CONSTRAINT-ORIENTED ARCHITECTURE

Building Software in the Age of AI Agents

(and why everything is going to be alright)

Pablo Bermejo

ACT 1: WHAT IS GOING ON?

Software Engineering in the age of Agentic AI

WHAT IS AN AI AGENT?

LLM

Probabilistic Engines. Stateless math functions predicting the next token with massive general knowledge

Reasoning LLM

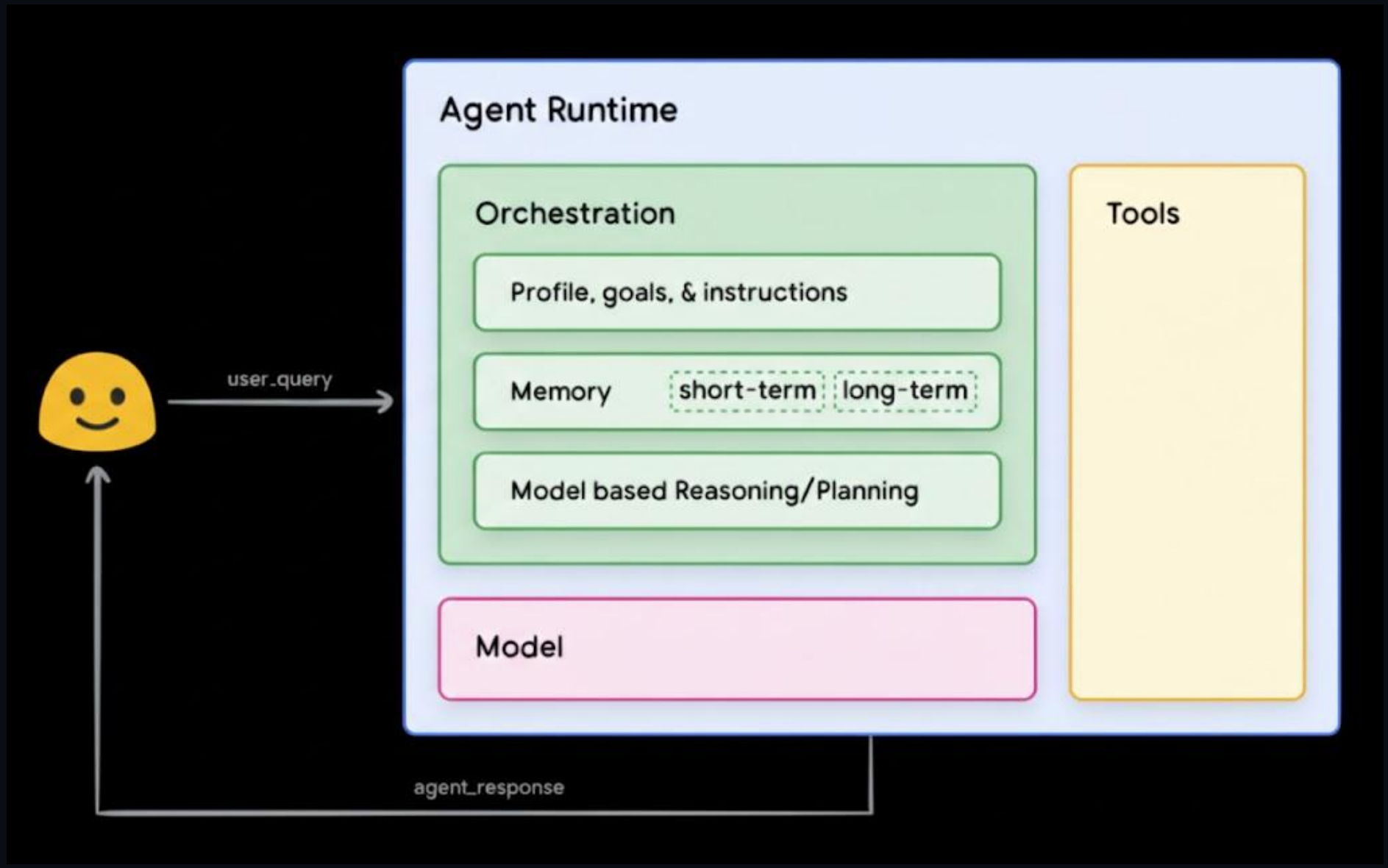
Inference-time compute. These models "think" before speaking, performing multi-step logical deduction

AI Agent

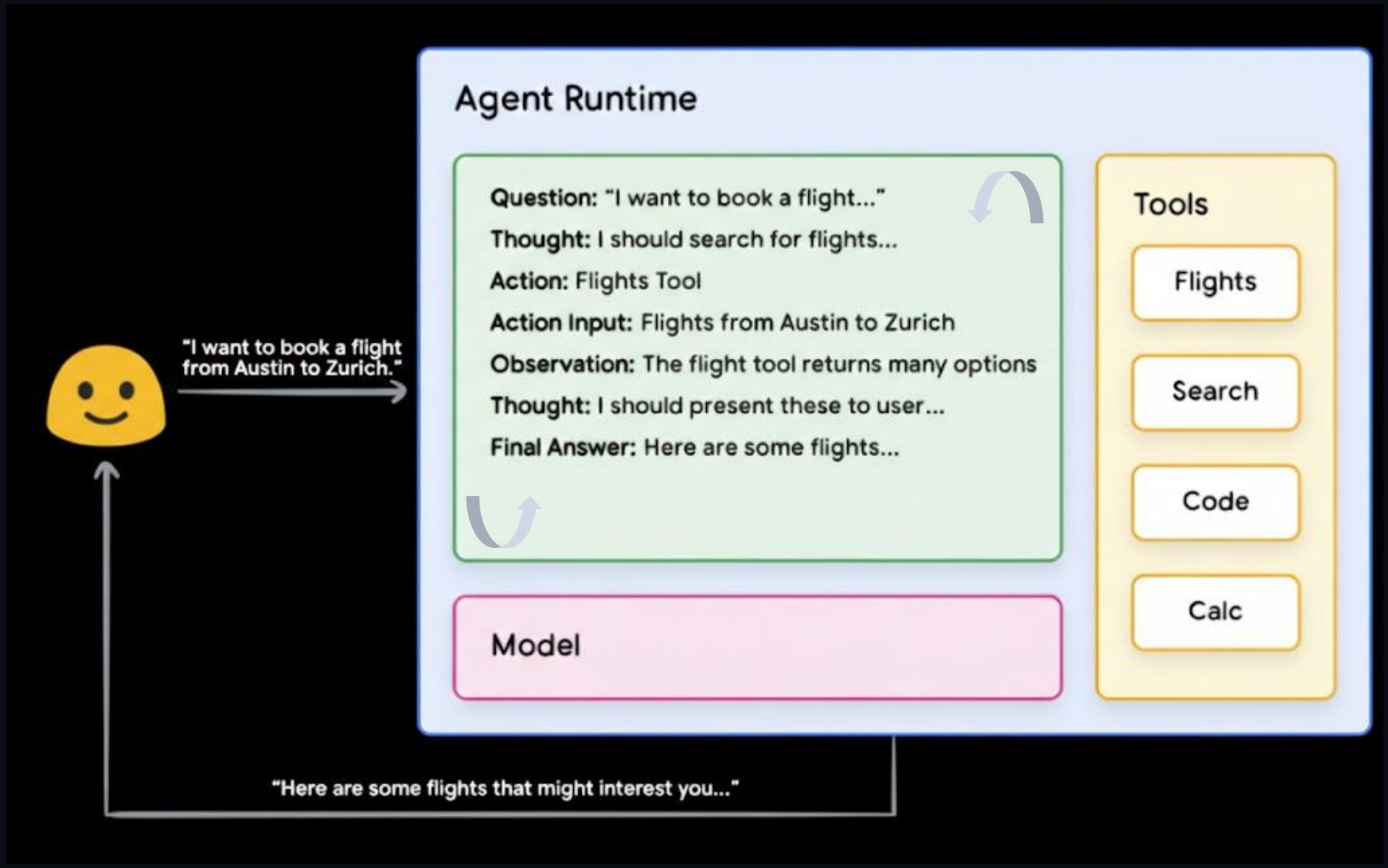
Loop-based Programs. Memory + Tools + Planning. They act autonomously to achieve a goal.

An AI agent is a WHILE loop calling LLMs and tools

AGENT ANATOMY



SIMPLE EXAMPE OF AN AGENT



MEET THE NEW DEV IN THE TEAM: CODING AGENTS

Literal Execution

Follows binary rules perfectly. Fails at nuance. **Needs strict data constraints**, not soft guidelines.

Visually Blind

Processes text and code, not pixels. Cannot "see" clutter. Relies on design tokens for visual quality.

Confident

Does not ask clarifying questions unless asked otherwise. Fills information gaps by **hallucinating** non-existent patterns.

A coding agent is an agent that ... writes code

THE ELEPHANT IN THE ROOM

> "AI Agents will write **100%** of code."

— **Dario Amodei, CEO of Anthropic**

This quote went everywhere. If you're studying software engineering right now... I get it. That's scary.

If this is true,

what is left us to do?

THE CRITICAL DISTINCTION

"AI DOES ENGINEERING"

Problem understanding, solution design, verification, and long-term system maintenance.

"AI WRITES 100% OF CODE"

Syntactic generation, boilerplate, unit tests, and routine implementation steps.

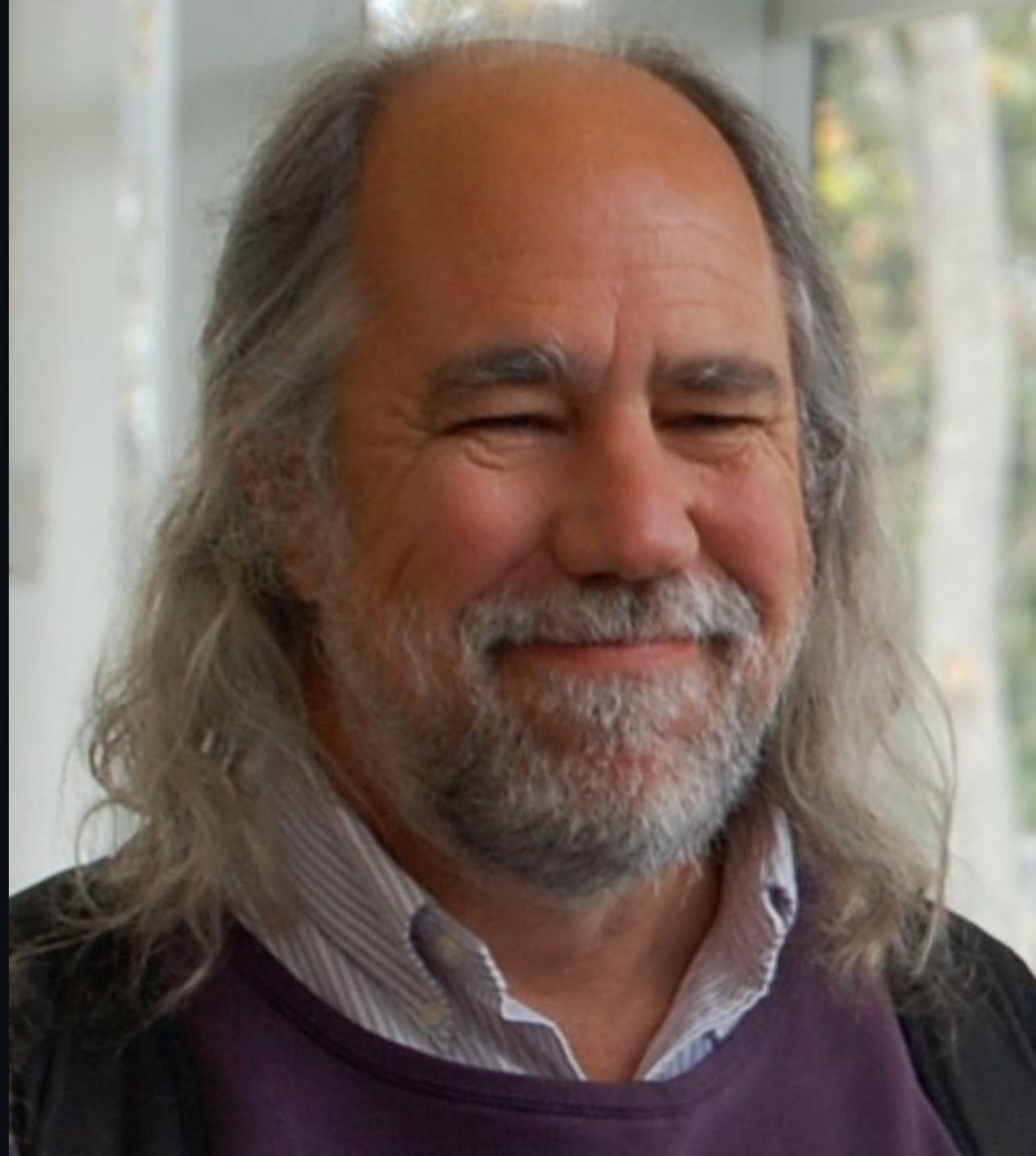
Writing code is one step in a very long chain.

WE'VE BEEN HERE BEFORE

> "This is not the first time developers have faced an existential crisis. Not even close."

— Grady Booch, Creator of the UML

In software for 40+ years. He's been watching this debate since before most of us were born.



THE ABSTRACTION LADDER

> "The entire history of software engineering is one of rising levels of abstraction. That's what we are seeing here. No more, no less"
>
> — **Grady Booch**

- **The Evolution**: Programmers moved from managing bits and registers to memory and data structures.
- **The Fear**: When Fortran arrived, assembly programmers feared the compiler would kill their craft!
- **The Reality**: Every new level of abstraction created MORE programmers, not fewer.
- **The Shift**: Demand for those who understand WHAT to build went UP.

WE ARE JUST CLIMBING THE ABSTRACTION LADDER

THE CLAIM

"Software Engineering is over! Just throw magic spells in plain english. Anyone can do it!"

THE REALITY

It's **Structured English**. You need to be precise. You're specifying behavior. That IS programming.

We are now moving from managing memory, data structures, and algorithms to manage natural language structures in plain english.

Where does software architecture fit in this picture?

DEFINITION OF ARCHITECTURE

> "Architecture is the set of significant **design decisions** that shape a system, where significance is measured by cost of change."
>
> — **Grady Booch**

Note: This definition mentions no language, no code, and no specific tools.

THE PERSPECTIVE OUT THERE (MANY COMPANIES)

99%

Code written by LLMs

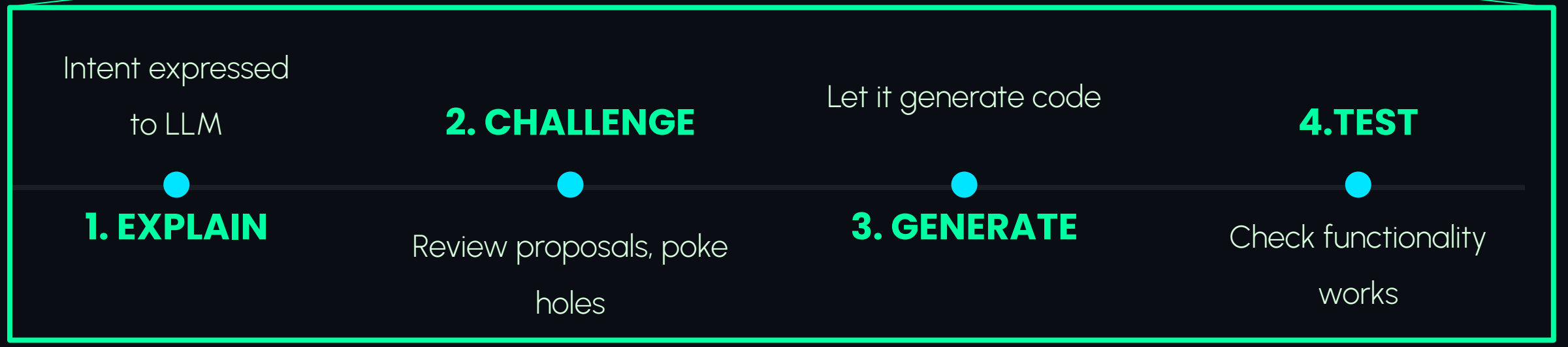
LAST STEP ENGINEERING

"When you're building high quality production software, writing code is always the last step."

The actual architecture and engineering happens before a single token is generated to write code.

THE MODERN DEVELOPMENT WORKFLOW

PROBLEM > BUILD > RELEASE



Output quality is a direct function of the LLM, the Coding Agent and user input

ACT 2: THE ARCHITECTURE PRACTICE IS REDEFINED

Constraint-Oriented Architecture

Designing the "Box" for the Agent.

CONSTRAINT-ORIENTED ARCHITECTURES

Drive architecture design through context for coding agents

OLD

Hide complexity so humans aren't overwhelmed

NEW

Enforce physical constraints so agents cannot touch what they shouldn't

Principle of Least Privilege, by architectural design.

WHAT CHANGED?

SAME PRINCIPLES

Coupling, Cohesion, Abstraction, Domain Boundaries, Security, Resilience, Cost of Change.

NEW ARTIFACTS

From 50-page design docs and UML to CLAUDE.md files, SKILL.md files, and guardrails

THE NEW PRACTICES

FITNESS FUNCTIONS

Evals as continuous validation of conceptual integrity.

POLICY-AS-CODE

Automated rules that the agent physically cannot bypass.

GUARDRAILS

Security and performance constraints as the primary artifact.

SPEC-DRIVEN DEVELOPMENT

> "The limit of the imperative to declarative transition... The spec IS the program."

— Andrej Karpathy

Architecture becomes the act of writing the perfect, unassailable spec

REFLECTION

If architecture is significant design, where significance is measured by cost of change …

… and …

… the cost of change drops thanks to LLMs and Coding Agents then …

… software architecture is also redefined and moved up the abstraction ladder.

ACT 3: THE ARCHITECT ROLE IS REDEFINED

The job of the Software Architect in the age of Agentic AI

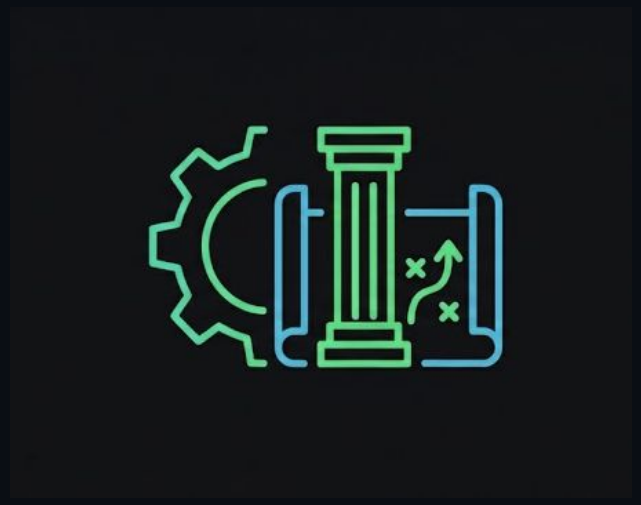
THE BLURRED CAREER PATH

Becoming an Architect of Intents

DEVELOPER

ARCHITECT

Becoming a Product Decision Maker



Becoming a System Strategist

PRODUCT MGR

THE UNCOMFORTABLE TRUTH

> "Ideas being expensive to implement was actually helping."
>
> — Creator of OpenCode

Now that coding is cheap, the bottleneck is deciding what's worth building.

TECHNICAL PRODUCT MANAGEMENT

A HIDDEN PATH FOR CS STUDENTS

Your domain is software. Your users are developers. You already know the "how".

Enjoying the "Why" more than the implementation? This is a high-demand, high-leverage career option.

ACT 4: THE PRINCIPLES-FIRST ARCHITECT

Hacking the Learning Curve

THE NEW PRIMITIVES IN SOFTWARE ENGINEERING

CLAUDE.md

Tokens

MCP

Vocabulary drops every week. It's normal to feel behind.

Agentic SDK

Token Windows

AGENTS.md

Suba-agents

Skills.md

Ralph Wiggum

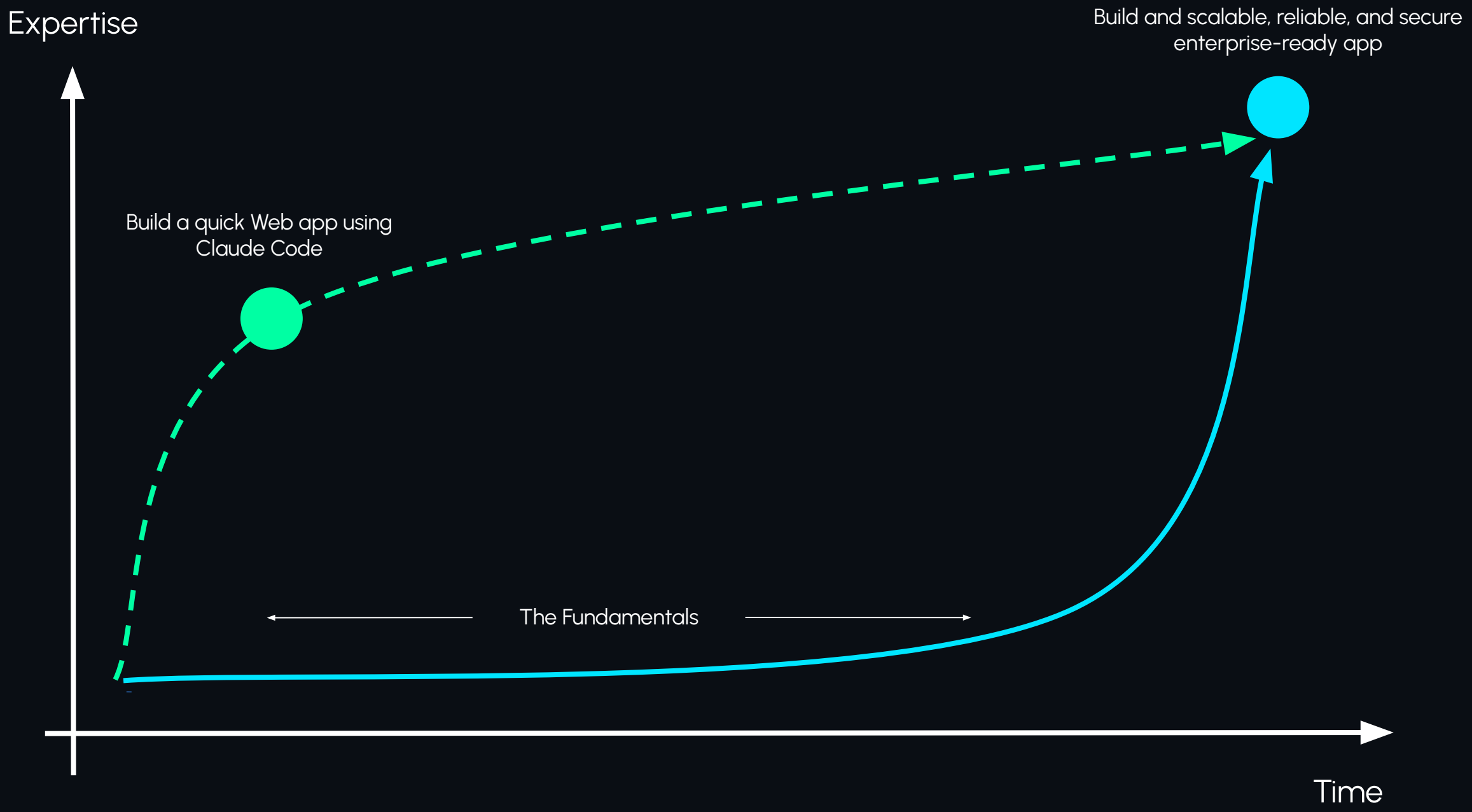
HACKING THE LEARNING CURVE

JAIME ALTOZANO

Learn theory through practice

The hack isn't skipping fundamentals.

It's using AI to accelerate learning


Expertise
Build and scalable, reliable, and secure enterprise-ready app
Build a quick Web app using Claude Code
The Fundamentals
Time

THE FUNDAMENTALS ARE THE HACK

SYSTEMS THINKING

Understand how domains decompose and interact.

AI-READY FUNDAMENTALS

High Cohesion, Low Coupling, Bounded Contexts

TECHNOLOGY

Understand how underlying technologies work

IRREPLACEABLE ENGINEERS

> "The people who use AI to skip learning will be replaced by AI. **The people who use AI to accelerate learning will be irreplaceable.**"

Use it as a Socratic Partner.

Ask it Why, not just How. Challenge its answers.

QUESTIONS?

Constraint-Oriented Architecture

THE "VIBE CODING" TRAP

Liability Warning

NOT FOR EVERYTHING

Vibe coding for side projects is great. Vibe coding for payment systems or healthcare is **negligence**.

Doing the math wrong isn't a bug; it's a civil responsibility failure.

THE PR FLOOD

AGENTS READING CODE

Hundreds of AI PRs every day. Open source maintainers are drowning.

The new frontier: **Custom verification agents** trained on your codebase to review work as a senior would.

HIRING IN THE AGE OF AI

ANTHROPIC'S REALITY

Claude writes 90% of their code.

They are hiring MORE engineers.

Searching for "Engineers good at prompting."

Prompting requires systems knowledge.

MORE OUTPUT ≠ LESS WORK

TASK EXPANSION

You take on more projects because they "feel" faster to start.

BLURRED BOUNDARIES

Work leaks into evenings because "it's just a prompt."

COGNITIVE LOAD

Constant multitasking disguised as 10x productivity.

THE TOKEN ANXIETY

"I replaced Netflix with Claude Code. I lie in bed thinking about what I can spin up before I fall asleep... The agents come with me now."

A new dopamine loop: We swapped social media for "productive" addiction.