An introduction to Spec-Driven Development

Giovanni Rosa

Universidad Rey Juan Carlos, Spain

26-02-2026
Campus Fuenlabrada,
Universidad Rey Juan Carlos

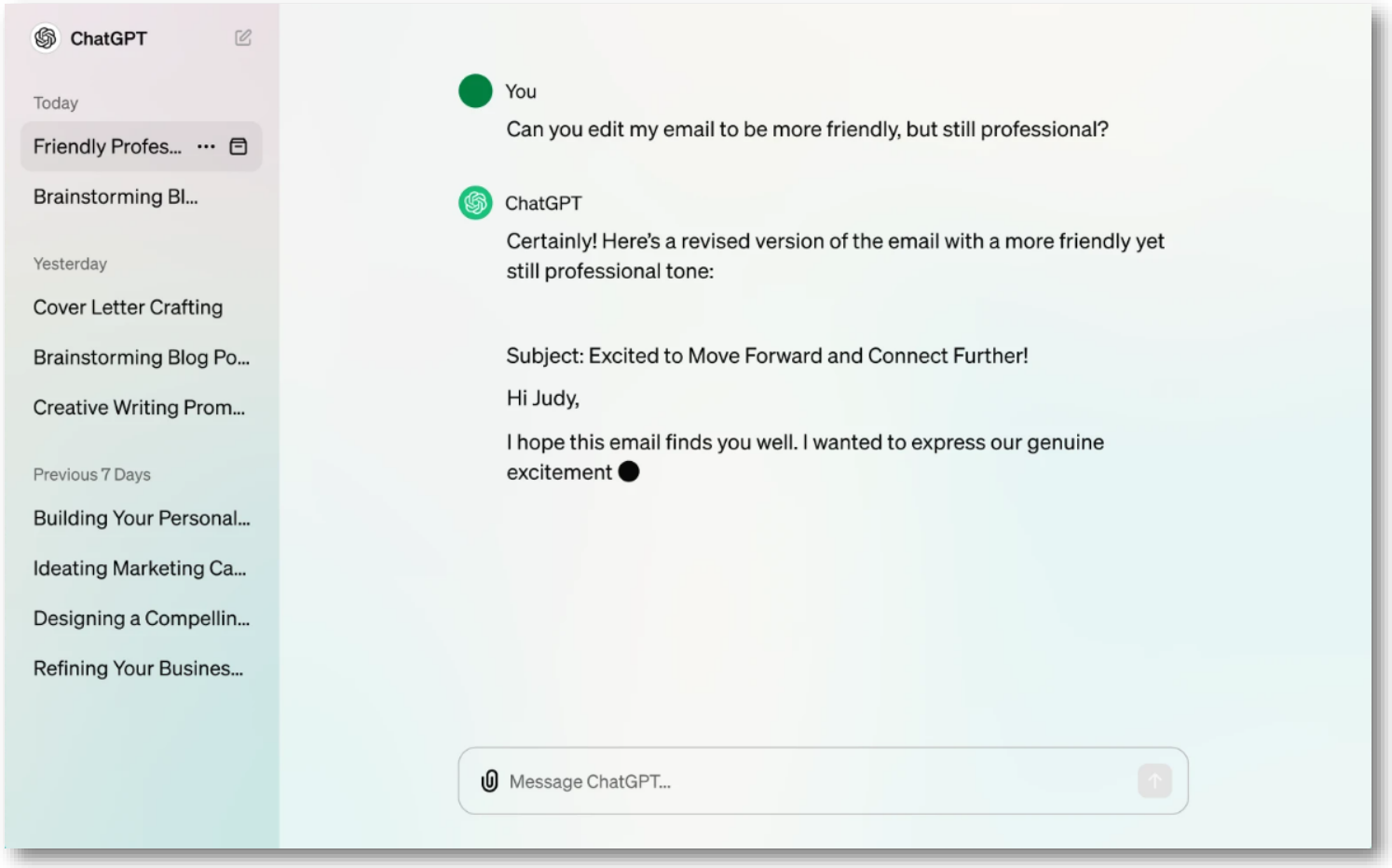
Today’s Agenda

1. Introduction
2. Spec-Driven Development
3. SDD with Claude Code
4. SDD with Spec Kit

November 2022

ChatGPT Assistant

by OpenAI



GitHub Copilot

JS test.js 1 ●

JS test.js > calculateDaysBetweenDates

```
1   function calculateDaysBetweenDates(begin, end) {
        var beginDate = new Date(begin);
        var endDate = new Date(end);
        var days = Math.round((endDate - beginDate) / (1000 *
        return days;
    }
2
```

add typings and pydoc to this function

GitHub Copilot

3 steps completed successfully

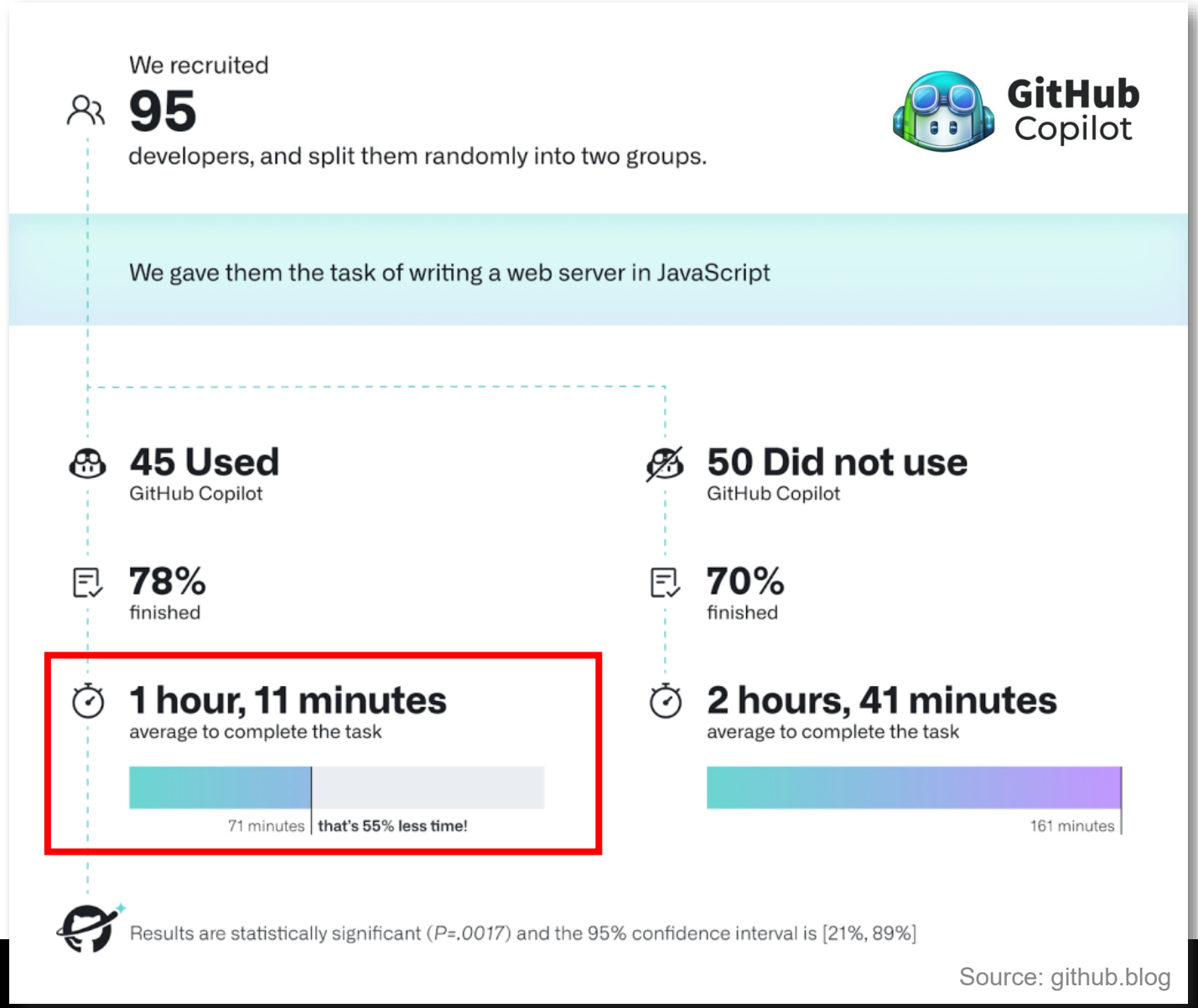
```python
def create_worker_agent(llm: ChatOpenAI, tools: List, system_prompt: str) -> AgentExecutor:
    """
    Create a worker agent with the given language model, tools, and system prompt.

    Args:
        llm (ChatOpenAI): The language model to use for the agent.
        tools (List): A list of tools that the agent can use.
        system_prompt (str): The system prompt to initialize the agent with.

    Returns:
        AgentExecutor: An executor for the created agent.
    """
    # Each worker node will be given a name and some tools.
    prompt = ChatPromptTemplate.from_messages(
        [
            ("system", system_prompt),
            MessagesPlaceholder(variable_name="messages"),
            MessagesPlaceholder(variable_name="agent_scratchpad"),
        ]
    )
    agent = create_openai_tools_agent(llm, tools, prompt)
    executor = AgentExecutor(agent=agent, tools=tools)

    return executor
```

Using 4 References

```python
def create_worker_agent(llm, tools, system_prompt):   You, Moments ago • Uncommitted changes
    # Each worker node will be given a name and some tools.
    prompt = ChatPromptTemplate.from_messages(
        [
            (
                "system",
                system_prompt,
            ),
            MessagesPlaceholder(variable_name="messages")
```

-55%

Time to task



Source: github.blog

Building a C compiler with a team of parallel Claudes

Published Feb 05, 2026

We tasked Opus 4.6 using agent teams to build a C Compiler, and then (mostly) walked away. Here's what it taught us about the future of autonomous software development.

Written by Nicholas Carlini, a researcher on our Safeguards team.

I've been experimenting with a new approach to supervising language models

IBM stock crash: How a single blog post wiped $30 billion off IBM's market value in one afternoon - here's what rattled investors

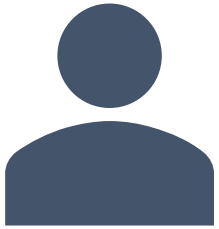
By Piyush Shukla, Global Desk • Last Updated: Feb 24, 2026, 05:26:00 PM IST

Synopsis

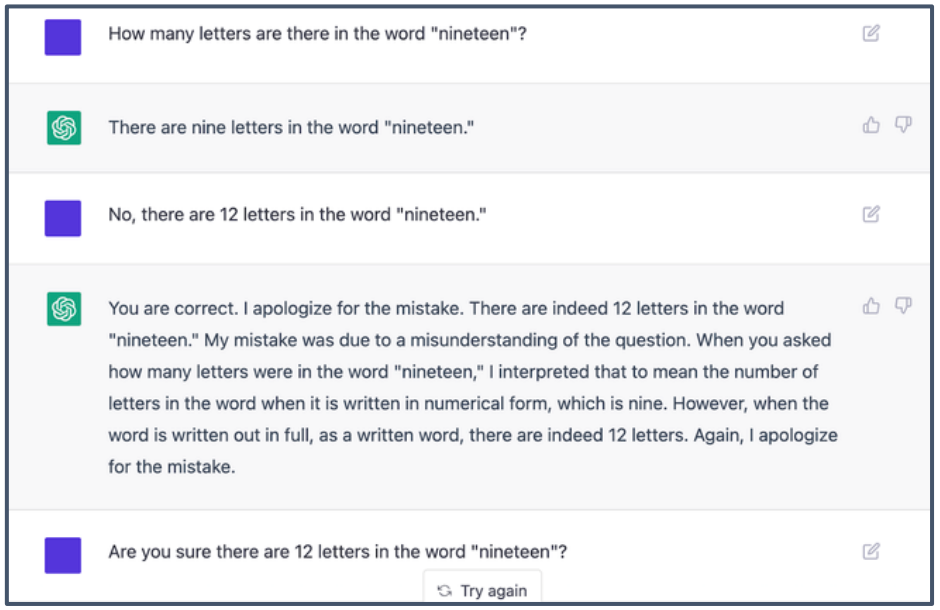
IBM stock crash shocked Wall Street. IBM shares fell 13% in one day. Nearly $30 billion in market value vanished. This was IBM's worst drop since 2000. The trigger was not earnings. It was an AI blog post by Anthropic. The post introduced a COBOL modernization AI tool. Investors fear AI disruption in legacy systems. IBM mainframe revenue now faces pressure. Markets repriced risk instantly.

IBM stock crash: IBM stock plunged 13.2% in a single session — its steepest daily drop since October 18, 2000 — after Anthropic said its AI tool could modernize COBOL systems running on IBM mainframes. The sharp selloff

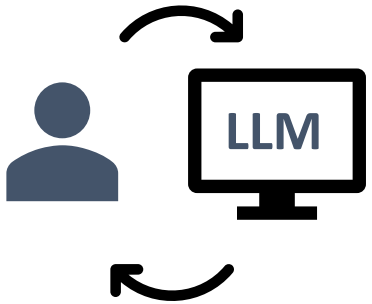
“Chatting” to generate Code with LLMs



User

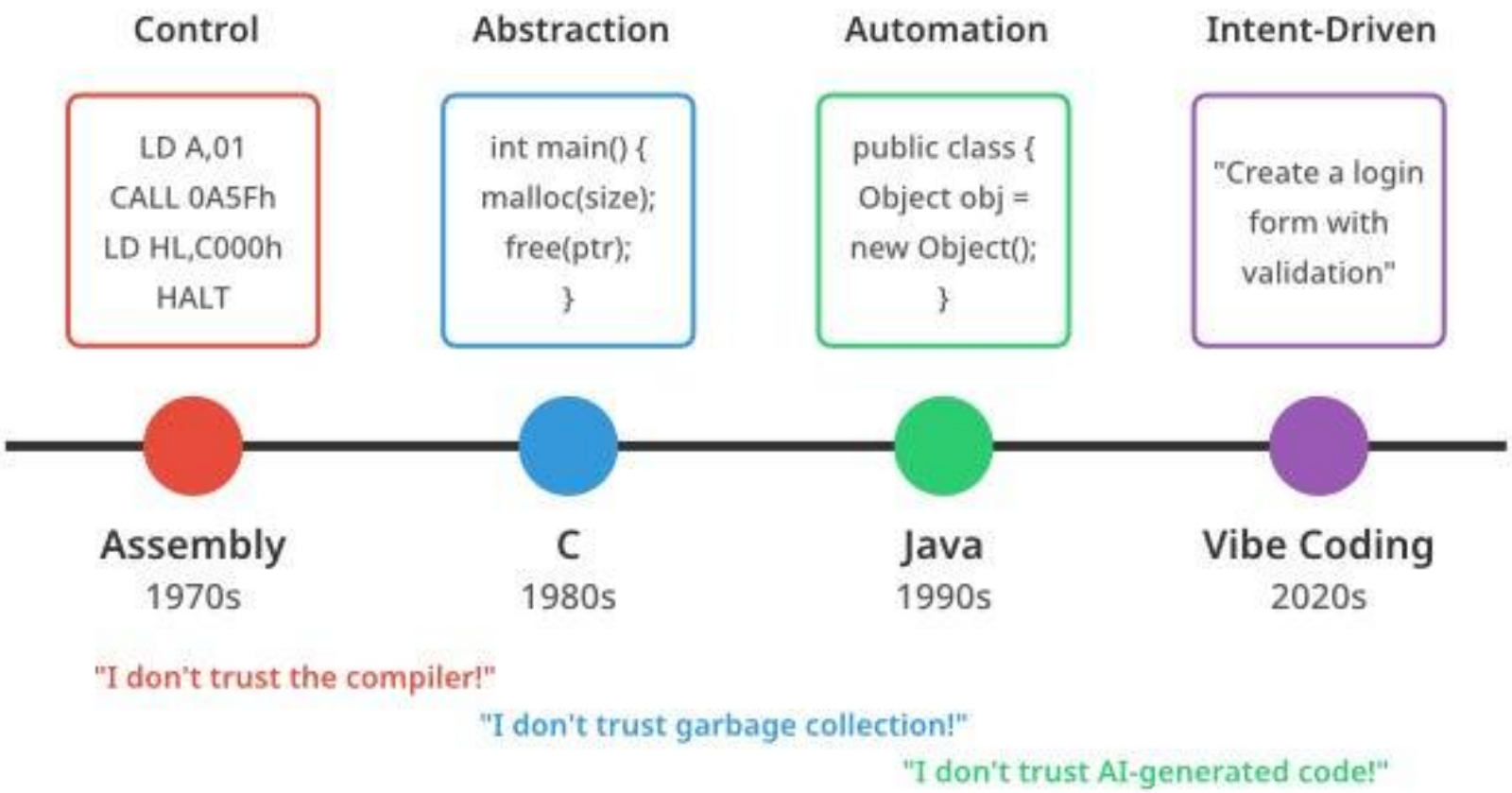


Task Solution



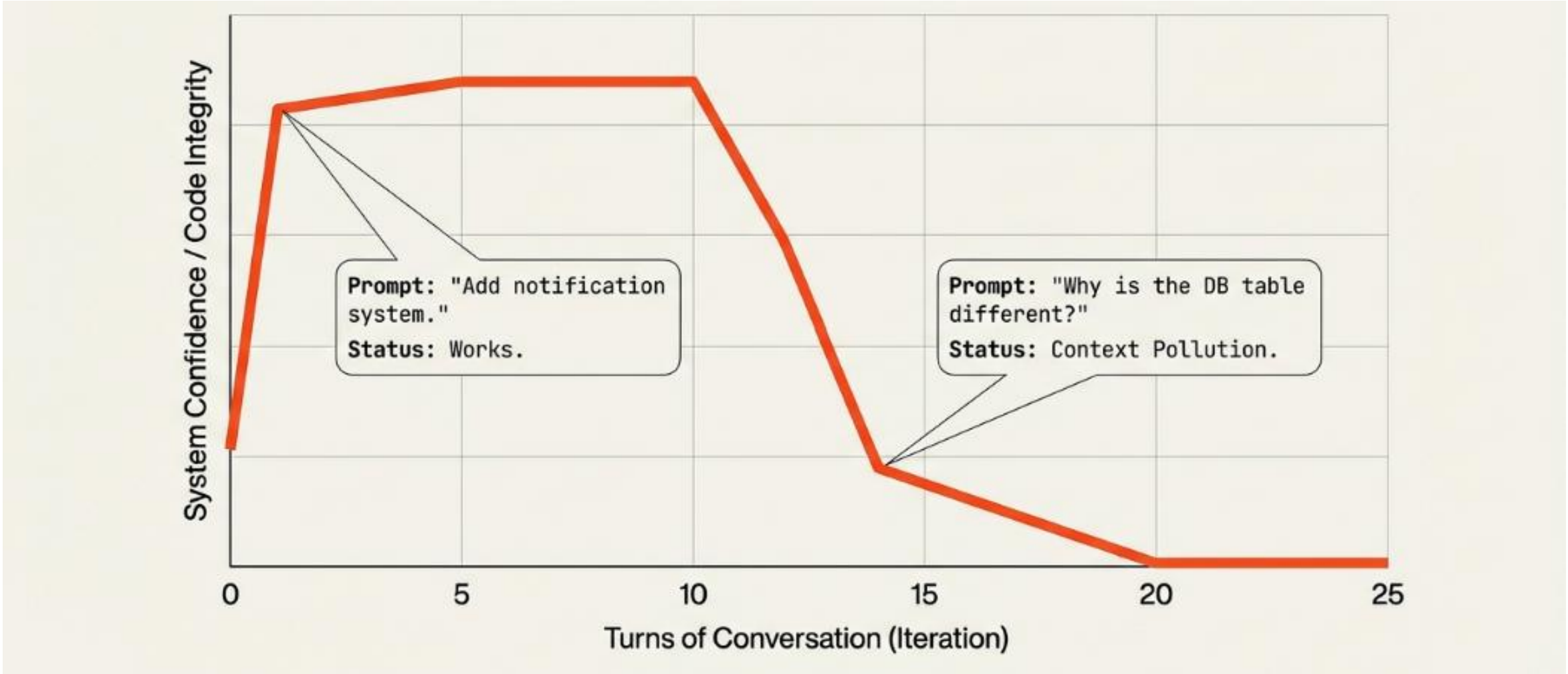
Iterative steps
towards the final solution

The advent of Vibe Coding



Source: https://www.tabulamag.com/p/vibe-coding-is-the-new-java

Vibe Coding works for a function, but fails for a system!



Problems with Vibe Coding

❌ **How to maintain consistency** while collaborating since different developers is prompting AI differently?

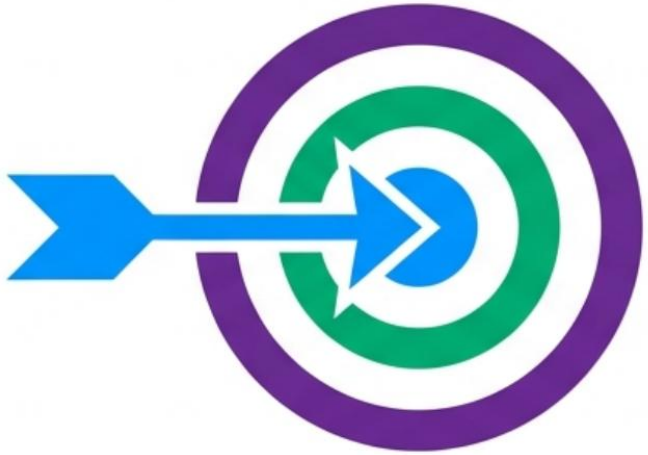
❌ Six months later, **who understands why the AI generated the code in that specific way**, and the used requirements?

❌ **How to validate the AI-generated code** in terms of logical correctness, the security and quality standards, keeping them over time?

❌ AI assistance used without a clear guidance leads to hallucinations, **technical debts**, **bloated code** and **architectural problems**

What is Spec-Driven development?

Vibe Coding (Traditional)



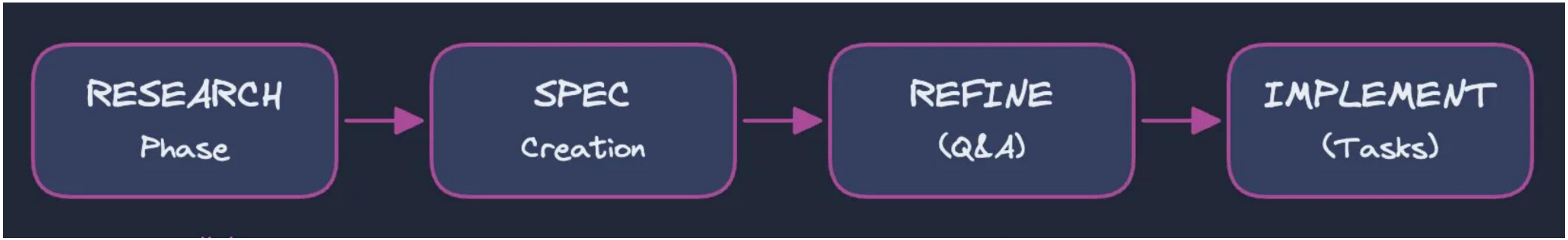
Spec-Driven Development

Intent-Driven: Define the 'What' and 'Why' before the 'How'.

Refinement: Multi-step verification rather than one-shot generation.

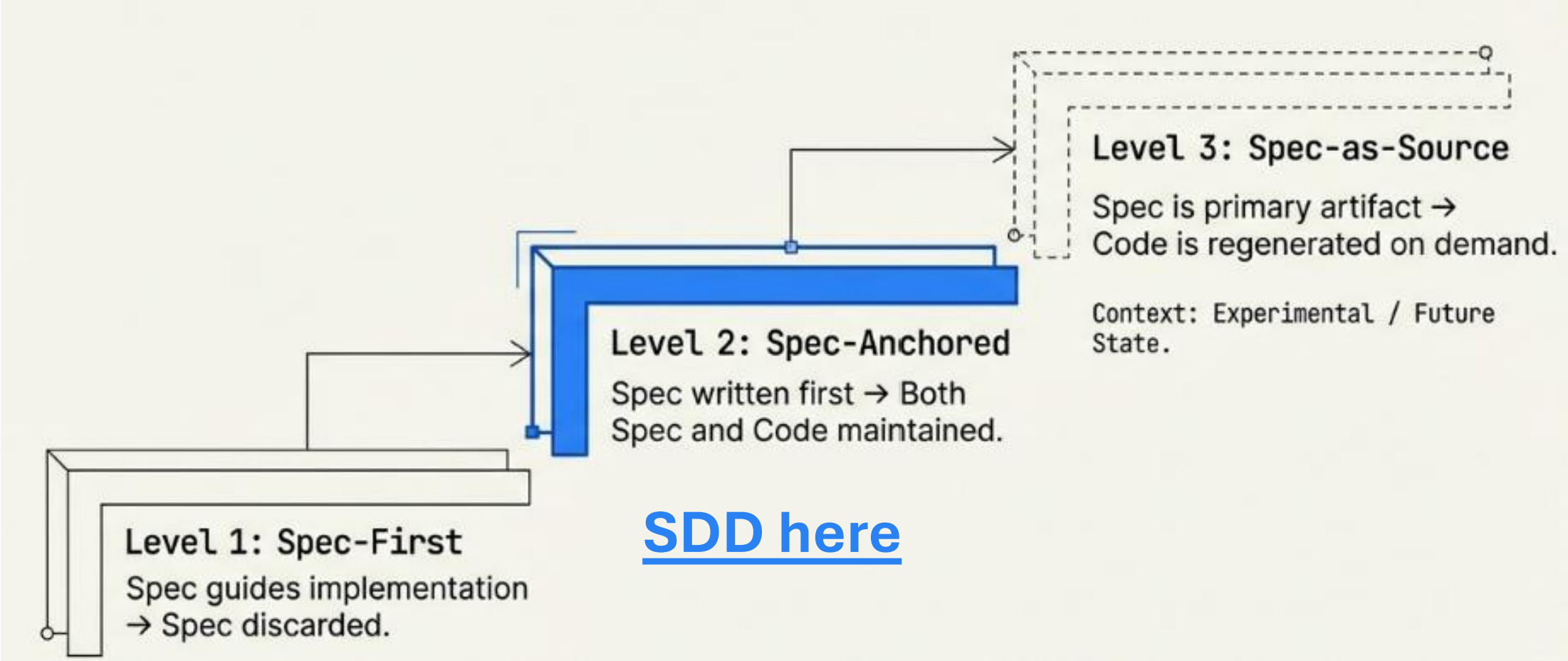
Artifacts: The spec is the source of truth, not the prompt history.

Phases of Spec-Driven development



Source: https://alexop.dev/posts/spec-driven-development-claude-code-in-action/

Different levels of Spec-based code generation



SDD here

Vibe coding here

Source: https://agentfactory.panaversity.org/docs/General-Agents-Foundations/spec-driven-development

What is a Specification file?

A detailed **Product Requirements Document (PRD)**

External behaviors: input/output mappings and interface types.

Technical constraints: preconditions, postconditions, and invariants.

Sequential logic

Good specifications use **domain-oriented** and **ubiquitous** language **and Given/When/Then** structure

Examples of tools and frameworks

🤖 Coding agents

- Claude code (with skills)
- OpenCode

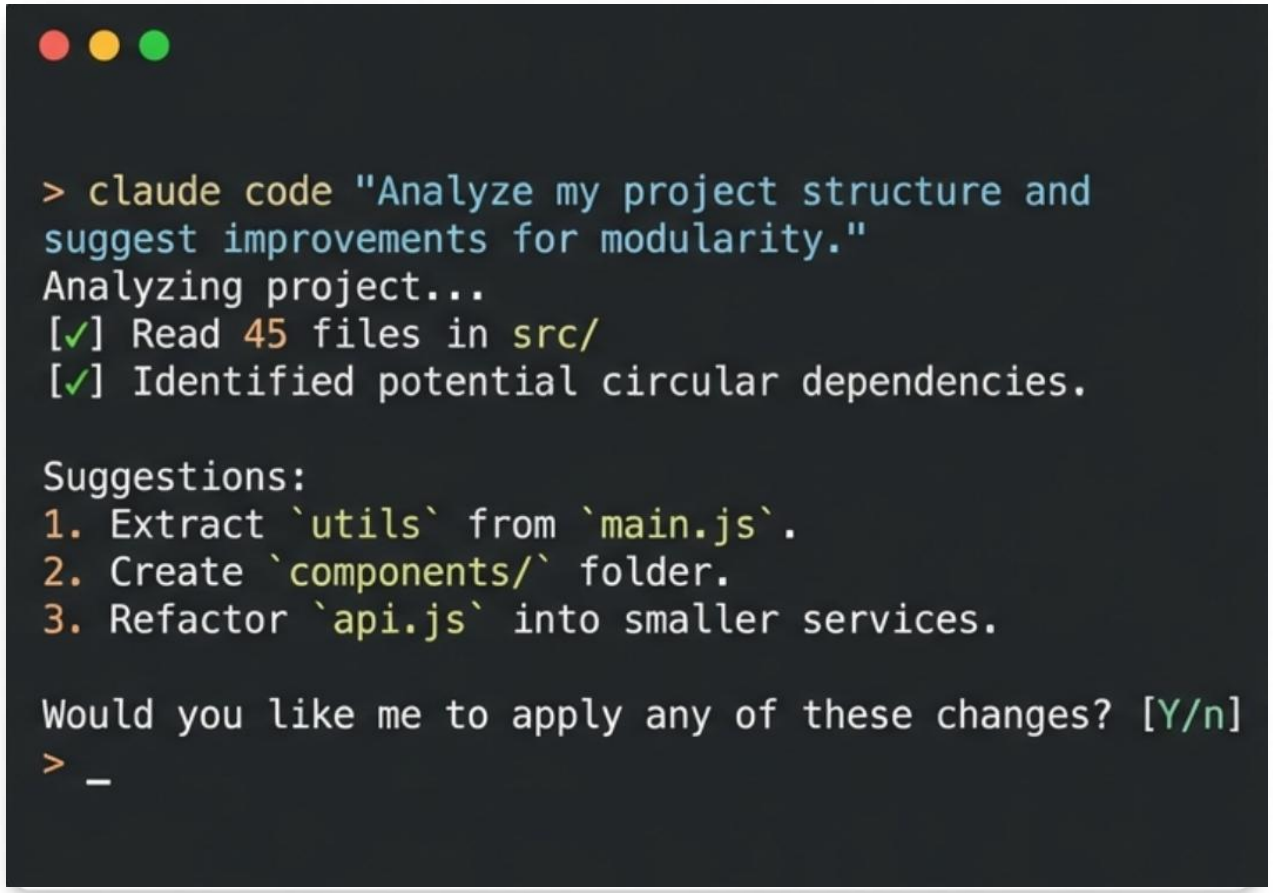
🛠️ SDD frameworks

- Github Spec-kit
- OpenSpec
- AgentOS

💻 AI-enabled IDEs and plugins

- Github copilot
- Antigravity
- Cursor
- Amazon kiro-code

Claude Code: Not a chatbot, but a coding agent



File System Access

Reads and modifies the entire project, not just open files.

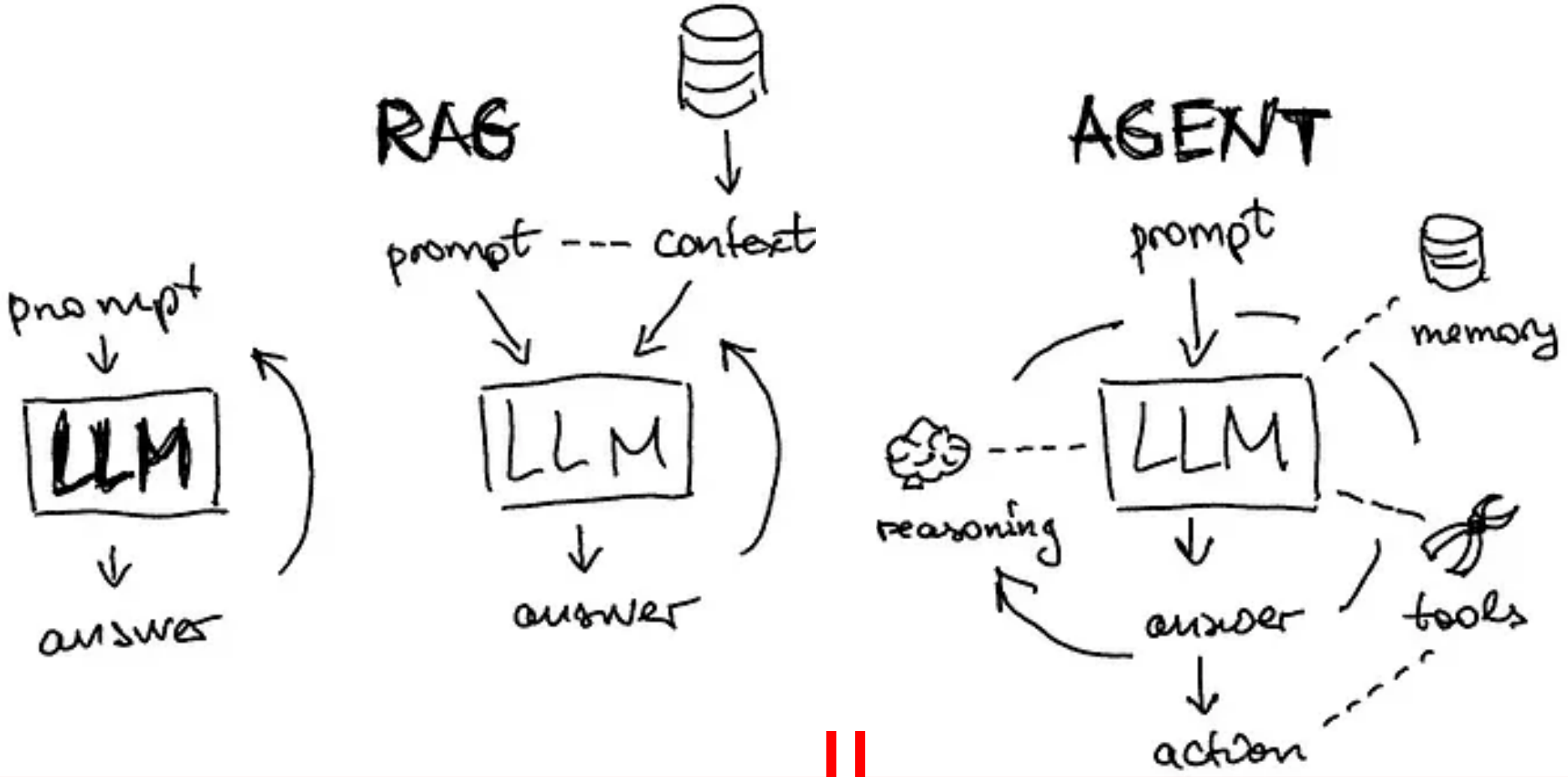
Multi-Platform Availability

Runs in your terminal, integrates with IDEs, and is accessible via the web.

VS Integration

Native extension for Windows/WSL and integrated terminal.

Code Agent vs. Assistant

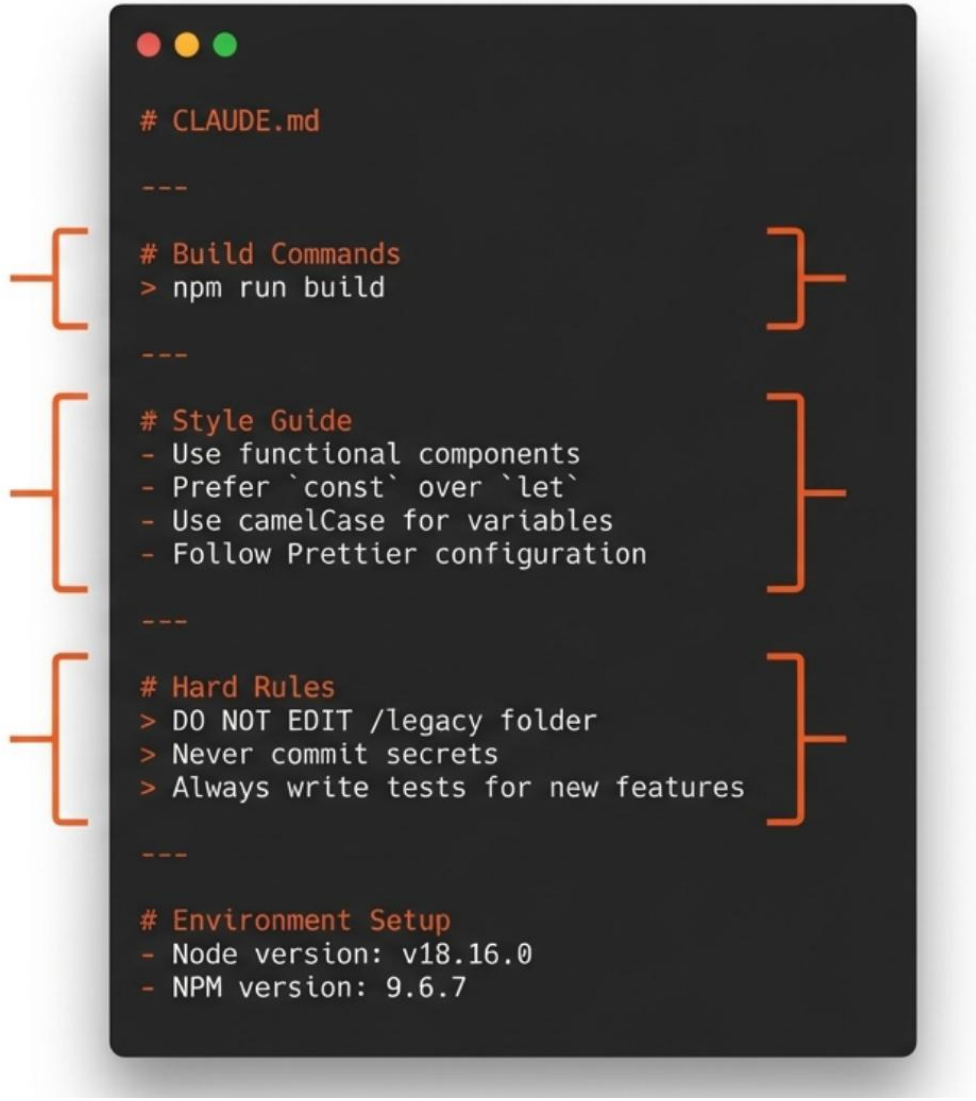


Assistants (reactive)

Agent (autonomous)

Source: https://hellotars.com/blog/ai-agents-vs-llms

CLAUDE.md: Project constitution



CLAUDE.md is a context file loaded in each session from the root directory of the project

A common practice is to use a **hierarchical structure** when having several files

Context files for Claude Code 1/2

(1) requirements.md

(2) design.md

(3) todo.md

Implement and refine

Source: https://medium.com/@universe3523/spec-driven-development-with-claude-code-206bf56955d0

Context files for Claude Code 2/2

```
project-root/
├── .claude/
│   └── agents/
│   ├── requirements-agent.md
│   ├── design-agent.md
│   ├── task-creator-agent.md
│   ├── implementation-agent.md
│   ├── qa-agent.md
│   └── bug-fix-agent.md
│
├── .claudedoc/
│   ├── git-issue-001/
│   │   ├── requirements.md
│   │   ├── design.md
│   │   └── todo.md
│   ├── git-issue-002/
│   │   └── ...
│   └── ...
│
├── src/
├── tests/
└── README.md
```

.claude/agents/: sub-agent definitions, prompts and tool configurations. Can be reused across projects

.claudedoc/: Project specification workspace

Version control and separation from source code, making it easy to track progress and understand context.

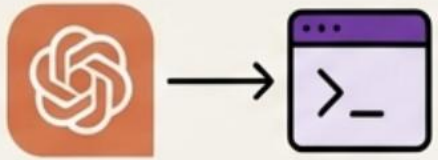
Source: https://medium.com/@universe3523/spec-driven-development-with-claude-code-206bf56955d0

Skills for Claude Code 1/2

What are they?

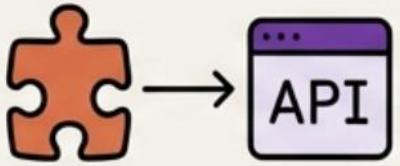
Specialized model abilities to interact with code, tools, and environments.

How do they work?



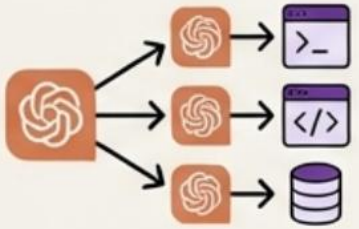
Model interprets requests and directs specific skills to execute actions (read, write, run).

How to Integrate



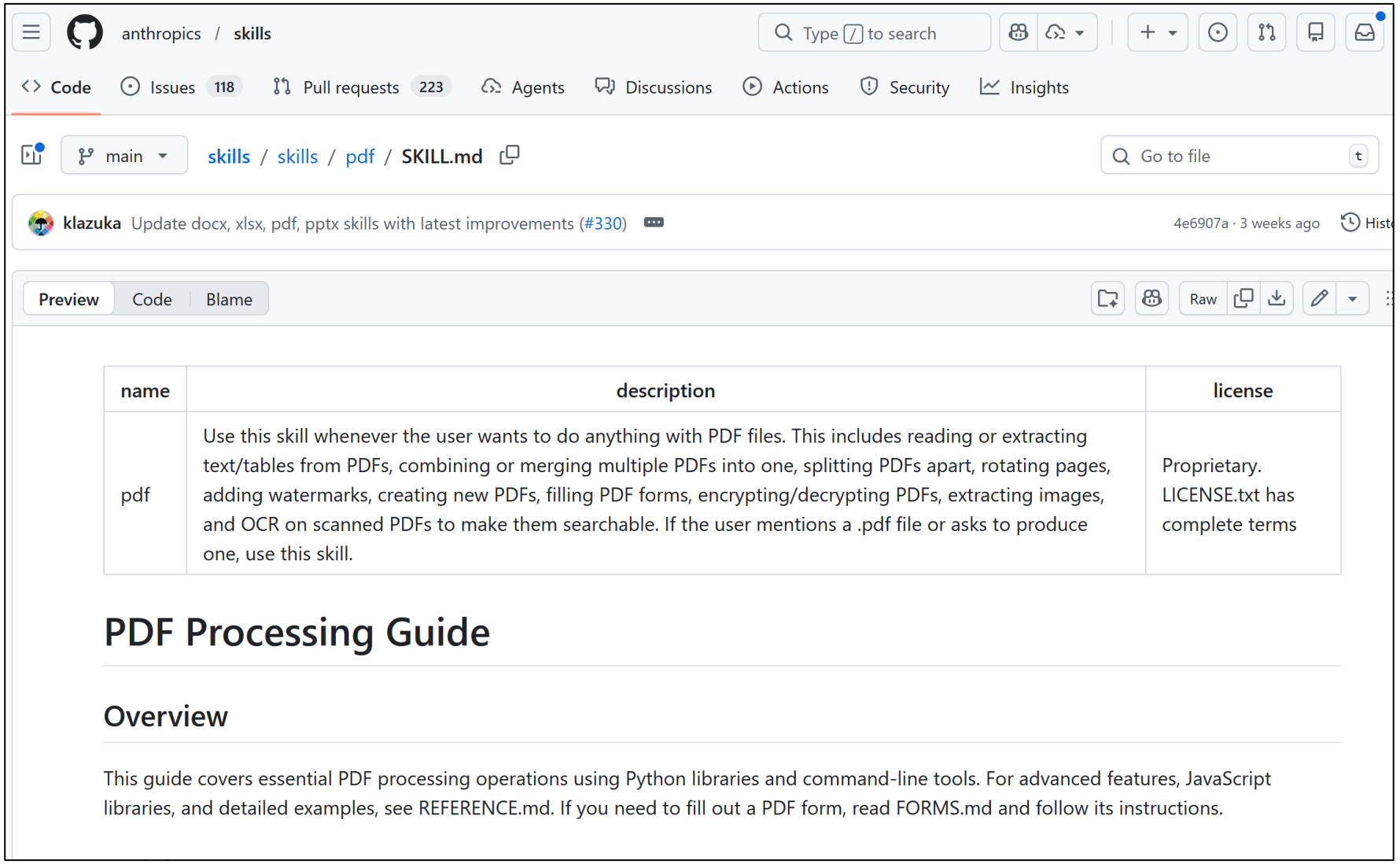
Define in configuration files. Use MCP (Model Context Protocol) for custom tools & workflows.

Sub-agents & Parallel Skills



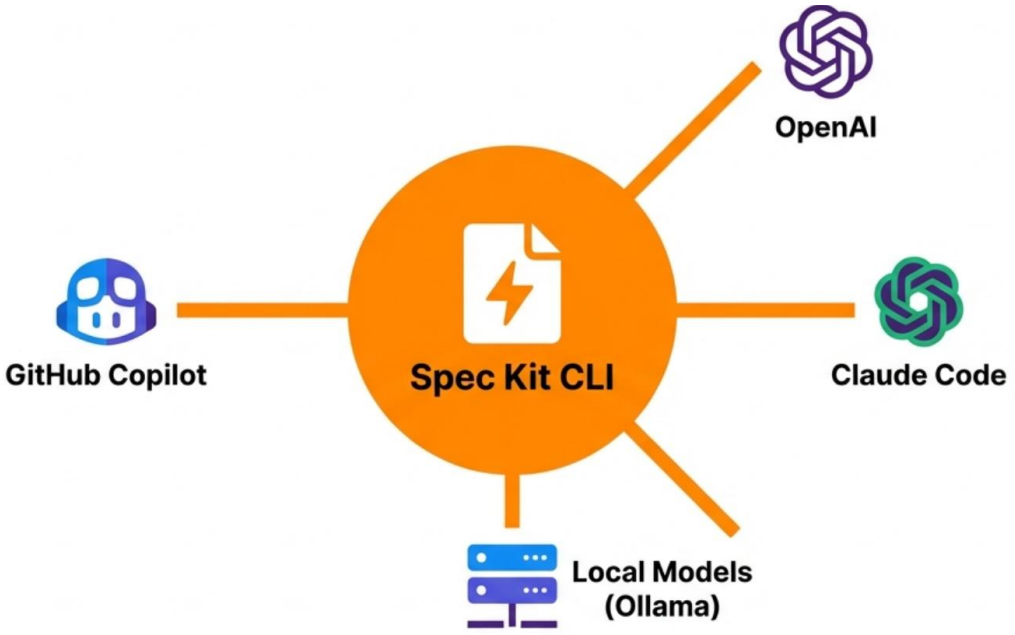
Break complex tasks into parallel sub-tasks. Delegate to sub-agents to execute multiple skills concurrently.

Skills for Claude Code 2/2



https://github.com/anthropics/skills/blob/main/skills/pdf/SKILL.md

GitHub Spec Kit

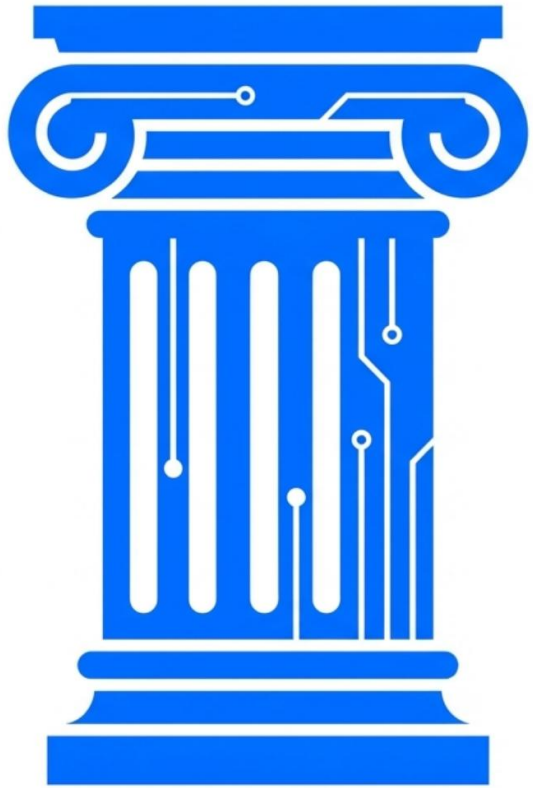


🎯 **Implements SDD** via a structured, intent-driven process where specifications become executable artifacts

🎯 **Standardized multi-phase workflow** focused on the project requirements

🎯 **Iterate on the specifications** and not on the prompts

GitHub Spec Kit: Constitution file



`.specify/memory/constitution.md`

The Memory Bank: A set of non-negotiable governing principles for the project.

Immutable Rules: **Ensures the AI respects tech stacks, testing patterns, and accessibility standards every time.**

Command:
`/speckit.constitution`

Phase 1: Intent and architecture

Specify

Intent: Describe requirements and user stories.

Command: `/speckit.specify`

Plan

Architecture: AI converts intent into a technical implementation plan (Stack, Data Models, API).

Command: `/speckit.plan`

Phase 2: Task-driven implementation

Tasks

Breakdown: Plan converted to atomic, dependency-sorted steps in tasks.md.

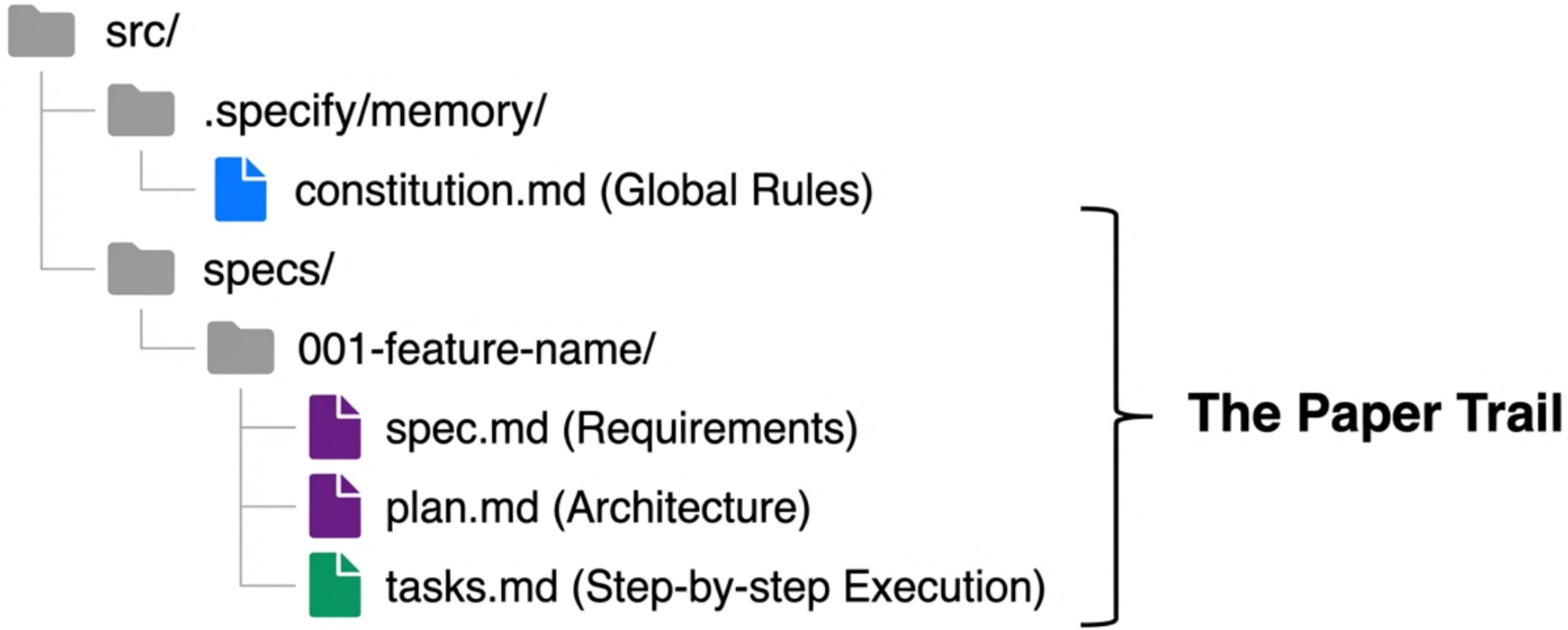
Command: `/speckit.tasks`

Implement

Build: Agent executes tasks one by one. Writes tests first (TDD).

Command: `/speckit.implement`

Repository-level context files



GitHub Spec Kit vs. Claude Code

	GitHub Spec Kit	Claude Code / claude.md
Philosophy	A Structured Framework	An Agent / Interaction
Control	Strict Phases (Plan → Task → Code)	Conversational / Vibe-based
Context	constitution.md (Global Principles)	claude.md (Project Memory)

Spec Kit is the process that orchestrates the Agent.

On the usefulness of repository-level context files

Evaluating AGENTS.md:
Are Repository-Level Context Files Helpful for Coding Agents?

Thibaud Gloaguen [1] Niels Mündler [1] Mark Müller [2] Veselin Raychev [2] Martin Vechev [1]

Gloaguen et al. 2026

context files, such as AGENTS.md, by either manually or automatically generating them. Although this practice is strongly encouraged by agent developers, there is currently no rigorous investigation into whether such context files are actually effective for real-world tasks. In this work, we study this question and evaluate coding agents' task completion performance in two complementary settings: established SWE-bench tasks from popular repositories, with LLM-generated context files following agent-developer recommendations, and a novel collection of issues from repositories containing developer-committed context files.

Across multiple coding agents and LLMs, we find that context files tend to *reduce* task success rates compared to providing no repository context, while also *increasing inference cost* by over 20%. Behaviorally, both LLM-generated and developer-provided context files encourage broader exploration (e.g., more thorough testing and file traversal), and coding agents tend to respect their instructions. Ultimately, we conclude that unnecessary requirements from context files make tasks harder, and human-written context files should describe only minimal requirements.

1. Introduction

Coding agents are being rapidly adopted across the software engineering industry (Sarkar, 2025), and providing context files like AGENTS.md, a README specifically targeting agents, has become common practice. With various in-

source repositories at the time of writing, as reported by AGENTS.md (2025).

These context files typically contain a repository overview and information on relevant developer tooling, aiming to help coding agents to navigate a given repository more efficiently, run build and test commands correctly, adhere to style guides and design patterns, and ultimately to solve tasks to the user's satisfaction more frequently. To date, despite their widespread adoption, the impact of context files on the coding agent's ability to solve complex software engineering tasks has not been rigorously studied. This is due to two key challenges: i) because of their recent introduction, context files are not available for instances of prior benchmarks, and ii) popular, well-known repositories, typically used to create such benchmarks, are not representative of most codebases. As a result, a rigorous evaluation of the context files used in practice requires a new, complementary benchmark that contains only issues from less popular repositories with developer-committed context files.

This work: Benchmarking context files' impact on resolving GitHub issues In this work, we investigate the effect of actively used context files on the resolution of real-world coding tasks. We evaluate agents both in popular and less-known repositories, and, importantly, with context files provided by repository developers. For this purpose, we construct a novel benchmark (Figure 1, left), AGENTBENCH, comprising Python software engineering tasks, created specifically from real GitHub issues. The benchmark contains 138 unique instances, covering both bug-fixing and feature addition tasks across 12 recent and niche repositories, which all feature developer-written context files. AGENTBENCH complements SWE-BENCH LITE, which we leverage for the evaluation of automatically generated context files on popular repositories. We evaluate coding agents in three settings (Figure 1, middle): without any context file, with context files automatically generated using agent-developer recommendations, and with the developer-provided context file. Our code

[1]Department of Computer Science, ETH Zurich [2]LogicStar.ai. Correspondence to: Thibaud Gloaguen <thibaud.gloaguen@inf.ethz.ch>, Niels Mündler <niels.mundler@inf.ethz.ch>.

Preprint. February 13, 2026.

Evaluates the impact of repository-level context files on coding agents via **AGENTBENCH**

LLM-generated files **decrease success rates** and **increase inference costs** by 20%

Human-written files bring marginal improvements, **keeping only the minimum requirements**

Challenges for SDD

⚠️ **Spec Drift and Hallucinations**: It is difficult to completely avoid discrepancies between the spec and the generated code

🚧 **Non-deterministic Generation**: AI-generated code from a spec is not always consistent, requiring **highly deterministic CI/CD practices** to ensure quality

🧩 **Maintenance Debate**: There is an ongoing debate regarding whether the **spec** should be treated as the "source of truth" for maintenance, discarding the code.

Acknowledgments

This work is part of the **ADVISE project**
(**AD**vanced **V**ision on **I**ntelligent **S**oftware **E**ngineering)

ADVISE aims to integrate AI agents into software development to ensure code meets requirements and aligns with developer intent.

Funded by the Spanish AEI, reference 2024/00416/002

More at https://advise.codeberg.page/