# The Year of the Orchestrator

The Red Queen is here

Pere Villega | Aracon Software SLU

28th Jan 2026

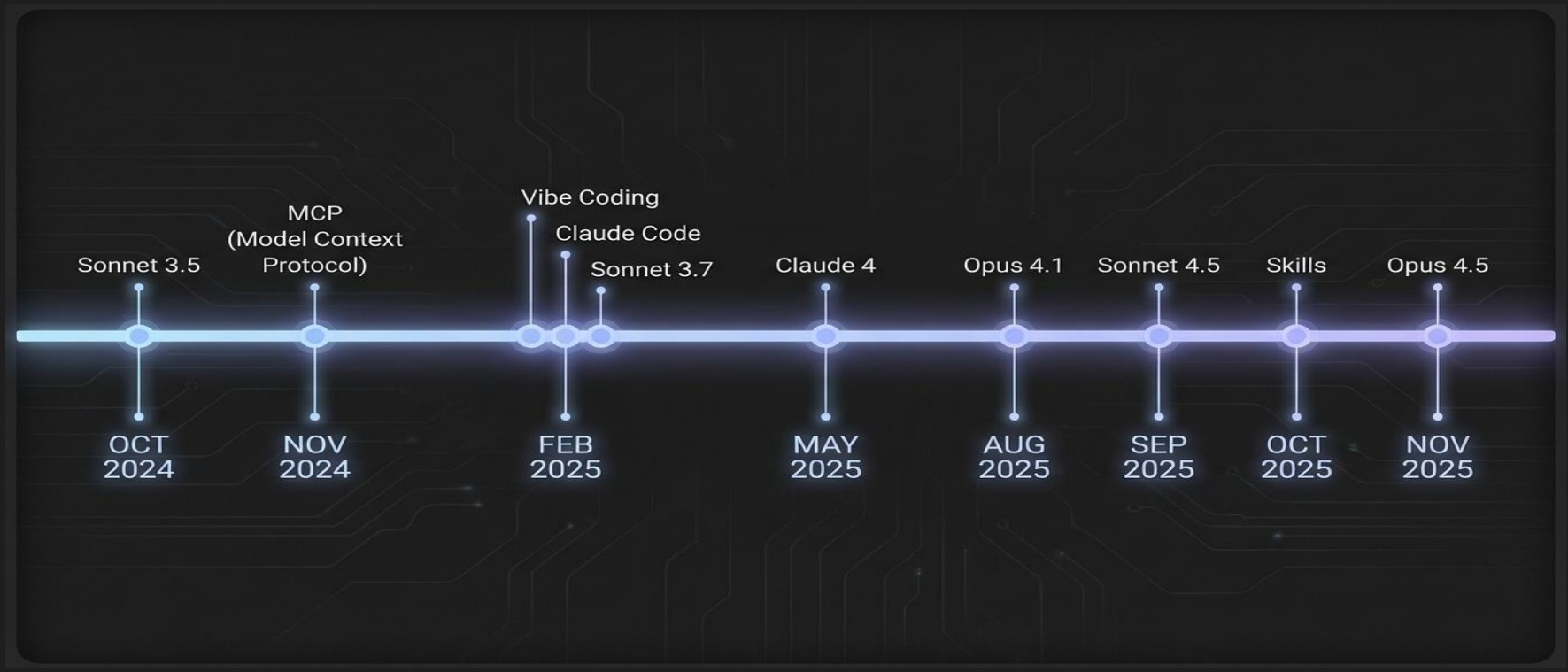
# Hi!

## I'm Pere Villega

- Suffering computers for 20+ years
- Believer on AI for Software Development
- Using it in real projects!

- And made too many slides...

# AI is Relentless




*And this is ONLY Anthropic…*

# …and now Orchestrators

## What we will do

- See how we got here
- Understand what they are

- Consider how will they impact us

DANGER LIES AHEAD

# DO NOT USE ORCHESTRATORS

*(They are coming. But they are not there yet.)*

# 01

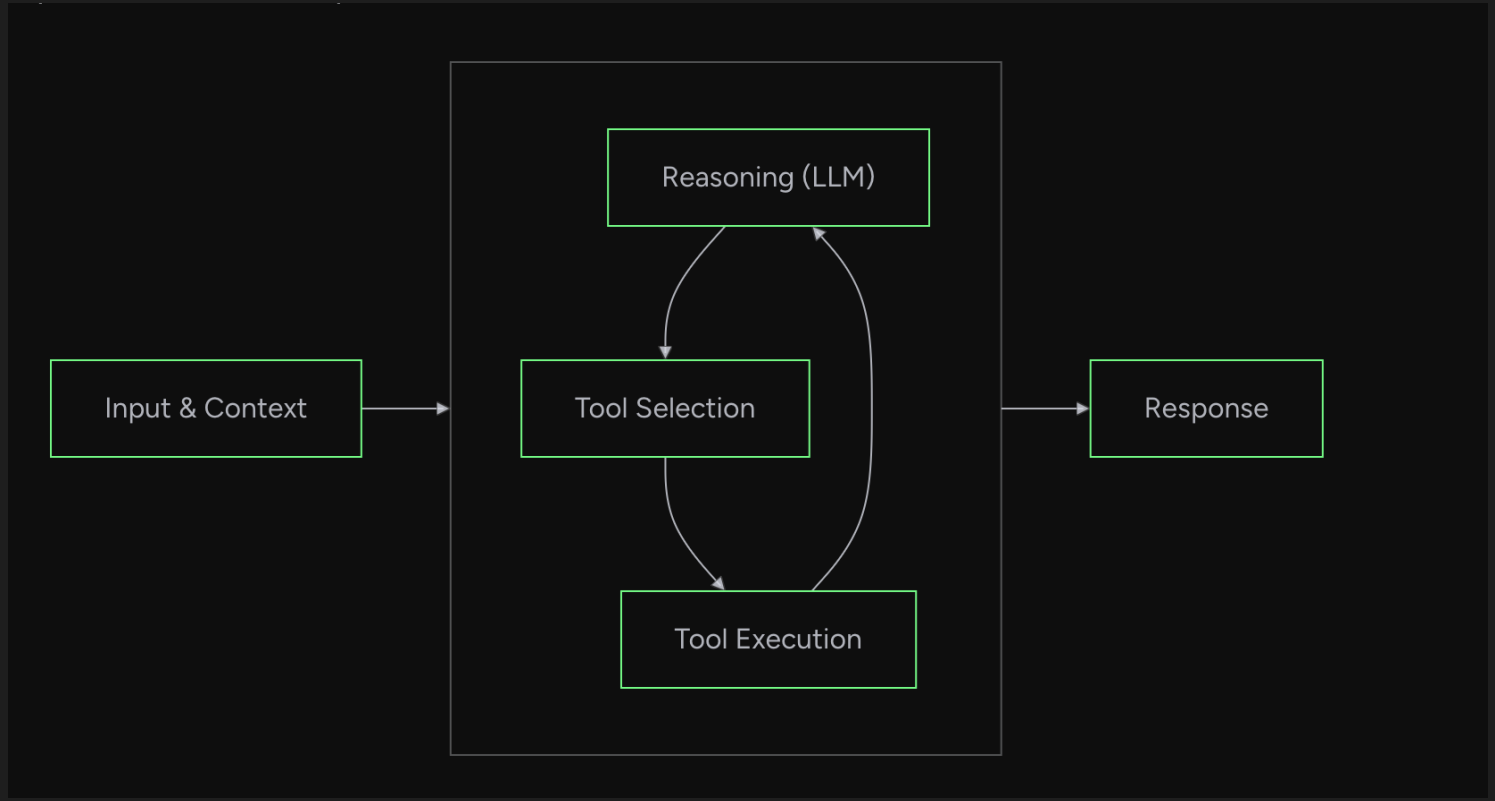
# ORCHESTRATORS

# Agents

## Agents have 4 parts

| | |
|---|---|
| **LLM** | The "brain" - reasons and decides next action |
| **Tools** | The "hands" - read files, run code, call APIs |
| **State/Memory** | Accumulated **context** across iterations |
| **Stop Condition** | When to exit (task done, max steps, error) |

# Agents


Reasoning (LLM)
Input & Context
Tool Selection
Response
Tool Execution

# Agents

```
const env = { state: initialState };
const tools = new Tools(env);

while (true) {
  // 1. LLM thinks based on prompt + current state
  const action = llm.run(`${system_prompt} ${env.state}`);

  // 2. Check if done
  if (action.type === "final_response") break;

  // 3. Execute tool and update state
  env.state = tools.run(action);
}
```

# Agents

## The problem: CONTEXT

- The LLM has a limited window (200k)
  - ~176K truly usable
  - Realistic:  ~80k to 120k usable
- **Solution**:
  - Use external markdown files
  - Ask the agent to read them at the start

# Ralph Wigum (July 2025)

# Ralph Wigum

## Automated Agent Loop for Tasks

- Two prompts, one loop
- New context window each loop
- Hands-off: runs until completion

- **DANGER**: use a sandbox. This requires YOLO mode.

# Ralph Wigum

## Phase 1

- Define Job to be done talking to the LLM
- Identify topics (authentication, scheduling, reports, etc.)

- **OUTPUT**: one spec file per topic.

# Ralph Wigum

## Phase 2

- Two prompt files: PLANING and BUILDING
- PLANNING: Generate/update IMPLEMENTATION_PLAN.md
- BUILDING: Implement 1 task from plan, commit, update plan as a side effect

- **IMPORTANT**: Backpressure = > Validation (compiler, tests, linter, etc.)

# Ralph Wigum

```
project-root/
├── loop.sh                     # Ralph loop script
├── PROMPT_build.md             # Build mode instructions
├── PROMPT_plan.md              # Plan mode instructions
├── AGENTS.md                   # Operational guide loaded each iteration
├── IMPLEMENTATION_PLAN.md      # Prioritized task list (generated/updated by Ralph)
├── specs/                      # Requirement specs (one per JTBD topic)
│   ├── [jtbd-topic-a].md
│   └── [jtbd-topic-b].md
├── src/                        # Application source code
└── src/lib/                    # Shared utilities & components
```

```bash
#!/bin/bash
# Usage: ./loop.sh [plan]

# Parse arguments
if [ "$1" = "plan" ]; then
    # Plan mode
    MODE="plan"
    PROMPT_FILE="PROMPT_plan.md"
else
    # Build mode, unlimited (no arguments or invalid input)
    MODE="build"
    PROMPT_FILE="PROMPT_build.md"
fi

CURRENT_BRANCH=$(git branch --show-current)

while true; do
    cat "$PROMPT_FILE" | claude -p \
        --dangerously-skip-permissions \
        --output-format=stream-json \
        --model opus \
        --verbose

    # Push changes after each iteration
    git push origin "$CURRENT_BRANCH"

done
```

# Ralph Wigum

## Does it work?

- It works. But it is sequential
- Wouldn’t it be great if we could parallelise it?

# Orchestrators

## More Ralphs, please

- Multiple Agents on the same task requires:
  - Task management & coordination
  - Observability
  - Recovery

- An Orchestrator is a tool that handles all that

# Orchestrators

## What are they not!

- Specific Agents and Commands in .Claude
  - Plenty of tooling built this way. That's not it
- Plugins for Claude
  - Sidenote: The Anthropic Ralph plugin is not Ralph!

# Orchestrators

**Example: Gas Town (by Steve Yegge)**

- Primary AI coordinator
- Watchdog for system health
- Agent recovery processes
- Agent for merge management
- Ephemeral workers
- Named agents for humans
- Agents at workspace and project level
- Project worktrees
- Agent mailboxes (communication)
- Task tracking (Beads)
- Workflow orchestration

# Orchestrators

```
# Assume Gas Town installed
gt rig add myproject https://github.com/you/your-repo.git
gt crew add my_name --rig myproject
gt mayor attach # This opens a Claude Code session

> I need to add a module to myproject that reads weather data
from the OpenWeatherMap API. It should fetch current conditions
and cache them for 5 minutes. Include error handling and tests.

# check status
gt mail inbox
gt convoy show
```

02

# WHY DOES THIS MATTER

# Orchestrators

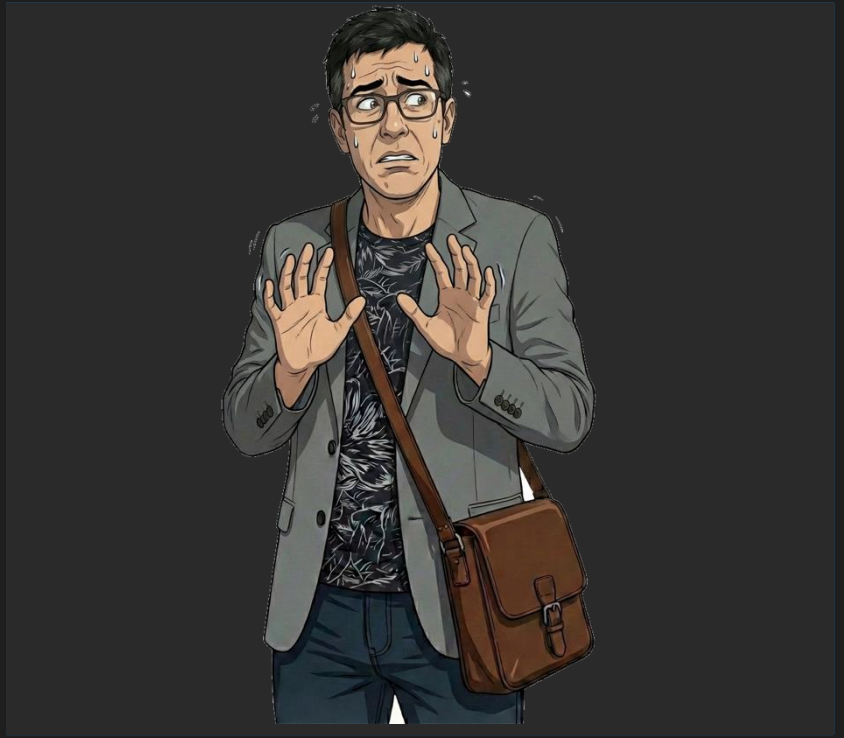
**Breaking news!**

- Claude Code TeammateTool (found in Claude Code v2.1.19)
  - https://gist.github.com/kieranklaassen/d2b35569be2c7f1412c64861a219d51f
- Provides: inter-agent messaging, plan approval workflows, coordination between agents, etc.
- A lot of the infrastructure that Orchestrators are building right now, is there

# What's coming

## This is happening

- More code written. Reviews?
- More agents. Monitoring?
- Coordination between Orchestrators?



The Question

# How do you adapt your processes for this

**The Answer**

# We don’t know (yet)

# It’s a new world

# How to use them

## We know what won't work

- PR review-based workflows
- Keep FE/BE/Ops split across multiple teams
- Tasks without acceptance criteria
- Lack of CI (backpressure)
- No automated E2E testing
- Lack of observability

# How to use them

## Implications of the previous list

- “Old” best practices are even more crucial
- Teams should own a vertical
- Need good architecture from the start
- Requirements can’t have major gaps
- Quality gates (ci, tests, alerting, observability) steer the outcome
- SWE role is changing, a lot

# How to use them

## New practices

- Embedded knowledge in the code is bad
  - It must be explicit in specs, tests
- Less people owning a vertical
- Microservices (decoupling) can be good
- SWE must understand/own product outcomes
- Black-box testing and workflow-based testing matter more
- More to be discovered during 2026

TL;DR

# Start adjusting your processes

# Questions?

**Pere Villega**
perevillega.com
Aracon Software SLU

*"We finally built a software architecture where 'asking nicely and retrying' is a legitimate error-handling strategy."*

# Links

- Agents: https://strandsagents.com/latest/documentation/docs/user-guide/concepts/agents/agent-loop/
- A good agent to inspect: https://shittycodingagent.ai
- Ralph: https://github.com/ClaytonFarr/ralph-playbook
- Not an orchestrator: https://github.com/glittercowboy/get-shit-done
- Gas Town: https://github.com/steveyegge/gastown
- Beads: https://github.com/steveyegge/beads