



AI-Assisted Coding Workflows

AI for software engineering 01 | Inline, chat, synchronous agent, async agent and headless modes; picking one per task; explore-plan-implement-verify; test-first with agents; task specs; parallel agents in worktrees and the review bottleneck

#AI4SE #Workflows #CodingAgents #TDD

Five ways to work with AI

Choosing an AI coding workflow

Newcomer

Small edits: let the editor finish your line. Bigger jobs: describe the job, let the agent try, then read every change before keeping it.

Practitioner

Match the mode to the task: inline for flow, chat to understand, a watched agent for multi-file work, an async agent for well-specified tickets with tests.

Lead

The scarce resource is verification and review. Spend agent time where a test or type checker can say 'done', and budget human review per change.

PICTURE IT

Kitchen help: a sous-chef hands you ingredients (inline), answers questions (chat), cooks a dish under your eye (agent), or preps overnight for you to taste in the morning (async).

One idea, three audiences: the workflow is chosen per task, not per team

Mode	Scope	Human role	Best for	Typical tools
Inline completion	a line to a function	accept or reject each suggestion	boilerplate, known patterns	Copilot, Cursor Tab, JetBrains AI
Chat assist	one question or snippet	copies and adapts the answer	explaining code, regex, SQL, errors	any chat model, IDE chat
Synchronous agent	multi-file change in one session	approves plan and commands, reads diffs	bug fixes, refactors, features with tests	Claude Code, Codex CLI, Cursor agent
Async agent	one ticket → one PR	writes the spec, reviews the PR	well-specified tickets, chores	Copilot coding agent, Codex cloud, Jules
Headless / CI	scripted batch	owns the script and the gates	triage, migrations, review bots	agent CLIs in GitHub Actions

The five modes as of 2026 (product names change; the shapes stay)

Before AI-assisted coding workflows, can you explain...

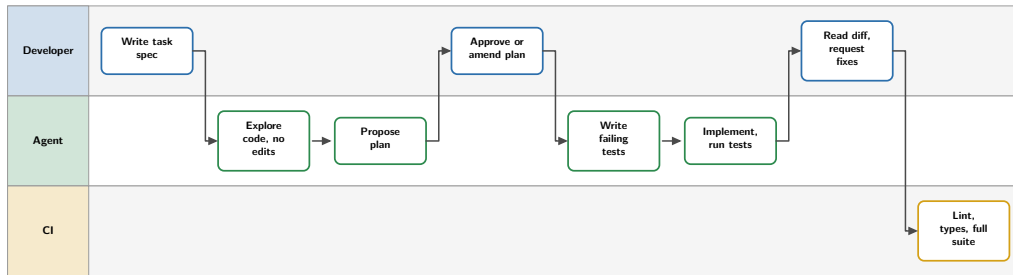
	Not yet	Partly	Yes
MUST KNOW Agent loop, permission modes, sandboxes Learn: agents 07	☐	☐	☐
MUST KNOW git branches, worktrees, PRs Learn: workflows 10	☐	☐	☐
MUST KNOW A test suite that runs in one command Learn: programming 08	☐	☐	☐
HELPFUL Prompt structure (role, context, output) Learn: llm 09	☐	☐	☐
HELPFUL Context budgets and compaction Learn: agents 04, llm 10	☐	☐	☐

Before running agents on real tickets

→ Pick the workflow per task

Conditions	Chore	Bug + repro	Feature	Design spike	Auth / crypto
Spec fits in 10 lines	Y	Y	N	N	--
A test can say 'done'	Y	Y	Y	N	--
Touches security-critical code	N	N	N	N	Y
Design still unknown	N	N	N	Y	--
Actions					
Async agent → PR	X				
Synchronous agent, test-first		X	X		
Chat to explore options, human writes code				X	
Human writes, AI reviews and explains					X
Human reviews every line	X	X	X	X	X
Notes	dependency bumps, lint fixes	failing test first	split into PR-sized steps	prototype, then discard	AI as second reader

Which workflow for which task. Read a column top to bottom: the conditions select the X actions



The explore → plan → implement → verify loop of a synchronous agent session (pattern from Anthropic's Claude Code best practices, 2025; works with any agent)

A task spec an agent can finish (paste into the session or the issue body)

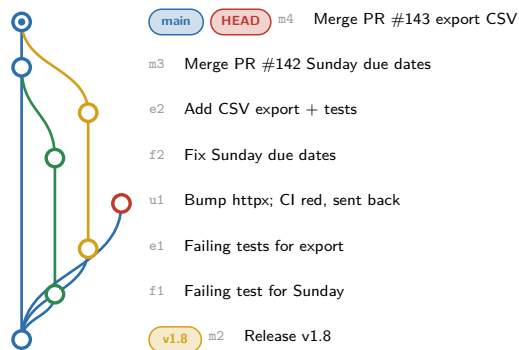
```

## Goal
Invoices due on a Sunday must move to Monday.
## Context
src/invoices/dates.py (due_date), tests/test_dates.py
## Constraints
- No new dependencies; keep the public signature
- Do not edit existing tests
## Acceptance criteria
- New test: issued 2026-09-20 + 14 days -> 2026-10-05
- uv run pytest -q and uv run ruff check . pass
## Out of scope
Bank holidays (ticket INV-212)
  
```

⚠ The agent must not grade its own homework

Agents under pressure to make tests pass sometimes weaken or delete the tests, special-case the test input, or mock the function under test. Commit the tests first, forbid edits to them in the spec, and check the diff of tests/ before anything else in review.

🧑‍🤖 Parallel agents and the review bottleneck



Three agents in three worktrees, each on its own branch from the same main; each lands as a separate small PR

PROBLEM One developer runs N parallel agents. Each agent produces one PR every $T_a = 40$ min; reviewing a PR takes $T_r = 12$ min; a share $p = 0.6$ of PRs merge after review, the rest go back. How many PRs merge per hour with $N = 3$ and $N = 4$?

- GIVEN**
- N = number of parallel agents
 - T_a = agent time per PR (min); production rate = $N \times 60 / T_a$ (PRs/h)
 - T_r = human review time per PR (min); review capacity = $60 / T_r$ (PRs/h)
 - p = share of reviewed PRs that merge (0-1)
 - Merged per hour = $p \times \min(\text{production}, \text{review capacity})$

1 Review capacity

$$60 / 12 = 5 \text{ PRs/h}$$

Why: the human can read at most five PRs an hour, doing nothing else

2 Production with $N = 3$

$$3 \times 60 / 40 = 4.5 \text{ PRs/h}$$

Why: below capacity: the reviewer is busy $4.5 \times 12 = 54$ min per hour

3 Merged with $N = 3$

$$0.6 \times \min(4.5, 5) = 2.7 \text{ PRs/h}$$

Why: rejected PRs consume review time too

4 Production with $N = 4$

$$4 \times 60 / 40 = 6 \text{ PRs/h} > 5$$

Why: one PR per hour piles up unreviewed

5 Merged with $N = 4$

$$0.6 \times \min(6, 5) = 3.0 \text{ PRs/h}$$

Why: +11 % output for +33 % agents; the queue grows by 1 PR/h

ANSWER 2.7 merged PRs/h at $N = 3$, 3.0 at $N = 4$ with a growing backlog: the reviewer, not the agents, is the limit

CHECK ☒ Raise p (better specs, test-first) or cut T_r (smaller PRs, AI pre-review in sheet 03) before adding agents

How many agents can one reviewer keep busy? Throughput is capped by the slowest stage

💡 TIP

Raising p is the cheapest lever: a spec with acceptance tests and a 'do not edit tests' rule turns many rejected PRs into merged ones. Cutting PR size cuts T_r roughly in proportion, and small PRs are also what reviewers approve with confidence.

✓ Habits that make it work

Before you merge an AI-assisted change (checked = always)

☒ baseline / mandatory ☐ recommended – verify in your context

- ✓ You can explain every changed line to a colleague
- ✓ Tests were written or confirmed failing before the fix
- ✓ No edits to tests/, CI config or lock files you did not ask for
- ✓ Every new dependency exists, is maintained and was intended
- ✓ Full suite, linter and type checker pass in CI, not only locally
- ☐ **SHOULD** One task per session; context cleared between tasks

- ☐ **SHOULD** PR description says which parts were AI-generated and how they were checked
- ☐ **COULD** Prompts that worked are saved as reusable commands or templates

YOU MAY HEAR

"A longer prompt with the whole app idea gives a better result"

BUSTED

✓ ACTUALLY

PR-sized tasks with acceptance criteria do

Large vague goals produce large unreviewable diffs; split into steps a test can verify.

YOU MAY HEAR

"Async agents remove the need for a developer"

BUSTED

✓ ACTUALLY

They move the developer to spec writing and review

The ticket quality and the review set the merge rate (worked example above).

YOU MAY HEAR

"Vibe coding is how professionals use agents"

BUSTED

✓ ACTUALLY

Professionals read and verify the diff

Karpathy's 2025 term describes accepting code unread: acceptable for throwaway prototypes, not for production.

YOU MAY HEAR

"More context in the session is always better"

BUSTED

✓ ACTUALLY

Stale context steers the model wrong

One task per session; reset between tasks; keep the rules file short (agents 07).

Common beliefs about working with coding AI

