SPEC-DRIVEN DEVELOPMENT

Specky turns the discipline of specifications into a programmable enforcement engine.

An open-source MCP server with 53 validated tools and a 10-phase pipeline that takes any input to production code, deterministically, in any AI IDE.

AUTHOR
Paula Silva

ROLE
Software Global Black Belt

DURATION
45 to 60 minutes

DATE
2026-06-01

AGENDA

Five parts, one argument.

PART I	The problem, why vibe coding produces fast code and slow teams
PART II	The engine, what Specky is and how it enforces determinism
PART III	The pipeline, ten phases, EARS requirements, and human review gates
PART IV	The ecosystem, any input, any IDE, no vendor lock-in
PART V	The decision, enterprise readiness and where to start Monday

PART

I

The problem.

AI writes code in seconds. But fast is not the same as correct, and nobody wrote down what correct meant.

THE COST OF SKIPPING THE SPECIFICATION

40%

of engineering time goes to rework when requirements were never written down.

You ask the assistant to build a login system. It generates code immediately, skips the requirements, guesses the architecture, and produces something that runs but matches nobody's intent. There is no way to verify the code against an intent that was never captured.

WHAT THEY CALL IT		WHAT IT PRODUCES
vibe coding	→	code without intent

TWO WAYS TO BUILD WITH AI

The fix is not a better prompt. It is a validated pipeline between intent and code.

VIBE CODING

Code from vibes

- The model generates code straight from a sentence.
- Requirements are never captured, acceptance criteria never defined.
- No way to verify the result against the original intent.
- Works in a demo, breaks in review and in production.

DETERMINISTIC DEVELOPMENT

Code from validated specs

- A structured specification describes what the system must do, first.
- Every requirement is validated before a line of code is written.
- Spec, design, and tasks stay aligned and traceable.
- The AI is the operator. Specky is the engine.

PART

II

The engine.

Specky reimplements the Spec-Kit methodology as 53 enforceable tools, with programmatic validation instead of good intentions.

WHAT SPECKY IS, IN THREE NUMBERS

An open-source MCP server that channels AI creativity through a validated pipeline.

VALIDATED MCP TOOLS

53

Across thirteen categories: input conversion, pipeline core, quality gates, diagrams, infrastructure, testing, and export.

PIPELINE PHASES

10

A state machine that blocks phase-skipping. You cannot jump from Init to Design without completing Specify first.

SUPPORTED IDES

any

Works in any AI IDE that speaks MCP. Use it in VS Code with GitHub Copilot or the GitHub Copilot CLI. One npx command.

THE 53 TOOLS, ORGANIZED

Fifty-three tools, thirteen categories, grouped into the seven families you actually touch.

01 · INPUT CONVERSION

Turns transcripts, documents, designs, and raw text into clean Markdown the pipeline can read.

02 · PIPELINE CORE

The ten phase tools, from sdd_init to sdd_release, that drive the state machine forward.

03 · QUALITY GATES

The EARS validator and cross-artifact analysis that block a phase until the work holds together.

04 · DIAGRAMS

Generators for seventeen software engineering diagram types, from C4 to sequence flows.

05 · INFRASTRUCTURE

Routing to Terraform and Docker so the spec reaches real environments, not just a document.

06 · TESTING

Tools that tie every requirement to a test, so verification can prove the code matches the spec.

07 · EXPORT

Routing to GitHub, Azure DevOps, and Jira, so the plan lands as issues and pull requests.

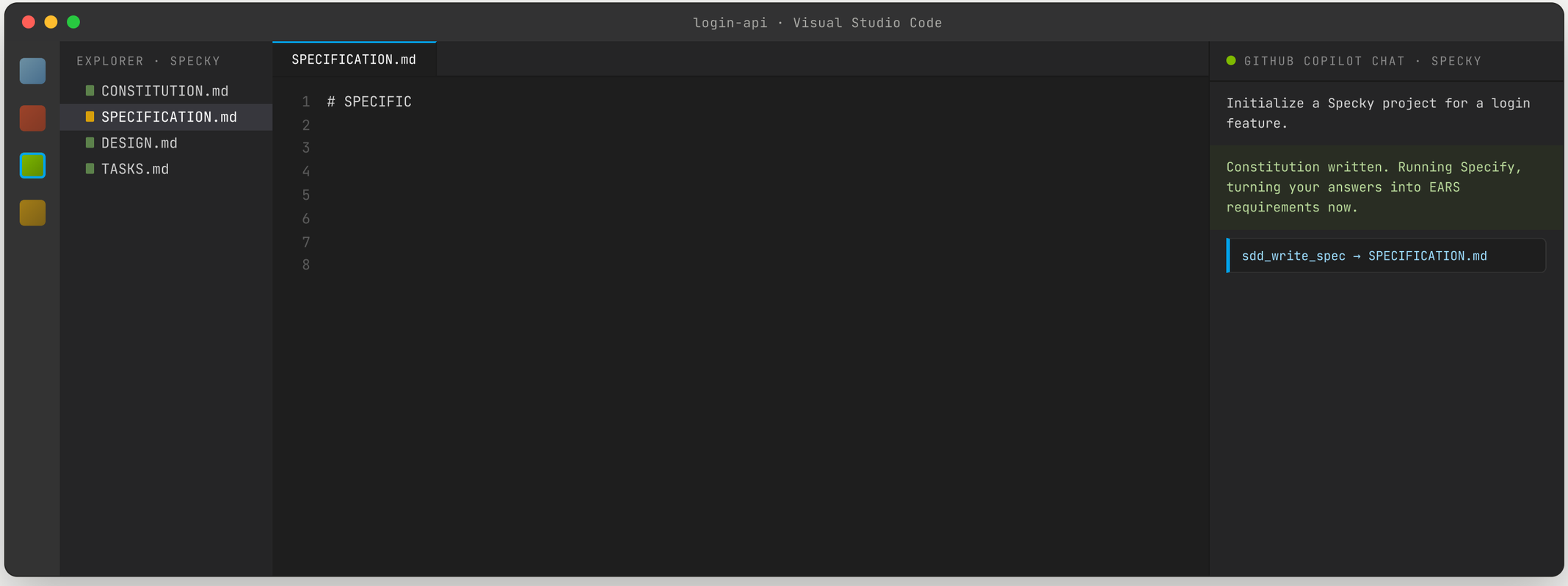
53

tools in all, each with strict input validation. You never call them by hand, the pipeline does.

You do not memorize tools. You move through phases, and each phase calls the tools it needs.

SIMULATION · SPECKY INSIDE VS CODE

It lives in the editor you already use. Here it writes a spec, live.



Illustrative reconstruction. Same GitHub Copilot Chat, same files, the requirements are validated as they are written.

METHODOLOGY AND ENGINE

Spec-Kit defines what to do. Specky enforces how to do it.

SPEC-KIT, THE METHODOLOGY

Prompt templates and agent definitions

EARS notation, gated phases, the constitution model, and Markdown templates. The AI reads them and tries to follow. Lightweight, ideal for learning Spec-Driven Development.

SPECKY, THE ENGINE

Tools that validate, enforce, and generate

A state machine, an EARS regex validator, and Zod schemas turn the methodology into enforcement. Install Specky with one command and the full Spec-Kit methodology comes built in.

No separate install of Spec-Kit is needed. The methodology ships inside the engine.

FOUR MECHANISMS THAT MAKE DETERMINISM REAL

Determinism is not a promise. It is enforced by code that refuses to let the pipeline drift.

MECHANISM 01

State machine

Phase enforcement

- Ten mandatory phases, no skipping
- Required files gate advancement

MECHANISM 02

EARS validator

Requirement quality

- Every requirement checked against 6 patterns
- Vague terms like fast or good are flagged

MECHANISM 03

Cross-artifact analysis

Alignment scoring

- Spec, design, and tasks checked for consistency
- Orphaned requirements flagged instantly

MECHANISM 04

MCP-to-MCP routing

No vendor lock-in

- Outputs structured JSON for any client
- Routes to GitHub, Azure, Jira, Terraform

PART

III

The pipeline.

Ten phases. No skipping. Human review at every gate. Every phase leaves a traceable artifact behind.

FROM INTENT TO RELEASE, ONE PHASE AT A TIME

The state machine turns a vague idea into a documented, reproducible journey.

01
Init
CONSTITUTION.md

02
Discover
7 questions

03
Specify
SPECIFICATION.md

04
Clarify
decision tree

05
Design
DESIGN.md

06
Tasks
TASKS.md

07
Analyze
ANALYSIS.md

08
Implement
checkpoints

09
Verify
VERIFICATION.md

10
Release
PR + work items

Frame the project · Specify and design · Plan and audit · Build, verify, ship

SIMULATION · THE STATE MACHINE RUNNING

Press go, and the ten phases execute in order, each leaving an artifact.

01 ✓ **Init** CONSTITUTION.md	02 ✓ **Discover** 7 questions	03 ✓ **Specify** SPECIFICATION.md	04 ✓ **Clarify** decision tree	05 ✓ **Design** DESIGN.md
06 ✓ **Tasks** TASKS.md	07 ✓ **Analyze** ANALYSIS.md	08 ✓ **Implement** checkpoints	09 ✓ **Verify** VERIFICATION.md	10 ✓ **Release** PR + work items

LGTM Paused after Specify. The machine will not advance to Design until you review and reply LGTM.

Each phase turns green only when its artifact exists and its gate passes. Skipping is impossible by design.

PHASE 01 OUTPUT, UP CLOSE

Every project opens with a constitution. This is what it pins down.

The constitution is the project's rulebook. The AI reads it before every phase, so the same principles, stack, and quality bars apply from the first requirement to the final pull request.

01 Principles the code must honor, like accessibility and privacy by default.

02 The technical stack, so the AI never guesses your language or framework.

03 Quality gates, the bar that every later phase is held to.

CONSTITUTION.MD

```
# Principles
- Accessibility is not optional. WCAG 2.2 AA.
- No personal data leaves the tenant.

# Stack
- TypeScript, React, PostgreSQL.
- Tests in Vitest. Lint with ESLint.

# Quality gates
- Every requirement maps to a test.
- No phase advances with an open question.
```

Illustrative sample. Write it once, and every phase inherits these rules automatically.

EARS NOTATION, THE SIX REQUIREMENT PATTERNS

Every requirement fits one of six shapes. The validator rejects anything that does not.

Ubiquitous	The system shall...	The system shall encrypt all data at rest. Always true, no trigger.
Event-driven	When [event], the system shall...	When a user submits login, the system shall validate credentials.
State-driven	While [state], the system shall...	While offline, the system shall queue requests.
Optional	Where [condition], the system shall...	Where 2FA is enabled, the system shall require a one-time code.
Unwanted	If [condition], then the system shall...	If the session expires, then the system shall redirect to login.
Complex	While [state], when [event]...	While in maintenance, when a request arrives, the system shall queue it.

EARS IN PRACTICE

The same requirement, rejected and accepted.

REJECTED BY THE VALIDATOR

"The login should be fast and secure."

- Fast names no number, so no test can check it.
- Secure has no trigger and no condition.
- It fits none of the six EARS shapes.

ACCEPTED BY THE VALIDATOR

"When a user submits login, the system shall respond within 500 ms. If credentials fail three times, then the system shall lock the account for 15 minutes."

- Event-driven shape, with a measurable 500 ms.
- An unwanted-behavior shape for the lockout.
- Every clause is now a test you can write.

The validator does not make you a better writer. It refuses to let an untestable sentence become a requirement.

ONE FEATURE, THROUGH THE PIPELINE

Watch a single login feature move from one sentence to a pull request.

Discover 7 questions	Seven questions surface the unspoken: SSO or password, lockout policy, session length, audit needs.
Specify SPECIFICATION.md	Each answer becomes an EARS requirement: REQ-001 validate credentials, REQ-002 lock after three fails.
Design DESIGN.md	The architecture: an auth service, a rate limiter, a session store, each traced back to a requirement.
Tasks TASKS.md	An ordered, checkable plan. Each task points at the requirement it satisfies and the test it needs.
Verify and release PR + work items	Verification confirms the code matches the spec, then opens a pull request with the work items attached.

One sentence in, a traceable trail out. Every box above is a real file a teammate can read next year.

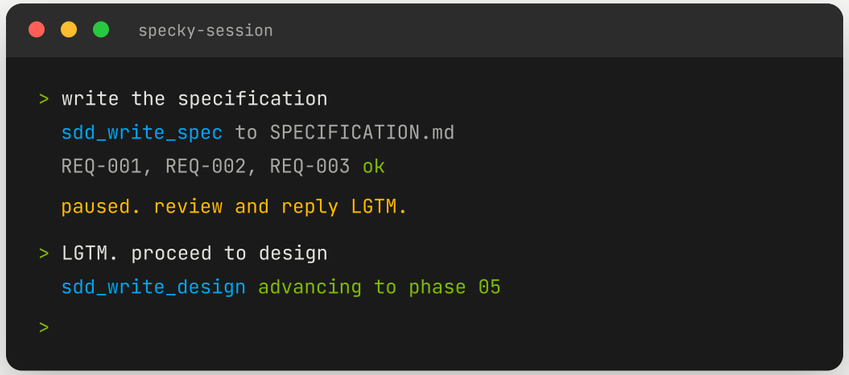
THE HUMAN STAYS IN THE LOOP

After every major phase, the pipeline pauses and waits for you to type LGTM.

Specify, Design, and Tasks each end at a review gate. The AI does not advance on its own. If verification later detects drift between the spec and the code, Specky routes you back to fix the divergence before proceeding.

3 review gates

0 phases skipped



PART

IV

The ecosystem.

Any input becomes a spec. Any IDE runs it. Specky routes work to the tools you already use, with no lock-in.

SIX WAYS TO START

Whatever you already have becomes the starting point of a specification.

01 · NATURAL LANGUAGE

A prompt in chat

Type the idea directly. sdd_init and sdd_discover structure it. Best for greenfield.

02 · MEETING TRANSCRIPT

VTT, SRT, TXT, MD

Extracts decisions, action items, and raw requirements from Teams, Zoom, or Meet.

03 · DOCUMENTS

PDF, DOCX, PPTX

Import RFPs, requirement docs, and decks. Converts to Markdown and feeds the pipeline.

04 · FIGMA DESIGN

Design to spec

Reads components and interactions through the Figma MCP. Best for design-first work.

05 · CODEBASE SCAN

Brownfield context

Detects language, framework, and structure before any new feature is specified.

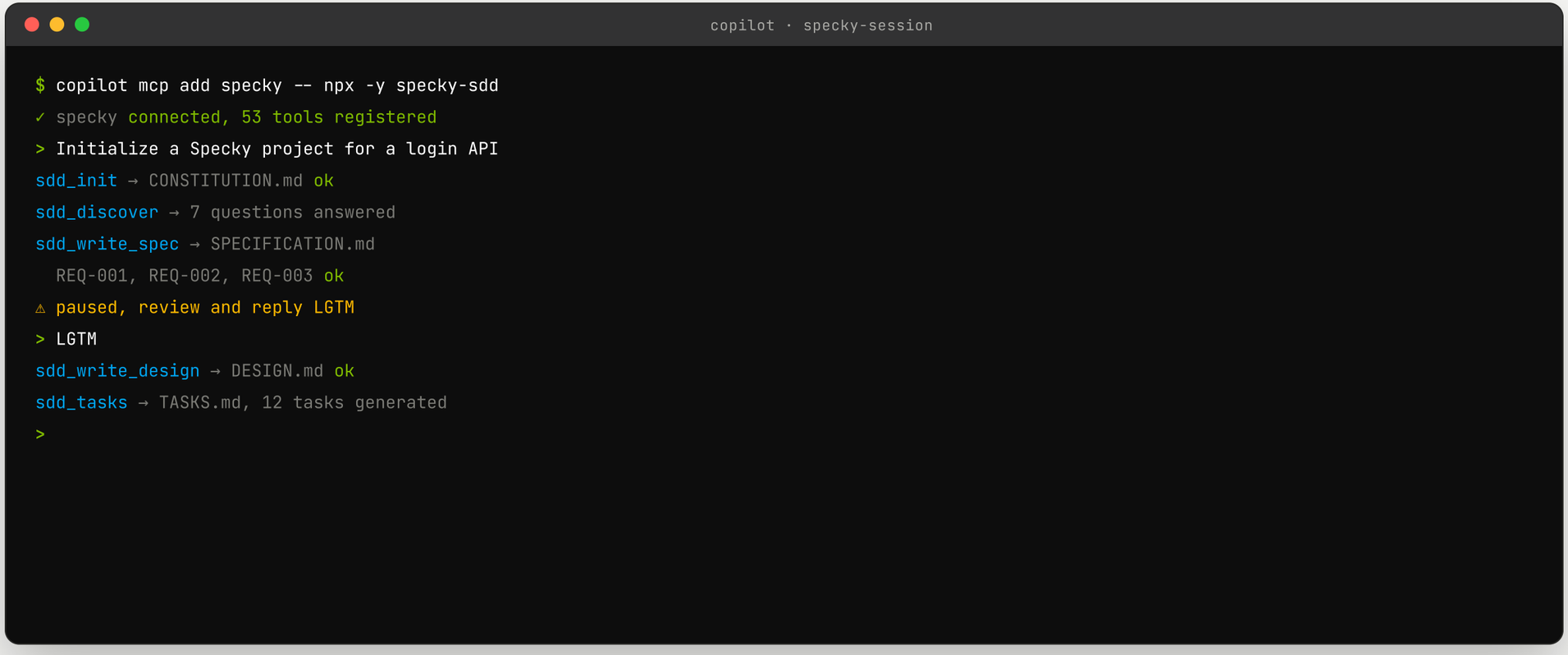
06 · RAW TEXT

Paste anything

No file needed. Paste a client email or notes and every import tool accepts it directly.

SIMULATION · THE SAME RUN IN THE GITHUB COPILOT CLI

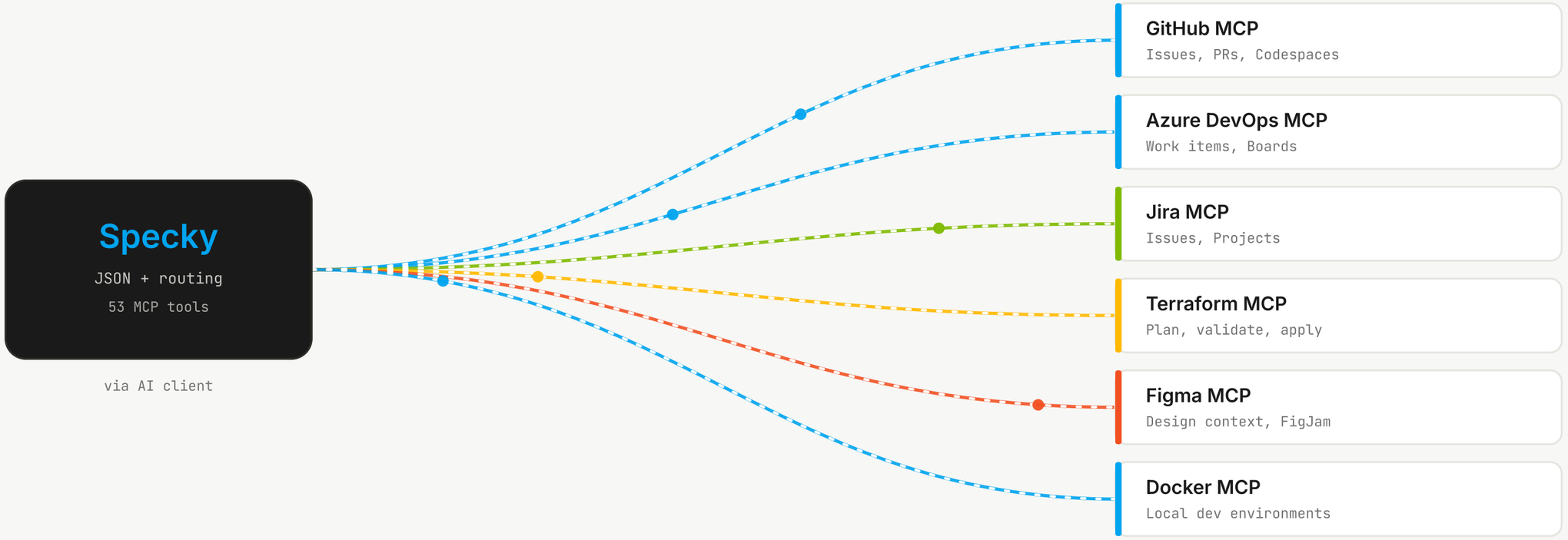
No IDE needed. The terminal runs the identical pipeline, line by line.



Illustrative reconstruction. Same tools, same artifacts, same LGTM gate. Only the surface changed.

MCP-TO-MCP ARCHITECTURE

Specky emits structured JSON. Your client routes it to the tools you already pay for.



Two runtime dependencies. Zero outbound network requests from Specky itself. Your data stays local.

ONE PIPELINE, THREE STARTING POINTS

The pipeline is the same. What changes is where you begin.

GREENFIELD

Start from scratch

A new application, no existing code. Init, discover with seven questions, specify, design, and ship.

BROWNFIELD

Add to existing code

Scan the codebase first. The discovery questions then carry your real stack as context, and drift checks keep code and spec aligned.

MODERNIZATION

Assess and upgrade legacy

Scan, batch-import old docs and transcripts, run compliance checks, and generate migration artifacts with rollback runbooks.

PART

V

The decision.

Why a deterministic SDD layer matters for an enterprise, and where to start on Monday.

BUILT FOR ENTERPRISE ADOPTION

Governance and a minimal attack surface, validated by tests, not by marketing.

COMPLIANCE FRAMEWORKS

HIPAA	Healthcare, 6 controls
SOC 2	SaaS and cloud, 6 controls
GDPR	EU data, 5 controls
PCI-DSS	Card handling, 6 controls
ISO 27001	Security management, 6 controls

SECURITY POSTURE

- Two runtime dependencies only. Minimal attack surface.
- Zero outbound network requests. All data stays local.
- No eval or dynamic code execution. Path traversal blocked.
- Strict schema validation on every tool input.
- **292 unit tests, 89 percent coverage, enforced on every push.**

THE SDD MARKET ANALYSIS 2026

SDD workflows are at parity. The enterprise platform around the layer is the differentiator.

GITHUB AND MICROSOFT

4.25

Enterprise readiness score, out of 5.0

WORKFLOW PARITY

4 : 4

PLATFORM DECIDES

KIRO, AWS

2.85

Enterprise readiness score, out of 5.0

Security, governance, multi-model freedom, and compliance are where the gap opens. Specky is the open engine that makes that layer real.

WHERE SPECKY IS GOING

Shipping today, planned next, and on the horizon.

V3.0, CURRENT	Enriched interactive responses, 17 software engineering diagram types, a 12-section C4 design template, and strengthened gate enforcement at every phase transition.	shipping
V2.4, PLANNED	Token-based HTTP authentication, OpenTelemetry observability, internationalized spec templates, and AI-powered shrinking feedback into spec refinement.	next
V3.1 AND BEYOND	Role-based access control, a persistent audit log, rate limiting, SSO and SAML, multi-tenant workspaces, and a specification quality analytics dashboard.	horizon

THE VOCABULARY

Six terms that unlock everything else in this talk.

SDD	Spec-Driven Development	Write and validate the specification first, then let the AI build against it.
EARS	Easy Approach to Requirements Syntax	Six sentence shapes that make every requirement testable, or rejected.
MCP	Model Context Protocol	The open standard that lets Specky run inside any AI IDE and route to other tools.
Constitution	The project rulebook	Principles, stack, and quality gates the AI honors in every phase.
Drift	Spec and code diverge	When the code stops matching the spec. Verification catches it and routes you back.
LGTM gate	The human checkpoint	After a major phase the pipeline pauses until a person types LGTM to proceed.

HANDS-ON, TODAY

Five steps from install to your first validated spec.

STEP 01	Add Specky to your AI IDE with one npx command. No new editor, no migration.
STEP 02	Ask the AI to initialize a project. It writes the constitution with you.
STEP 03	Answer the seven discovery questions. Your feature idea becomes structured intent.
STEP 04	Let it write the specification, then read it and reply LGTM at the review gate.
STEP 05	Continue through design and tasks. You now have a traceable spec, before any code.

Ten minutes to a validated specification. The first time is the slowest, and it is still faster than the rework you skip.

CLOSING

Specifications reward engineering discipline.

The fun name, the serious engine. The AI is the operator. Specky is the engine.

CONTACT

Paula Silva
Software Global Black Belt

paulasilva@microsoft.com

GET STARTED

Open source, MIT
Specky · open source

`npm: specky-sdd`

INSTALL SPECKY, START MONDAY

```
# GitHub Copilot CLI, inside the repo
copilot mcp add specky -- npx -y specky-sdd

# VS Code with GitHub Copilot
# add specky to .vscode/mcp.json

# then, in the AI chat
> Initialize a Specky project
> and help me define the scope
```