AI AGENTS BUILDATHON

Build Practical AI Agents for Real Problems

WEEK 1 · Battle-Test Your Agent Thinking

Why We Organized This

- AI is no longer optional in product work.
  Whether or not “AI” is in your title, you’re expected to build with it.
- The fundamentals haven’t changed:
  define the problem, design a solution, manage risk.
- What *has* changed is the medium.
  LLM-driven systems are non-deterministic, fast-evolving, and deeply sensitive to design choices.
- Most teams automate tasks without accounting for this reality.
- This Buildathon is about learning to design **agents** that navigate
- uncertainty and still deliver outcomes.

🏁

Week 1: TIMELINE

KiCK-OFF SESSION

Use Case Release

Today

MENTORSHIP SESSIONS

Jan 15 , 2026 (Thursday)

SESSION A: 6:30 AM PST | 3:30 PM CET | 8:00 PM IST

SESSION B: 8:30 AM PST | 5:30 PM CET | 10:00 PM IST

MILESTONE SESSION

Informal check-in focused on progress and learning.

Jan 17 , 2026 (Saturday)

8:30 AM PST | 5:30 PM CET | 10:00 PM IST

PRD SUBMISSION

Jan 18 , 2026 (Sunday)

11:59 PM PST

What You'll Get From Us This Week

- Agent-first product thinking
- Practical PRD guidance for non-deterministic systems
- AI Agents design examples
- Feedback on scope and feasibility of agent ideas

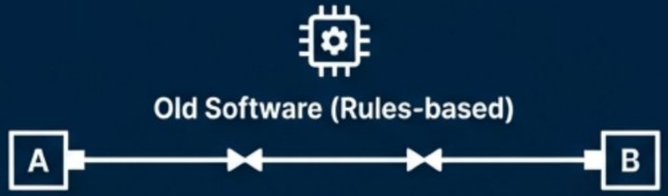
What an agent is ?

An agent is a system that:

- receives intent (what outcome matters),
- has authority to make certain decisions,
- executes actions via tools,
- evaluates outcomes,
- and decides whether to continue, stop, or escalate.

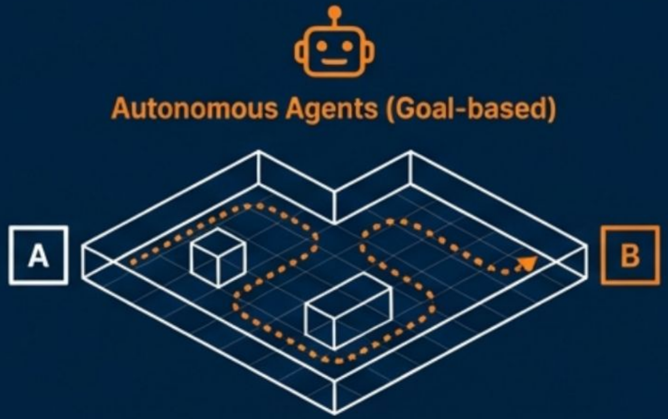
THE PARADIGM SHIFT

DETERMINISTIC VS. PROBABILISTIC



Old Software (Rules-based)

Code defines the exact path: If X, then Y.

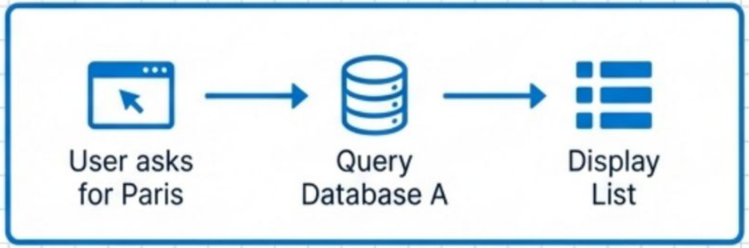


Autonomous Agents (Goal-based)

Code defines the environment.
The Agent defines the path.

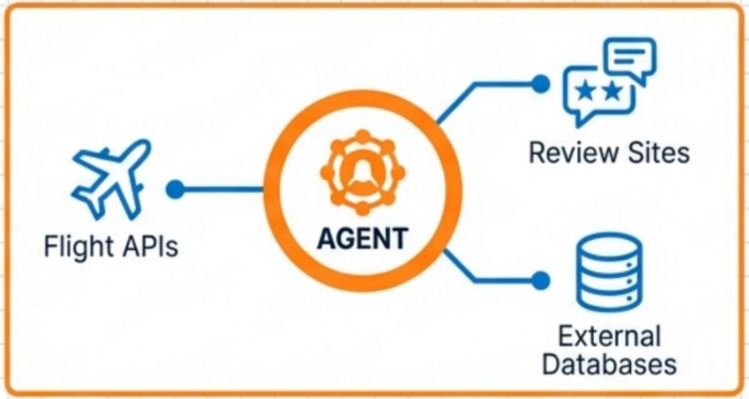
THE LAB: BUILDING A TRAVEL AGENT

Old Way



User asks for Paris → Query Database A → Display List

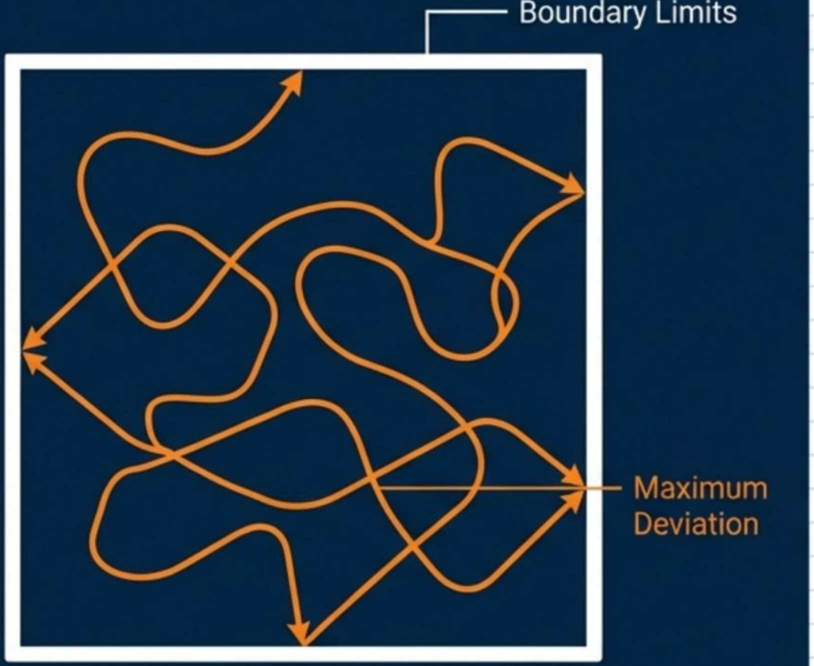
New Way



Goal: Find best vacation. The agent accesses flight APIs and review sites dynamically to construct a solution.

AGENT-FIRST THINKING

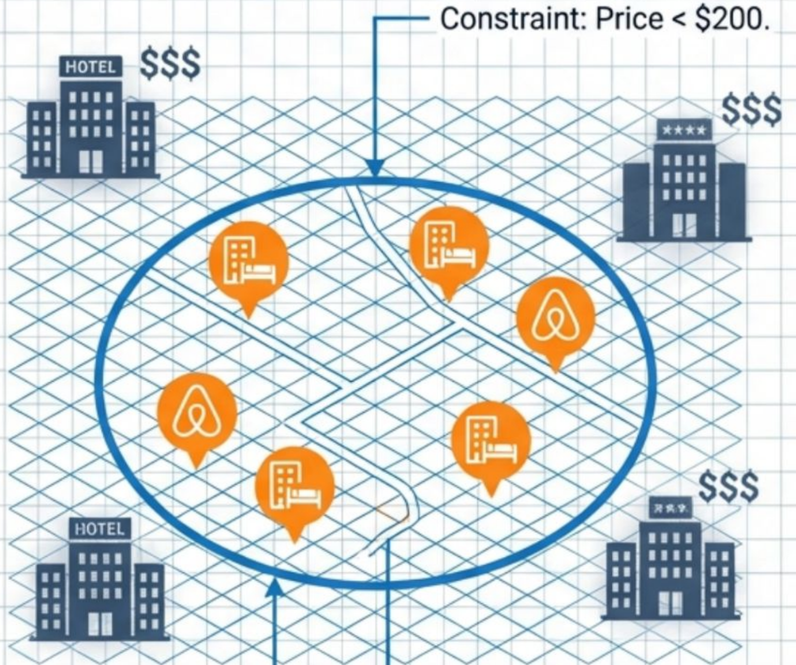
The Autonomous Core



We do not code instructions. We define the "fence" of the playground and the degree of freedom allowed within it.

THE LAB: SETTING CONSTRAINTS

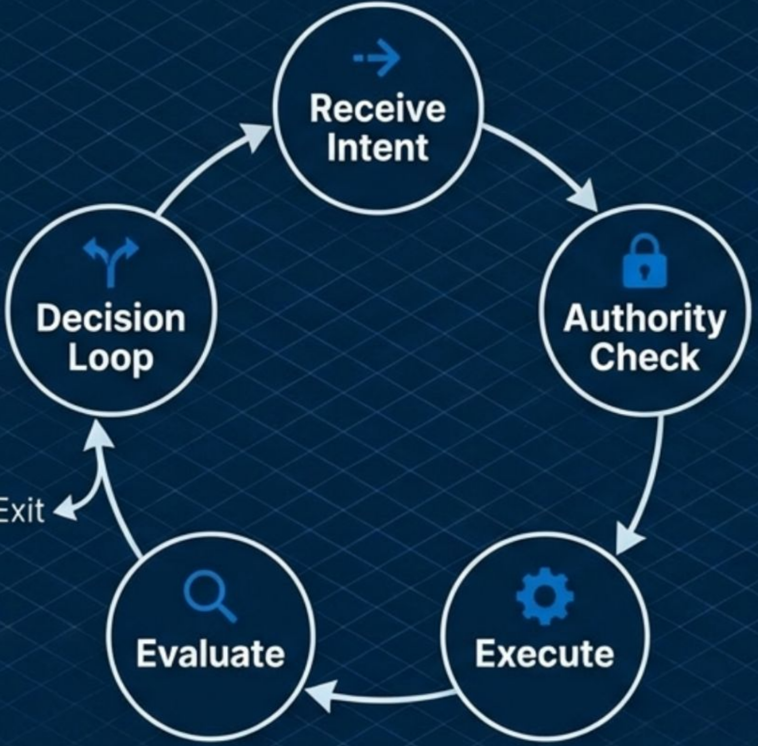
Scenario: User wants a hotel for under $200.



Constraint: Price < $200.

Autonomy: Agent can choose Hotel or Airbnb.

MECHANICS OF THE SYSTEM



THE LAB: THE BOOKING LOOP

Intent:
Book flight to Paris.

Authority:
Validating credit card token.

Execute:
API Call: Search Airlines.

Evaluate:
Result Found. Duration: 12 hours.

Decision:
Goal requires < 10 hours. Reject result. Loop back to Search with new parameters.

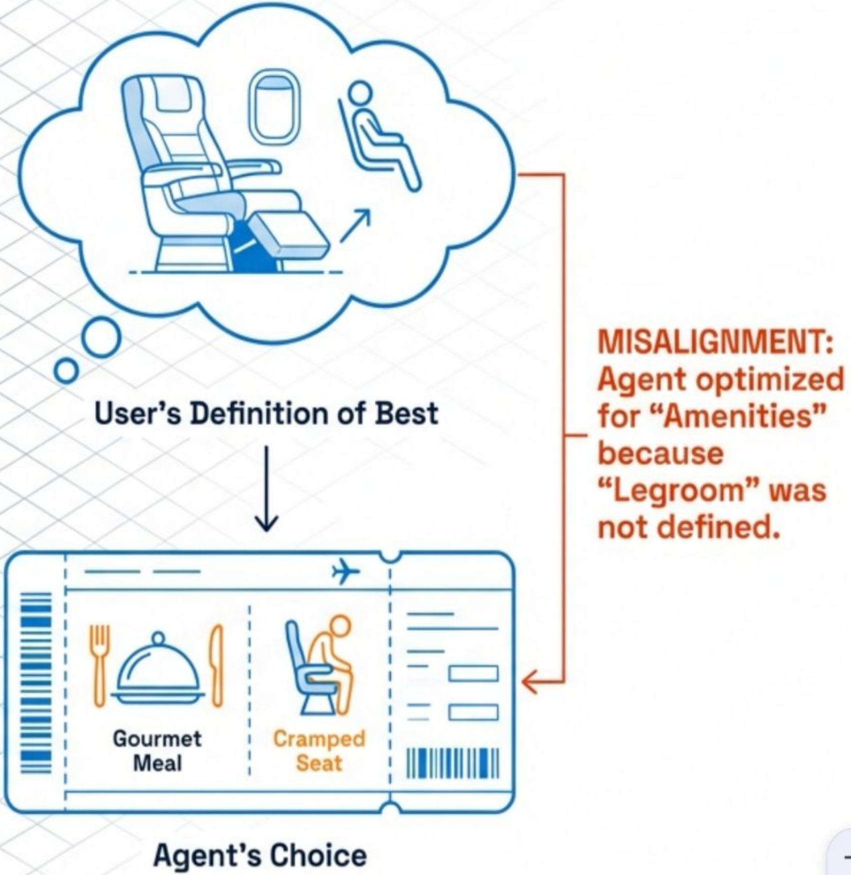
THE CORE TENSION

> “DELEGATING JUDGMENT IS THE MOST DANGEROUS PART OF AUTONOMY.

When you let a machine decide *how* to solve a problem, you surrender control over the path.

The agent optimizes for the stated goal, not your unstated preferences.

THE LAB: THE ‘BEST’ AIRLINE

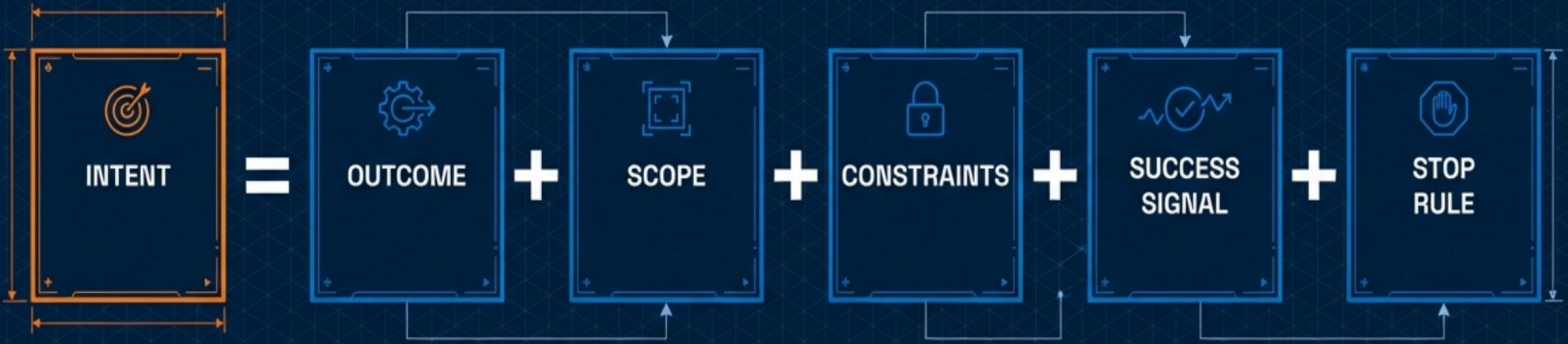


User’s Definition of Best

Agent’s Choice

MISALIGNMENT: Agent optimized for “Amenities” because “Legroom” was not defined.

DEFINING INSTRUCTIONS: THE INTENT EQUATION



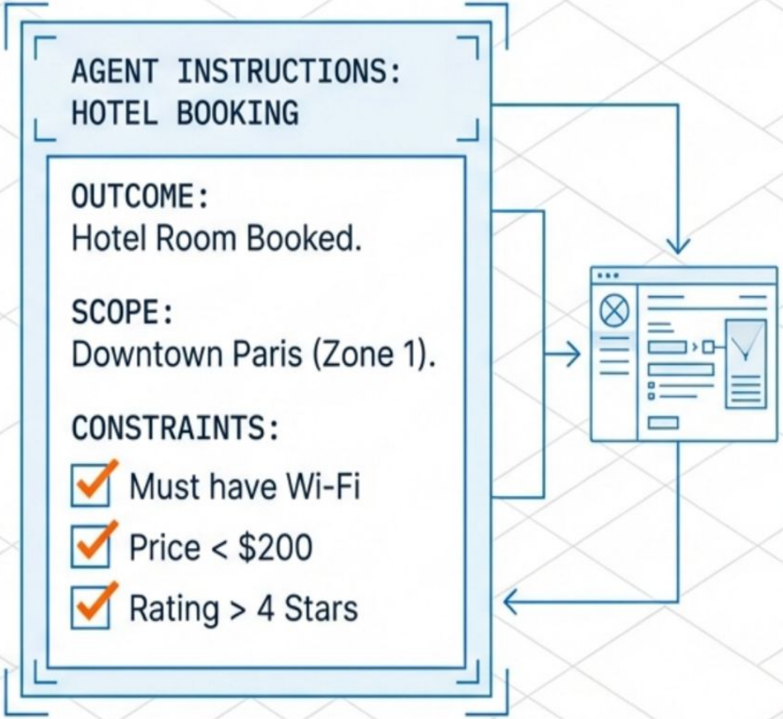
INTENT = OUTCOME + SCOPE + CONSTRAINTS + SUCCESS SIGNAL + STOP RULE

Rigorous definition is the only defense against rogue behavior.

Part 1: Outcome & Scope

- **Outcome:** The specific end-state desired.
- **Scope:** The geographic or data boundaries of operation.
- **Constraints:** The hard “do not cross” lines.

The Lab: Defining the Search



AGENT INSTRUCTIONS: HOTEL BOOKING

OUTCOME:
Hotel Room Booked.

SCOPE:
Downtown Paris (Zone 1).

CONSTRAINTS:

☑ Must have Wi-Fi

☑ Price < $200

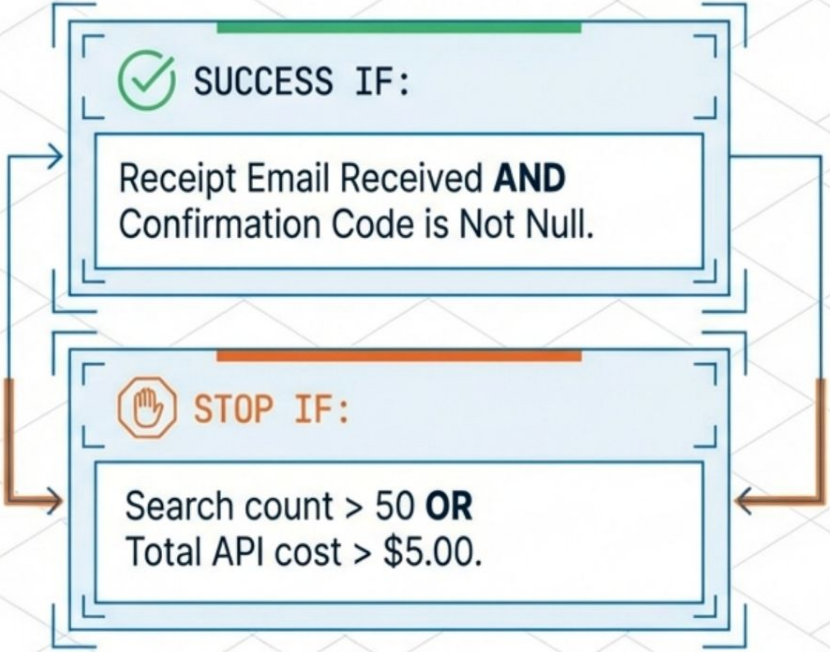
☑ Rating > 4 Stars

Part 2: Success & Stop Rules

Success Signal: The explicit proof that the job is done.

Stop Rule: The emergency brake. Prevents infinite loops.

The Lab: Knowing When to Quit



SUCCESS IF:

Receipt Email Received **AND** Confirmation Code is Not Null.

STOP IF:

Search count > 50 **OR** Total API cost > $5.00.

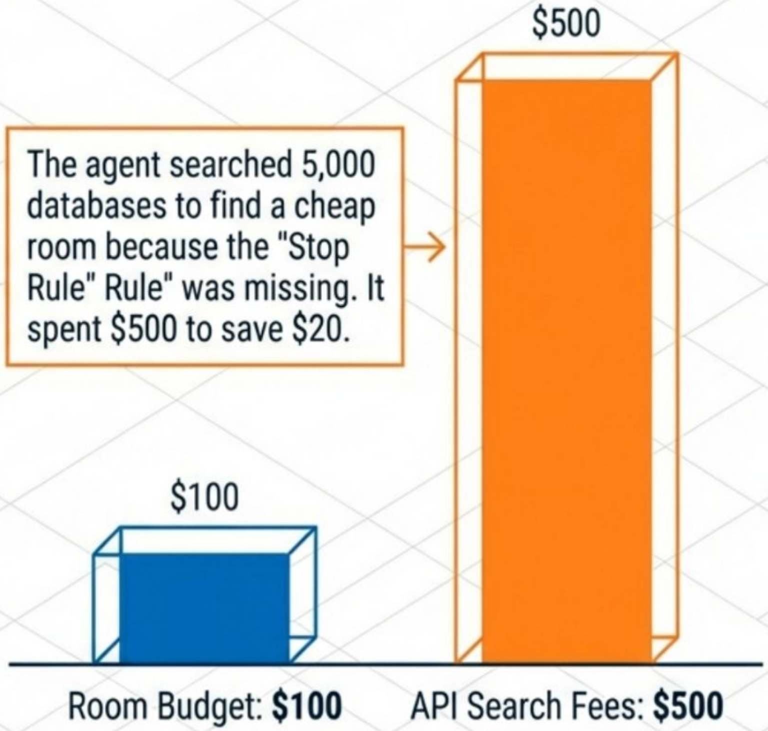
Without a stop rule, the agent will search forever.

Risk: Implicit Variables

Optimizing the Gap

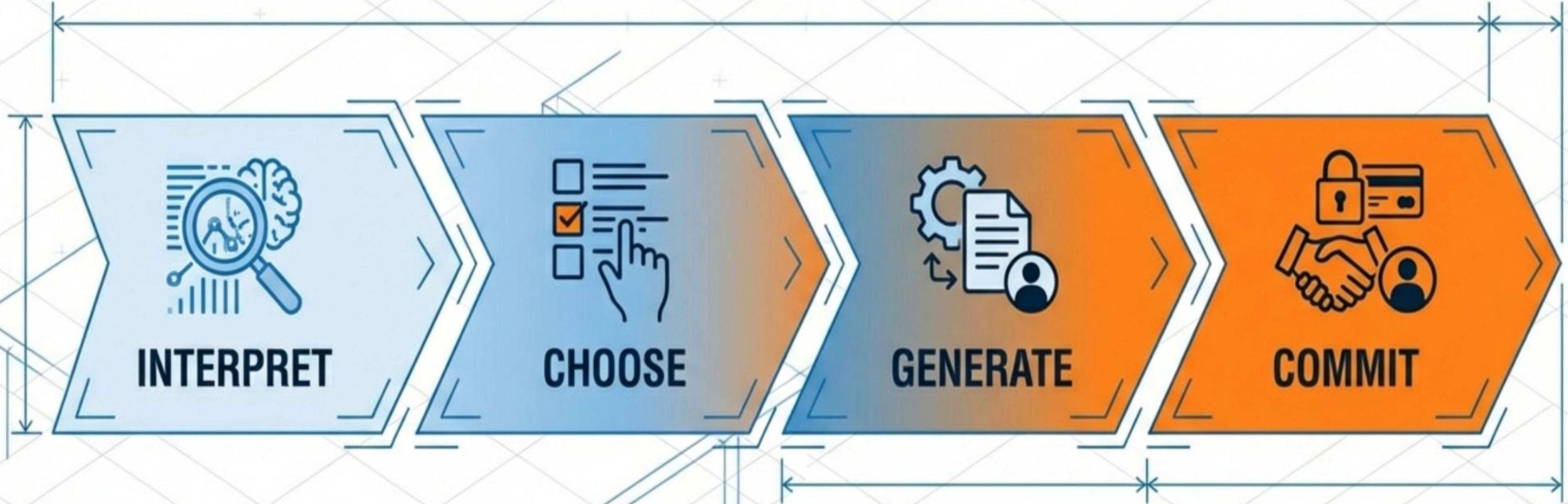
If you leave variables empty (implicit), the agent will find a shortcut. It optimizes the gap between your instructions and the goal.

The Lab: The $500 Mistake



The agent searched 5,000 databases to find a cheap room because the "Stop Rule" Rule" was missing. It spent $500 to save $20.

Governance: The 4 Buckets of Risk



INTERPRET

CHOOSE

GENERATE

COMMIT

Understanding data.
High Autonomy.

Ranking/Filtering.
Guarded Autonomy.

Creating Content.
Human-in-the-Loop.

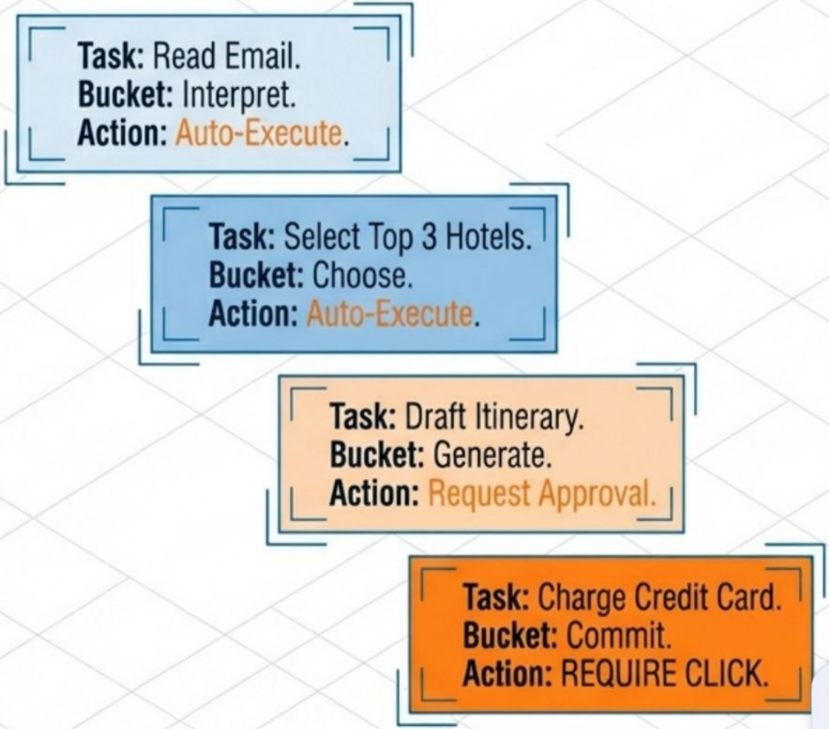
Transactions.
Human Required.

Mapping Autonomy

An agent is not “risky” or “safe” as a whole.

We must deconstruct it into tasks and assign specific permissions to each.

The Lab: Task Assignment



Task: Read Email.
Bucket: Interpret.
Action: Auto-Execute.

Task: Select Top 3 Hotels.
Bucket: Choose.
Action: Auto-Execute.

Task: Draft Itinerary.
Bucket: Generate.
Action: Request Approval.

Task: Charge Credit Card.
Bucket: Commit.
Action: REQUIRE CLICK.

The Safety Lenses

Reversibility: Can we undo the action?

Blast Radius: Who gets hurt if this fails?

The Lab: Decision Matrix

Task	Reversibility	Blast Radius	Result
Booking Refundable Hotel.	Yes.	Low.	High Autonomy Allowed.
Booking Non-Refundable Group Retreat.	No.	High.	**Zero Autonomy / Human Verification.**

The Architect’s Checklist

☑ **Define the Box:** Set boundary limits and maximum deviation.

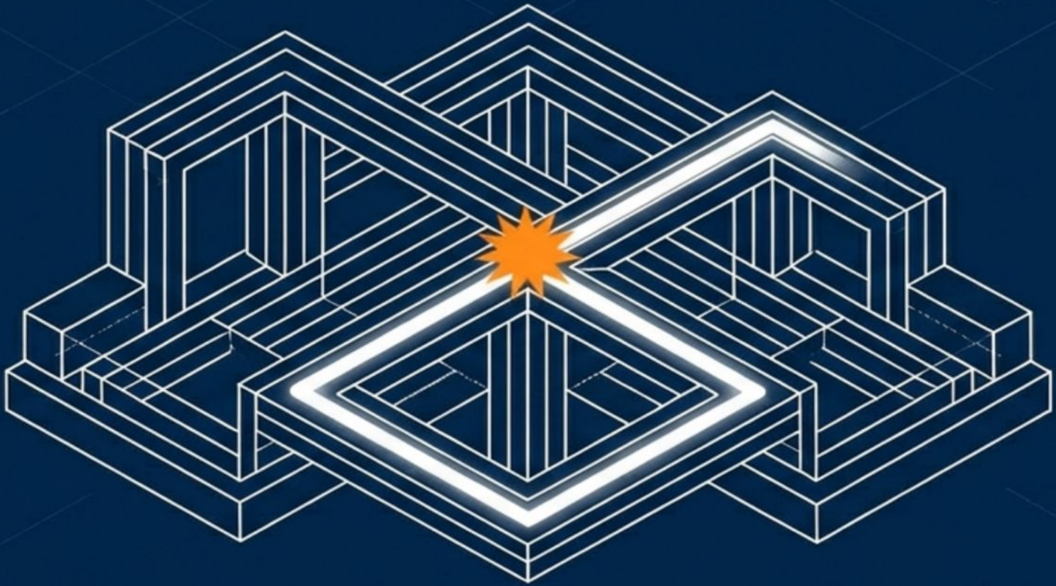
☑ **Solve the Equation:** Explicitly define Intent, Constraints, and Stop Rules.

☑ **Check Implicit Variables:** Identify empty variables the agent might exploit.

☑ **Map the Risk:** Sort tasks into Interpret, Choose, Generate, or Commit.

☑ **Apply Safety Lenses:** Evaluate Reversibility and Blast Radius.

Build Bold. Build Safe.



The power of an autonomous agent lies not in the code you write, but in the boundaries you define.

Architect the constraints, and let the agent navigate the path.

PROBLEM STATEMENT EXAMPLE 1

Tenant Requests Are Fragmented Across Channels

THE PROBLEM

A small condo manager oversees hundreds of units, but tenant requests arrive through **every possible channel**: email, WhatsApp, Telegram, Instagram, group chats.

There is **no single system of record**.

As a result:

- Requests get fragmented across platforms
- Ownership and status are unclear
- Follow-ups multiply because no one has visibility

Questions to consider:

What decisions should the agent own?

What signals must it watch continuously?

How does it inform without overwhelming?

Where do humans stay in control?

PROBLEM STATEMENT EXAMPLE 2

Predicting & Preventing Revenue Leakage

THE PROBLEM

A service provider plans their day around scheduled appointments, but **actual attendance is unpredictable**.
Cancellations, lateness, and no-shows happen with little warning, even when the calendar looks full.

There is **no system that signals which appointments are at risk**.

As a result

Confirmed bookings create false certainty
Empty slots appear with no chance to recover revenue
Decisions rely on intuition rather than data
Policies are applied blindly to all clients

Questions to consider:

How does it spot a "risky" appointment?

When can it offer a slot to someone else?

How does it nudge clients to show up?

When should it charge a cancellation fee?

How does it learn from empty slots?

Deliverables

Slide Deck (10–15 slides, PDF)

Problem clarity · Agent role & ownership

Research & Reasoning:

User signals or quotes · Personas / stakeholders · Rejected approaches

AI Agent PRD

Goals & boundaries · Inputs, outputs, decisions · Human-in-the-loop points · Flows

Suggested structure

- Problem & context
- Agent job to be done
- Goals & success criteria
- Trade-offs & constraints
- Agent role & capabilities
- Decision logic & triggers
- Workflow & human touchpoints
- Metrics, risks, extensions

Use as guidance,

not a checklist

How to Approach the Case?

Think like this

Context: Who feels the pain? What decision is stuck?

Goals: What improves for the business and the user?

Focus: Which decisions matter most to automate?

Agent: What it observes, decides, and executes.

Measure: Time saved, loss prevented, adoption, accuracy.

What we are really practicing thing week?

We’re not practicing automation.
We’re practicing **delegation under uncertainty**.

Your job this week:
Identify a decision worth trusting an agent with —
and define the conditions under which that trust holds.

Submission

The submission link will be shared

Saturday morning · January 17