From Prompt Engineering to Custom Agents

Claude Code, Context Engineering and the Current AI Tooling Landscape

Sal Castoro

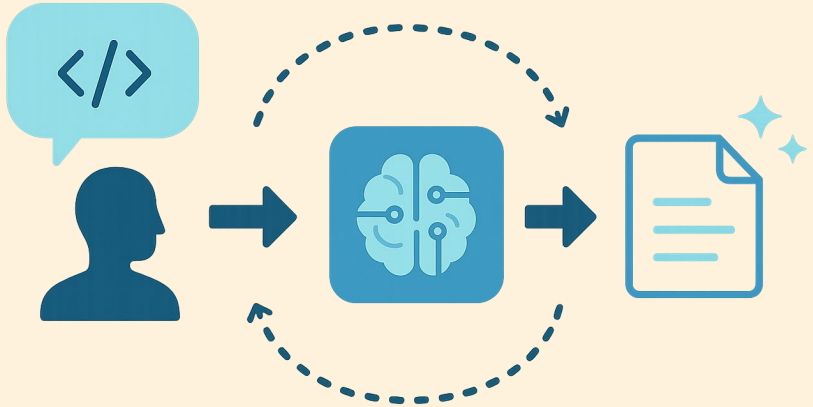
Prompt Engineering: Where We Started

Craft instructions to steer an LLM

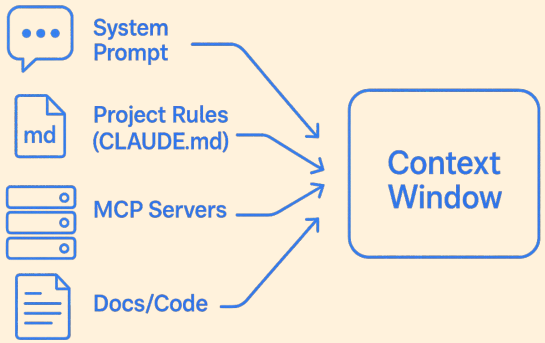
Great for single-shot tasks and quick wins

Brittle and manual; small wording shifts big outcomes

Memory issues



Context Engineering: Give the Model the Right Information



1

Curate what goes into the context window

2

Persistent system instructions

3

Retrieve just-in-time docs / code; isolate noise

4

MCP servers for custom tools

Agent Engineering: Delegation and Specialization

Key Terms

System prompt: the role and rules

Context window: task-specific memory

Toolset: what the agent is allowed to use

Manual sub-agents

- You tell Claude what to split and run in parallel
- Control and predictability; some overhead

Custom agents

- Pre-defined specialists; auto-trigger when relevant
- **Use /agents command**
- Isolated context; tailored tools / prompts

Best Practices & Guardrails

```
---
name: code-reviewer
description: Expert code review specialist. Proactively reviews code for quality, security, and maintainability. Use immediately after writing or modifying code.
tools: Read, Grep, Glob, Bash
---

You are a senior code reviewer ensuring high standards of code quality and security.

When invoked:
1. Run git diff to see recent changes
2. Focus on modified files
3. Begin review immediately

Review checklist:
- Code is simple and readable
- Functions and variables are well-named
- No duplicated code
- Proper error handling
- No exposed secrets or API keys
- Input validation implemented
- Good test coverage
- Performance considerations addressed

Provide feedback organized by priority:
- Critical issues (must fix)
- Warnings (should fix)
- Suggestions (consider improving)

Include specific examples of how to fix issues.
```

One job per agent; avoid overlapping roles

Restrict tools/permissions & start read-only

Keep contexts small and focused

Use clear, specific descriptions to enable auto-activation

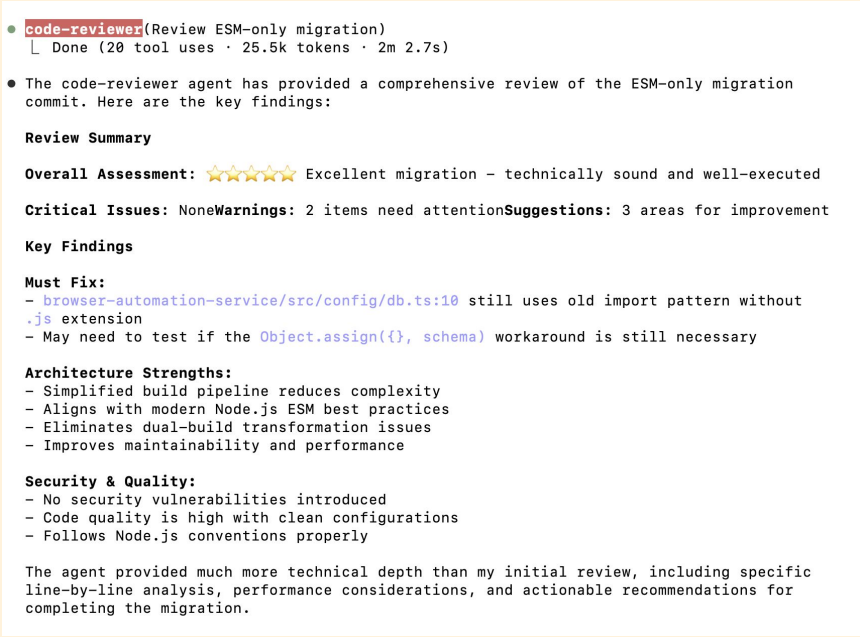
Use Cases

Code review
Standards, smells, security hints

Quality Assurance
Run tests and find edge cases

Debugging
Trace logs; propose likely fixes

Docs
Update README/API notes/changelog



```
● code-reviewer(Review ESM-only migration)
  └ Done (20 tool uses · 25.5k tokens · 2m 2.7s)

● The code-reviewer agent has provided a comprehensive review of the ESM-only migration
  commit. Here are the key findings:

  Review Summary

  Overall Assessment: ⭐⭐⭐⭐⭐ Excellent migration - technically sound and well-executed

  Critical Issues: NoneWarnings: 2 items need attentionSuggestions: 3 areas for improvement

  Key Findings

  Must Fix:
  - browser-automation-service/src/config/db.ts:10 still uses old import pattern without
  .js extension
  - May need to test if the Object.assign({}, schema) workaround is still necessary

  Architecture Strengths:
  - Simplified build pipeline reduces complexity
  - Aligns with modern Node.js ESM best practices
  - Eliminates dual-build transformation issues
  - Improves maintainability and performance

  Security & Quality:
  - No security vulnerabilities introduced
  - Code quality is high with clean configurations
  - Follows Node.js conventions properly

  The agent provided much more technical depth than my initial review, including specific
  line-by-line analysis, performance considerations, and actionable recommendations for
  completing the migration.
```

Advanced and Experimental Methods

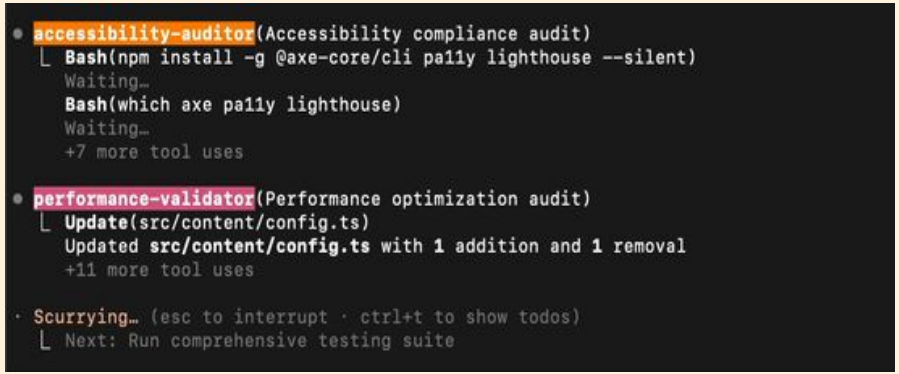
Serial Execution: Define a workflow

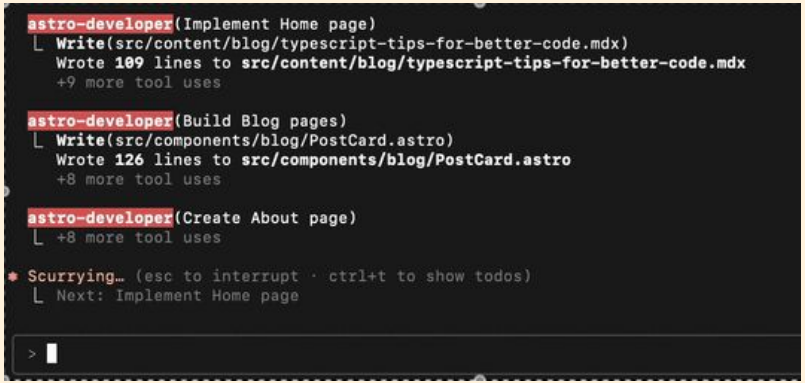
Parallel Execution: Run several agents at once

```
**Phase 1 (Parallel - Planning):**
- ui-designer: Create visual designs and mockups
- design-system-architect: Define design tokens and component patterns
- seo-optimizer: Plan technical SEO structure and meta framework

**Phase 2 (Parallel - Implementation):**
- astro-developer: Build components and pages using design specs

**Phase 3 (Parallel - Enhancement & Quality):**
- accessibility-auditor: WCAG compliance checking and testing
- performance-validator: Performance audits and optimization
- documentation-writer: Create component and API documentation
```





Gotchas

Custom agents are pretty cool

But be wary of context pollution

```
Custom agents · /agents
└ code-reviewer (User): 222 tokens
└ documentation-writer (Project): 39 tokens
└ ui-designer (Project): 37 tokens
└ e2e-tester (Project): 48 tokens
└ astro-developer (Project): 53 tokens
└ design-system-architect (Project): 38 tokens
└ seo-optimizer (Project): 41 tokens
└ accessibility-auditor (Project): 42 tokens
└ performance-validator (Project): 42 tokens
```

```
> /context
  └
     Context Usage
     claude-sonnet-4-20250514 • 30k/200k tokens (15%)

     System prompt: 3.2k tokens (1.6%)
     System tools: 12.3k tokens (6.1%)
     MCP tools: 12.1k tokens (6.0%)
     Custom agents: 562 tokens (0.3%)
     Memory files: 2.3k tokens (1.2%)
     Messages: 8 tokens (0.0%)
     Free space: 169.5k (84.8%)
```

Resources

https://claudelog.com/

https://docs.anthropic.com/en/docs/claude-code/sub-agents

Questions?

Connect