BUILDING AGENTS LIKE A PRO

Professional agents are built on tests, tools and context, not three-page prompts

Anyone can write a long prompt. Agents that hold up on real deals are designed, tested and maintained like any critical process. No code is needed to start; some code is needed to finish.

KEY MESSAGE

Professional agents come from tests, tools and context, not from longer prompts.

3 things to know

1. **Start simple, evaluate, then add autonomy** [1].
2. **Context and tools matter more than prompt wording** [1][2].
3. **No test set, no agent:** ten past cases with known answers is the minimum.

EXHIBIT 1 **The build ladder (adapted from Anthropic guidance [1]; dots = complexity)**

LEVEL	WHAT IT IS	COMPLEXITY
Prompt	One request, one answer	● ○ ○
Skill	Method, template, checks	● ○ ○
Workflow	Predefined steps and tools	● ● ○
Agent	Model plans its own steps	● ● ●
Multi-agent	Lead agent with sub-agents	● ● ●

Climb only when the level below fails on your test set [1].

1–2K
tokens a sub-agent returns after exploring tens of thousands [2]

80–90%
resolution rate of a financial-services support agent [3]

Dec 2024
Anthropic's Building effective agents [1]

Why long prompts fail

A three-page prompt mixes method, data, format and exceptions in one block. It is hard to test, breaks when the model changes, and nobody knows which line matters.

Professionals split it: a short instruction, a skill holding the method, templates for the output, tools for actions, and checks at the end.

Context engineering

Anthropic calls context engineering the natural progression of prompt engineering [2]. The question is not the wording but what enters the model's limited attention: which documents, which examples, which tool results, in which order.

Sub-agents can explore widely and return 1,000 to 2,000 tokens of distilled findings [2], keeping the main context clean.

Evaluation, the professional habit

Build a test set from past deals where you know the answer: ten teasers with the memo your team wrote, five data rooms with the issues found. Run every new version against it. Score accuracy, sources, format and time.

Agents bring higher costs and compounding errors; test them in a sandbox with guardrails [1].

Sources on page 3.

CarveLines © 2026

No code, but not without code

Skills, templates, connectors and assistants let a non-coder build most of a deal-team agent. Code enters at three points: connecting a data source no one has packaged, automating evaluation runs, and checking outputs mechanically.

Do the numbers add up, are all sources live, do dates match? A fund does not need engineers in every team; it needs one person who can write that code.

EXHIBIT 2 What a professional agent contains (CarveLines view; dots = effort share)

PART	CONTENT	EFFORT
Instruction	Short goal and constraints	● ○ ○
Skill	Method, templates, examples	● ● ○
Tools	Search, files, Excel, connectors	● ● ●
Checks	Totals, dates, live sources	● ● ○
Test set	Ten past cases with answers	● ● ●
Owner	Reads failures, updates	● ● ○

CarveLines view, consistent with [1][2].

Maintaining agents

Models change every few months. Each change can improve one task and degrade another. Rerun the test set at every model or skill update, and keep the previous version until the new one passes.

Assign an owner to each agent, as to any process: someone who reads the failures and updates the method.

> "While building our agent for SWE-bench, we actually spent more time optimizing our tools than the overall prompt."
>
> **Erik Schluntz and Barry Zhang**, Anthropic · Dec 2024 [1]

> "At Anthropic, we view context engineering as the natural progression of prompt engineering."
>
> **Anthropic Applied AI team**, Effective context engineering for AI agents · Sep 2025 [2]

5 hints: the average fund vs full power

	THE AVERAGE FUND	FULL POWER
1	Writes a three-page prompt	**Splits instruction, skill, template, tools and checks**
2	Judges on a demo	**Judges on a test set of past deals**
3	Keeps agents unchanged when models change	**Reruns tests at every update**
4	Avoids code	**Uses a little code for checks and evaluation**
5	Leaves agents ownerless	**Gives each agent an owner who reads failures**

Watch out

Judging an agent on one good demo. Agents fail on the fifth case, not the first. Only a test set shows the error rate.

Questions for the partners' meeting

1. Which of our AI routines has a written test set?
2. Who owns each agent, and who reads its failures?
3. What happens to our agents when the model changes?
4. Who in the firm can write the few lines of code we need?

Sources on page 3.

CarveLines © 2026

7 recipes: who, how, what you get

01 Test set in a day

CHAMPION

- Pick one repetitive deliverable
- Collect ten past cases with the final human version
- Write what good looks like for each
- Store them as the evaluation set

Output evaluation set
Time 1 day

02 Skill split

CHAMPION

- Take the long prompt
- Split it into instruction, skill, template and checks
- Move examples into the skill
- Retest on the set

Output maintainable agent
Time half a day

03 Mechanical checks

BUILDER

- Write a short script with an AI coding assistant
- Check totals, dates and that every link resolves
- Run it on every output
- Block outputs that fail

Output automatic QA
Time 2 hours

04 Eval run

CHAMPION

- Run the agent on the ten cases
- Score accuracy, sources, format and time
- Compare with the human version
- Log the score with the version number

Output score sheet
Time 1 hour

05 Model-change protocol

AI LEAD

- When a new model ships, rerun the evaluation set
- Compare scores with the current model
- Switch only if nothing regresses
- Record the decision

Output no regressions
Time 1 hour per update

06 Tool tuning

CHAMPION

- List the tools the agent calls
- Improve their descriptions and outputs
- Remove tools it never needs
- Retest

Output fewer errors
Time half a day

07 Agent register

AI LEAD

- One line per agent: owner, purpose, test score, last update
- Review monthly
- Retire agents without an owner
- Share with champions

Output one-page register
Time monthly

Monday morning

Choose one repetitive deliverable. Collect ten past examples with the final human version. That set is your evaluation: nothing is deployed until it passes.

Sources: [1] Anthropic, Building effective agents, Dec 2024; [2] Anthropic, Effective context engineering for AI agents, Sep 2025; [3] Anthropic, Building effective AI agents: architecture patterns and implementation frameworks (Gradient Labs case). Quotes verbatim from the sources cited. Figures as published; survey scopes and dates vary. Vendor-reported figures are labelled; [E] = CarveLines estimate; 'CarveLines view' = our interpretation.

CarveLines © 2026