Governing Code with Specification-Driven Development

Giovanni Rosa
Postdoctoral Researcher
Universidad Rey Juan Carlos
giovanni.rosa@urjc.es

LLMA4SE School

University of Extremadura, Cáceres, Spain
Sep 9-11, 2026

URJC ADVISE

Where am I from?

SoftDev Research Group

https://softdev.codeberg.page/

Giovanni Rosa
Postdoctoral researcher,
AI for Software Engineering

David Moreno Lumbreras

Gregorio Robles

Jesús M. González-Barahona

ADVISE

Spanish AEI
Ref. 2024/00416/002

MADRID

Today’s Agenda

1

Part 1

Spec-Driven Development

2

Part 2

Exploring SDD adoption in the open-source

3

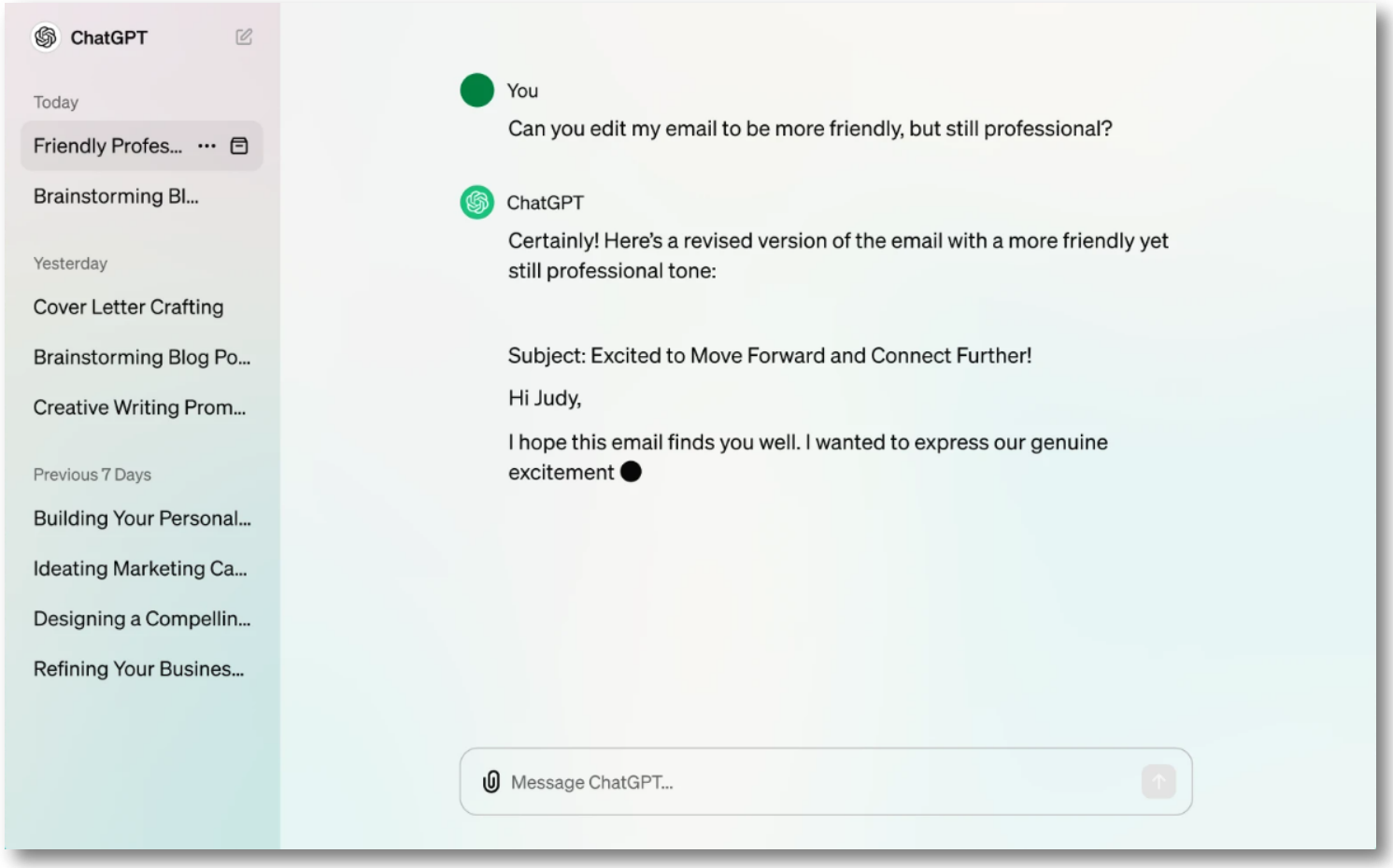
Part 3

Re-imagining Software Visualization for SDD

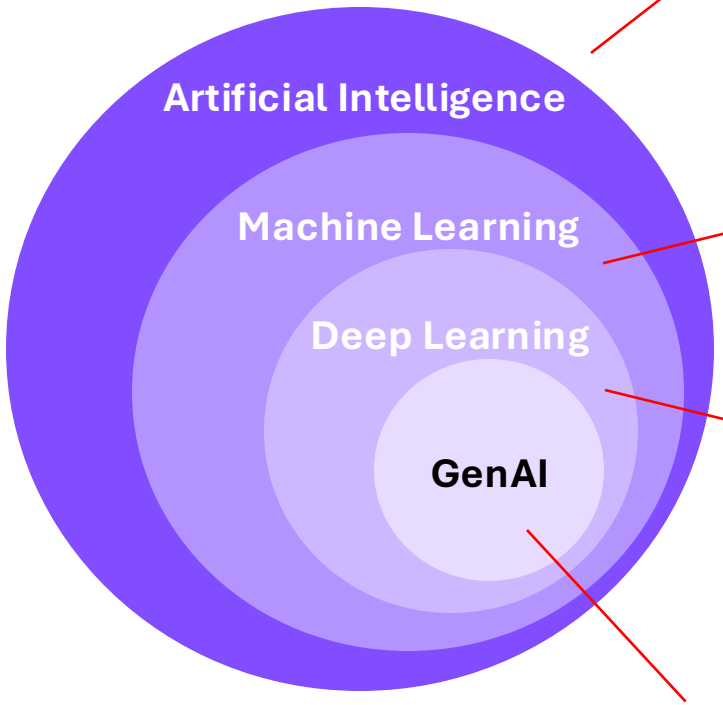
November 2022

ChatGPT Assistant

by OpenAI



AI vs. Generative AI



Intelligent autonomous machines

Pattern recongition
Learn general patterns form data

Neural Networks
Learn general patterns in unstructured data

Generative AI
Learn to understand the data (es. natural language)

The Economist

Fox News. Now what?
Ukraine's game plan
Why civil wars are lasting longer
A special report on the car industr

APRIL 22ND-28TH 2023

AI

THE DAILY AI NEWS

www.dailyainews.com FOR THE LATEST FICTIONAL AI WORLD NEWS Dec 2022

Is this the Death of Google Search and the rise of ChatGPT?

How would you describe the front page of a newspaper that reads "Google search is

"Why is ChatGPT going to replace google search"?

It is not specified why ChatGPT is

ChatGPT vs software developers: is generative AI the end of the road for developers?

News in focus

The artificial-intelligence chatbot ChatGPT is disrupting

much as people have listed pets,

CHATGPT LISTED AS AUTHOR ON RESEARCH PAPERS

Many scientists disapprove of articles crediting the AI tool as a co-author.

By Chris Stokel-Walker

The artificial-intelligence (AI) chatbot ChatGPT that has taken the world by

papers. But some publishers say that an AI's contribution to writing papers can be acknowledged in sections other than the author list. (*Nature*'s news team is editorially independ-

tools. "We are not new to this field,

He says that *Oncoscience* peer reviewed this paper after he asked its editor to do so. The journal did not respond to *Nature*'s request for comment.

A fourth article[4], co-written by an earlier chatbot called GPT-3 and posted on French preprint server HAL in June 2022, will soon be published in a peer-reviewed journal, says co-author Almira Osmanovic Thunström, a neurobiologist at Sahlgrenska University Hospital in Gothenburg, Sweden. She says one journal rejected the paper after review, but a second accepted it with GPT-3 as an author after she rewrote the article in response to reviewer requests.

6 DAILY NATION TUESDAY, JANUARY

HealthyNation | Tech

Opinion

How ChatGPT will transform medicine this year

Though still in its initial phase,the platform is already cutting down the time needed to conduct medical scientific research

BY FAUSTINE NGILA

Technologists have long discussed

That is just one among billions of medical answers for many health questions stored inside this artificial brain. Its accu-

solutions,treatments and consultations.

And with Kenya's tech ecosystem booming, where over 130 health tech startups have sprouted over the last seven years,

This means it can improve efficiency in the pharmaceutical supply chain by automating the process of generating purchase orders, invoices, and delivery notes

velop clinical decision deliver pre-operative sist with the manage tive pain.

The Freeman

10 Lifestyle

EDITOR: YASUNARI RAMON SUAREZ TAGUCHI

SATURDAY | February 18, 2023

Jobs that Could be Replaced by AI

By Yasunari Ramon Suarez Taguchi

Since its rollout in November last year, the artificial intelligence matrix developed by the OpenAI group known as ChatGPT has been used to do all sorts of things – from writing cover letters to coming up with well-written essays.

Essentially a chatbot, many have come to see it as more than a program that's designed to automate queries in customer service chats – with Google even alleged to have said that it could "hire" ChatGPT as an entry level coder if it applied for a position at the company without the knowledge that it was a program and not a real person.

With this development, the question as to what real-world jobs an AI matrix could replace have been brought up. Here are takes on five professions which industry experts and tech pundits are saying could be replaced by AI.

Graphic Designers

Customer Service Agents

Though the prospect of "talking" to a machine, as opposed to a human customer service agent, is not appealing to most, the customer service field is largely at risk due to advances in the field of AI.

Chances are, you have already

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

We recruited

95

developers, and split them randomly into two groups.

We gave them the task of writing a web server in JavaScript

45 Used
GitHub Copilot

78%
finished

1 hour, 11 minutes
average to complete the task

71 minutes | **that's 55% less time!**

50 Did not use
GitHub Copilot

70%
finished

2 hours, 41 minutes
average to complete the task

161 minutes

Results are statistically significant (*P=.0017*) and the 95% confidence interval is [21%, 89%]

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.

Enabling long-running Claudes

Running Claude in parallel

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

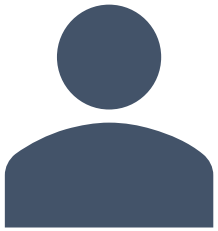
By Piyush Shukla, Global Desk • Last Updated: Feb 24, 2026, 05:26:00 PM IST

Synopsis

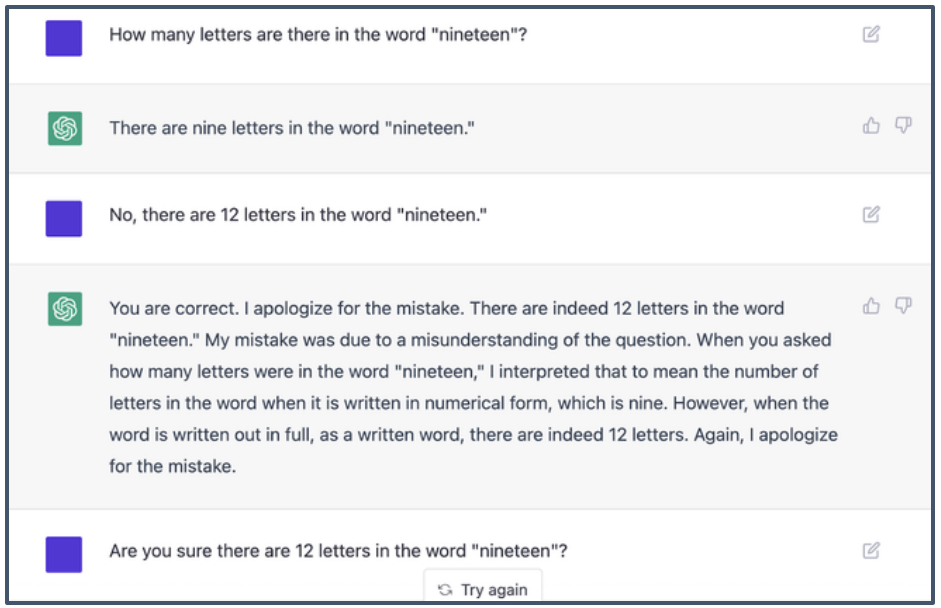
IBM stock crash shocked Wall Street. IBM shares fell 13% in one day. Nearly $30 billion in market value vanished. This was IBM's worst drop since 2000. The trigger was not earnings. It was an AI blog post by Anthropic. The post introduced a COBOL modernization AI tool. Investors fear AI disruption in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

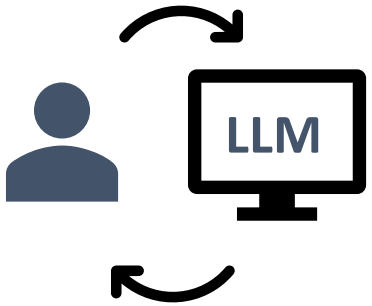
“Chatting” to generate Code with LLMs



User

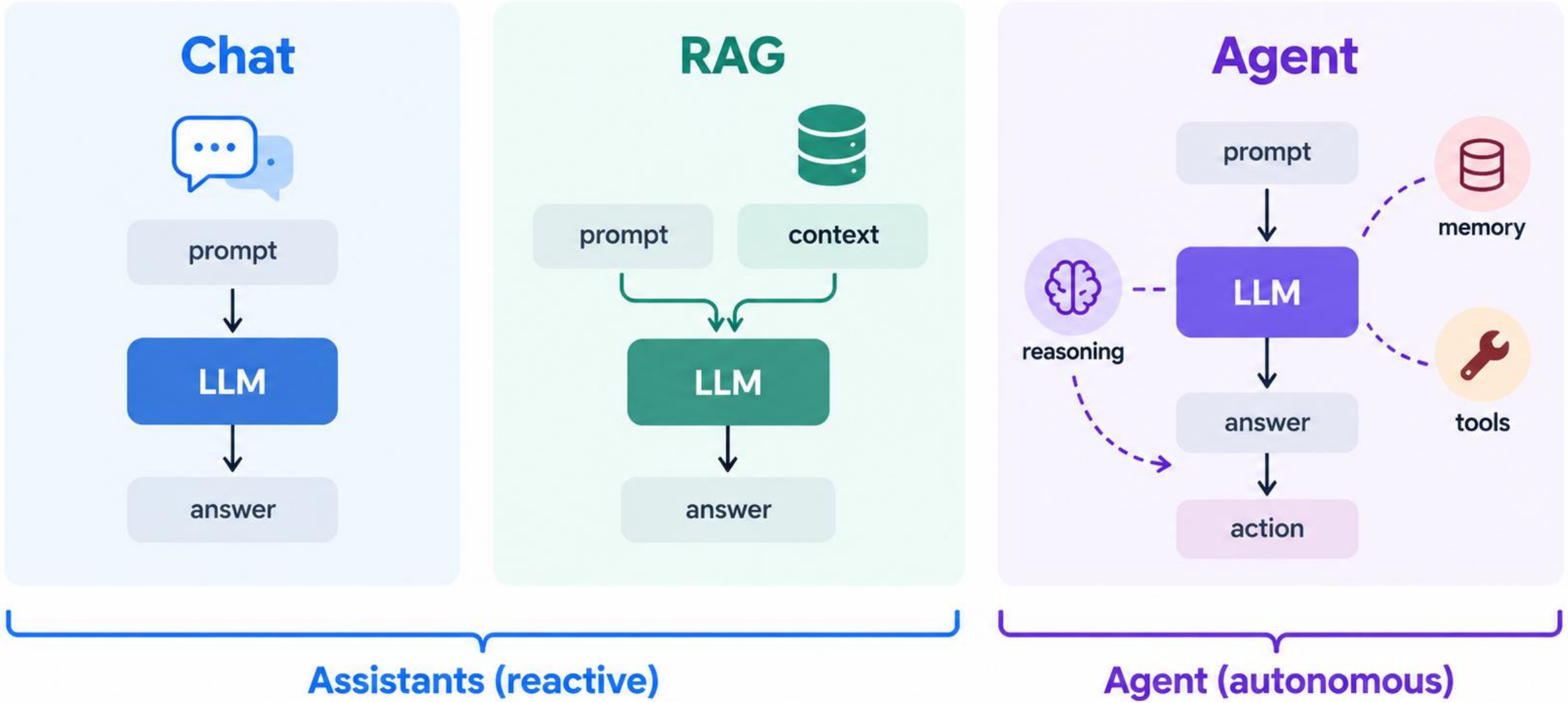


Task Solution



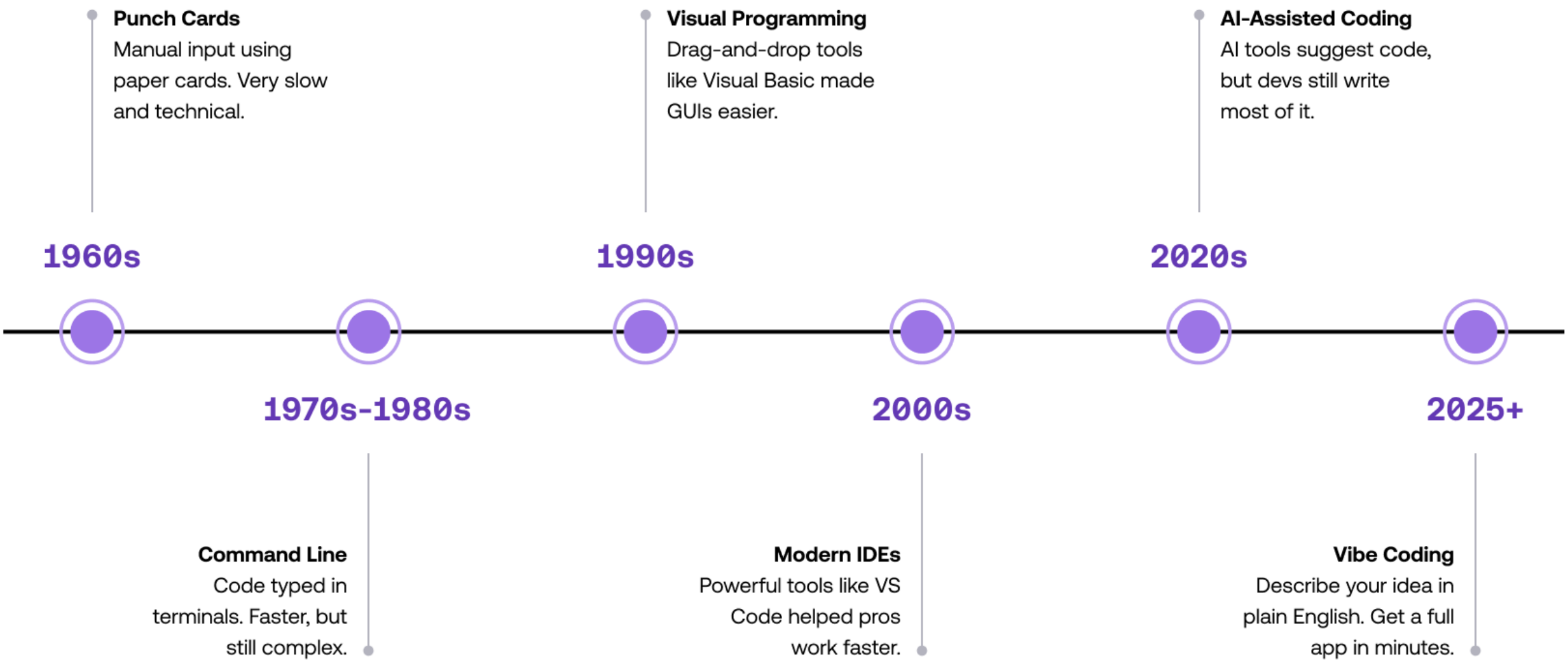
Iterative steps
or
«looping»

Code Assistant != Code Agent



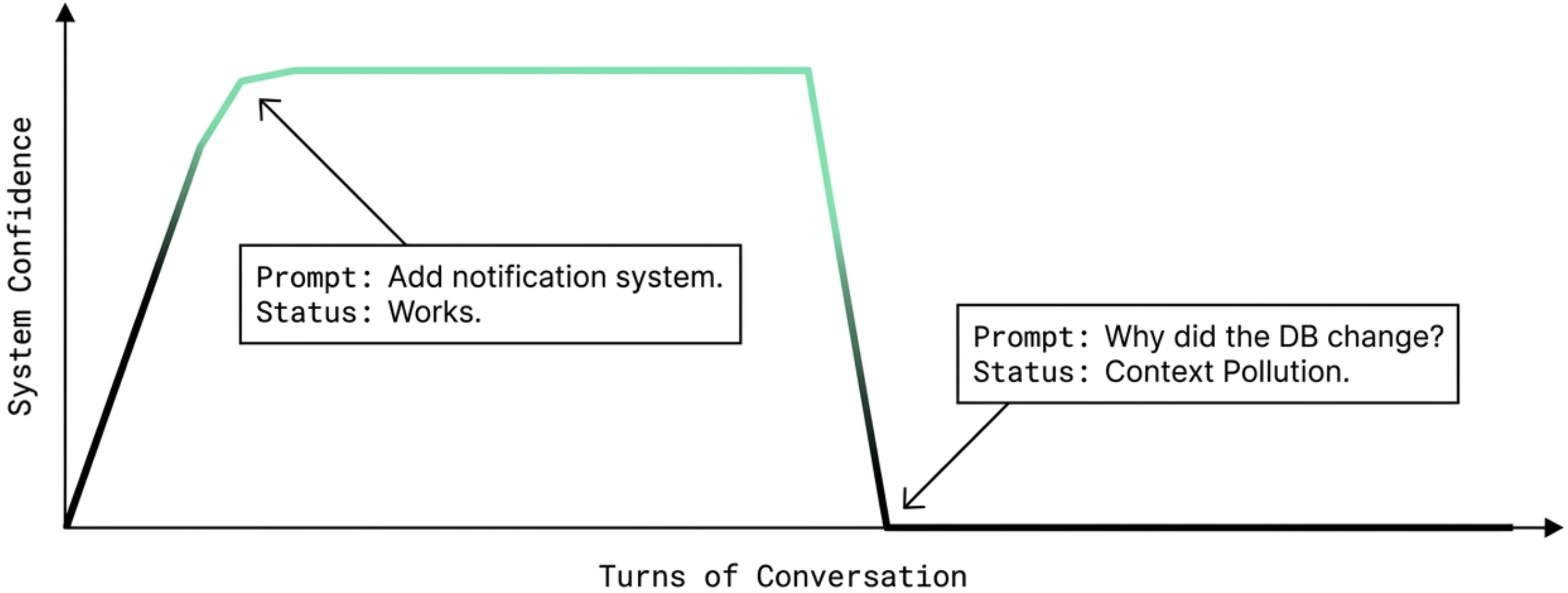
Source: https://hellotars.com/blog/ai-agents-vs-llms

The advent of Vibe Coding

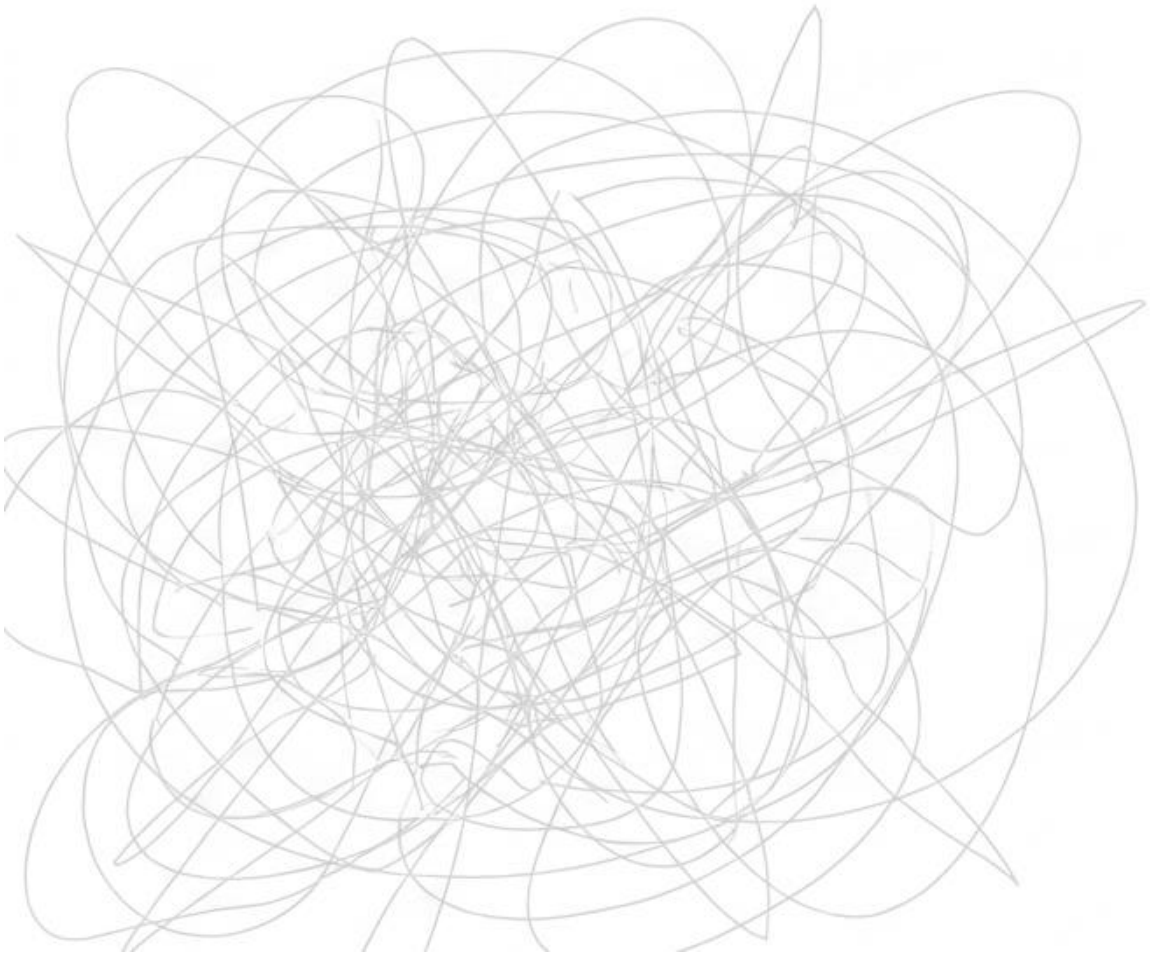


Source: https://mimo.org/blog/the-rise-of-vibe-coding

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



Problems with Vibe Coding



Architectural Drift

AI passes the immediate unit tests but quietly breaks cross-service contracts and API boundaries.

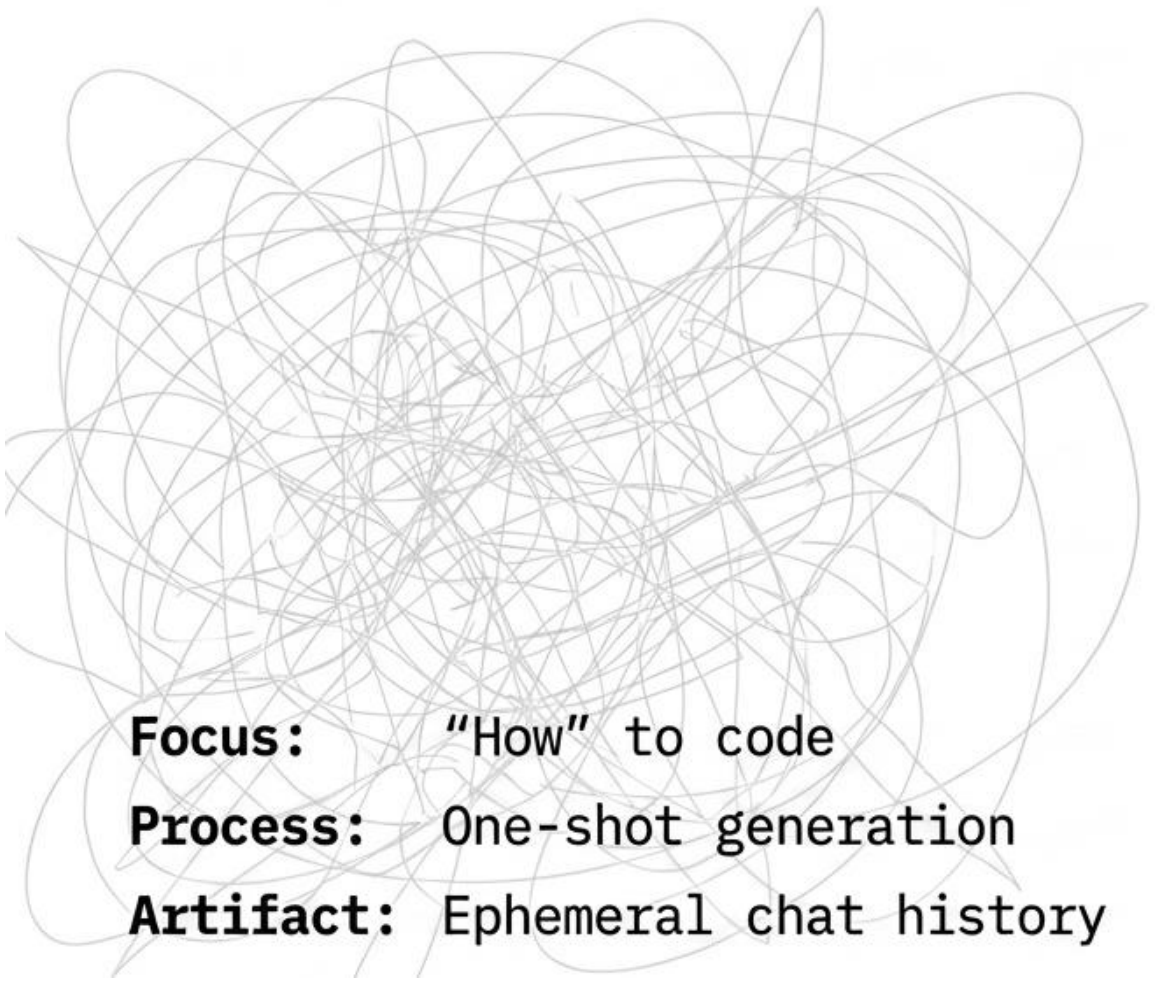
Orphaned Logic

Six months later, no human engineer understands the underlying requirements or why the AI generated the code.

Context Bloat

Endless conversation turns generate unmanageable complexity, validating persistent rises in static analysis warnings.

Towards Spec-Driven Development



Focus: "How" to code
Process: One-shot generation
Artifact: Ephemeral chat history

Ad-hoc AI

Focus: "What & Why" to build
Process: Multi-step validation
Artifact: Executable Specification

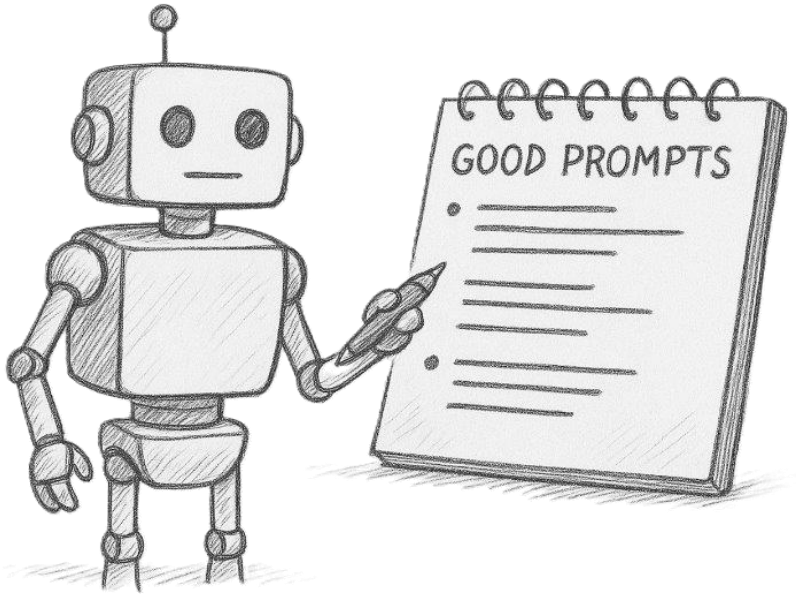
SDD

What is Spec-Driven development?

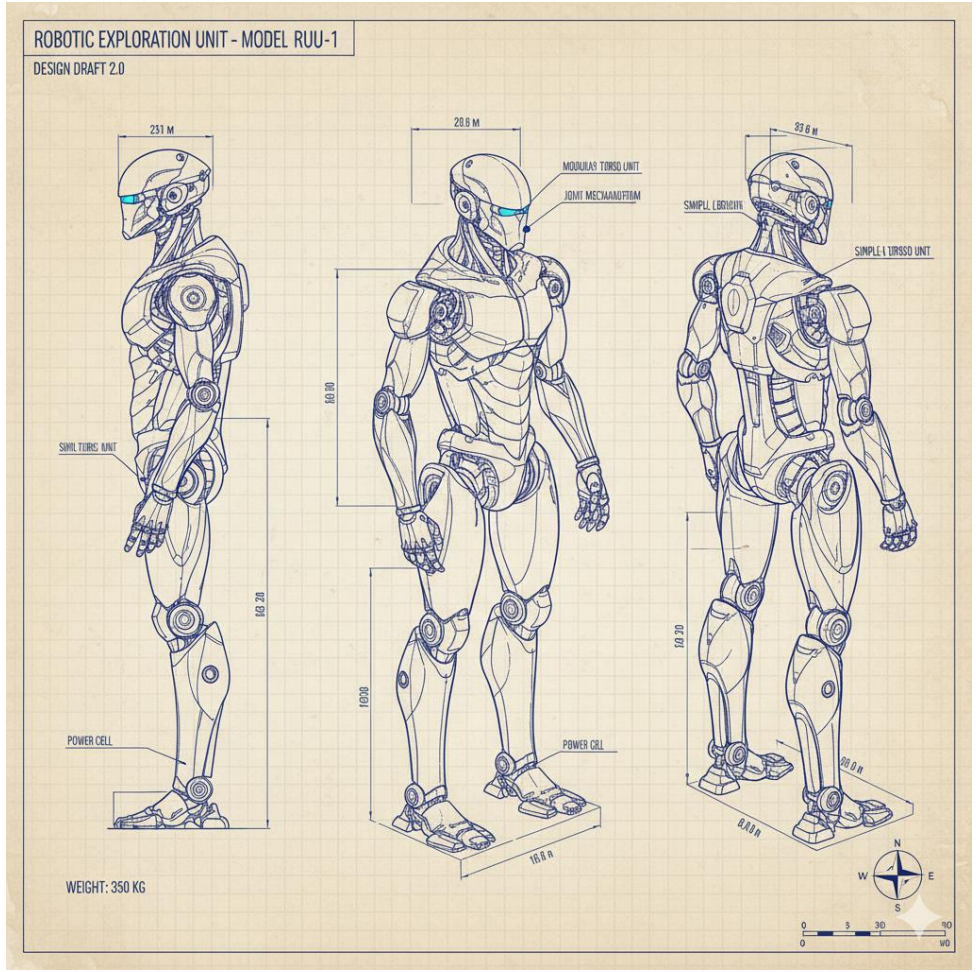
Utilizes formal **requirement specifications as prompts** for AI coding agents to generate executable code following the **Given/When/Then** structure

Context engineering is the key

e.g., Scratchpads, Progress Logs



The Anatomy of a Specification File 1/2



Source of Truth for the project requirements. Usually in **Markdown** format

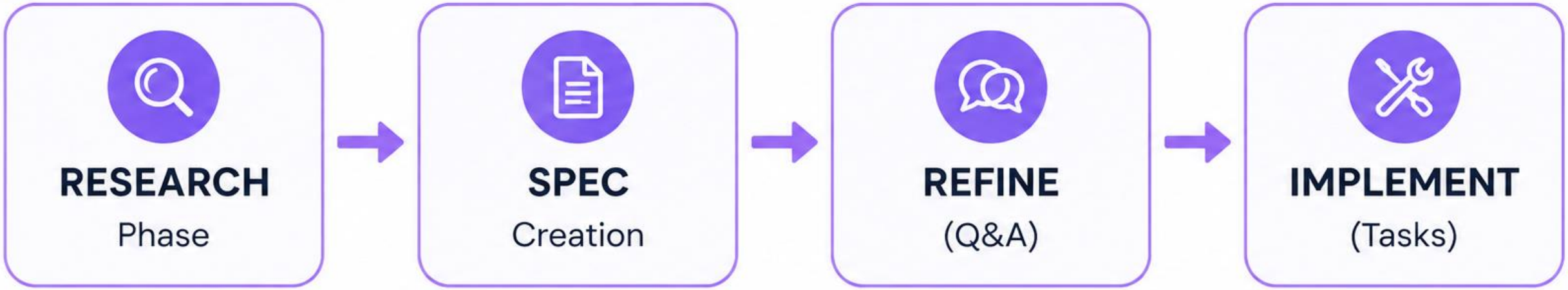
An **executable blueprint** that guides both the developer and the AI agents, allowing collaboration

Maintained together with the code implementation

The Anatomy of a Specification File 2/2

1. Outcomes The exact ‘Done’ state.	**2. Boundaries** Strict definitions of what not to touch.	**3. Constraints** Technical preconditions and invariants.
4. Decisions Explicitly stated prior architectural choices.	**5. Breakdown** Expected sequential logic flow.	**6. Verification** Executable Given/When/Then criteria.

Phases of Spec-Driven Development



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

Examples of tools and frameworks

Coding Agents

Claude Code

OpenCode

SDD Frameworks

GitHub Spec-kit

OpenSpec

AgentOS

AI IDEs

GitHub Copilot

Cursor

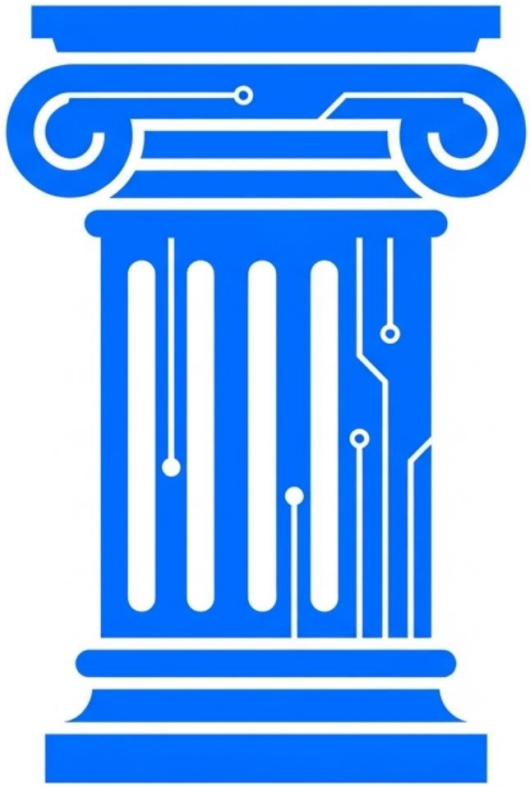
Tessl

Amazon Kiro-code

The Newcomer:

GitHub Spec-Kit

The Core: 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`

An example of constitution.md 1/2

Core Principles

I. Code Quality

All code MUST meet these non-negotiable quality standards:

- **Readability**: Code MUST be self-documenting with clear naming conventions. Variable and function names MUST express intent without requiring comments to understand basic functionality.
- **Maintainability**: Functions MUST have a single responsibility. Files MUST NOT exceed 300 lines without architectural justification. Cyclomatic complexity MUST remain below 10 per function.
- **Consistency**: All code MUST follow the project's established style guide and formatting rules. Linting MUST pass with zero warnings before merge.
- **Documentation**: Public APIs MUST have documentation. Complex algorithms MUST include inline comments explaining the "why" not the "what".
- **No Dead Code**: Unused imports, variables, and functions MUST be removed. Commented-out code MUST NOT be committed.

Rationale: High code quality reduces technical debt, accelerates onboarding, and minimizes bug introduction during changes.

II. Testing Standards

Testing is mandatory and MUST follow these requirements:

- **Coverage Threshold**: New code MUST have minimum 80% line coverage. Critical paths (authentication, data mutations, payments) MUST have 100% coverage.
- **Test Types Required**:
  - Unit tests for all business logic and utility functions
  - Integration tests for API endpoints and database operations
  - Contract tests for external service integrations
- **Test Quality**: Tests MUST be deterministic (no flaky tests). Tests MUST be independent and runnable in isolation. Test names MUST clearly describe the scenario being tested.
- **Test-First Encouraged**: For complex features, write tests before implementation (Red-Green-Refactor). All bug fixes MUST include a

An example of constitution.md 2/2

CORE PRINCIPLES

What we believe / must follow

- Configuration-as-Code
- Extensibility
- PASS FAIL pytest Test-First
- Performance & Correctness
- MAJOR.MINOR.PATCH Simplicity
- AI-First Development

TECHNOLOGY CONSTRAINTS

What technologies we may use

- Python 3.12+ "Legacy Exception"
- Jinja2
- pytest
- Data sources
- Recipients

DEVELOPMENT WORKFLOW

How changes must be developed

1. Feature Spec
2. Failing Tests
3. Implementation
4. Output Review
5. Full Pytest Suite
6. Descriptive Commits
7. 7. Changelog

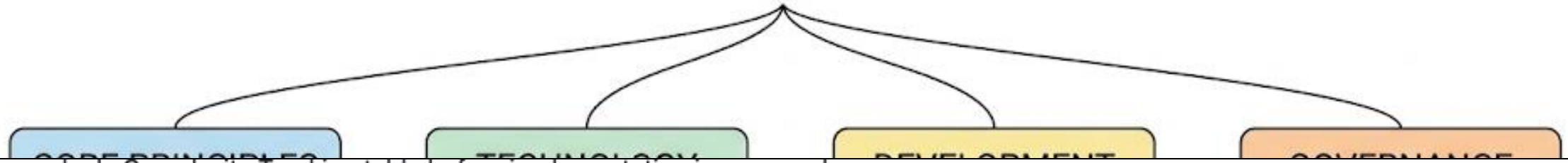
GOVERNANCE

How the rules are enforced and changed

- Amendments require written proposal and approval.
- Versioning policy for updates.
- Complexity Tracking
- AI Transition Baseline for additional notion and approval.

https://github.com/lausser/coshsh/blob/9e729a2/.specify/memory/constitution.md

An example of constitution.md 2/2



plan's Complexity Tracking table before implementation is approved.

AI Transition Baseline

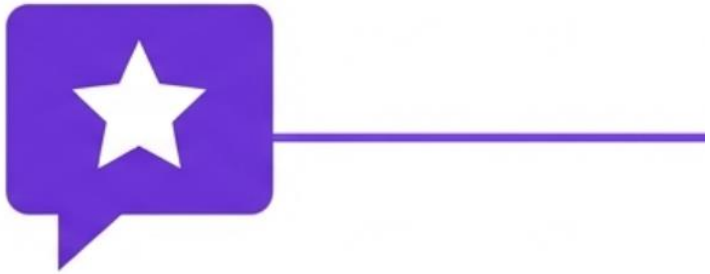
The state of the coshsh codebase as of **February 2026** constitutes the official baseline: the last version designed, written, and maintained exclusively by human contributors. From this point forward:

- All new code, bug fixes, refactoring, and enhancements MUST be authored by AI agents.
- Human contributors MUST NOT directly write or modify source code; their role is to specify intent, review AI-authored output, and approve or reject changes.
- The February 2026 baseline is preserved in git history and serves as the reference point for measuring AI-driven evolution of the project.

Version: 1.2.0 | **Ratified**: 2026-02-17 | **Last Amended**: 2026-07-03

https://github.com/lausser/coshsh/blob/9e729a2/.specify/memory/constitution.md

Phase 1: Intent Definition



Specify

Intent: Describe requirements and user stories.

Command: `/speckit.specify`

```
# Feature: User Authentication

Given an unregistered user
When they submit valid signup details
Then create account and return session
token.
```

Phase 2: System Design

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 3: Tasks breakdown

Tasks

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

Command: `/speckit.tasks`

```
# Task: Implement JWT Middleware

[x] Read constitution rules on auth.
[ ] Write `verifyToken` function.
[ ] Add Pytest unit tests for edge cases.
```

Phase 4: Task-driven implementation

Tasks

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

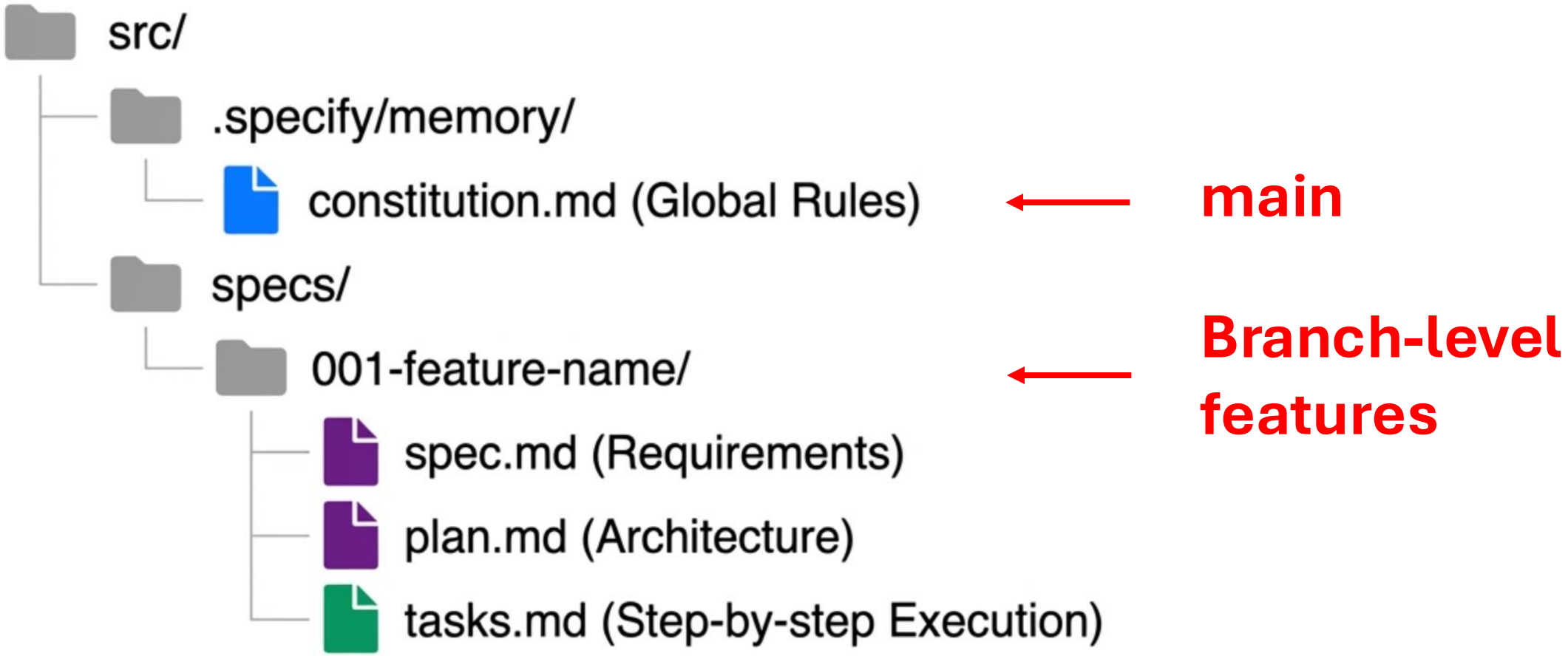
Command: `/speckit.tasks`

Implement

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

Command: `/speckit.implement`

Repository-level context files



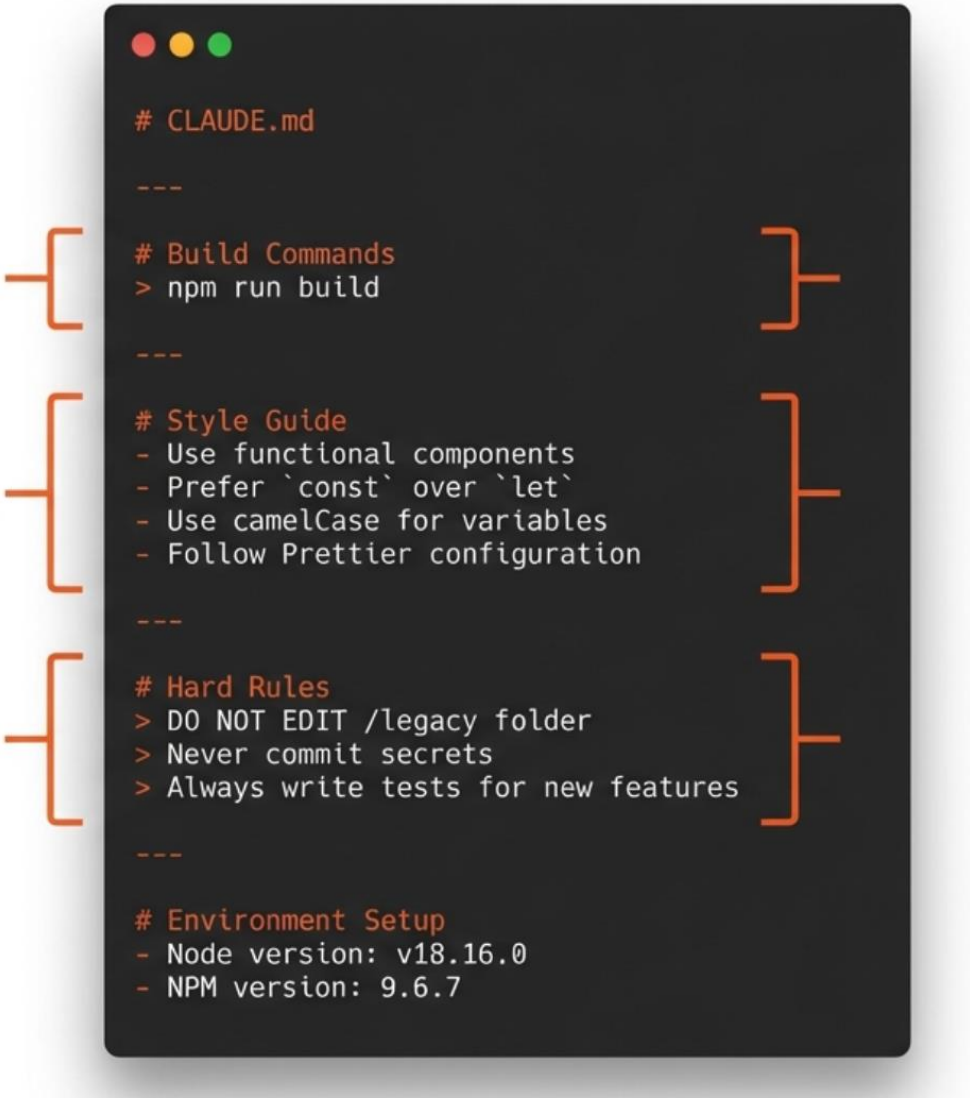
Claude Code: The Beloved One


```
> claude code "Analyze my project structure and
suggest improvements for modularity."
Analyzing project...
[✓] Read 45 files in src/
[✓] Identified potential circular dependencies.

Suggestions:
1. Extract `utils` from `main.js`.
2. Create `components/` folder.
3. Refactor `api.js` into smaller services.

Would you like me to apply any of these changes? [Y/n]
> _
```

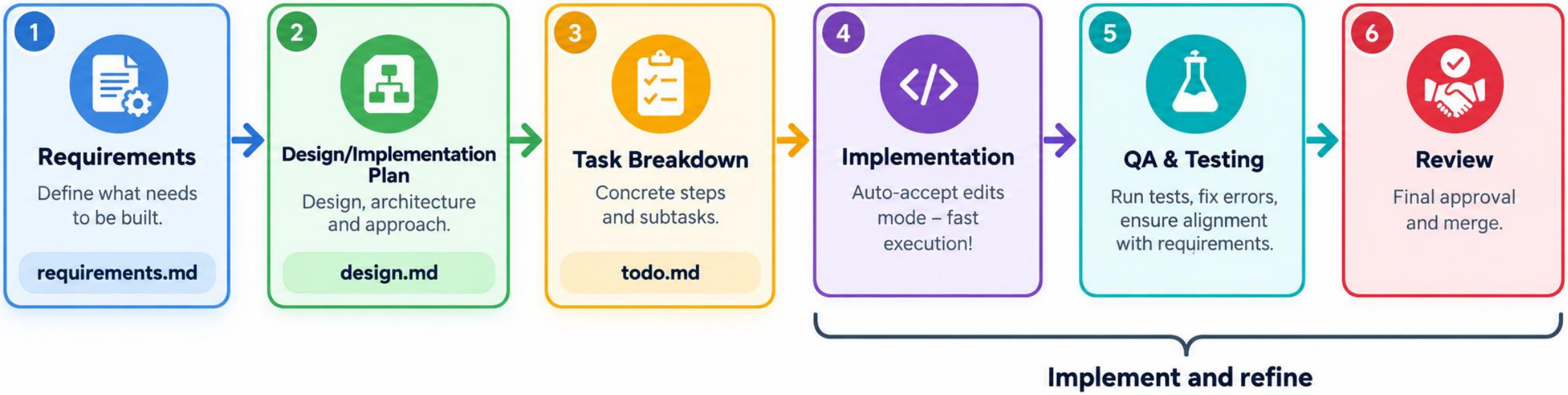
CLAUDE.md: Project constitution



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

SDD with Claude Code 1/2



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

SDD with 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

GitHub Spec Kit vs. Claude Code

	GitHub Spec Kit	Claude Code
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)

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 open 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**

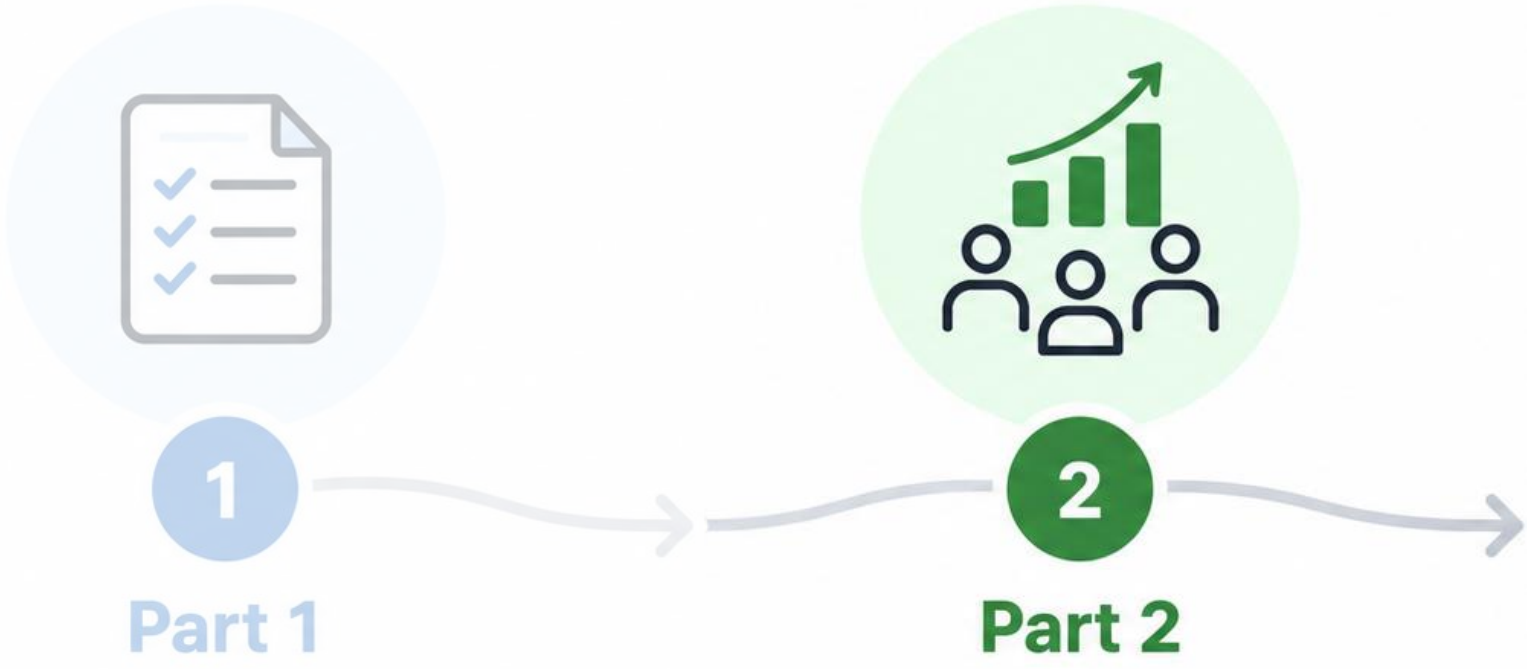
LLM-generated files **decrease success rates** and **increase inference costs** by 20%

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

Are people actually doing Spec-Driven Development?

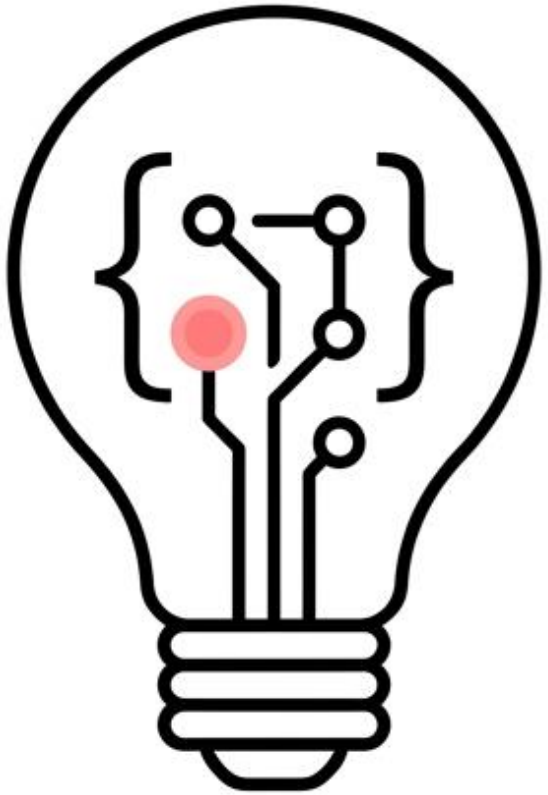
or is it just hype?

Today’s Agenda



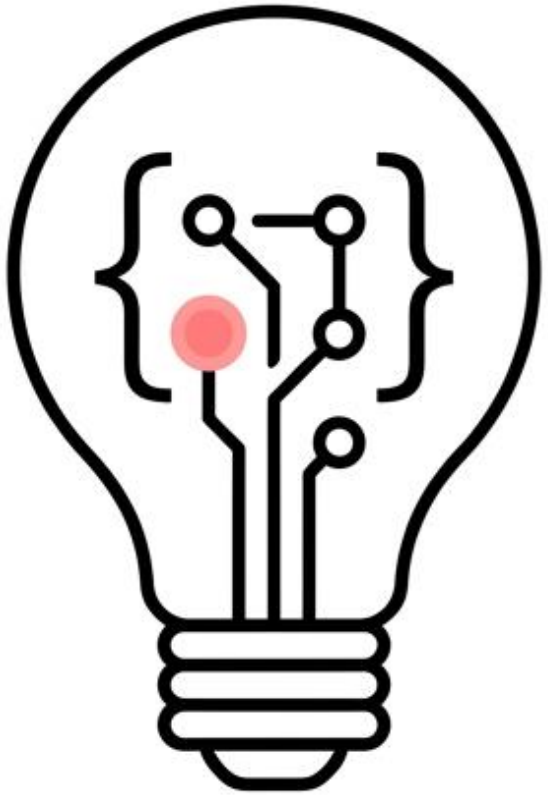
Spec-Driven Development

Exploring SDD adoption in the open-source



How to identify repositories using SDD?

or at least tried?

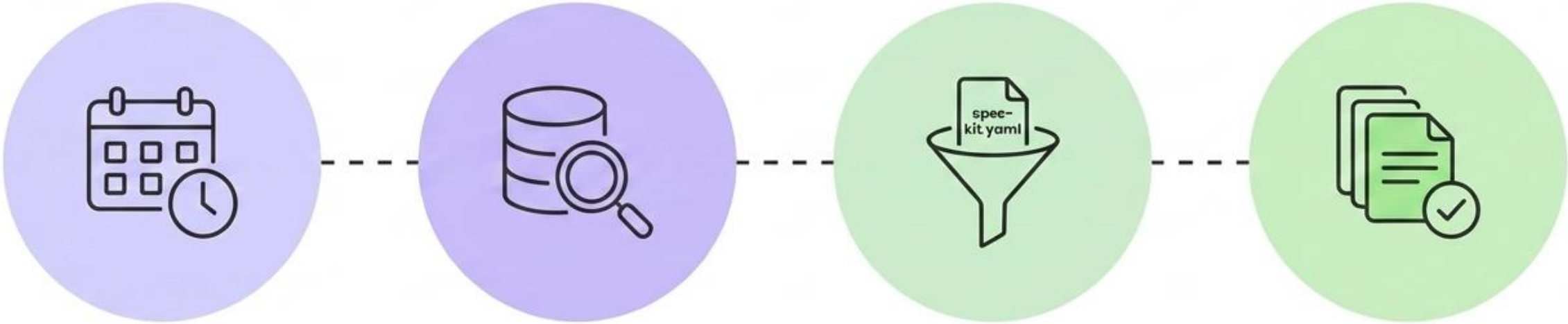


How to identify repositories using SDD?

or at least tried?

Let’s focus on GitHub Spec Kit!

Mining Process Outline



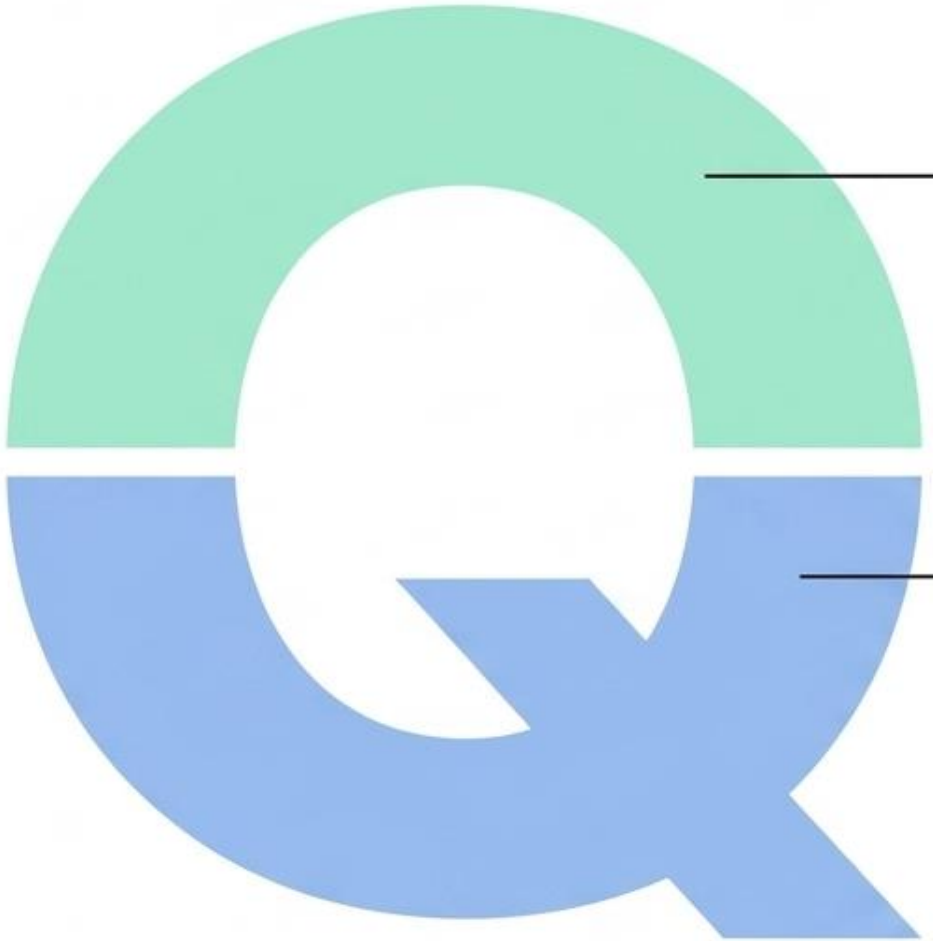
Time window
until May 2026

Code Search API
6,782 having
constitution.md

Filter Criteria
> 8 commits

Matching found
4,642 repos

Framing the Explorative Analysis



RQ1: Demographics & Usage Status

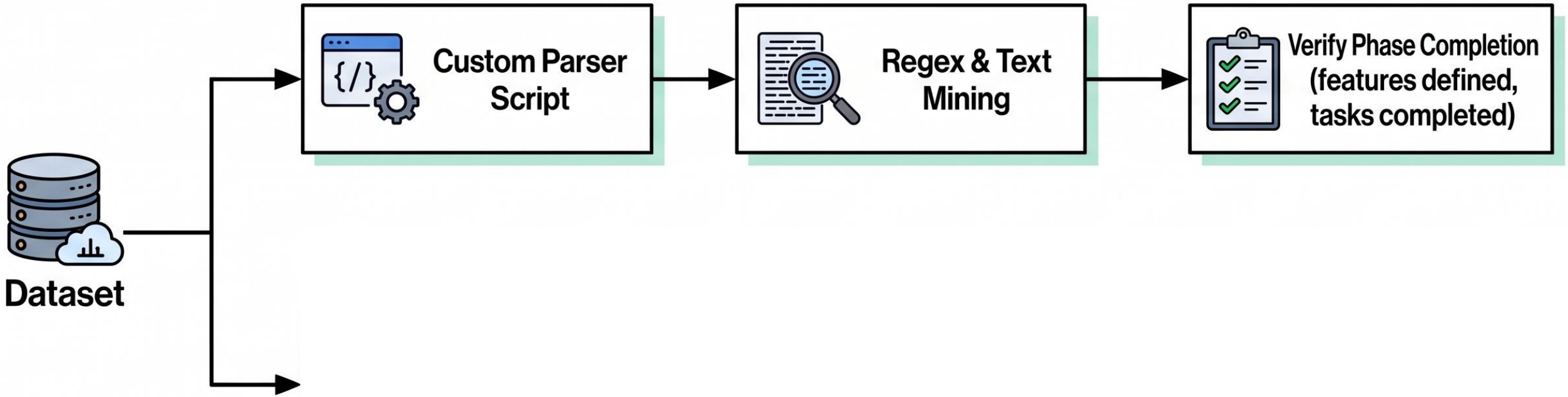
What are the characteristics of open-source projects adopting Spec Kit? How far do they progress in the structured workflow?

RQ2: Constitution Structure & Customization

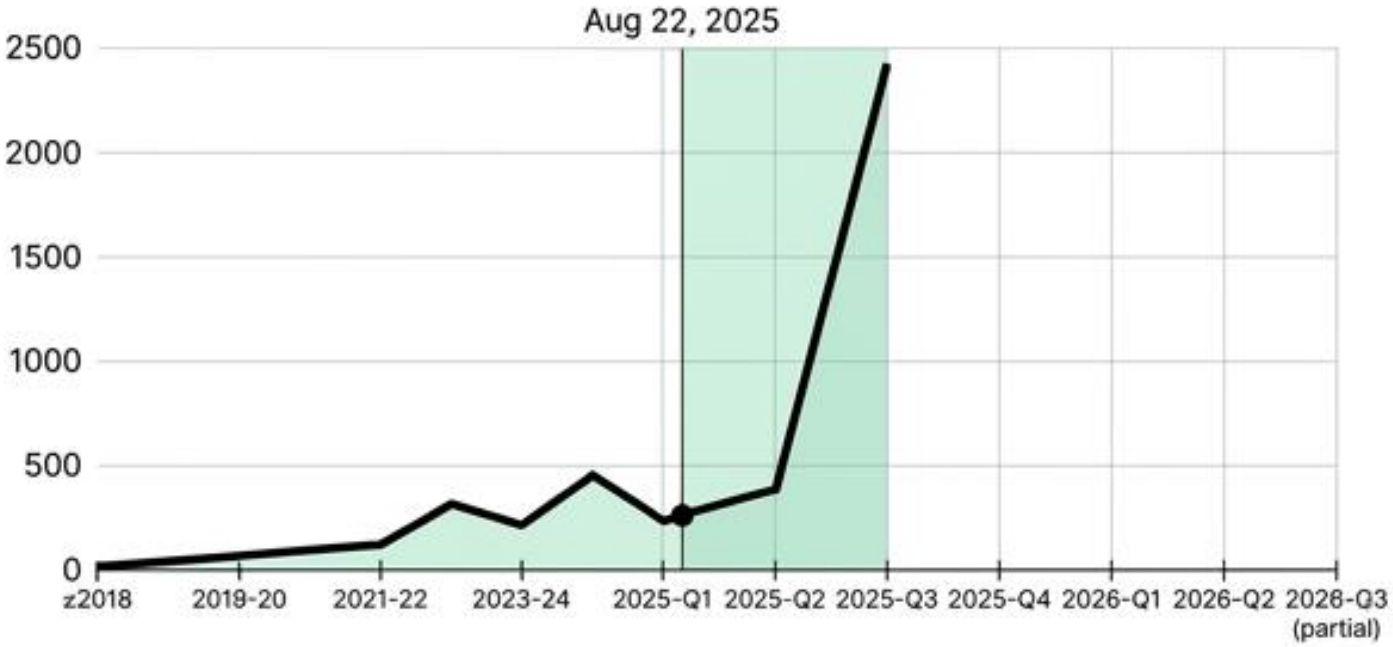
How do developers structure and customize the **constitution.md** core artifact to guide their AI agents?

Analysis Outline 1/2

Track 1: Demographics

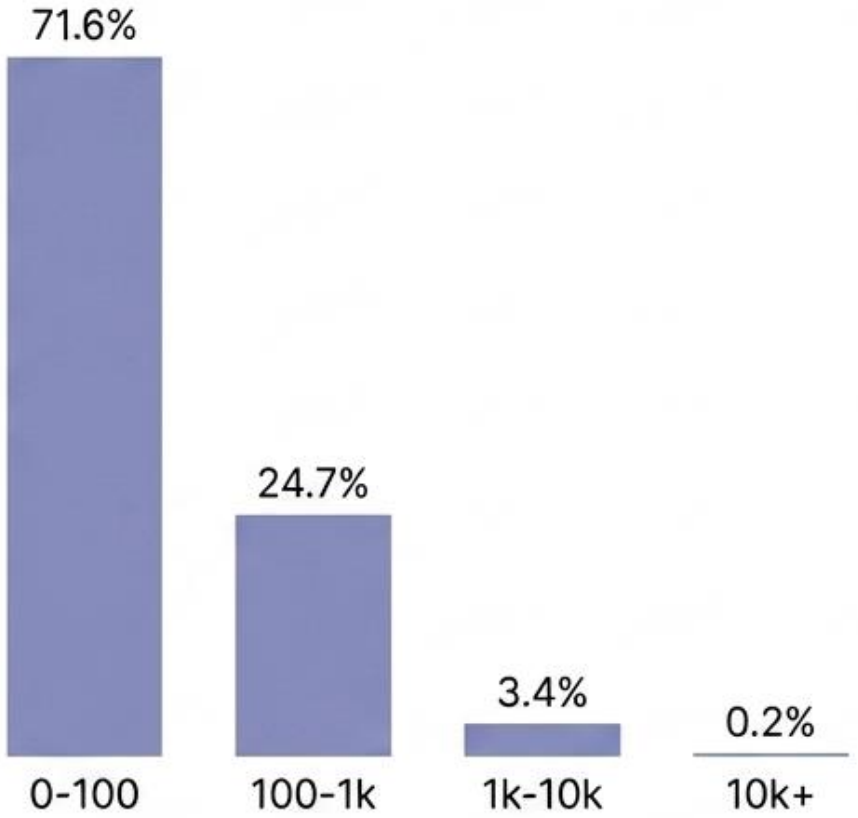


Fast growing and early-adopters



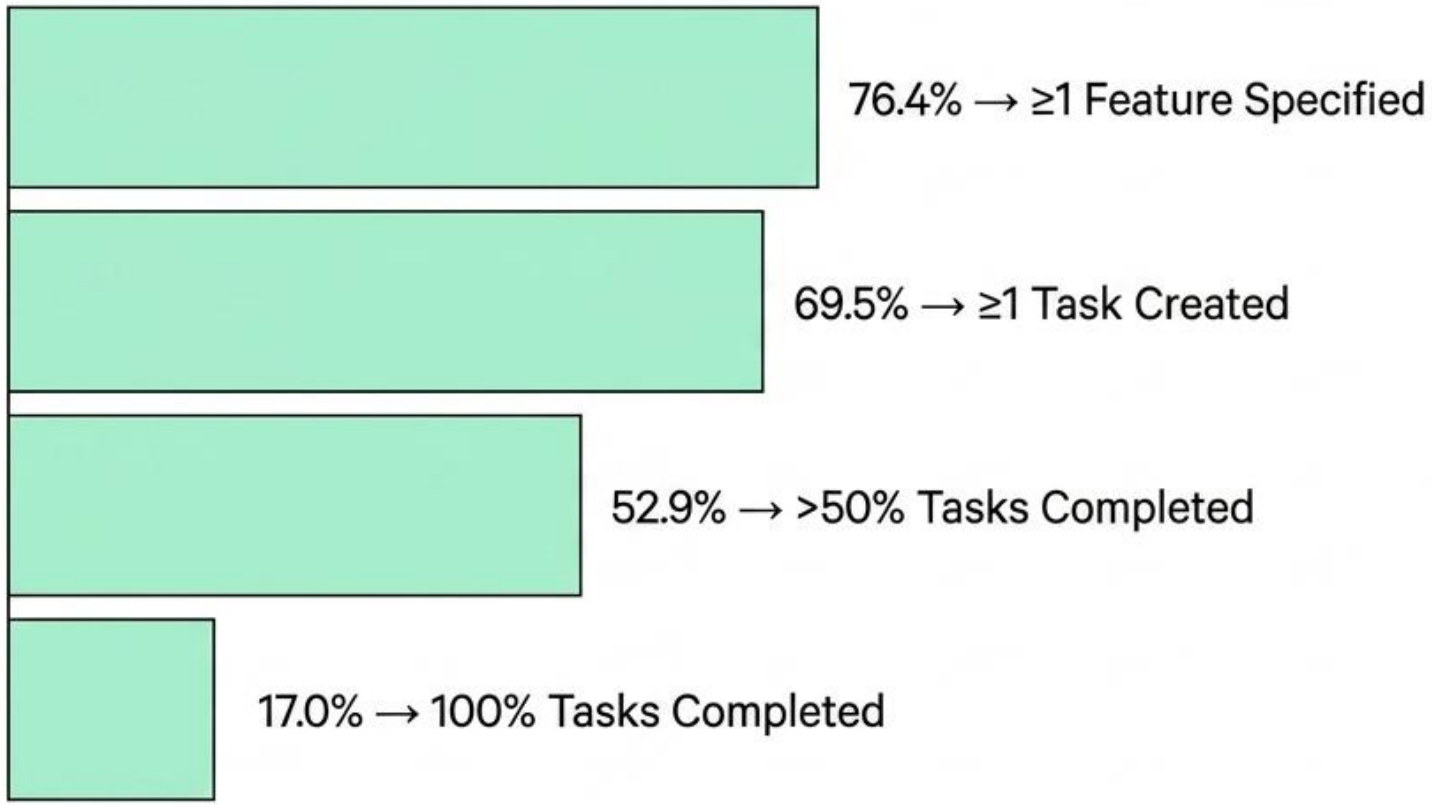
Project creation trend

avg. project age is 50 days



Project size by LOC

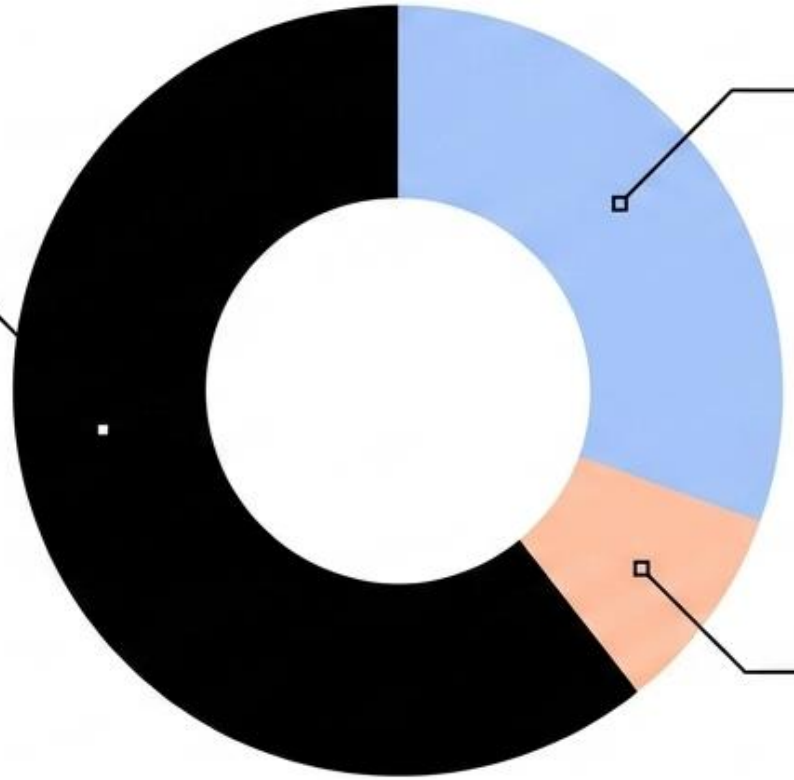
SDD implementation status



~50%

projects are at the task implementation phase

Agent Customization



33.1%
GitHub Copilot
(Default)

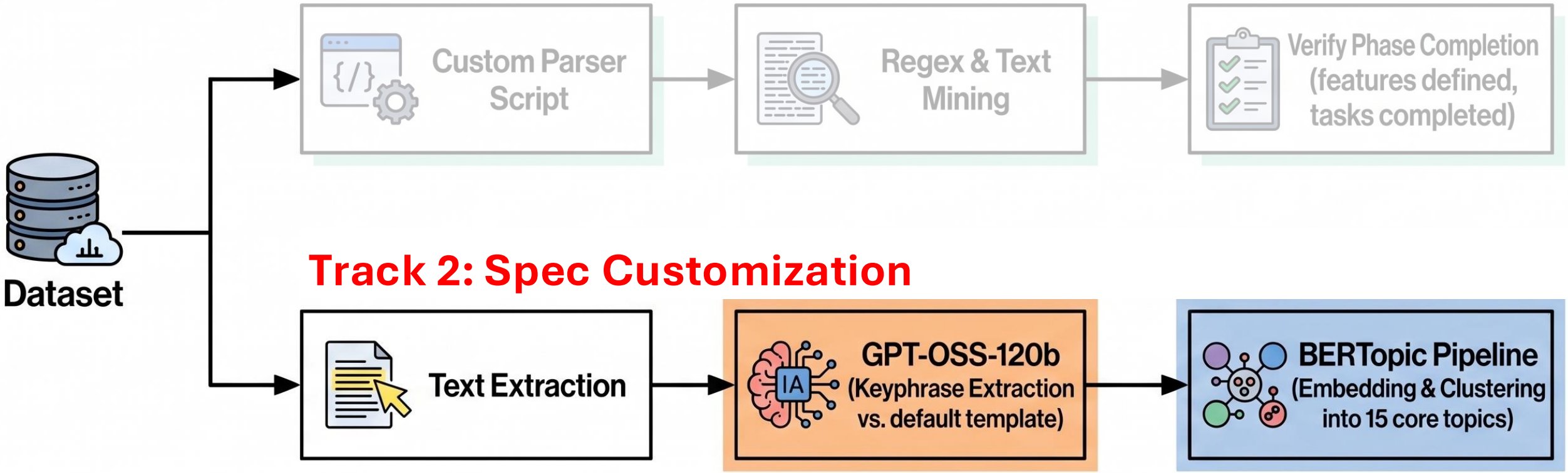
63.0%
Custom / Other
Non-Default Agents

9.6%
Claude Code

13%

of the projects feature custom skills

Analysis Outline 2/2



Sections

Principles

Core Principles

I. Code Quality

All code MUST meet these non-negotiable quality standards:

- **Readability**: Code MUST be self-documenting with clear naming conventions. Variable and function names MUST express intent without requiring comments to understand basic functionality.
- **Maintainability**: Functions MUST have a single responsibility. Files MUST NOT exceed 300 lines without architectural justification. Cyclomatic complexity MUST remain below 10 per function.
- **Consistency**: All code MUST follow the project's established style guide and formatting rules. Linting MUST pass with zero warnings before merge.
- **Documentation**: Public APIs MUST have documentation. Complex algorithms MUST include inline comments explaining the "why" not the "what".
- **No Dead Code**: Unused imports, variables, and functions MUST be removed. Commented-out code MUST NOT be committed.

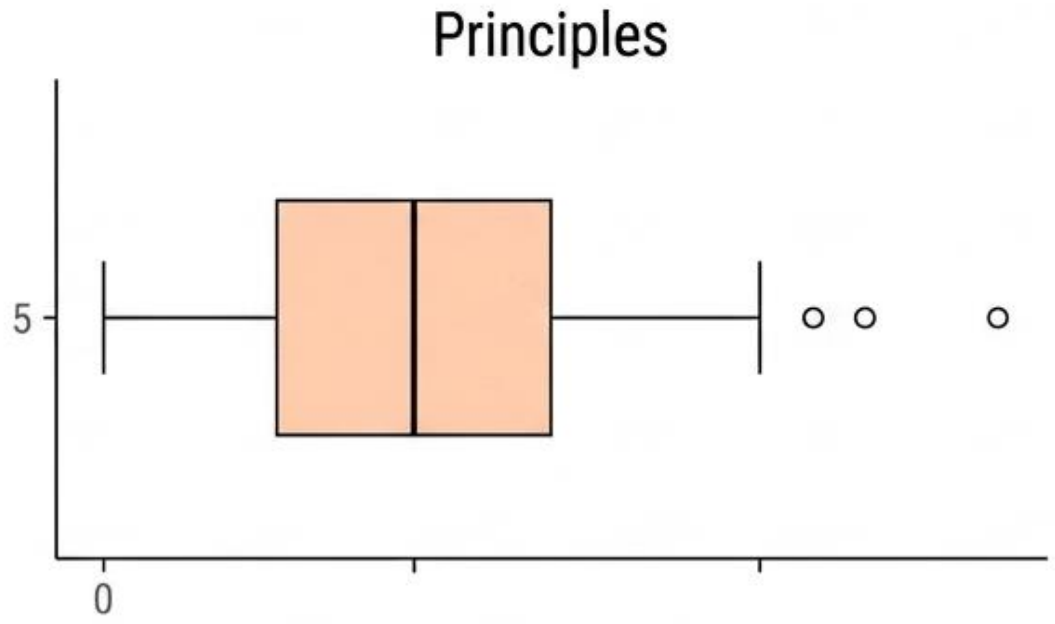
Rationale: High code quality reduces technical debt, accelerates onboarding, and minimizes bug introduction during changes.

II. Testing Standards

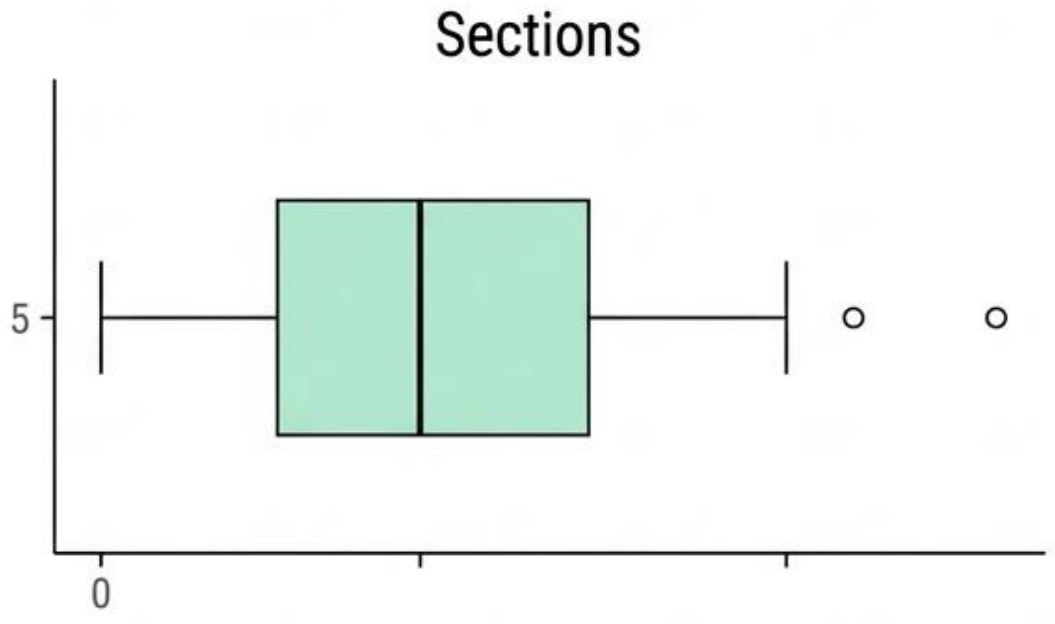
Testing is mandatory and MUST follow these requirements:

- **Coverage Threshold**: New code MUST have minimum 80% line coverage. Critical paths (authentication, data mutations, payments) MUST have 100% coverage.
- **Test Types Required**:
  - Unit tests for all business logic and utility functions
  - Integration tests for API endpoints and database operations
  - Contract tests for external service integrations
- **Test Quality**: Tests MUST be deterministic (no flaky tests). Tests MUST be independent and runnable in isolation. Test names MUST clearly describe the scenario being tested.
- **Test-First Encouraged**: For complex features, write tests before implementation (Red-Green-Refactor). All bug fixes MUST include a

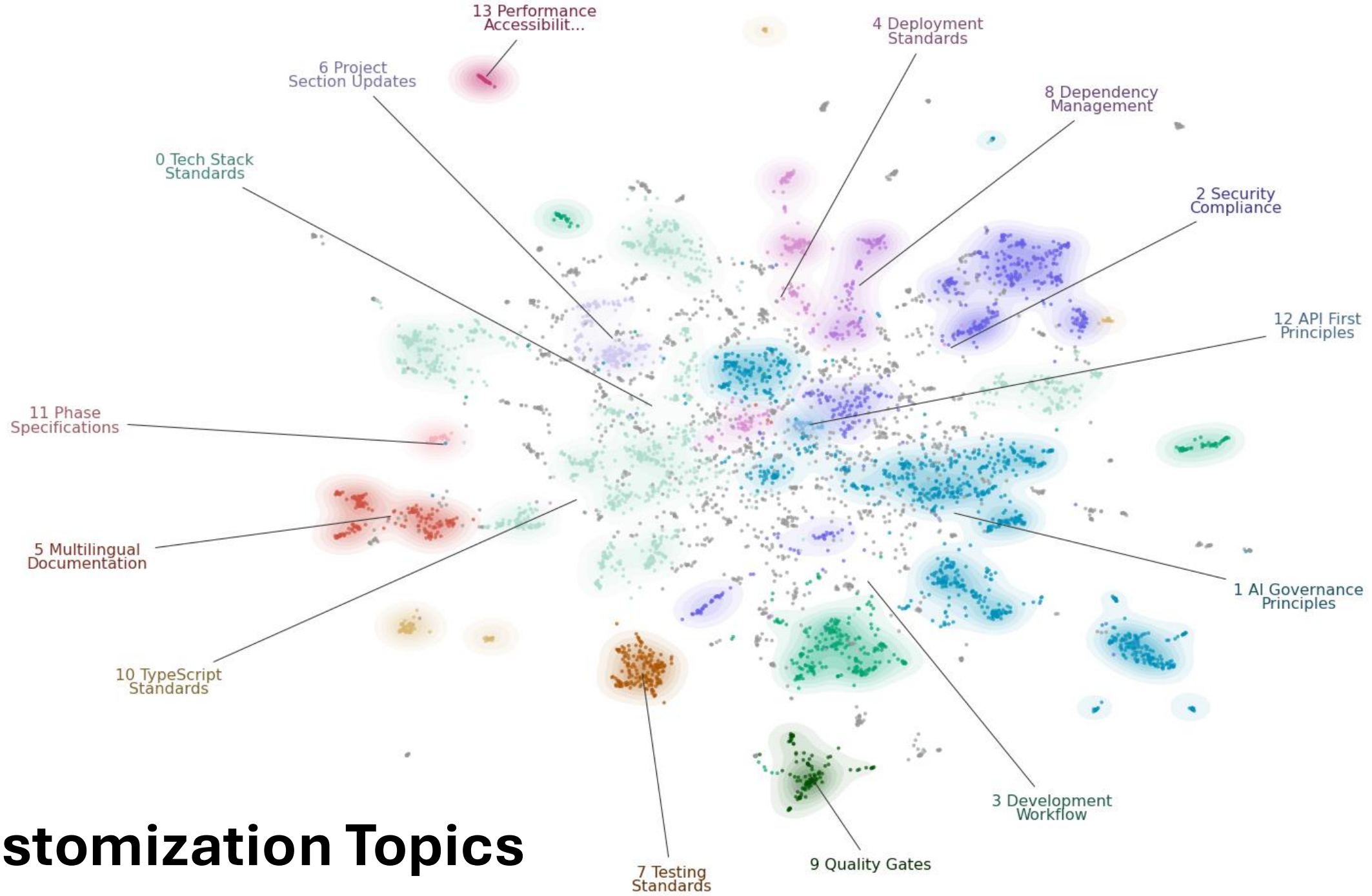
Constitution files structure



Median of 5 principles



4 to 5 sections in mature projects



Customization Topics

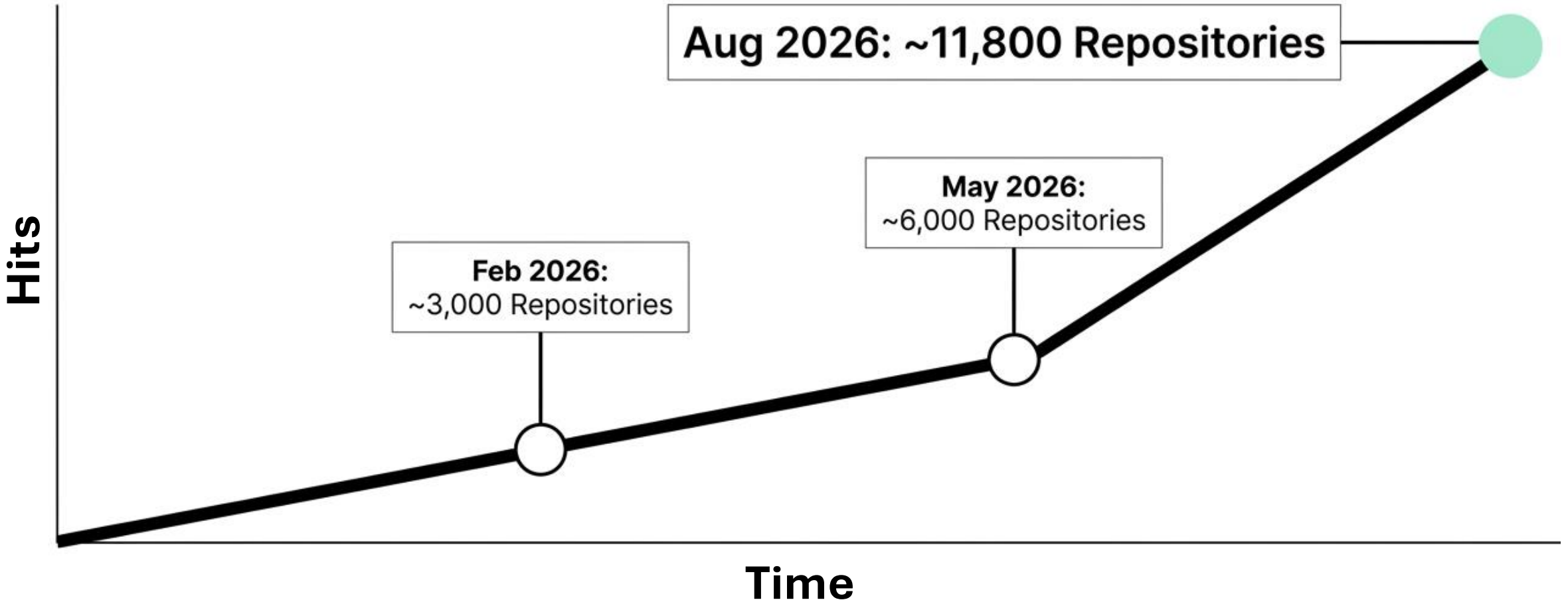
Core Customization Topics

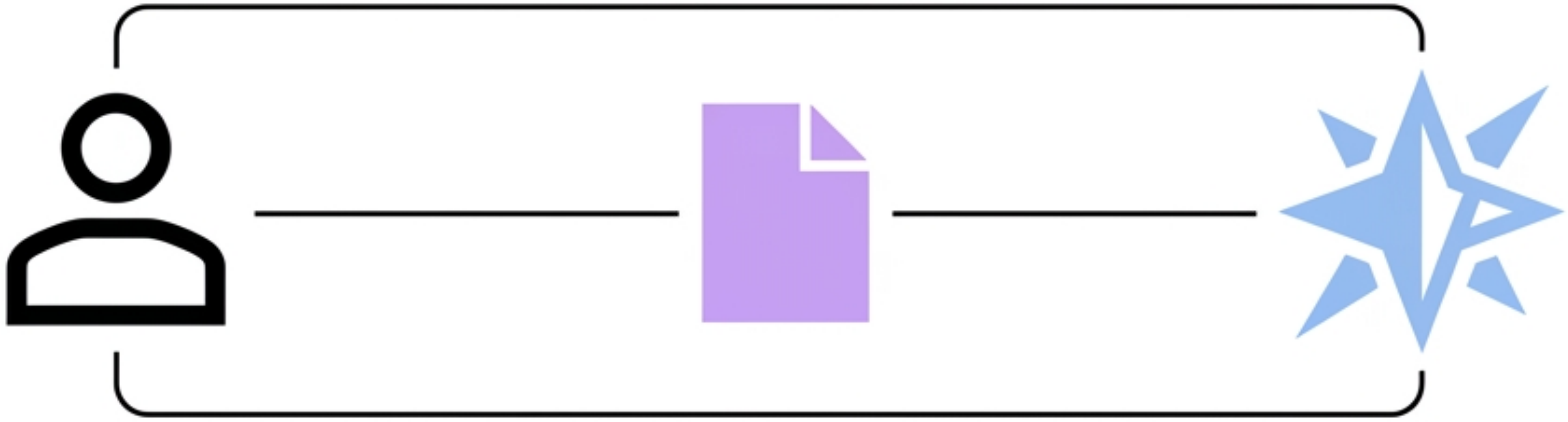
2,445	**Tech Stack Standards** Guidelines for the specific technology stack.
2,097	**AI Governance Principles** Defining rules for human and AI agent interaction.
1,087	**Development Workflow**
711	**Security Compliance**

Customization Topics by Group

	0-100 Commits	100-1K Commits	1K-10K Commits	>10K Commits
Tech Stack Standards	●	●	●	●
AI Governance Principles	●	●	●	●
Multilingual Documentation			●	

More and more repositories over time

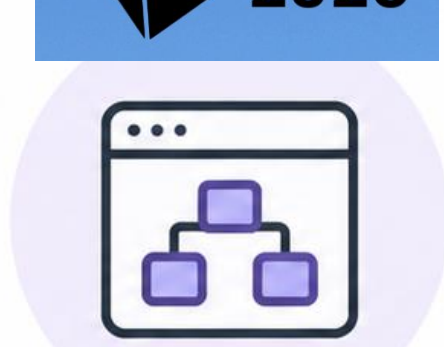




Code is not a first-class citizen anymore

Specs are authored, reviewed and evolved
as deliberately as code

Today's Agenda



1 — Part 1

Spec-Driven Development

2 — Part 2

Exploring SDD adoption in the open-source

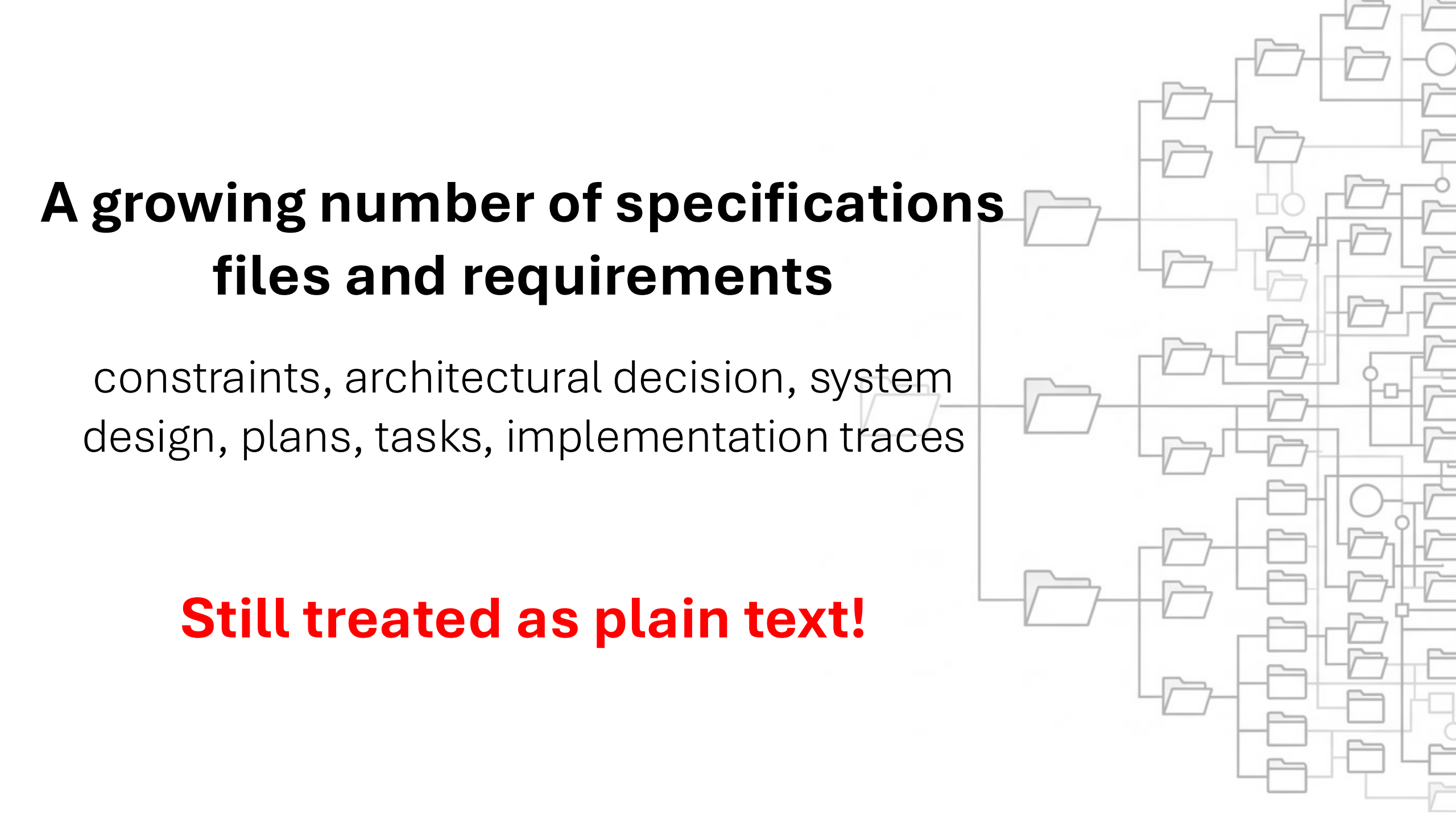
3 — Part 3

Re-imagining Software Visualization for SDD

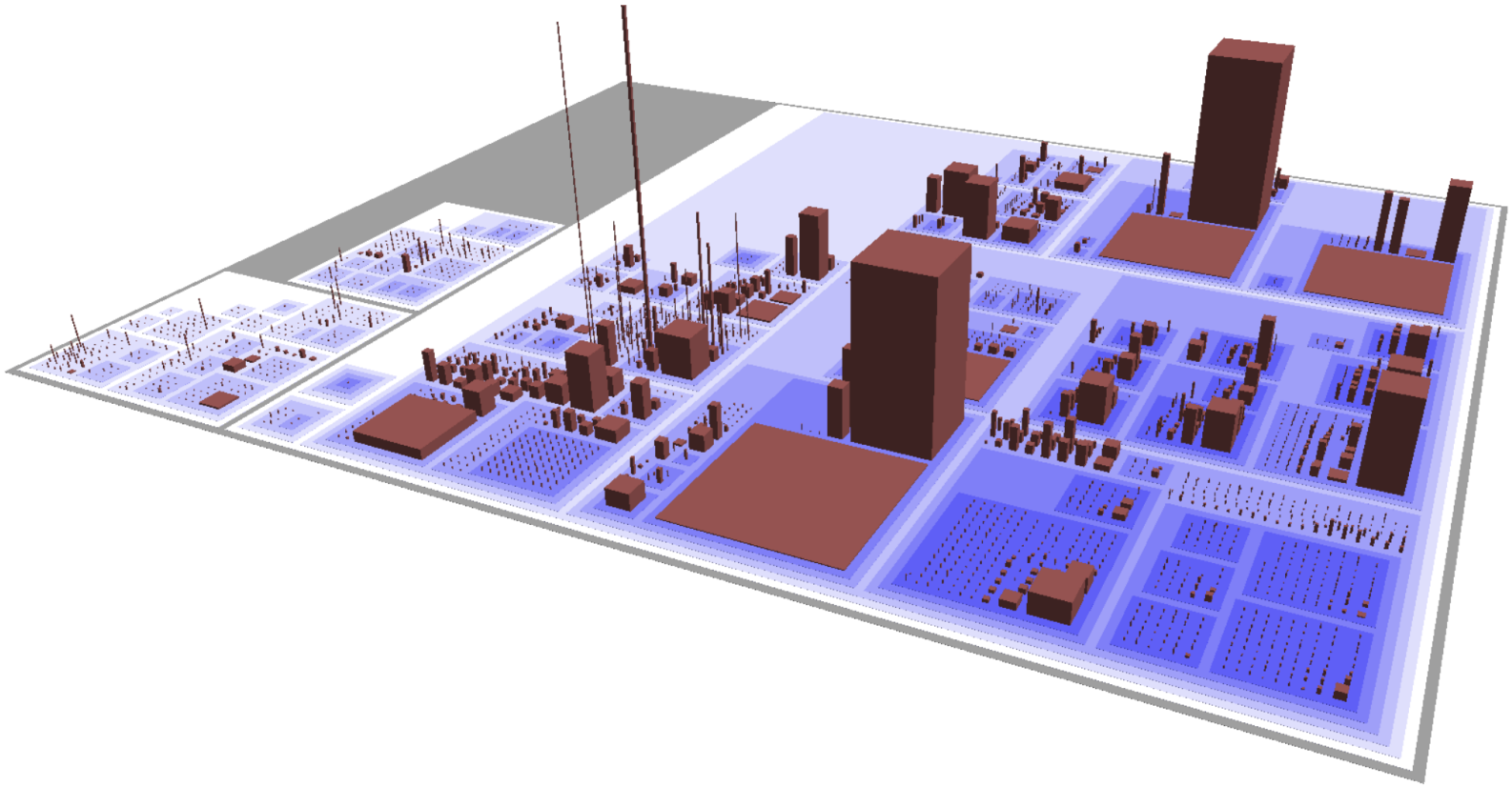
A growing number of specifications files and requirements

constraints, architectural decision, system design, plans, tasks, implementation traces

Still treated as plain text!

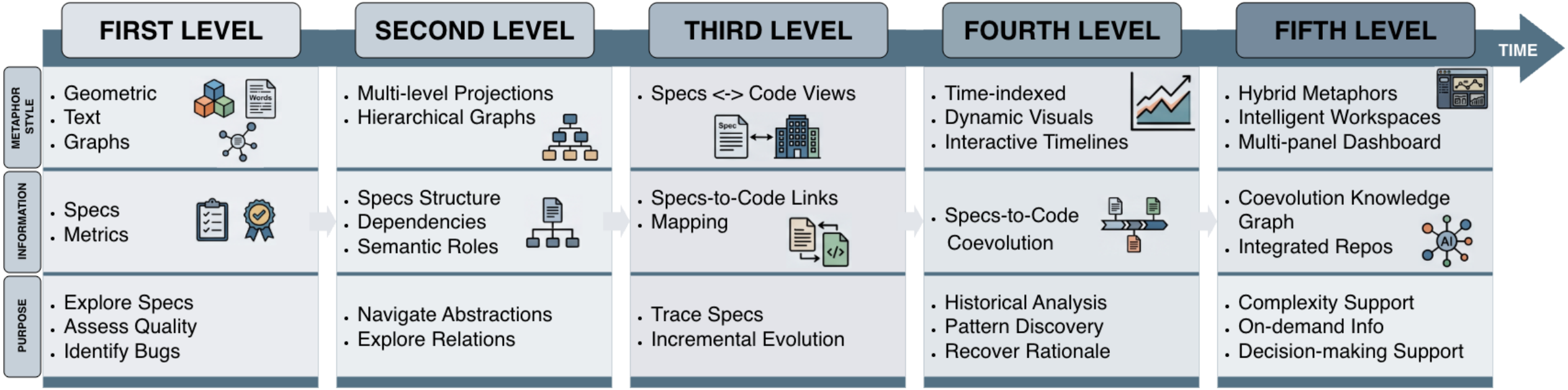


How to adapt visualization metaphors to the co-existence of code and specifications?



Wettel, R., & Lanza, M. (2007, June). Visualizing software systems as cities. In 2007 4th IEEE International workshop on visualizing software for understanding and analysis (pp. 92-99). IEEE.

Envisioning a multi-level visualization framework



Rosa, G., Linares-Vásquez, M., & Robles, G. (2026). Re-imagining software visualization for spec-driven development. In 2026 IEEE 14th Working Conference on Software Visualization (VISSOFT). IEEE.

Level 1 : Assessing Structure

	FIRST LEVEL	SECOND LEVEL	THIRD LEVEL	FOU
METAPHOR STYLE	• Geometric • Text • Graphs	• Multi-level Projections • Hierarchical Graphs	• Specs <-> Code Views	• Time-in • Dynami • Interact
INFORMATION	• Specs • Metrics	• Specs Structure • Dependencies • Semantic Roles	• Specs-to-Code Links • Mapping	• Specs-t Coevol
PURPOSE	• Explore Specs • Assess Quality • Identify Bugs	• Navigate Abstractions • Explore Relations	• Trace Specs • Incremental Evolution	• Historic • Pattern • Recove

Rosa, G., Linares-Vásquez, M., & Robles, G. (2026). Re-imagining software visualization for spec-driven development. In 2026 IEEE 14th Working Conference on Software Visualization (VISSOFT). IEEE.

Level 2 : Semantic Roles

EVEL

SECOND LEVEL

THIRD LEVEL

FOURTH LEVEL

- Multi-level Projections
- Hierarchical Graphs

- Specs Structure
- Dependencies
- Semantic Roles

- Navigate Abstractions
- Explore Relations

- Specs <-> Code Views

- Specs-to-Code Links
- Mapping

- Trace Specs
- Incremental Evolution

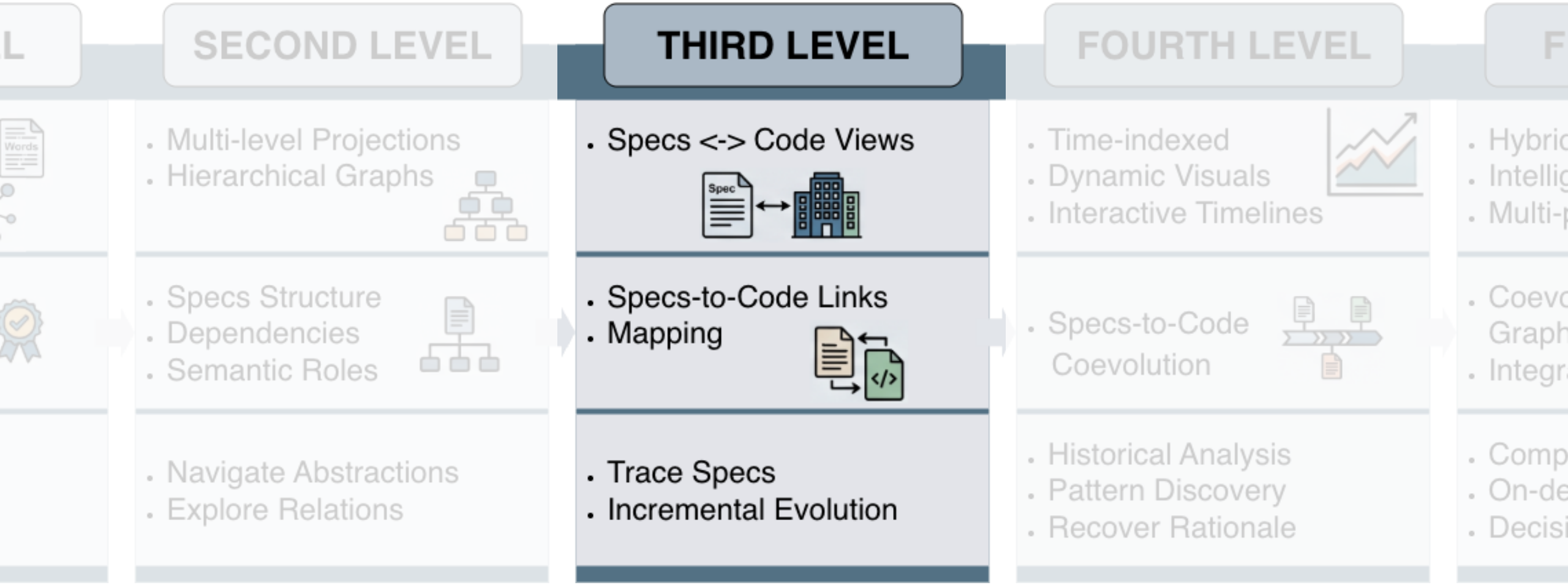
- Time-indexed
- Dynamic Visuals
- Interactive Timelines

- Specs-to-Code Coevolution

- Historical Analysis
- Pattern Discovery
- Recover Rationale

Rosa, G., Linares-Vásquez, M., & Robles, G. (2026). Re-imagining software visualization for spec-driven development. In 2026 IEEE 14th Working Conference on Software Visualization (VISSOFT). IEEE.

Level 3: Mapping Specs to Code



Rosa, G., Linares-Vásquez, M., & Robles, G. (2026). Re-imagining software visualization for spec-driven development. In 2026 IEEE 14th Working Conference on Software Visualization (VISSOFT). IEEE.

Level 4: Historical Analysis

SECOND LEVEL

- Multi-level Projections
- Hierarchical Graphs

- Specs Structure
- Dependencies
- Semantic Roles

- Navigate Abstractions
- Explore Relations

THIRD LEVEL

- Specs <-> Code Views

- Specs-to-Code Links
- Mapping

- Trace Specs
- Incremental Evolution

FOURTH LEVEL

- Time-indexed
- Dynamic Visuals
- Interactive Timelines

- Specs-to-Code Coevolution

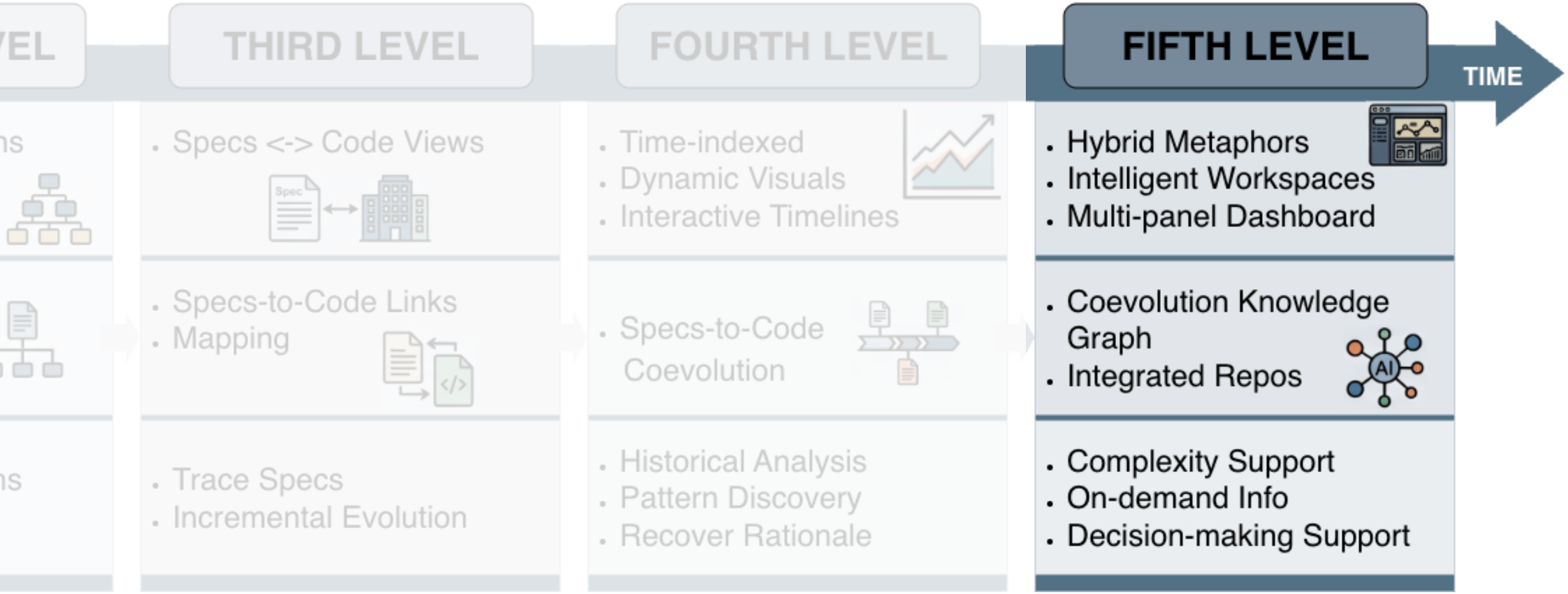
- Historical Analysis
- Pattern Discovery
- Recover Rationale

FIFTH

- Hybrid Metap
- Intelligent Wo
- Multi-panel D
- Coevolution
  Graph
- Integrated Re
- Complexity S
- On-demand
- Decision-mal

Rosa, G., Linares-Vásquez, M., & Robles, G. (2026). Re-imagining software visualization for spec-driven development. In 2026 IEEE 14th Working Conference on Software Visualization (VISSOFT). IEEE.

Level 5: Intelligent Workspaces

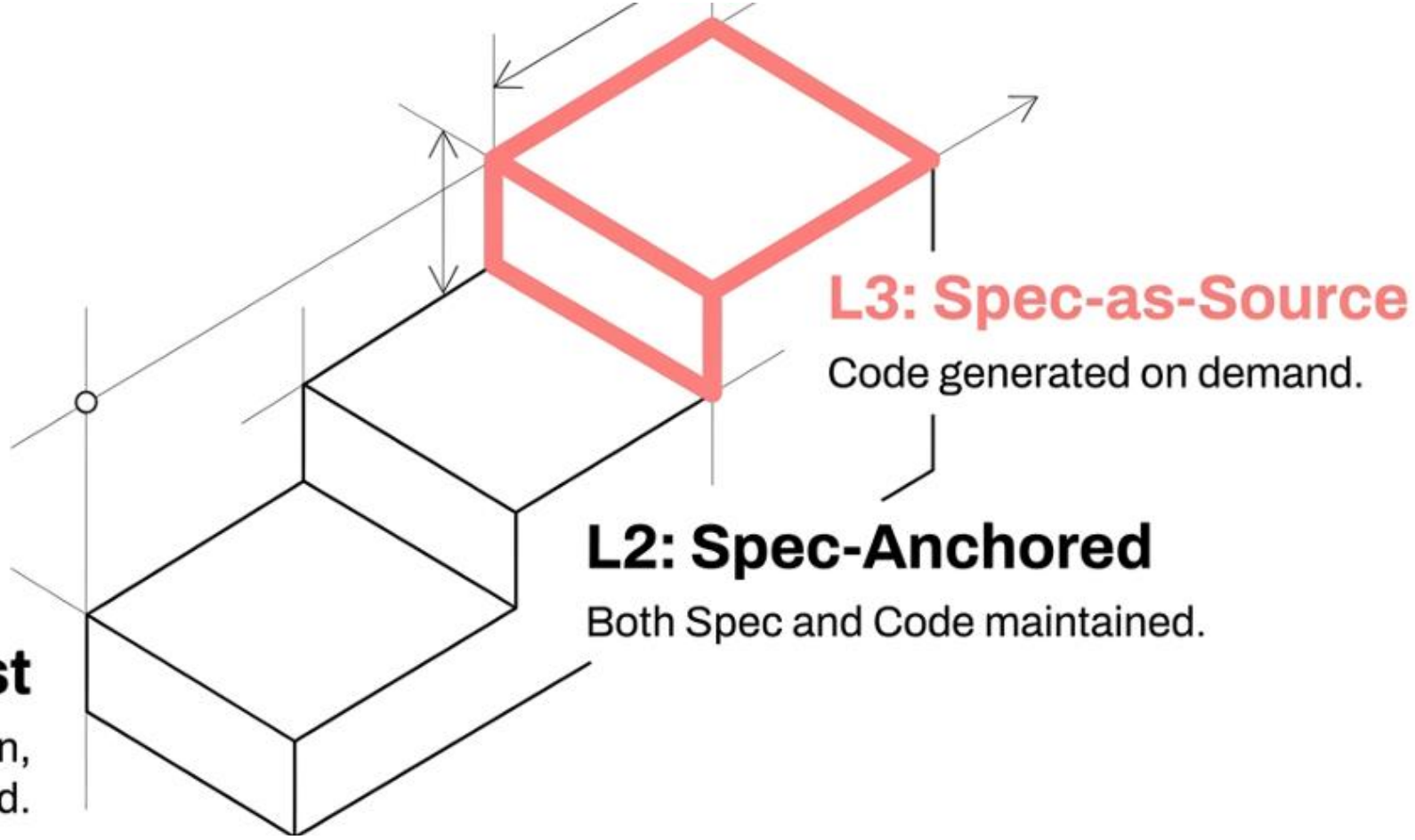


Rosa, G., Linares-Vásquez, M., & Robles, G. (2026). Re-imagining software visualization for spec-driven development. In 2026 IEEE 14th Working Conference on Software Visualization (VISSOFT). IEEE.

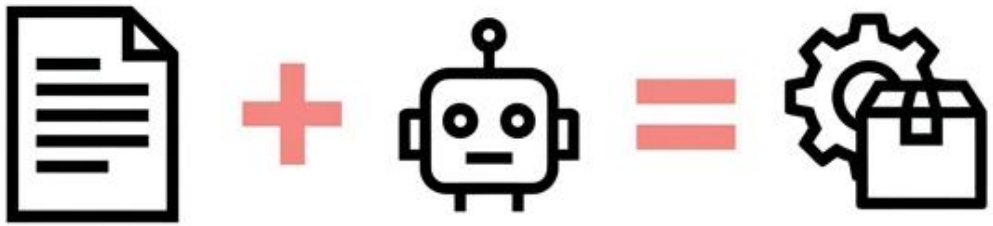
Traceability links are still a weak point!

Agent traces are not generated by default

Different Levels of Spec-based Code Generation



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

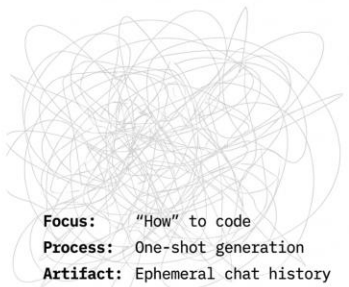


The Spec is the new source of truth

is the code becoming a disposable product?

Summary

Towards Spec-Driven Development



Focus: "How" to code
Process: One-shot generation
Artifact: Ephemeral chat history

Ad-hoc AI

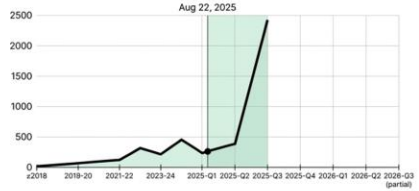
Focus: "What & Why" to build
Process: Multi-step validation
Artifact: Executable Specification

SDD

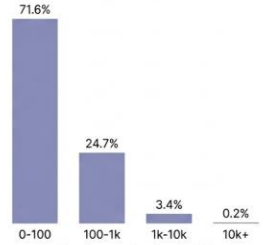
GitHub Spec-Kit

Constitution → Specify → Plan → Tasks

Fast growing and early-adopters



Project creation trend
(avg. age: 50 days)

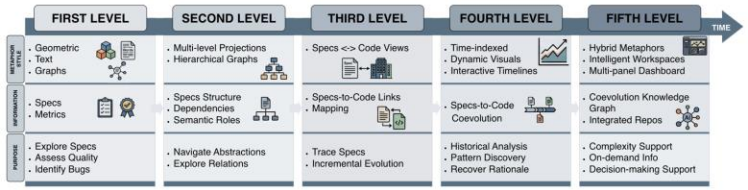


Project size by LOC

Core Customization Topics

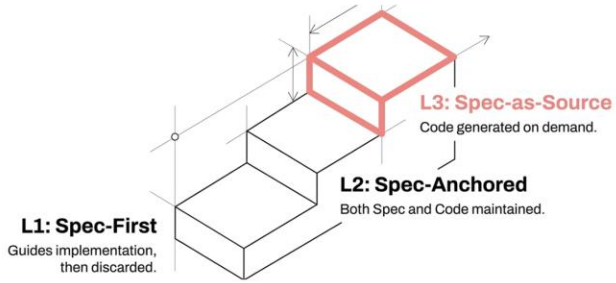
2,445	**Tech Stack Standards** Guidelines for the specific technology stack.
2,097	**AI Governance Principles** Defining rules for human and AI agent interaction.
1,087	**Development Workflow**
711	**Security Compliance**

Envisioning a multi-level visualization framework



Rosa, G., Linares-Vásquez, M., & Robles, G. (2026). Re-imagining software visualization for spec-driven development. In 2026 IEEE 14th Working Conference on Software Visualization (VISSOFT). IEEE.

Different Levels of Spec-based Code Generation



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

Giovanni Rosa
Postdoctoral Researcher @ URJC
More at **giovannirosa.com**

SoftDev

URJC

ADVISE

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/

AI Notice: AI tools were used to support the creation of text and images. All content was reviewed and revised by the author.