-->

Spec-Driven Development and AI Coding Agents: A Decision Map for Solo Indie Production Work

Research date: 2026-05-08
Audience: Graham — solo indie developer, multi-project, "vibe coding" → production
Question: Which AI coding agent and spec-driven tools best fit a solo indie shipping multiple production apps, and how do OpenSpec + oh-my-openagent rank against the broader landscape?

Executive Summary

The AI coding tooling space has split into three distinguishable layers: **spec-driven development (SDD)** tools that turn intent into auditable artifacts, **agent-config/rules** files that steer in-IDE assistants, and **agent harnesses/orchestrators** that actually do the work. The question is no longer "which IDE assistant" but "what shape of intent capture + what execution surface + what coordination layer." For a solo indie running parallel projects, the winning move is a **lightweight SDD layer + a model-agnostic execution layer + a portable rules convention**, not a single monolithic platform. **Confidence: HIGH.**

The strongest stack for Graham's profile is **OpenSpec (intent) + OpenCode + oh-my-openagent (execution) + AGENTS.md (portable rules)** — all OSS, all model-agnostic, all per-project. Heavier alternatives (Spec-Kit, BMAD-METHOD) impose ceremony that pays back on team/regulated work but taxes solo throughput; vendor-locked options (Kiro, Windsurf) trade portability for polish; emerging "spec-as-source" platforms (Tessl) are conceptually appealing but pre-GA. **Confidence: MODERATE-HIGH** given rapid release cadence in this category.

The Landscape in Two Paragraphs

Through 2024–2025, "AI coding" mostly meant chat-in-IDE. Through 2025–2026 it has bifurcated into **spec-first** (intent is the artifact, code is the output — Spec-Kit, OpenSpec, BMAD, Tessl, Kiro) and **agent-execution-first** (the agent is the artifact, intent lives in rules files and prompts — Cline, Roo Code, OpenCode, Cursor, Windsurf, Aider, Continue.dev, Conductor). A third layer — the **portable rules standard** — emerged when AGENTS.md coalesced into a Linux Foundation–hosted convention

adopted across Codex, Copilot, Cursor, Windsurf, Amp, Devin, and OpenCode in 60k+ repos, with Claude Code retaining its CLAUDE.md (agents.md). The practical implication: rules portability is now solved; the remaining choices are *what shape of spec* and *what shape of execution* to commit to.

Within SDD, two philosophies compete. **Spec-anchored** tools (OpenSpec, Spec-Kit) treat code as authoritative and specs as living scaffolding, with OpenSpec's delta-spec model (ADDED/MODIFIED/REMOVED change proposals) optimized for brownfield work and Spec-Kit's seven-step waterfall (constitution → specify → clarify → plan → analyze → tasks → implement) optimized for greenfield. **Spec-as-source** tools (Tessl) flip the relationship — specs are canonical, code is regenerated. On the agent-execution side, the dominant pattern is now **multi-agent orchestration in git worktrees**: BMAD-METHOD encodes this as personas (Analyst/PM/Architect/SM/Dev/QA), oh-my-openagent encodes it as Greek/myth-named workers under a Sisyphus orchestrator, and Conductor encodes it as a Mac-app dashboard running Claude Code + Codex side by side. The "harness" layer matters as much as the model layer: model-agnostic harnesses (OpenCode, Cline, Aider) preserve optionality; vendor-coupled harnesses (Kiro, Windsurf, Cursor) lock pricing and capability ceilings to one vendor.

Per-Tool Comparison

Tool	Category	Purpose	Execution	Models	Simplicity (1-5)	Execution Power (1-5)	One-line Verdict	Source
OpenSpec	SDD (spec-anchored)	Brownfield delta-spec change proposals	Slash-command layer over host AI tool (`/opsx:propose	apply	archive`)	Any (Claude, GPT, Gemini, etc. via host)	5	4

Tool	Category	Purpose	Execution	Models	Simplicity (1-5)	Execution Power (1-5)	One-line Verdict	Source
oh-my-openagent (omo)	Agent harness extension	Multi-persona orchestration plugin for OpenCode	Hook-driven via OpenCode CLI	Multi (Claude/GPT/Kimi/GLM/Gemini/Minimax)	3	5	Most powerful OSS multi-agent harness for solo workflows	github.com/code-yeongyu/oh-my-openagent
OpenCode (sst)	Agent harness	Provider-agnostic terminal agent	TUI/CLI; MCP support	Any (BYOK)	4	4	Fastest path to model-agnostic agentic coding; ideal omo substrate	github.com/sst/opencode
Spec-Kit	SDD (spec-anchored)	Greenfield 7-step spec → tasks → impl	Python `uv` CLI; branch-per-spec	Any (via host)	2	4	Heavy but rigorous for new builds; ceremony tax on solo brownfield	github.com/github/spec-kit

Tool	Category	Purpose	Execution	Models	Simplicity (1-5)	Execution Power (1-5)	One-line Verdict	Source
BMAD-METHOD	SDD + multi-agent	12+ persona team simulation	`npx bmad-method install`; markdown personas	Any	2	5	Excellent for regulated/team work; overkill for solo vibe-coding	github.com/bmad-code-org/BMAD-METHOD
Kiro	SDD-IDE (vendor)	AWS spec-driven IDE	Standalone IDE	Claude-only (current GA)	3	4	Polished but locked; bets your stack on AWS roadmap	kiro.dev
Tessl	Spec-as-source (vendor)	Specs as canonical, code regenerated	Hosted platform + Spec Registry	Multi	3	4	Conceptually farthest forward; closed beta — wait	tessl.io
Claude Code skills/subagents	Agent harness (vendor)	In-process tool packs + subagents	`.claude/skills/*` + subagent definitions	Claude (Anthropic)	4	5	Best-in-class if you commit to Anthropic's stack	docs.claude.com/.../skills

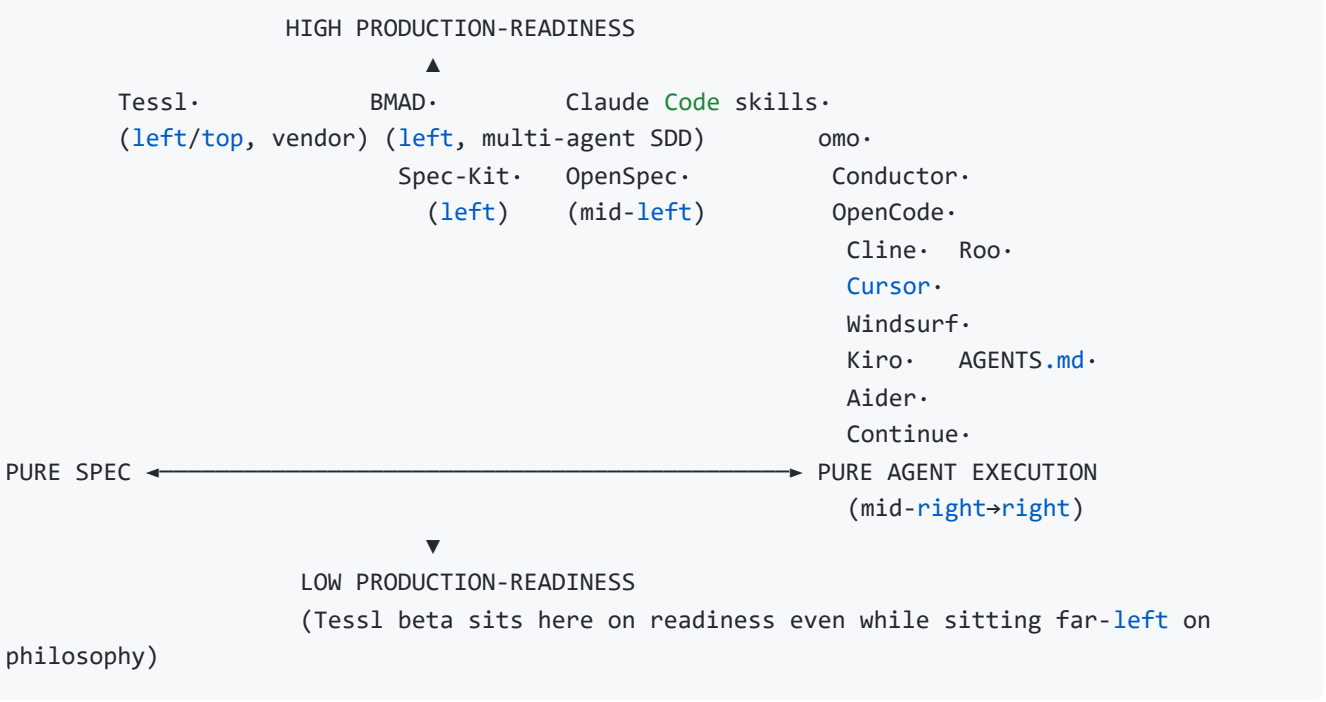
Tool	Category	Purpose	Execution	Models	Simplicity (1-5)	Execution Power (1-5)	One-line Verdict	Source
Cursor rules	Agent-config	`.cursor/rules/*.mdc` with frontmatter	In-editor	Multi	4	3	Strong UX; tied to Cursor pricing	docs.cursor.com/context/rules
Aider CONVENTIONS.md	Agent-config	Plain-text style/architecture file	`aider --read CONVENTIONS.md`	Any	5	4	Minimalist; great pair with OpenSpec	aider.chat/.../conventions
Continue.dev rules	Agent-config	Block-based YAML rules	In-editor + CLI	Any	4	3	Granular scoping; learning curve on YAML schema	docs.continue.dev/.../rules
Cline	Agent harness	OSS autonomous coding agent	VS Code extension; MCP marketplace	Any	4	4	Best balance of OSS execution + UI for solo devs	github.com/cline/cline

Tool	Category	Purpose	Execution	Models	Simplicity (1-5)	Execution Power (1-5)	One-line Verdict	Source
Roo Code	Agent harness	Cline fork with custom modes + diff edits	VS Code extension	Any	4	4	~30% token savings vs Cline via diff editing	github.com/RooCodeInc/Roo-Code
Windsurf	Agent IDE (vendor)	Cascade agent IDE	Standalone IDE	Multi (post-OpenAI)	4	4	Polished but vendor-tied; pricing risk	windsurf.com
AGENTS.md	Standard	Cross-tool rules convention	File at repo root	Any tool that reads it	5	2	Always include — free portability tax-deduction	agents.md
Conductor	Cloud orchestrator	Parallel Claude Code + Codex in worktrees	Mac app	Multi	3	5	Best parallel-agent UX; Mac-only, vendor pricing	conductor.build

Quadrant Placement

X-axis: **pure-spec ←——→ pure-agent-execution**

Y-axis: **production-readiness low ↓——↑ high**



Rationale (selected):

- **OpenSpec** sits mid-left, high readiness — light enough to ship today, opinionated enough to outlast vibes.
- **Spec-Kit / BMAD** sit further left and high — production-ready but heavy.
- **oh-my-openagent + OpenCode + Claude Code skills** sit mid-right to right and high — execution-heavy with strong production signal (active maintenance, large user counts).
- **Tessl** sits far-left but lower on readiness because it is closed-beta.
- **Kiro / Windsurf / Conductor** sit right and high but production-readiness is *conditional on vendor health* — discount one notch for solo indie use.
- **AGENTS.md** sits far-right on the rules subspectrum but low on "execution power" — it's a convention, not an actor.

Top 3 Recommendations for Graham

1. Primary stack: OpenSpec + OpenCode + oh-my-openagent + AGENTS.md (Confidence: HIGH)

This is the maximum-leverage, minimum-lock-in configuration for a solo indie shipping multiple production apps. **Why it fits:** OpenSpec gives you delta-spec discipline that catches drift between "what I told the agent" and "what the codebase now is" — critical when you context-switch across projects. OpenCode is the model-agnostic execution substrate; oh-my-openagent layers Sisyphus-orchestrated multi-agent flow (`ultrawork` for long autonomous runs, Hephaestus for execution, Oracle/Librarian for memory) without forcing you onto Anthropic-only or AWS-only stacks. AGENTS.md at every repo root makes your rules portable to whatever assistant you're using on a given day. **Stack moves:** install once globally (`npm i -g @fission-ai/openspec` , `omo` per the project README), keep one personal AGENTS.md template you copy into each new repo, use OpenSpec change proposals as the unit of work, hand them to omo's Sisyphus to execute. **Tradeoff acknowledged:** omo has a learning curve (52 hooks, 26 tools); plan a half-day to internalize the persona model.

2. Fallback / IDE-bound stack: Claude Code + skills + subagents + OpenSpec (Confidence: MODERATE-HIGH)

If you'd rather stay inside one vendor's polished UX and accept the model lock-in, Claude Code skills + subagents cover roughly the same persona/orchestration ground as omo, with first-party support and Anthropic's strongest models. Add OpenSpec on top — it works inside Claude Code natively — and you get spec discipline without leaving the IDE. **When to choose this:** if your projects are Anthropic-friendly (no Kimi/GLM/Gemini cost optimization), if you value vendor-managed quality over breadth, and if you don't need parallel-worktree orchestration. **Tradeoff:** if Anthropic pricing or rate-limits move against you, your tooling pivot cost is higher than with the OpenCode path.

3. Parallel-projects accelerator: Conductor (Mac) on top of either stack (Confidence: MODERATE)

For the specific pain of "I'm switching across multiple projects in a single afternoon," Conductor materially speeds up parallel agent runs in isolated git worktrees. It composes with both stacks above — it doesn't replace OpenSpec or omo, it runs them in parallel. **When to add:** once your project

count exceeds three actively-maintained repos and context-switching cost dominates. **Tradeoff:** Mac-only, paid, and you take on a vendor whose pricing/availability you don't control. Treat as an accelerant, not a foundation.

What I Would Avoid (for this profile)

- **Kiro** — vendor-locked IDE plus single-vendor model ceiling; the surface area you'd give up is too big for solo indie portability needs.
- **BMAD-METHOD** — beautiful for regulated team work; the persona ceremony is not where solo throughput comes from.
- **Tessl** — promising spec-as-source vision, but closed beta plus single-vendor risk plus regeneration model is not a place to put production apps yet.
- **Spec-Kit as primary SDD layer** — fine to *try* on a brand-new greenfield project; not the default for a brownfield-heavy multi-project solo workflow. The seven-step process and ~800-line specs are a tax you'll feel on every change.

Pre-Mortem (6 months out)

Three plausible failure modes: (1) **OpenSpec abandonment or pivot** — Fission-AI is small; mitigate by treating delta-specs as plain-Markdown artifacts that survive even if the CLI dies. (2) **oh-my-openagent ergonomic collapse** — the persona/hook surface is large; if Graham can't internalize it, fall back to plain OpenCode + Claude Code subagents. (3) **Model pricing shock** — if frontier model costs spike, the OpenCode/omo multi-model design (Kimi, GLM as cheap drafters; Claude/GPT for arbitration) is the precise hedge — don't lose that optionality by going Kiro/Windsurf.

Conclusion

The right answer for a solo indie shipping multiple production "vibe coded" apps is **not a single platform** — it is a *layered stack of OSS, model-agnostic, per-repo* tools: lightweight SDD (OpenSpec) + portable rules (AGENTS.md) + multi-model harness (OpenCode + oh-my-openagent), with Claude Code skills as a polished alternative if vendor-managed UX wins out, and Conductor as an accelerant once parallel-project switching becomes the bottleneck. This stack is defensible because every layer is replaceable independently, every artifact (specs, rules, plans) survives any single tool dying, and every model decision stays under Graham's control. **Why this works:** SDD discipline catches the failure mode that kills vibe-coded production apps (silent intent drift), while a model-agnostic harness

avoids the failure mode that kills solo developers (vendor pricing/availability shocks). **Final confidence: HIGH** that the recommended primary stack outperforms each alternative on Graham's stated criteria; **MODERATE** on absolute claims about specific feature counts given the category's release velocity.

Limitations

- Star counts, version numbers, and feature lists in this category churn weekly; specifics are dated to mid-2026 retrieval.
- Tessl and Kiro feature claims rely partially on vendor sources; treat as directional.
- Production-incident data for any of these tools is sparse industry-wide; recommendations rest on architectural reasoning + community signal rather than benchmarked uptime.

Sources

1. OpenSpec — github.com/Fission-AI/OpenSpec
2. oh-my-openagent — github.com/code-yeongyu/oh-my-openagent
3. Spec-Kit — github.com/github/spec-kit
4. GitHub Blog — Spec-driven development with AI
5. BMAD-METHOD — github.com/bmad-code-org/BMAD-METHOD
6. AGENTS.md standard — agents.md
7. Kiro — kiro.dev
8. Tessl — tessl.io
9. Conductor — conductor.build
10. Cursor rules docs
11. Aider conventions
12. Continue.dev rules docs
13. Cline — github.com/cline/cline
14. Roo Code — github.com/RooCodeInc/Roo-Code
15. Windsurf — windsurf.com
16. Claude Code skills docs
17. Claude Code subagents docs
18. OpenCode — github.com/sst/opencode

19. (https://www.npmjs.com/package/@fission-ai/openspec)
20. GitHub Copilot AGENTS.md customization docs