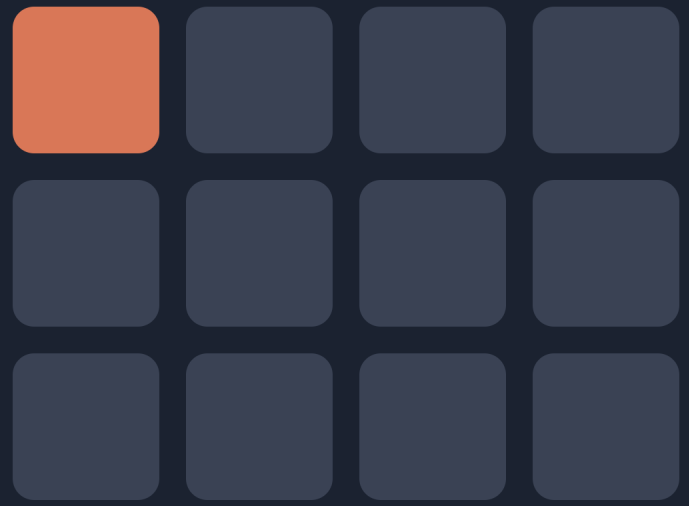


ARCHITECTING WITH CLAUDE

The Principles

A short guide to designing AI systems that work

Daniel Wanja

Nouvelles Solutions · October 2026

Two teams. One model.

Team A: demo first

- A dazzling demo in ten days
- Not in production three months later
- Nobody could say if it was accurate enough
- Cost per claim: unknown

Team B: outcome first

- Measured how long claims take today
- Gathered 200 past claims with outcomes
- Agreed what “good enough” means
- Shipped less; saved time within a quarter

Same model. The difference was the architecture.

Four ideas run through everything

Simple first

Every step up in complexity must be earned by a measured need.

Measure before you argue

Define the measure and the baseline, then test every change against it.

Enforce what matters

“Never do X” in a prompt is a request. Real limits live in code and permissions.

Design for the people

Sponsors, reviewers, lawyers and operators decide whether a system succeeds.

The life of an AI system

1 FRAME
What outcome are we buying?

2 SHAPE
How much autonomy is needed?

3 CONFIGURE
Which model, and what context?

4 CONNECT
What may it touch, and how?

5 GROUND
Where do the facts come from?

6 PROVE
How will we know it works?

7 OPERATE
What does it cost, and how fast?

8 GOVERN
What must never happen?

9 ALIGN
Do the people agree?

10 ENABLE
How does the team move fast, safely?

The decisions arrive in this order, from the first conversation to the day someone else runs the system.

01 · FRAME

Start from the outcome, not the model

Decide which number must move, for whom and by how much, before you choose a model.

Problem statement: “For [actor], when [unit of work] arrives, produce [artifact] so [metric] moves from [baseline] to [target].”

Split the work

Code does rules, lookups and arithmetic. Claude does the messy judgment.

One pillar leads

Efficiency, transformation, productivity, cost or performance. The rest become guardrails.

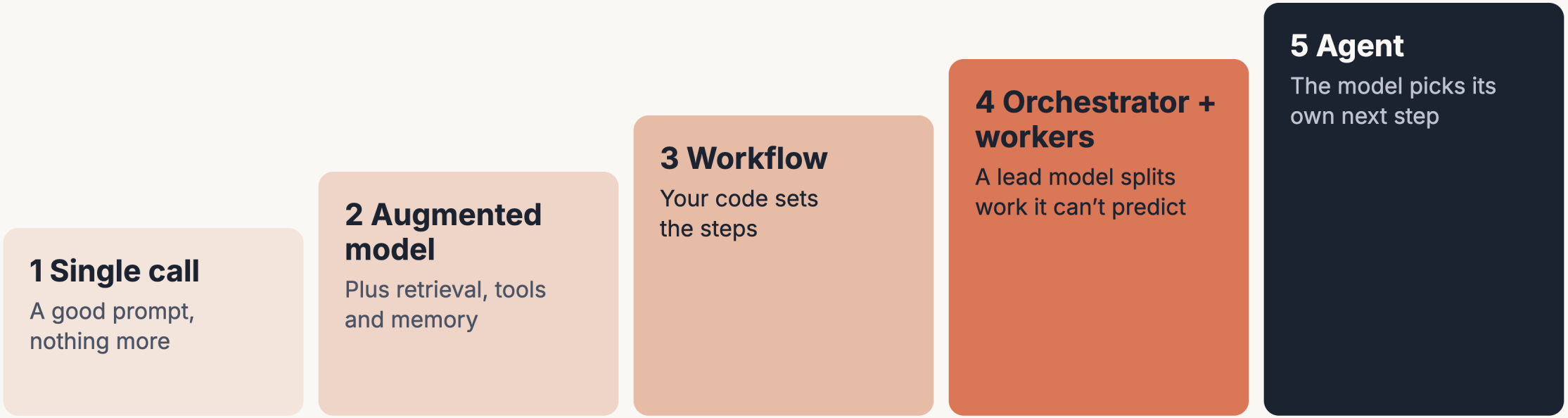
Price the outcome

Model cost × attempts + review time + escaped errors. Not the token.

02 · SHAPE

Use the simplest shape that works

Climb one rung at a time, and only when evidence says the rung below falls short.



02 · SHAPE

Workflows follow code. Agents choose.

Workflow

Your code picks the next step.

- **Chaining:** steps known in advance
- **Routing:** distinct input categories
- **Parallelization:** independent parts, or votes
- **Evaluator-optimizer:** clear criteria

Agent

The model picks the next step. Every agent needs:

- Ground truth from its environment
- Stopping rules and caps
- Bounded tools
- Human checkpoints

Decompose first. The number of agents follows from the problem; multi-agent runs used about 15× the tokens of chat.

03 · CONFIGURE

Pick the model, then spend context

Fit the model with evidence; give it the smallest set of high-signal tokens.

Tier	Best for
Haiku	Speed and volume: classify, extract, route
Sonnet	Everyday coding, analysis, tool use
Opus	Complex, accuracy-critical work
Above Opus	Hours-long agents, deep research

Choose by the hardest common case. Unsure? Start on Opus; step down while evals pass.

Tune effort before you switch models.

Stable first, variable last so the cache can reuse it.

04 · CONNECT

Give agents less

Grant the least capability, and check who the agent acts as at every hop.

Remove → **Narrow** → **Gate** → **Detect**

The order of controls. Never reverse it: logging a tool is weaker than removing it.

Hint vs boundary

If the model reads it, it's a hint. If the runtime enforces it, it's a boundary.

Cut one leg

Never give one agent private data, untrusted content and an outbound channel.

04 · CONNECT

Choose by what's on the other side

On the other side	Connect with
Another team's or company's agent	Agent-to-agent (A2A)
A capability several apps reuse	An MCP server
Scripts in a sandboxed shell	CLI or code execution
One tool for one app	A direct tool definition

Load on demand. Tool choice degrades past roughly 30–50 tools; tool search and Skills load the long tail when needed.

Always load policy. Rules that must apply on every request never wait to be discovered.

05 · GROUND

Ground answers in the right data

Send each question where its answer lives, and make every answer show its evidence.

The question needs	Send it to
Facts in prose	Hybrid search + reranking
Totals, counts, joins	A query tool, never search
Current state	A live call to its system
A small corpus (about 500 pages)	The whole corpus, cached

67%

fewer failed retrievals with Contextual Retrieval plus reranking

06 · PROVE

Define “good” before you build

EVERY CRITERION IS

Metric + **Threshold** + **Test set** + **Grader**

Measure five things: accuracy · latency · cost · safety · security.

ROLL OUT, ONE CHANGE AT A TIME

Offline eval → Shadow → Canary → **A/B test**

Fifty cases cannot prove a four-point gain.

±11 points
margin with 50 cases at an 80% pass rate

±4 points
margin with 400 cases at the same rate

4× the cases
to halve the margin of error

07 · OPERATE

Make cost and speed visible

Find where the time and money go, attack the largest term, and watch quality as you do.

When the big cost is	Reach for
A long prefix on every call	Prompt caching
Work nobody is waiting on	Batch: half price
Many easy requests	A smaller model
Long outputs	Length caps, structure
Users feel the wait	Streaming

Halving a 5% cost line saves less than trimming 10% off a 60% one.

Streaming fixes the wait people feel, not the bill.

One trace per request. Log structure always, content rarely, secrets never.

08 · GOVERN

Enforce, don’t request.

The ignore test: if the model ignored this rule, what would still stop the harm?

Prompts steer. Permissions, tool handlers and sandboxes enforce. Structured outputs guarantee shape, not truth.

08 · GOVERN

Route human review by stakes

LOW STAKES

Automate

Sample outputs to watch quality.

MEDIUM STAKES

Review when unsure

A person checks when the system is unsure.

HIGH OR IRREVERSIBLE

A person approves

Before the action happens, every time.

A real approval gate shows the exact action, is bound to that payload, treats silence as “no”, and is logged.

Users approved about 93% of permission prompts. Seed known errors to keep reviewers alert.

09 · ALIGN

Bring people along

Say the constraints, trade-offs and failure rates out loud, early.

When they ask for	Agree on
“100% accurate”	A threshold on a test set, plus human review
“Instant”	p50 and p95 latency at the client, plus time to first token
“Always available”	An internal SLO, and a looser SLA backed by fallbacks

HAND OVER EVERYTHING THAT SHAPES BEHAVIOR

Architecture · Decision records · Prompt registry · Eval suite · Runbooks · Known limits

10 · ENABLE

Configure the team, not just the tool

Informs

- CLAUDE.md, kept under 200 lines
- Skills that load when needed
- Team conventions and examples

Enforces

- Managed settings beat every other scope
- Permissions: deny, then ask, then allow
- Hooks and a sandboxed shell

DEBUG ONE LAYER AT A TIME

Replay the trace → Isolate the layer → Smallest fix → Regression test

The twelve principles

1 Start from the outcome, not the model

2 Verify it or contain it

3 Use the simplest shape that works

4 Fit the model with evidence; spend context deliberately

5 Grant the least capability; authorize every hop

6 Send each question where its answer lives

7 Define “good” before you build

8 Change one thing and measure it

9 Attack the dominant cost; trace every hop

10 Enforce, don’t request

11 Say it out loud, and write it down

12 Configure the team, not just the tool

Start with one real problem

The fastest way to own these principles is to apply them once, end to end.

1

Write the outcome

Actor, unit of work, baseline, target, cost of an error.

2

Choose the shape

The lowest rung of the ladder that can meet the target.

3

Define “good”

Metrics, test cases and graders before any build.

4

Measure

Change one thing at a time and trace every request.

Go deeper: Anthropic’s documentation and engineering articles

platform.claude.com/docs · code.claude.com/docs · anthropic.com/engineering