# A practical guide to AI-assisted development.

Sol for implementation, Luna for bounded routine work, Astra and Fable for demanding reasoning, and Opus for implementation or a second perspective. Use a clear contract and observed evidence throughout delivery.

UPDATED 23.09.2026 | Software · web · mobile · APIs · data | Recommended workflow



## Five working principles

1. **Sol is the default:** use it for implementation with clear interfaces and acceptance criteria.
2. **Astra resolves hard uncertainty:** use it for architecture, trust boundaries, complex concurrency, and recovery.
3. **Luna handles bounded work:** give it repeatable tasks with an inexpensive, reliable check.
4. **Opus provides another perspective:** use a separate review task with explicit adversarial questions.
5. **Fable for demanding extended work:** assign a substantial, bounded package when difficult reasoning must remain coherent across many steps.

## What each model is for

COMPLEX DECISIONS

## GPT 6 Astra

Architecture, security boundaries, difficult debugging, data migrations, distributed behavior, and integration across several systems.

`High for consequential changes; Extra High for unresolved, interacting constraints. Use lower effort for a narrow, well-understood task.`

EVERYDAY IMPLEMENTATION

## GPT 6 Sol

Features, UI and API work, refactoring, meaningful tests, automation, and bug fixes within a defined scope.

`Medium as the working default; High for complex logic and cross-module changes.`

FOCUSED EXECUTION

## GPT 6 Luna

Documentation from evidence, structured extraction, small mechanical edits, translation checks, and execution of fixed test scripts.

`High as a starting point. Escalate decisions about permissions, business meaning, or destructive operations.`

IMPLEMENTATION & INDEPENDENT REVIEW

## Opus 5.5

Can own a bounded implementation package. Particularly useful as a separate reviewer challenging another agent's assumptions and tests.

`Medium for well-defined implementation; High for critical reviews. Verify available settings in the Claude client.`

DEMANDING EXTENDED DEVELOPMENT

## Claude Fable 5.1

Complex packages spanning many steps, challenging architectural investigations, or a deep independent review when simpler approaches leave material questions unresolved.

`High as a starting point. Use clear checkpoints and a usage budget; validate the benefit on your own tasks.`

### Allocate work across the development lifecycle

| WORK | PRIMARY MODEL / EFFORT | REQUIRED CHECK |
|---|---|---|
| Requirements and scope | Sol Medium; Astra High for ambiguity | Examples, exclusions, unresolved decisions, acceptance criteria |
| Architecture and public contracts | Astra High | Alternatives, failure modes, compatibility, independent review |
| UI, accessibility and localization | Sol Medium | Real interaction, keyboard use, relevant languages and screen sizes |
| Business logic and API implementation | Sol High for complex work | Public-interface tests, validation, errors and retries |
| Authentication and tenant isolation | Astra High / Extra High | Negative cross-tenant, role, session and object-access tests; independent review |
| Database migrations and concurrency | Astra High | Real database, contention, failed transactions, restore and rollback |
| Small mechanical refactors | Luna High or Sol Medium | Diff review and a focused regression check |
| Difficult or intermittent defects | Sol High → Astra High if unresolved | Reproduction, hypothesis, isolated cause, regression test |
| Fixed browser / device acceptance | Sol Medium; Luna for records | Actual target device/browser and exact build identity |
| Benchmark and evaluator design | Astra High or Fable High + independent reviewer | Independent oracle, held-out cases, meaningful thresholds, no target-score tuning |
| Extended multi-module development | Fable High or Astra High; Sol for bounded subtasks | Milestone evidence, fixed interfaces, one integration owner, independent review |
| Deployment and production recovery | Astra High for planning and high-risk changes | Bound artifact, actual target, backup/restore, health checks and authorized rollout |

| WORK | PRIMARY MODEL / EFFORT | REQUIRED CHECK |
| --- | --- | --- |
| Release notes and maintenance records | Luna High | Trace every claim to executed evidence; Sol checks completeness |

## One delivery workflow, clear ownership

1. **Define the contract.** State the outcome, scope, interfaces, acceptance tests, budget, and stop conditions. Read repository instructions before editing.
2. **Assign ownership.** Give each agent a bounded package and distinct files. One integration owner resolves shared changes. Parallelize only independent work.
3. **Implement the whole flow.** Handle authorization, loading, empty states, failure, cancellation, retry, and recovery where relevant. Preserve unrelated changes.
4. **Test the actual risk.** Use public entry points and real dependencies when practical. Separate simulated results from database, device, provider, and production evidence.
5. **Review independently.** Ask for counterexamples and dangerous failure directions. Use independently derived expectations; a different model alone is not independence.
6. **Integrate and release.** Validate the combined artifact, migrations, and recovery path. Respect deployment authorization and change windows; do not infer approval from passing tests.
7. **Observe and maintain.** Check the deployed version and critical flows. Record limitations, operational signals, ownership, and follow-up work.

## When to change model or effort

ESCALATE THE UNCERTAINTY

Move from Luna to Sol when the task needs interpretation across files. Move to Astra when trust boundaries, irreversible effects, conflicting evidence, or complex system interactions remain unresolved. Give it the failing case and evidence, not just “try harder.” Fable High is an alternative for demanding work over

KEEP EFFORT PROPORTIONAL

A clear task with reliable checks can stay on a smaller model. High and Extra High are effort settings, not equivalent capability levels across models. Increasing effort does not replace missing requirements, an independent oracle, or a real device.

many steps or a second deep investigation. If Astra implements, Fable can review; if Fable implements, Astra can review. Assign separate files and independent acceptance evidence.

> Production data changes, destructive commands, secrets, paid services, and external publication need explicit scope and the applicable authorization. A model recommendation is not permission to perform an action.

## Reusable task brief

```
Outcome: [observable user or system result]
Scope: [included work and explicit exclusions]
Contract: [repository instructions, plan, interfaces]
Ownership: [allowed files; integration owner]
Model / effort: [recommendation and reason]
Acceptance: [real flows and objective success criteria]
Negative cases: [permissions, stale data, retry, concurrency]
Evidence: [commands, outputs, artifact/build identity]
Constraints: [time/credit budget, dependencies, data boundaries]
Escalation: [uncertainty that requires review or user input]
Release: [destination, authorization, migration and rollback]
Finish: [reviewable changes, test evidence, limits, next steps]
```

## Basis and limits

Vendor documentation informs the broad model roles. The allocation above is an engineering recommendation, not a measured model ranking or a guarantee of safety. Validate it on representative tasks from your own codebase; availability and effort controls vary by account and client.

- OpenAI · Models & reasoning effort
- Anthropic · Models overview

GUIDE ONLY · NO MODEL CONFIGURATION OR DEPLOYMENT CHANGES