d786d9. A bounded provenance note is included in the companion evidence package. This record was not independently rerun.

Source-to-claim map

Claim group	Evidence	Interpretation boundary
Current model positioning and effort settings	[1-5]	Official descriptions, not RSB speed measurements.
Service tiers, availability, and optimization	[6-9]	Endpoint-specific; no universal latency guarantee.
Evaluation design principles	[10]	Adapted methodology; proposed study is unexecuted.
RSB roles, model preferences, ownership rules	[11]	Verified design at one source revision.
Historical instruction-layer finding	[12]	Retrospective observation; no causal estimate.
Cascade arithmetic and parallel bound	Section 5	Analytical examples using hypothetical inputs.

Suggested citation

Al Salem, J. (2026). OpenAI Models in Practice: Speed, Task Fit, and Mixed-Model Workflows. RSB AI Research Lab, Research Report RSB-AI-002, version 1.0. Local technical report, 3 October 2026.