Spec-Driven Development

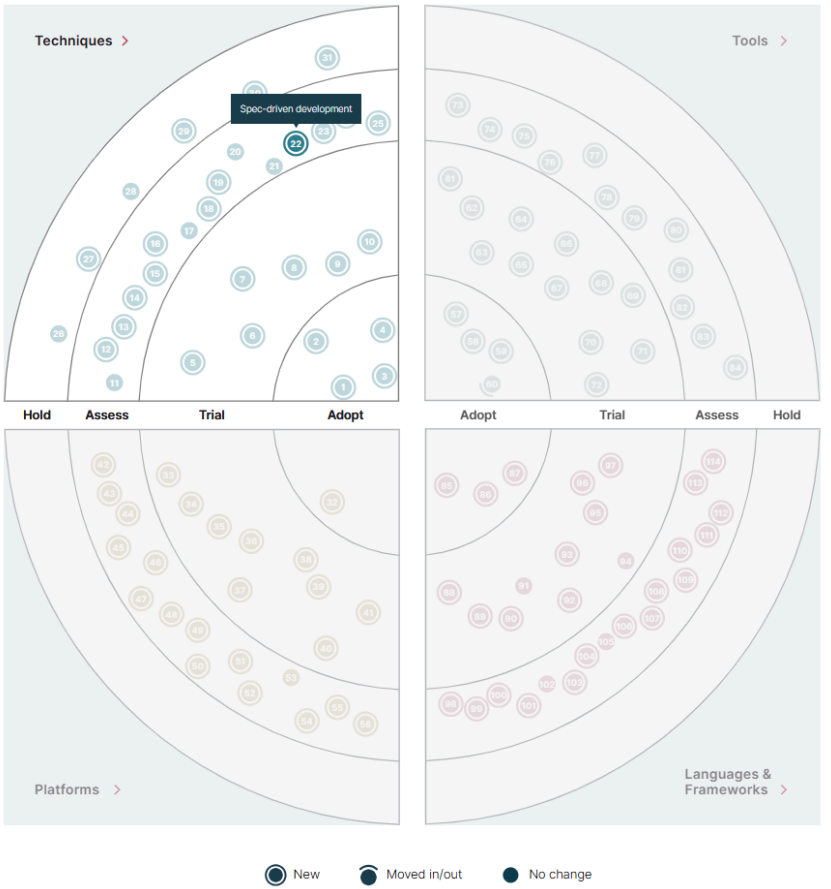
A New Era?

Artem Vegera, 2025-11-25

Agenda

1. Vibe Coding → SDD
2. Spec-Driven Tooling
    - Kiro
    - Spec Kit
    - OpenSpec
3. SDD Tools Comparison
4. Future of SDD

Hot Topic On Radar



https://www.thoughtworks.com/radar

I. Vibe Coding → SDD

Vibe Coding

OVERVIEW

- You open an AI chat
- ”Hey, let’s just build a service real quick…”
- You improvise your way through
- Code emerges from a **conversation**, **not** from a **plan**

Vibe Coding

OVERVIEW

PROS

- Zero friction, instant start
- Great for quick spikes & exploration
- Fast idea-to-code loop

Vibe Coding

OVERVIEW

PROS

CONS

- Context lives only in your chat history
- Must re-explain everything next session
- Process is not reproducible and not team-friendly

Vibe Coding

OVERVIEW

PROS

CONS

WHAT IF?

Two pains:

- Context state
- Change management

What if we move the state from chat to our repo?

Vibes → Context Rules

Context Rules

OVERVIEW

- Store context in markdown-files:
  - `.cursor/rules/context.mdc`
  - `AGENTS.md`
  - ...
- AI loads rule-files by condition (before responding)

https://cursor.com/docs/context/rules

Context Rules

OVERVIEW

PROS

- Context lives **inside the repo**, not in the chat
- Reusable across sessions, agents and team-members
- Reduces the re-explaining loop
- Makes work more predictable

Context Rules

OVERVIEW

PROS

CONS

- Requires manual maintenance
- Easy to forget or miss outdated rules
- **No standard structure** for rules content

Context Rules

OVERVIEW

PROS

CONS

WHAT IF?

Two pains:

- (±) Context state
- (-) Change management

What if we create a dedicated place where all **context and progress** lives and evolves?

Context → Memory Bank

Memory Bank

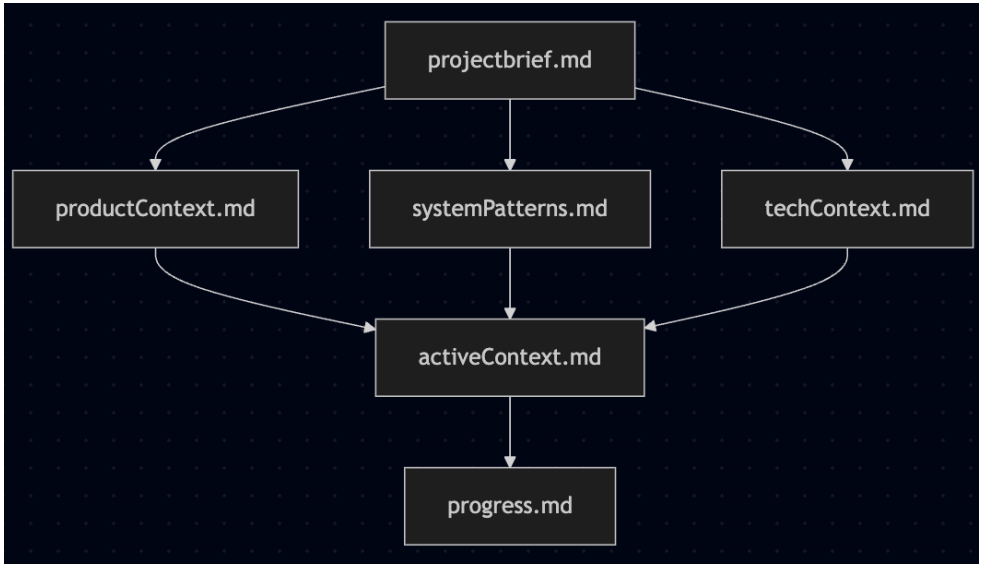
OVERVIEW

- Store evolving context in project:
  - memory.md
  - decisions.md
  - tasks.md
  - ...
- AI reads these files to continue where you left off

Memory Bank

OVERVIEW

WORKFLOW



Memory Bank

OVERVIEW

WORKFLOW

PROS

- **Context** + **tasks** finally moves inside the project, not the chat
- Works across agents and IDEs
- Stable even if the chat is interrupted
- Enables team onboarding: everyone shares the same state

Memory Bank

OVERVIEW

WORKFLOW

PROS

CONS

- Hard to scale across team-members
- Easy to forget or drift out-of-sync
- No unified structure across projects

Memory Bank

OVERVIEW

WORKFLOW

PROS

CONS

WHAT IF?

Two pains:

- (±) Context state
- (–) Change Management

What if context and change management were part of a single, guided workflow?

Memory Bank → SDD

Spec-Driven Development

OVERVIEW

- Specify a ”spec” before code
- Spec is source of truth for devs and AI
- Change Management become first-class project artifacts:
  - changes/
    - proposal.md
    - design.md
    - ...

Spec-Driven Development

OVERVIEW

PROS

- Reproducible development process
- Teams align on a shared spec
- Natural integration with PRs, audits, arch decisions

Spec-Driven Development

OVERVIEW

PROS

CONS

- Needs discipline to keep specs accurate (no silver bullet)
- Bad specs → bad results
- Team must agree on the spec format

Spec-Driven Development

OVERVIEW

PROS

CONS

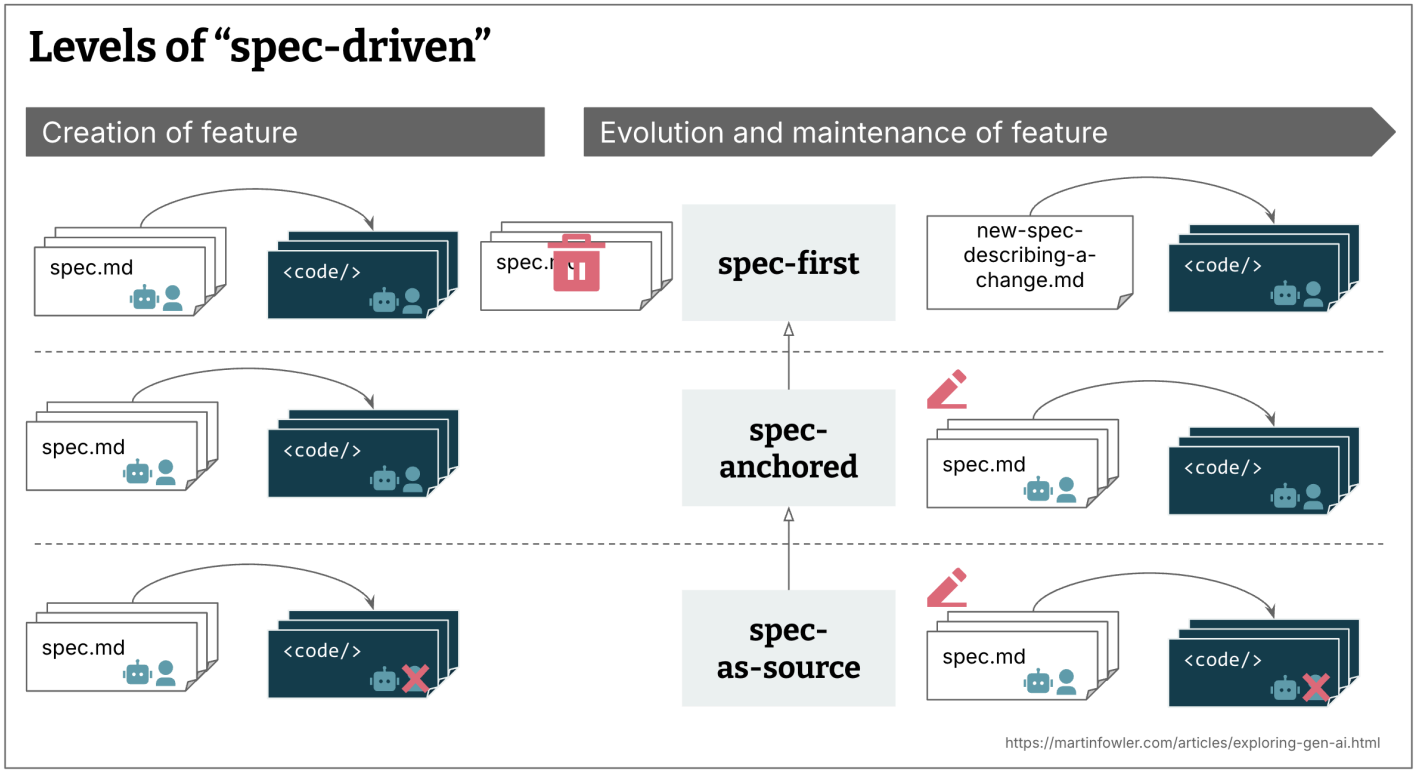
WHAT IF?

Two pains:

- (+) Context state
- (+) Change management

What if there were dedicated tools that guide SDD workflow?

SDD Tooling Levels



II. Spec-Driven Tooling

Kiro

OVERVIEW

July 14, 2025 (Amazon)

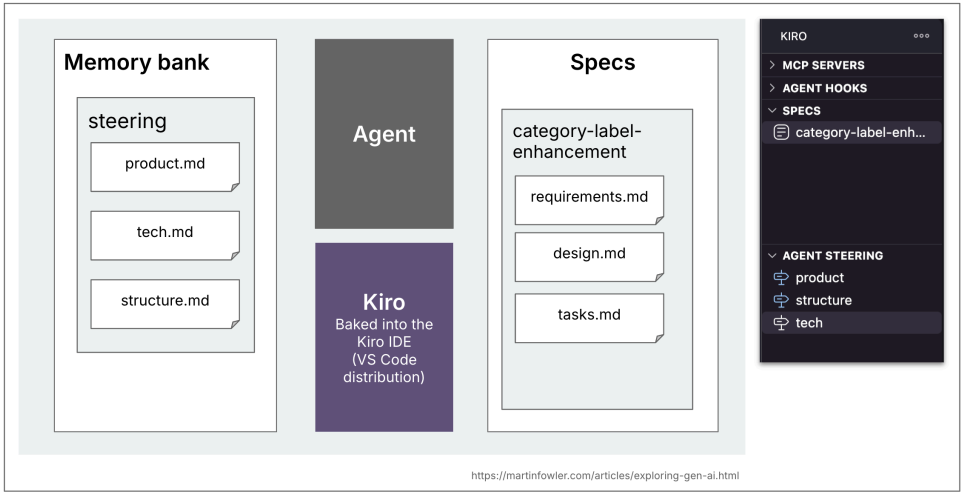
- AI-powered IDE with Spec Mode
- Specs written before implementation
- IDE stores and guides your workflow inside the project

Kiro

OVERVIEW

WORKFLOW

Requirements → Design → Tasks



Kiro

OVERVIEW

WORKFLOW

PROS

- Persistent state inside the IDE
- Guides you step-by-step through a SDD-process
- Structured, controlled workflow
- Requirements can be refined and regenerated

Kiro

OVERVIEW

WORKFLOW

PROS

CONS

- Proprietary / vendor lock-in
- Works only inside Kiro IDE

Kiro

OVERVIEW

WORKFLOW

PROS

CONS

WHAT IF?

Kiro is **IDE-locked** and not portable for teams with diverse tooling

What if we introduce **IDE-agnostic** approach?

Kiro → Spec Kit

Spec Kit

OVERVIEW

September 2, 2025 (GitHub)

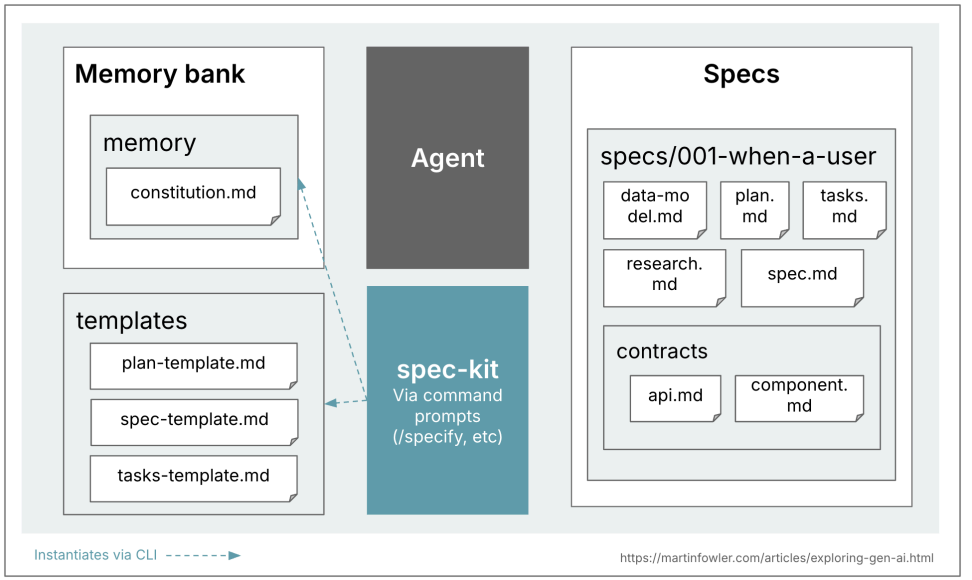
- A set of scripts and templates stored in the repo
- Workflows for implementing features
- Steps can be executed, repeated, automated
- Works in any IDE or with any AI agent

Spec Kit

OVERVIEW

WORKFLOW

Specify → Plan → Tasks → Implement



Spec Kit

OVERVIEW

WORKFLOW

PROS

- IDE-agnostic & portable
- Highly reproducible

Spec Kit

OVERVIEW

WORKFLOW

PROS

CONS

- Highly **verbose** spec output
- Even for simple changes:
  - **token-intensive**
  - **time-consuming**
- Very opinionated flow (separate branches, etc.)
- Not active project support

Spec Kit

OVERVIEW

WORKFLOW

PROS

CONS

WHAT IF?

Spec Kit feels **too heavy** and **opinionated** even for simple projects.

What if specs could stay **lightweight**, **flexible**, and become first-class citizens?

Spec Kit → OpenSpec

OpenSpec

OVERVIEW

September 6, 2025 (Fission AI)

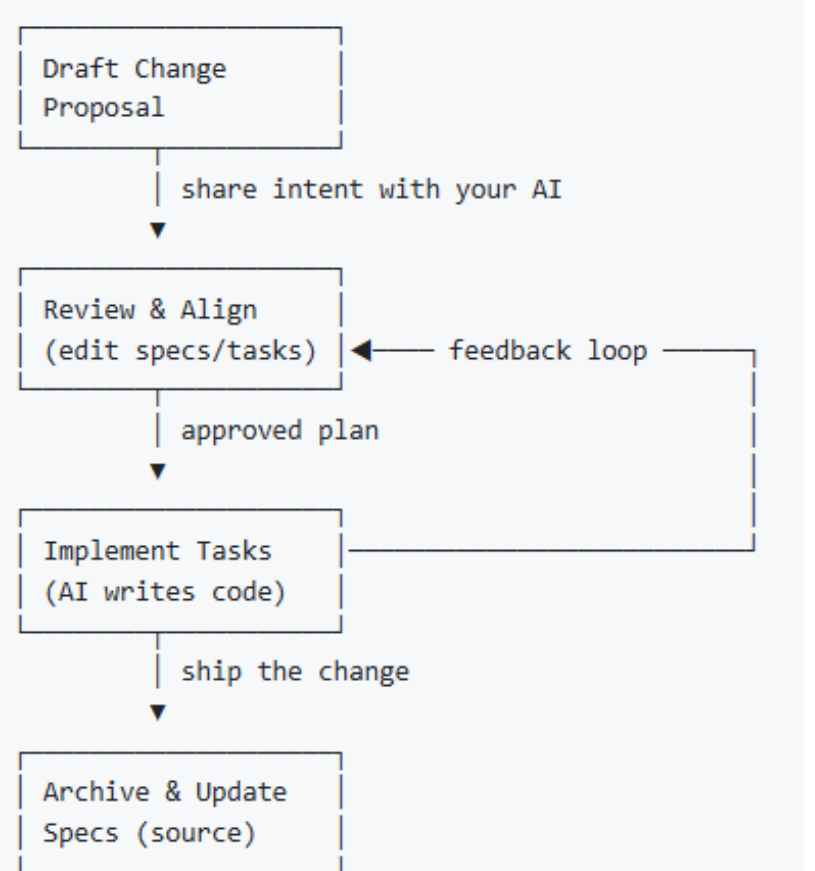
- Set of scripts + templates in your repo
- Splits **specs** and **change management** into separate workflows/dirs:
  - changes/
  - specs/
- Change output is **diff** for specification

OpenSpec

OVERVIEW

WORKFLOW

Proposal → Apply → Archive



OpenSpec

OVERVIEW

WORKFLOW

PROS

- Agile-friendly workflow
- Supports a lot of AI agents
- Easy to customize md-templates
- Works great for pull-requests review
- Finally, spec is first class citizen

OpenSpec

OVERVIEW

WORKFLOW

PROS

CONS

- Not extensible workflow engine (no custom steps)

OpenSpec

OVERVIEW

WORKFLOW

PROS

CONS

WHAT IF?

- Kiro — IDE-native
- Spec Kit — heavy structured
- OpenSpec — lightweight

What if we could take the best from all three — and build something: **lightweight**, **extensible**, and **universal**?

III. SDD Tooling Comparison

Tooling Steps - Mapping

Action	Kiro (IDE)	Spec Kit	OpenSpec
Init	N/A	`init`	`init`
Change	Requrements	`specify`	`proposal`
Tech Plan	Design	`plan`	N/A
Tasks List	Tasks List	`tasks`	N/A
Implement	Start task	`implement`	`apply`
Archive	N/A	N/A	`archive`

Tool Comparison

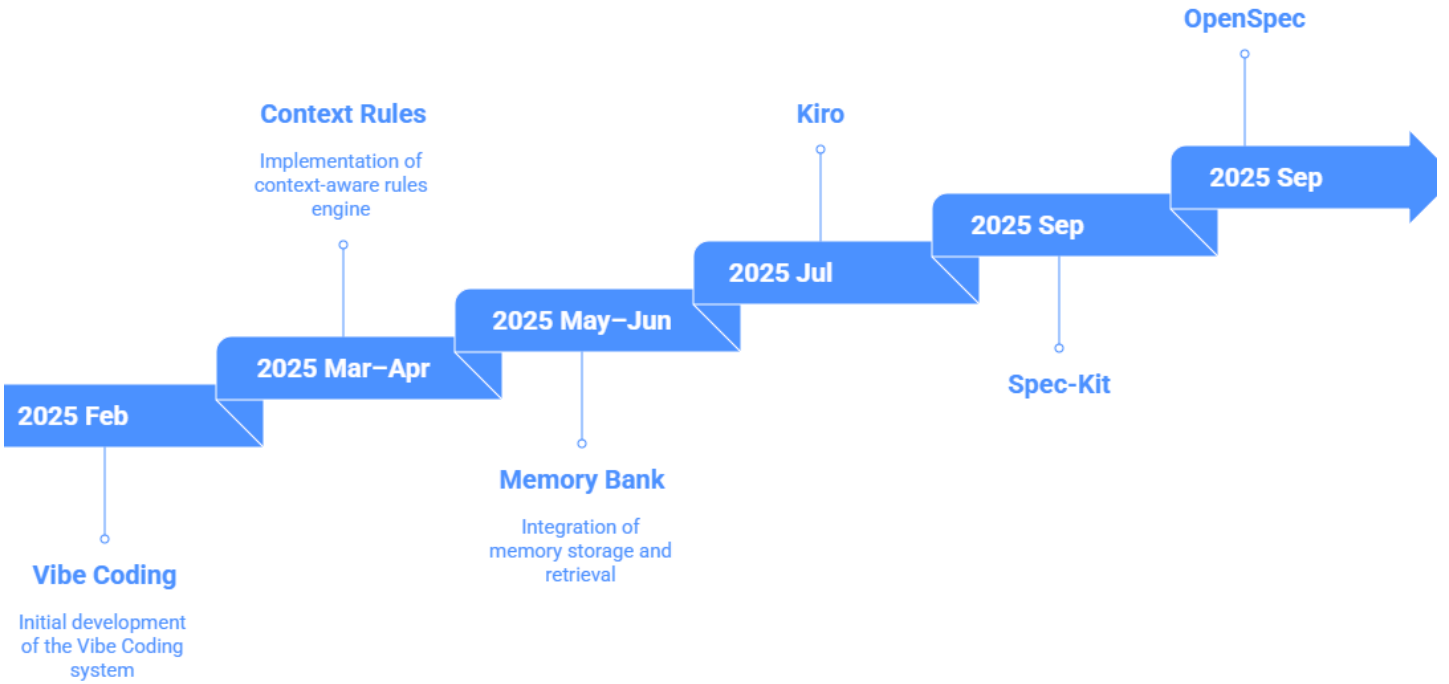
	Vibe	Kiro	Spec Kit	OpenSpec
State	Chat	Files	Files	Files
IDE-lock	No	Yes	No	No
Strictness	Low	Medium	High	Medium
Dogfooding	-	?	No	Yes
Team-ready	No	Part	May be	Yes

When It Doesn't Work?

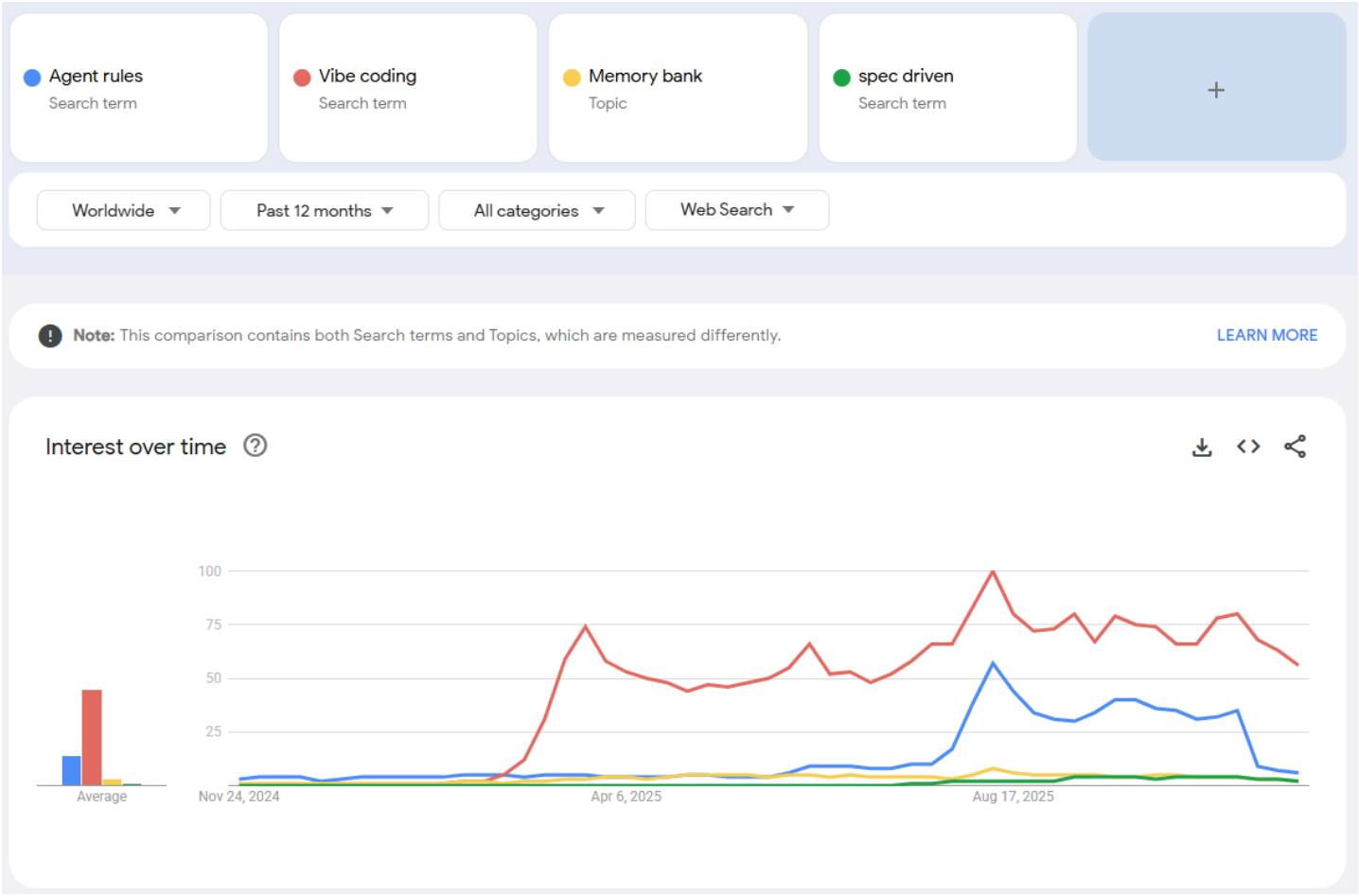
When to Use What

- **Vibe coding** → exploration
- **Agent Rules** → always
- **Memory bank** → deprecated as process flow
- SDD Tooling:
  - **Kiro** → solo work inside a single IDE
  - **Spec Kit** → not ready for prod, overengineering
  - **OpenSpec** → good choice for existing projects

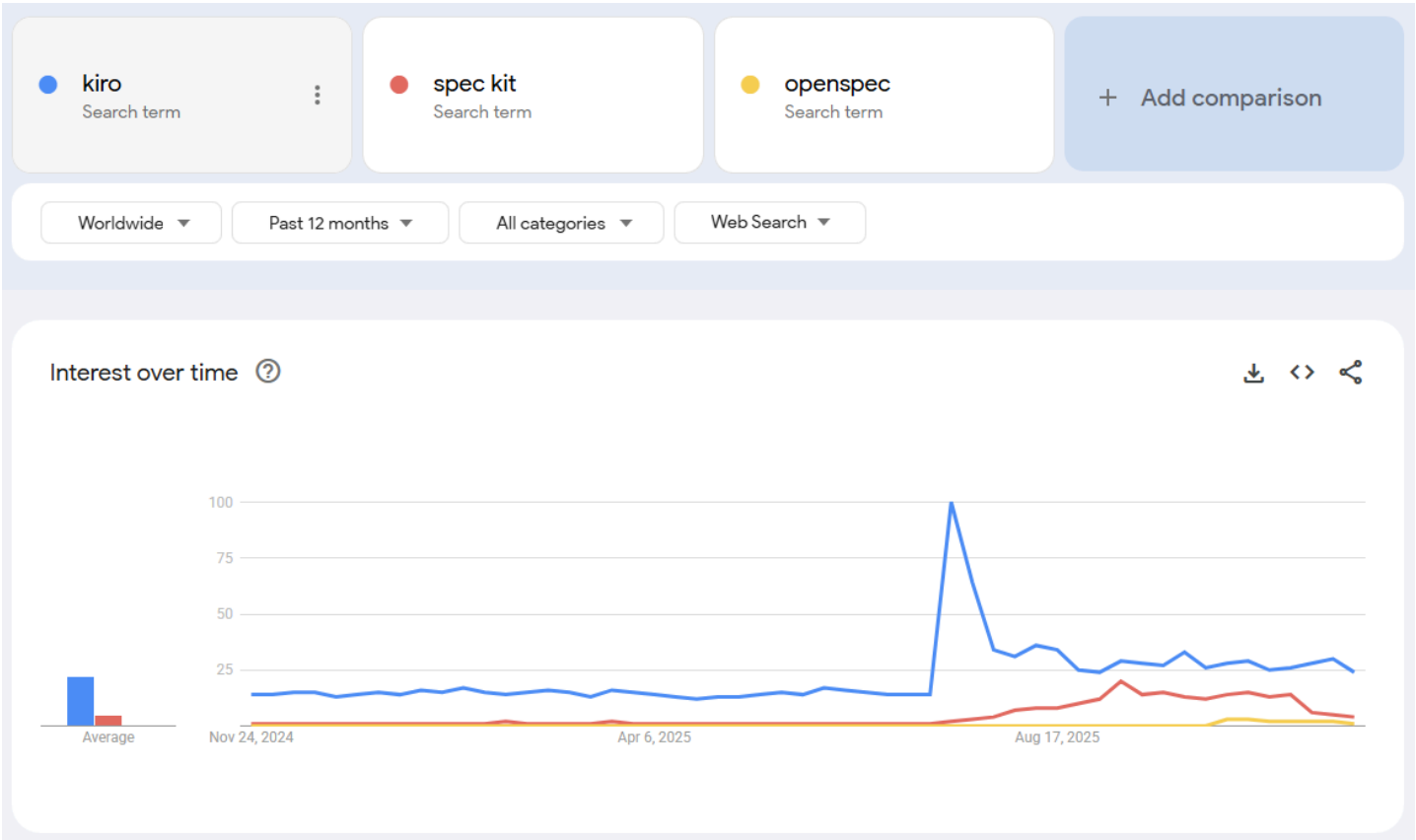
SDD - Year 2025



Trends: Approaches



Trends: Tooling



Future

- Steps **standardization**
- **Customization** of:
  - Dev workflows
  - Spec formats

Who is Next?

Ticket-systems as code?

Question Time :)