

EXPERIMENT REPORT

# The AI Engineering Experiment

A Technical Deep Dive into Agentic Software Development

AUTHOR

Steffen Eichenberg

METHOD

OpenCode Agentic System

LLM MODEL

Claude Sonnet 4.5

# Executive Summary: The 1000€ Bet

---

Can a single engineer, leveraging an agentic AI system, replicate the output of a full development team in a fraction of the time? The constraints were strict.

10

DAYS DURATION

20

TOTAL WORK HOURS

65k

TOTAL LINES OUTPUT

470

GIT COMMITS

✓ Result: Production-ready software, not a prototype.

# The Enemy: "Vibe Coding"

---

## The Definition

Vibe coding is the standard interaction mode with current LLMs:

- ✗ **Non-deterministic:** Different results every time.
- ✗ **Non-repeatable:** Cannot be automated or scaled.
- ✗ **Unmaintainable:** "Spaghetti code" that works only once.

## The Goal

Move from **stochastic generation** to **deterministic engineering**.

---

We need a repeatable process where the code is a side-effect of a structured thought process.

# The Hypothesis

---

## Scientific Method

If we treat AI as an **agent** rather than a **chatbot**, we can enforce:

- Strict architectural patterns
- Test-Driven Development (TDD)
- Documentation-first methodology

## Control Theory

By implementing a **Supervisor-Worker** model:

- High-level logic is separated from implementation details.
- Context drift is minimized by scoping tasks.
- Validation loops catch hallucinations early.

# Target Architecture: Proof of Complexity

This was not a "To-Do List" tutorial app. The target was a complex, production-grade desktop application.

## Frontend

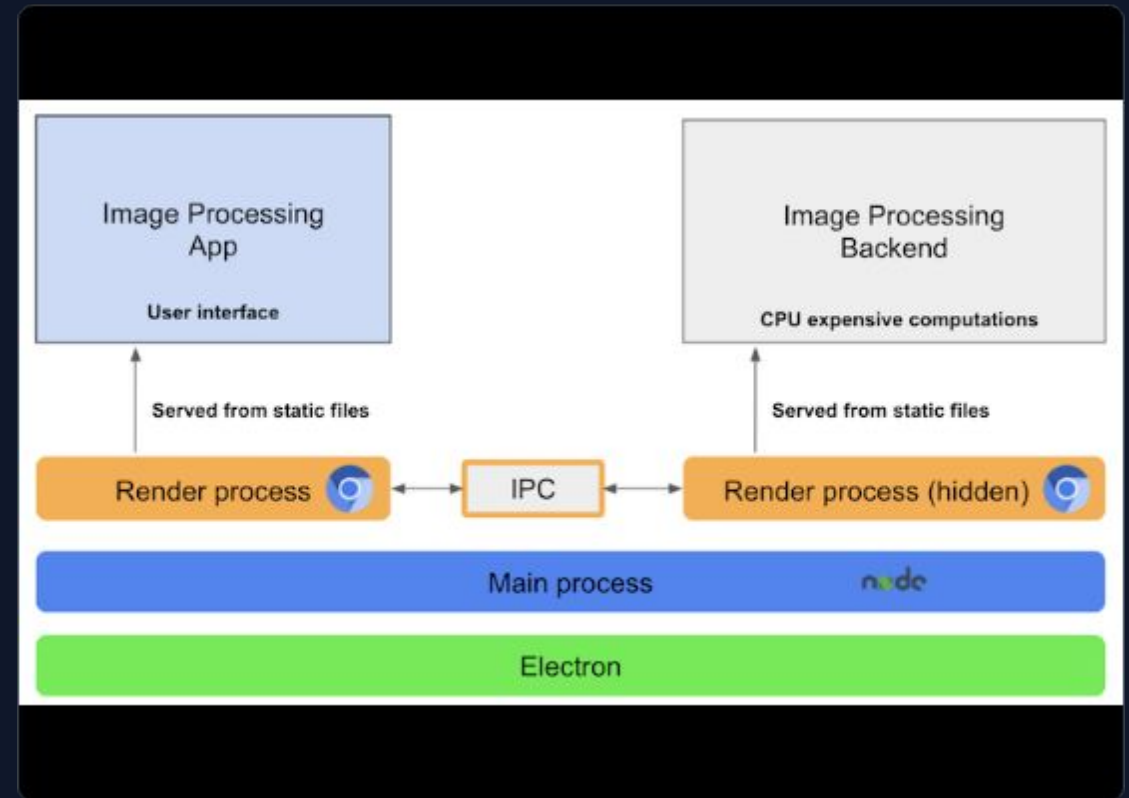
React 18 • Vite 6 • Tailwind 4 • XState

## Desktop Shell

Electron (Main/Renderer Process) • IPC Security

## Data Layer

SQLite • Kysely (Type-safe SQL) • Dual-client setup



# The Orchestrator: OpenCode

---

Why OpenCode? In an ecosystem of proprietary "Magic Boxes" (Cursor, GitHub Copilot), OpenCode offers strict control:



## Model Agnostic

Swap between OpenAI, Anthropic, or Local LLMs without changing the toolchain.



## No Vendor Lock-in

Open Source community driven. The process defines the success, not the tool.



## Agentic Control

Allows defining custom "Tools", "Skills" and "Agents" that the AI can execute autonomously.

# Methodology: The "Triangle of Truth"

---

## Code is NOT the Documentation

The core failing of most AI projects is relying on the code to be the single source of truth. We inverted this.

### 1. Documentation

The "Why" and "What". The intent must exist before the code.

### 2. Validation

Tests define the boundary conditions of success.

### 3. Structure

Architecture and patterns allow for the necessary abstraction.



# The Agents

---

**OpenCode is process agnostic. Bring-You-Own-Agent is the core philosophy**

The Agents to model the process being used in this experiment had to be developed first.

They have been developed within a few hours using the only two build-in agents of OpenCode (plan and build).



# Phase 1: The Planning Chain

We do not start coding. We start thinking. A chain of specialized agents transforms abstract ideas into concrete tasks.

## 💡 Idea Agent

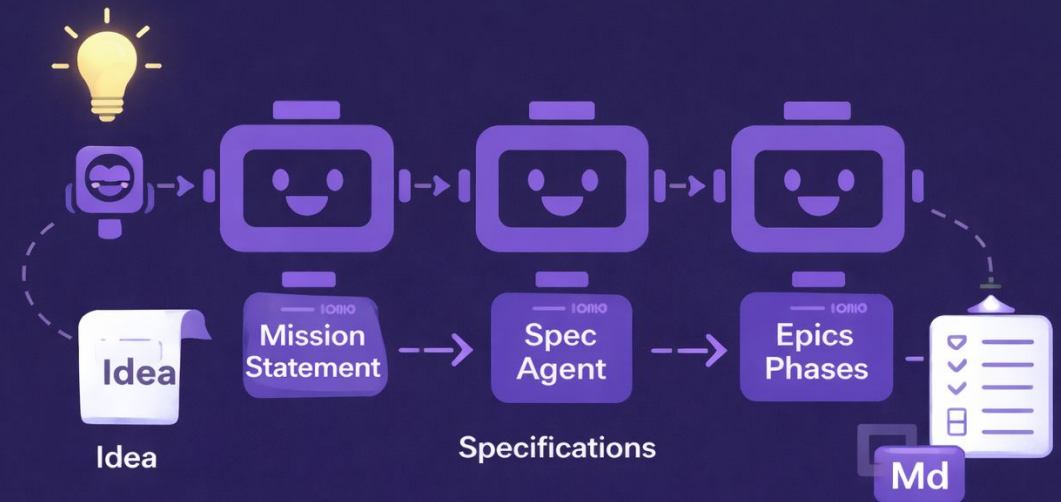
Input: Rough notes.  
Output: **Mission Statement**.

## 📋 Spec Agent

Input: Mission Statement.  
Output: **Technical Specifications**.

## 👥 Epic Agent

Input: Specs.  
Output: **Epics & Phases (Markdown)**.



# Deep Dive: The Specification Agent

## Why it matters




LLMs are notoriously bad at implicit constraints. The Spec Agent forces **Acceptance Criteria** to be explicit.

Instead of "Make it fast", the spec becomes:

- Must render 1000 rows in < 100ms
- Must support offline mode
- Must pass WCAG 2.1 AA

## Artifact Generation

All output is saved to the **thoughts/** directory.

-  thoughts/shared/mission.md
-  thoughts/shared/specs.md
-  thoughts/shared/epics/\*.md

\* This creates a permanent context memory for future agents.

# Phase 2: The Implementation Loop

---

A recursive cycle executed for each Phase of an Epic.



## Research

Find libraries, patterns & existing code.



## Plan

Create step-by-step implementation plan.



## Implement

Execute plan via sub-agents.



## Validate

Run tests & static analysis.

# Agent Hierarchy: The Manager Layer

---

The "Managers" hold the high-level context and delegate actual work. They do not write code directly.

## The Planner

Takes an Epic + Research inputs. Outputs a detailed, step-by-step plan.md.

## Implementation Controller

Reads the plan. Instantiates "Task Executor" sub-agents for each step. Monitors progress.

## QA Planner

Analyzes output from QA agents. Creates a "fix-it" plan for the Controller.

# Agent Hierarchy: The Worker Layer

---

## The Researcher

Specialized in information retrieval. It does NOT hallucinate code; it finds facts.

- Scouts internet for library documentation.
- Identifies patterns in existing project codebase.
- Finds root causes for test failures.

## QA Agents

Language-specific quality assurance. They act as the "Bad Cop".

- Run linters and static analysis.
- Execute test suites.
- Check for "Code Smells" and structural weakness.

# Sub-Agents: The Specialists

---

## Task Executor

The "Hands". Edits files, runs shell commands. Extremely short lifespan.

## Codebase Locator

The "Cartographer". Maps file structures to find relevant context.

## Codebase Analyzer

Follows data paths and execution flows to understand impact of changes.

## Pattern Finder

Ensures new code matches the style of existing code.

**Why separate them?** To prevent Context Pollution.

# The Context Challenge

---

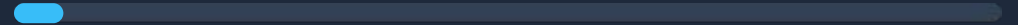
## The Token Limit Reality

Even with Claude Sonnet 4.5's **200k context window**, usability degrades after ~40% usage.

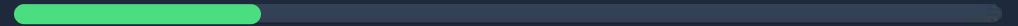
### Symptoms of Drift:

- Forgetting earlier instructions.
- Hallucinating files that don't exist.
- Lazy coding (implementing placeholders).

System Prompt 10k tokens



Project Context 50k tokens



Conversation History (Danger Zone) 100k+ tokens



# Strategy 1: Sub-Agent Delegation

Single Agent + Tools + Router



## Ephemeral Intelligence

Instead of one long conversation, we spawn ephemeral sub-agents.

- 1 **Primary Agent** defines a specific, isolated task.
- 2 **Sub-Agent** is spawned with MINIMAL context (only what's needed).
- 3 **Sub-Agent** performs the task (thinking, searching, coding).
- 4 **Sub-Agent** returns the *result* only.
- 5 **Sub-Agent** dies (discarding its internal thought process tokens).

Result: Primary context stays clean.



# Strategy 2: State File Persistence

---

## The 30% Reset Rule

Whenever context usage hits 30%, the session is **terminated**.

How do we not lose progress?

- We do not rely on Chat History.
- We rely on **State Files** on disk.

```
# thoughts/shared/plans/epic-1-phase-2-state.md

## Current Status
- [x] Task 1: Scaffolding created
- [x] Task 2: Database migration applied
- [ ] Task 3: API Endpoint Implementation ← PAUSED HERE

## Context for Next Session
- Migration file: src/db/migrations/001.ts
- Zod schema: src/types/schema.ts
```

The AI updates this file before terminating. The new session reads it to resume.

# Artifacts: The "Thoughts" Directory

The hidden iceberg of the project. This folder contains the **intelligence**.

Directory	Files	Lines of Content	Purpose
thoughts/shared/plans	42	30,000	Detailed step-by-step execution guides
thoughts/shared/research	20	6,700	Knowledge gathered from web/codebase
thoughts/shared/qa	8	2,600	Bug reports and fix strategies
thoughts/shared/epics	3	1,020	High level project roadmap

 **Insight:** 44,000 lines of planning vs 8,300 lines of production code. This ratio (5:1) is typical of Senior Engineering.

# Artifacts: Testing Pyramid

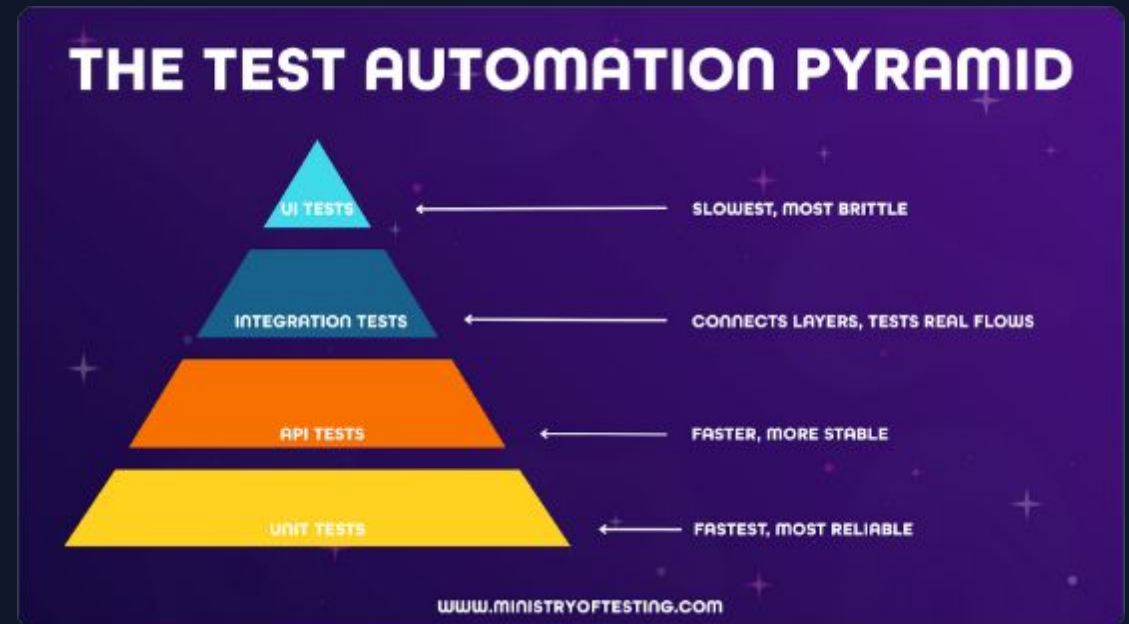
Strict TDD was enforced. The agent was not allowed to mark a task complete without a passing test.

UNIT TESTS

9,050  
LOC

E2E (PLAYWRIGHT)

910  
LOC



# Human Equivalent Estimate



## The 1000 Hour Estimate

Estimated by GPT-5.2 based on complexity.

- **Full-time equivalent:** 6-9 months
- **Actual Time Spent:** 20 hours
- **Efficiency Gain:** ~50x

*"This project does not look like 'someone just started coding', but like clear mental models and deliberate trade-offs."*

# Cost Analysis

---

The Experiment Cost

€1,000

API COSTS (CLAUDE)

Traditional Dev Cost

€100,000+

6 MONTHS @ SENIOR RATE

Even if the API costs were 10x higher, the ROI remains orders of magnitude positive. The constraint is no longer budget, but **architectural oversight**.

# Conclusion for Experts

---

## Shift in Skillset

The role of the Senior Engineer shifts from **Syntactical Production** (typing code) to **System Orchestration** (managing agents).

## Governance is King

Success depends on **Specification** and **Validation**. If you cannot describe it explicitly, the Agent cannot build it.

# Image Sources

---



<https://blog.logrocket.com/wp-content/uploads/2021/07/high-complexity-electron-app-architecture-1.png>

Source: [blog.logrocket.com](https://blog.logrocket.com)

---



[https://cdn.prod.website-files.com/61e1d8dcf4a5e16aab73f6b4/658098de6bc511c3a629283f\\_vvw-G-EAc5mgpFEYfA5V3nTWPJwXxTlyBZdWG34qHNUc\\_ri2IU3nVapGhREVq7Eu6JeVY0Ke\\_AzboD4RAc75rnjLnjBZId7HEX8S4TsxPJSNTstnrNOEAqw9R2sjvK-FPqjiAAjBOGVmkly2ddLMj3U.png](https://cdn.prod.website-files.com/61e1d8dcf4a5e16aab73f6b4/658098de6bc511c3a629283f_vvw-G-EAc5mgpFEYfA5V3nTWPJwXxTlyBZdWG34qHNUc_ri2IU3nVapGhREVq7Eu6JeVY0Ke_AzboD4RAc75rnjLnjBZId7HEX8S4TsxPJSNTstnrNOEAqw9R2sjvK-FPqjiAAjBOGVmkly2ddLMj3U.png)

Source: [www.codesees.io](https://www.codesees.io)

---



[https://substackcdn.com/image/fetch/\\$s!JhHO!,f\\_auto,q\\_auto:good,fl\\_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fc15b76ba-b6c2-4941-8841-be276858d491\\_1102x522.png](https://substackcdn.com/image/fetch/$s!JhHO!,f_auto,q_auto:good,fl_progressive:steep/https%3A%2F%2Fsubstack-post-media.s3.amazonaws.com%2Fpublic%2Fimages%2Fc15b76ba-b6c2-4941-8841-be276858d491_1102x522.png)

Source: [www.productcompass.pm](https://www.productcompass.pm)

---



<https://d8iqbmvu05s9c.cloudfront.net/m9ynnnh05lvw1n45nnw0vxjpwvcy>

Source: [www.ministryoftesting.com](https://www.ministryoftesting.com)