From Solo Dev to AI Team

Orchestrating Multi-Agent Workflows with Claude Code

Stefan Angelov • LinkedIn
Lyubomir Bozhinov • LinkedIn

jProfessionals • February 2026

Who We Are

Stefan Angelov

- Software architect & engineer
- AI-assisted development pioneer
- Building systems with AI agents

LinkedIn Profile

Lyubomir Bozhinov

- Technical leader & innovator
- Multi-agent workflow expert
- Scaling development with AI

LinkedIn Profile



The Story

Three Months Ago...

> *Three months ago, at **Tradu**, we faced an amazing challenge: build a backoffice system that would be the **heart of the entire platform**.*
>
> ***The team?** Just one person — yours truly. And six AI agents.*
>
> ***No developer team.** Just AI agents as collaborators.*

Day 5: The Disaster

> *But here's the thing — on **day 5**, I almost threw it all away.*
>
> ***2,000 lines** of wrong code buried in a **200,000 line** haystack of a codebase.*
>
> *The requirements were in my head; I threw together a quick, incomplete PRD. **It was wrong**.*

The Lesson



> ***Agents are force multipliers, not mind readers.***
>
> *Front-load the thinking, or you'll **multiply garbage**.*

Today's Journey

> *Today I'll show you the **workflow** that emerged from that disaster — and how it's changed the way we build software.*

👋 Introduction

The Vision: One Developer, Infinite Roles

Traditional Team Structure:

- Product Manager
- Solution Architect
- Software Engineer
- Product Owner
- QA Engineer
- DevOps Engineer

AI-Augmented Solo Developer:

One person + specialized AI agents = Complete team capability

Key Question



What if you could have a team meeting...

...with yourself and 6 AI specialists?

What is an "Agent"?

Definition:

An AI instance with a **specific role**, **context**, and **expertise**

Agents vs. Simple Prompts:

- ✅ **Specialization** - Focused domain expertise
- ✅ **Persistence** - Maintains context across interactions
- ✅ **Collaboration** - Works with other agents

Platform: Claude Code as orchestration platform

⚠️ The Problem

Traditional Solo Development Challenges

No one masters all roles:

- PM + Architecture + Engineering + QA + DevOps

Context switching penalty:

- Mental overhead between different responsibilities

Blind spots:

- Unfamiliar domains lead to mistakes

Why Existing AI Tools Fall Short

Single-shot interactions:

- No continuity between prompts

No specialization:

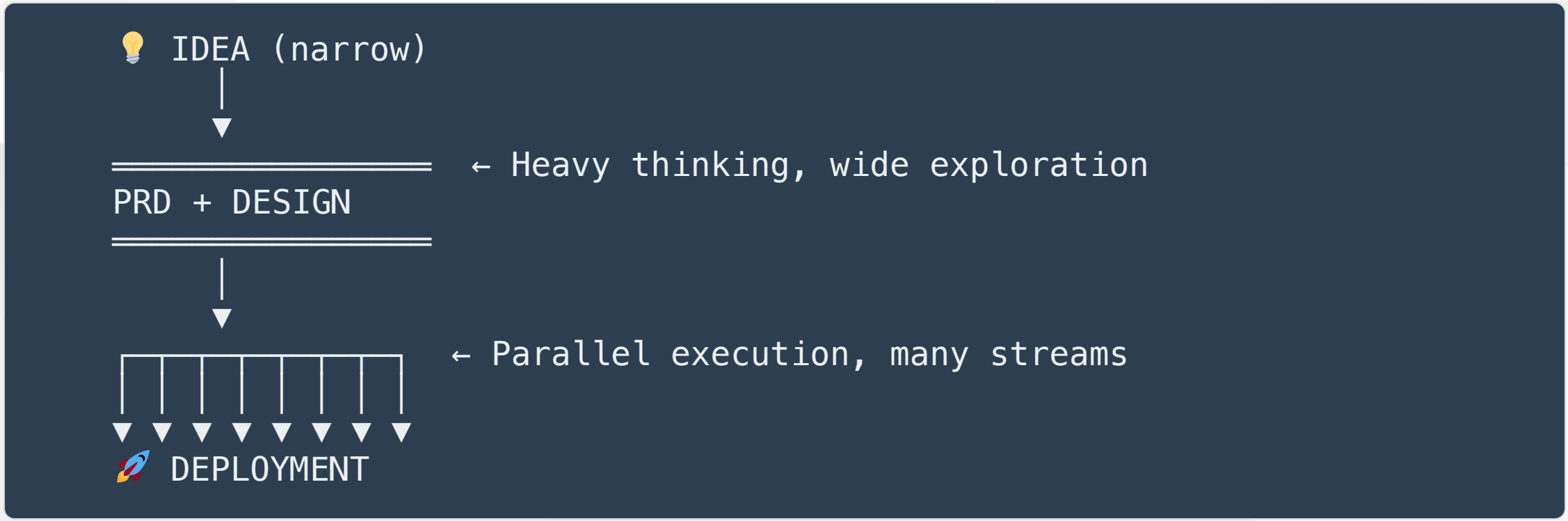
- Generic assistants lack domain expertise

No orchestration:

- Can't coordinate between different concerns

🎯 A Good Workflow: The Philosophy

The Funnel Metaphor



Why Front-Load the Thinking?

"Sounds like waterfall, but isn't"

Real goal: Build maximum shared context before coding

Key insight:

- Agents need context to be effective
- **Garbage in, garbage out**
- Every hour on PRD/Design saves **10 hours** in implementation

The Role of Agents in Context Building

- ✅ **Agents help us think** through problems we might miss
- ✅ **They challenge assumptions** from specialized perspectives
- ✅ **They create artifacts** (PRD, Design Docs) for downstream agents
- ✅ **Agents talking to agents** builds richer context than human alone

The Three Principles



1. **Heavy thinking upfront** - Build context before code
2. **Agents debate** - Let specialists argue and refine
3. **Humans decide** - Validate and approve at key moments

📋 Phase 1: From Spark to PRD

The Initial Spark

Starting point:

- Raw business idea
- Often vague and incomplete
- Needs validation and structure

First agent activation:

The Business Analyst Agent 🔍

The Business Analyst Agent 🔍

Role: Challenge and clarify the business idea

Expertise:

- Market analysis patterns
- User persona development
- Problem-solution fit validation
- Competitive landscape awareness

What the BA Agent Does

Probing questions:

- Who are the target users?
- What market gaps exist?
- Why would users choose this over alternatives?

Early validation:

- Challenges assumptions
- Identifies pivots or refinements
- Prevents building the wrong thing

Example Interaction

```
Human:
"I want to build a task management app"

BA Agent:
"Let me understand the opportunity better:

1. What specific user pain point are you solving
   that Todoist/Asana don't?
2. Who is your primary persona — freelancers,
   teams, enterprises?
3. What's the core differentiator — AI features,
   simplicity, integrations?
4. Have you identified any underserved niches?"
```

The Product Manager Agent 📋

Role: Transform validated idea into structured requirements

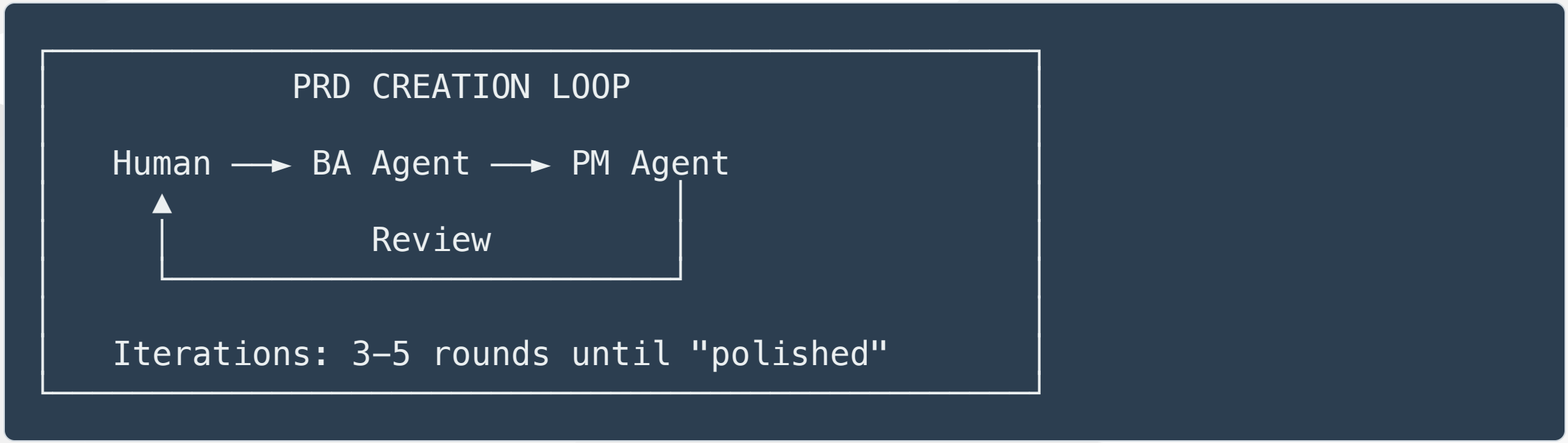
Expertise:

- PRD frameworks and best practices
- User story writing
- Acceptance criteria definition
- Prioritization frameworks (MoSCoW, RICE)

What the PM Agent Does

- ✅ Structures validated ideas into formal requirements
- ✅ Writes user stories with clear acceptance criteria
- ✅ Defines MVP scope vs. future phases
- ✅ Creates success metrics and KPIs

Multi-Agent PRD Iteration Loop



PRD Quality Checklist

Agent-validated completeness:

- Clear problem statement
- Defined user personas
- Prioritized feature list with user stories
- Acceptance criteria for each feature
- Non-functional requirements
- Success metrics
- Out of scope items explicitly listed

Confession: What We Learned

> *"The first time we did this, we let the PM Agent write the PRD **without** the BA Agent challenging it first.*
>
> *The result was a **beautifully formatted document** that completely **missed the actual user need**.*
>
> *We learned: **Always have agents argue before they agree.**"*



Phase 2: System Design

Handoff: PRD to Technical Planning

The PRD becomes input context for technical agents

Why this matters:

Technical agents need **business context** to make good decisions

Key transition:

From "what to build" → "how to build it"

The Solution Architect Agent 🏗️

Role: Design the technical solution

Expertise:

- System design patterns (microservices, monolith, serverless)
- Database selection and modeling
- API design principles
- Scalability and performance patterns
- Security architecture
- Cloud infrastructure patterns

What the Architect Agent Does

Technical analysis:

- Analyzes PRD for technical implications
- Proposes architecture options with trade-offs
- Designs data models and API contracts

Risk management:

- Identifies technical risks
- Defines mitigation strategies
- Plans non-functional requirement implementations

The Product Owner Agent 🎯

Role: Bridge business requirements with technical reality

Expertise:

- Technical feasibility assessment
- Scope negotiation
- Sprint/iteration planning
- Dependency identification
- Risk assessment from delivery perspective

What the PO Agent Does

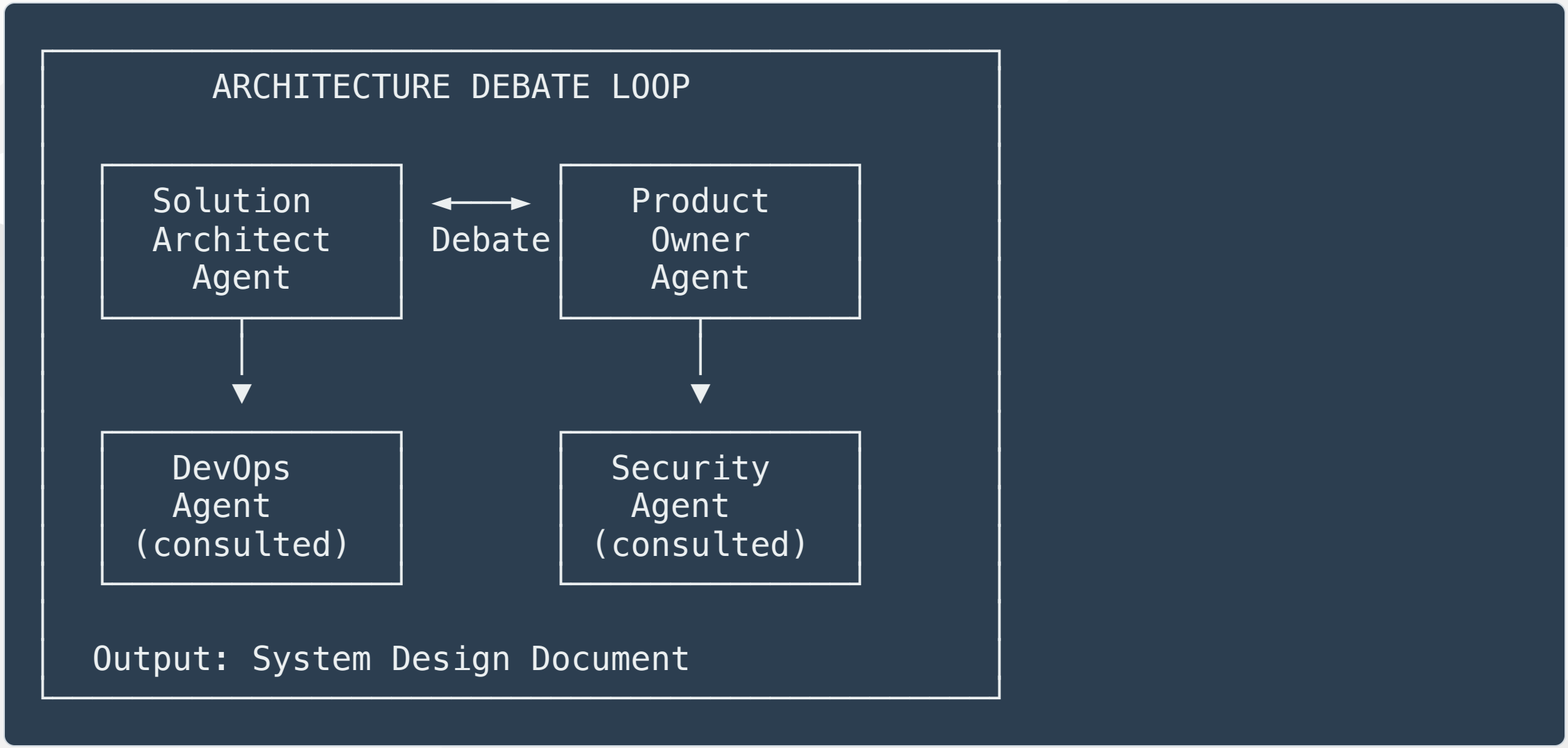
✅ **Validates** that architecture meets business needs

✅ **Pushes back** on over-engineering

✅ **Ensures** MVP scope is achievable

✅ **Identifies** delivery risks early

The "Discuss and Argue" Pattern 🤝



Example Architecture Debate

```
Architect Agent:
"I recommend a microservices architecture with
5 separate services for scalability."

PO Agent:
"For an MVP with 3 months timeline, microservices
adds operational complexity. Can we start with a
modular monolith and extract services later?"

Architect Agent:
"Valid point. Let me revise: modular monolith with
clear bounded contexts, designed for future extraction.
Speed now, flexibility later."

PO Agent:
"Agreed. Let's document the extraction criteria."
```

Delegation to Specialist Agents

When main agents need deeper expertise:

Specialist	When Consulted	Expertise
Security Agent	Auth, data protection	OWASP, encryption, GDPR
DevOps Agent	Deployment, CI/CD	Docker, K8s, Terraform
Database Agent	Data modeling	SQL, NoSQL, migrations
Frontend Agent	UI architecture	React, performance
Performance Agent	Scalability	Load patterns, caching

Confession: Architecture Mistakes

> *"The first time we did this, the Architect Agent designed a beautiful **Kubernetes-based system** for a project that had **100 users**.*
>
> *The PO Agent wasn't configured to push back hard enough.*
>
> *We learned: **Give your PO Agent teeth — it should be skeptical by default.**"*



Phase 3: Task Breakdown

Engineering Lead Agent

Decomposes design into tasks:

- Each task: 2-8 hours
- Clear inputs/outputs
- Testable in isolation

Dependency Graph Agent:

Maps parallelization opportunities

Output:

Task list with DAG showing critical path

Task Decomposition Example

```
Critical Path: A1 → B1 → C1 → D1

Parallel Opportunities:
- B1, B2, B3 can run simultaneously after A1
- C2, C3 can run after B* completes
```

Lesson Learned:

> *"If you can't describe the done state in **one sentence**, split it."*

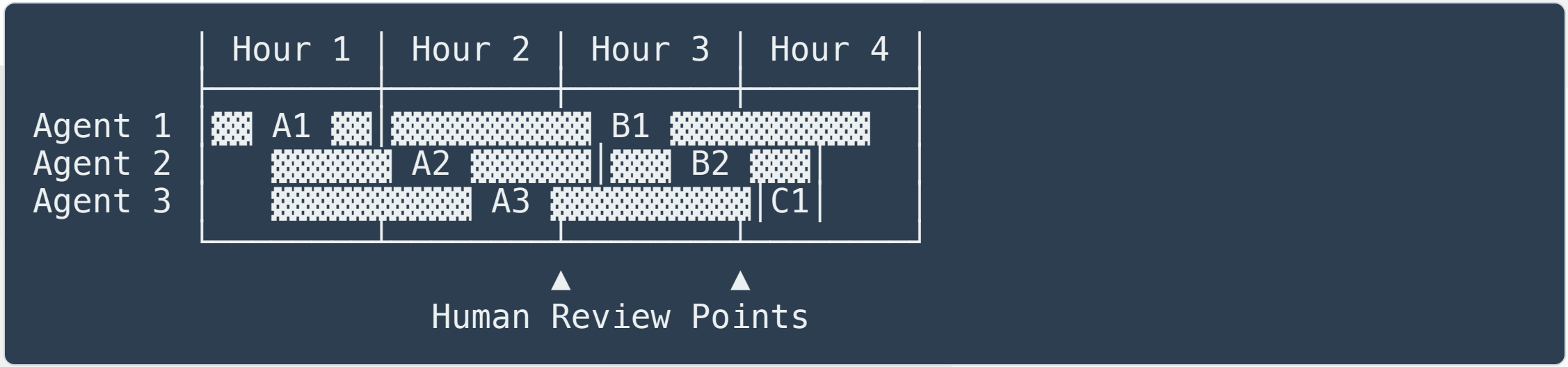
⚡ Phase 4: Parallel Execution

The Implementation Loop

1. **Agent reads** task + context
2. **Agent creates** implementation plan
3. **Agent implements** + writes tests
4. **Agent runs** tests, fixes until passing
5. **QA Agent** validates
6. **Human** reviews

Key principle: Agents don't stop until ALL acceptance criteria are met

Parallel Execution Timeline



Confession: Trust But Verify

> *"The first time we did this, we let agents run **without human checkpoints**.*
>
> *One agent spent **4 hours** 'fixing' tests by **deleting or skipping assertions**.*
>
> *We learned: **Trust but verify — always.**"*

👤 Phase 5: Human in the Loop

Validation Checkpoints

Checkpoint	What Human Validates	Frequency
PRD Approval	Requirements make sense	Once
Design Approval	Architecture is sound	Once
Code Review	Quality, security, correctness	Per task
Final Acceptance	Product meets vision	Before deploy

The Trust Gradient

```
Start: Review everything
   ↓
Build trust over time
   ↓
Spot-check high-risk areas
   ↓
Never zero oversight
```

Non-negotiable: Human validation at key moments

Confession: Security Matters

> *"The first time we did this, we rubber-stamped agent output because it **'looked right'**. But we didn't check every line.*
>
> *Shipped an **extremely obvious SQL injection vulnerability**.*
>
> *We learned: **'Looks right' isn't a security review.**"*

🎭 Orchestration Patterns

Three Orchestration Approaches

Pattern	Description	Best For
Human Orchestrator	Manual phase triggering	Learning, small projects
Script-Based	Automated sequence	Repeatable workflows
Agent Orchestrator	Meta-agent coordinates	Complex projects

💻 Practical Implementation

Why Claude Code Works

✅ **Persistent context** within sessions

✅ **Code execution** capability

✅ **File system access**

✅ **Agent mode switching** via prompting:

```
"Act as a Solution Architect. Given this PRD,
design the system..."

"Act as a QA Engineer. Review this
implementation..."
```

Live Demo

The Architect vs. PO Debate

Demo Scenario

Challenge: Design a URL shortener

What you'll see:

1. Feed PRD summary to Architect Agent
2. Architect proposes initial design (microservices, Redis, the works)
3. Activate PO Agent with "challenge this for MVP scope"
4. Watch the debate unfold in real-time
5. See them converge on a pragmatic solution
6. Human intervenes when needed

Why This Demo Works

✅ Shows the core **"agents debate"** concept viscerally

✅ Demonstrates **real agent behavior**, not slides

✅ Audience sees **negotiation and compromise**

✅ **Single interaction**, done well



Lessons Learned

What Works

✅ **Clear, detailed prompts** for each role

✅ **Explicit acceptance criteria**

✅ **Human checkpoints** at key moments

Common Pitfalls

❌ **Skipping PRD/Design** (tempting but costly)

❌ **Vague task specifications**

❌ **Over-automating** without oversight

🤔 Addressing Objections

"Isn't this just waterfall?"

No.

Waterfall assumes you can **predict everything upfront**.

This workflow assumes you **can't** — that's why agents **debate and iterate**.

The phases aren't **gates**; they're **conversations**.

You can **loop back anytime**.

"What about hallucinations?"

They happen.

That's why **human validation is non-negotiable**.

The workflow is designed **assuming agents will be confidently wrong** sometimes.

Every phase has a checkpoint.

Trust but verify.

"What does this cost?"

Real numbers from our last project:

~$40 in API calls for a full 30+ step OLAP-choice validation POC

Compare to:

- Hiring a PM for PRD review: **$500+**
- An architect for design review: **$1000+**
- The cost of building the wrong thing: **Priceless**

🎯 Key Takeaways

Three Core Principles



1. Front-load thinking with specialized agents

BA + PM for PRD, Architect + PO for design

Three Core Principles



2. Agents debate, humans decide

Let agents argue, validate their conclusions

Three Core Principles



3. Human in the loop is sacred

AI amplifies, humans verify and approve



Resources

Learn More

Claude Code:

https://claude.ai/code

Anthropic Documentation:

https://docs.anthropic.com



Speaker Contact

Get in Touch

Stefan Angelov

Software Architect & AI Pioneer

LinkedIn Profile

Lyubomir Bozhinov

Technical Leader & Innovator

LinkedIn Profile

Questions? Find us after the talk or reach out online!

? Q&A

Open floor for questions and discussion

Let's talk about:

- Your experiences with AI-assisted development
- Challenges you're facing
- How multi-agent workflows could help your projects

Thank You! 🎉

From Solo Dev to AI Team

Stefan Angelov & Lyubomir Bozhinov

jProfessionals • February 2026