Rethink the Work

What coding agents teach us about building AI for the rest of the business

Ken Tabor

Watch the Shape of the Work

The lesson is revealing. Many useful design patterns have wider application than programming.

Look for This

Orient → Plan → Act with Tools → Verify →
Keep the human in the loop

Orient Before Acting

The agent builds a map before changing anything

It verifies the environment is healthy

It looks for the system's "seams"

Turn Intent Into a Plan

```
› I like the web app a lot. It's very useful, and provides a human-centric user experience that's easy for me to navigate. All
  current information, pages, and UI design in the webapp is very good for me.
  Highly useful, relevant, and well organize.
  I imagine a new panel widget that covers the stocks that are most important to me because they are a part of my investment
  portfolio.
  It is a new UI component, but shall match and reuse the webapp's overall visual design.
  It can be added to the home, about, getting started pages.
  This is the list of stocks that I currently care most about, and want to have top-of-mind, and easy access to their data in the
  webapp:
  * ASTS, AUR, CRWV, EOSE, GENB, JOBY, SECZ, SPCX, TE, WULF

• I'll add a reusable portfolio-focus panel with these ten companies, styled from the existing app components and placed on Home,
  About, and Getting Started. I'm using the site-building guidance because this is a cross-page webapp UI addition.

• Explored
  └ Read SKILL.md (sites:sites-building skill), CompanyJumpBar.jsx, CompanyCard.jsx, SummaryStat.jsx, Shell.jsx,
  DashboardPage.jsx,
        AboutPage.jsx, GettingStartedPage.jsx, index.css, themeArtwork.js, main.jsx

──────────────────────────────────────────────────────────────────────────────────────────────────────────────────

• I found a clean fit: a single reusable "Portfolio focus" panel that pulls current company names and coverage status from the
  live catalog, so it stays accurate as reports arrive. I'll place it immediately after each page's opening context, with direct
  company links and zero-report names clearly marked as coverage-ready.

• Explored
  └ Read GettingStartedPage.jsx, AboutPage.jsx, formatters.js

──────────────────────────────────────────────────────────────────────────────────────────────────────────────────

• The panel will preserve your list order and show each company's current report count, with "Coverage ready" for names still
  waiting on source material. It will be a direct path into each company page—not a portfolio tracker or a recommendation tool.

• Edited 5 files (+86 -0)
  └ ema_analysis_webapp/src/components/PortfolioFocusPanel.jsx (+70 -0)
     1 +import { Link } from "react-router-dom";
```

The human describes the outcome, not the implementation

The agent translates intent into a concrete plan

It reasons about product boundaries, not just the code

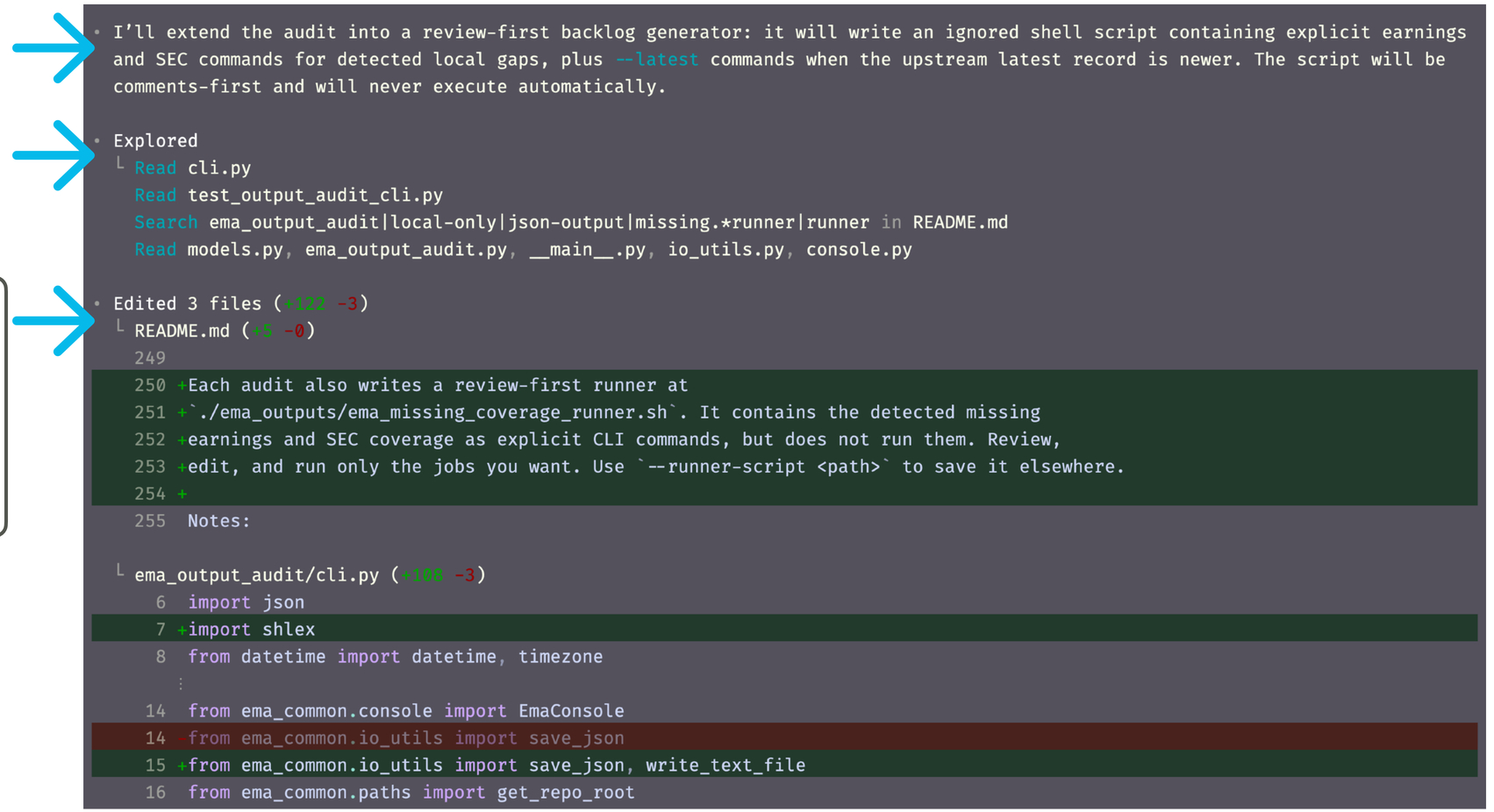
KenTabor

Use Tools to do Real Work (pt1)

The agent turns intent into a concrete work plan

It inspects before it changes

It leave an auditable trail of what changed



Use Tools to do Real Work (pt2)

The agent knows when its built-in knowledge is not enough

Specialized skills extent the agent's behavior

It scopes the changes and shows the delta.

```
› update that output helper runner script to use the "terra" 5.6 model

• I'll change only the generated audit runner's fallback model from Luna to Terra, while leaving the project-wide processing
  default unchanged. I'm checking the official model reference before making that scoped configuration update.

• Explored
  └ Read SKILL.md (openai-docs skill)

• Searching the web

• Searched the web for site:developers.openai.com/api/docs/models gpt-5.6-terra Responses structured outputs

• Searching the web

• Searched the web

──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────────

• Terra supports the existing Responses API structured-output workflow. I'll make the generated runner explicitly pass gpt-5.6-
  terra so an ambient EMA_OPENAI_MODEL value cannot silently change the reviewed commands.

• Explored
  └ Search MODEL=|gpt-5\.6-(luna|terra)|ema_missing_coverage_runner in ema_output_audit

• Edited 3 files (+5 -3)
  └ README.md (+3 -2)
    251  `./ema_outputs/ema_missing_coverage_runner.sh`. It contains the detected missing
    252 -earnings and SEC coverage as explicit CLI commands, but does not run them. Review,
    253 -edit, and run only the jobs you want. Use `--runner-script <path>` to save it elsewhere.
    252 +earnings and SEC coverage as explicit CLI commands using `gpt-5.6-terra`, but does
    253 +not run them. Review, edit, and run only the jobs you want. Use
    254 +`--runner-script <path>` to save it elsewhere.
    255
```

Act, then Verify

The agent chooses tools to verify its own work

It narrows first, then broadens

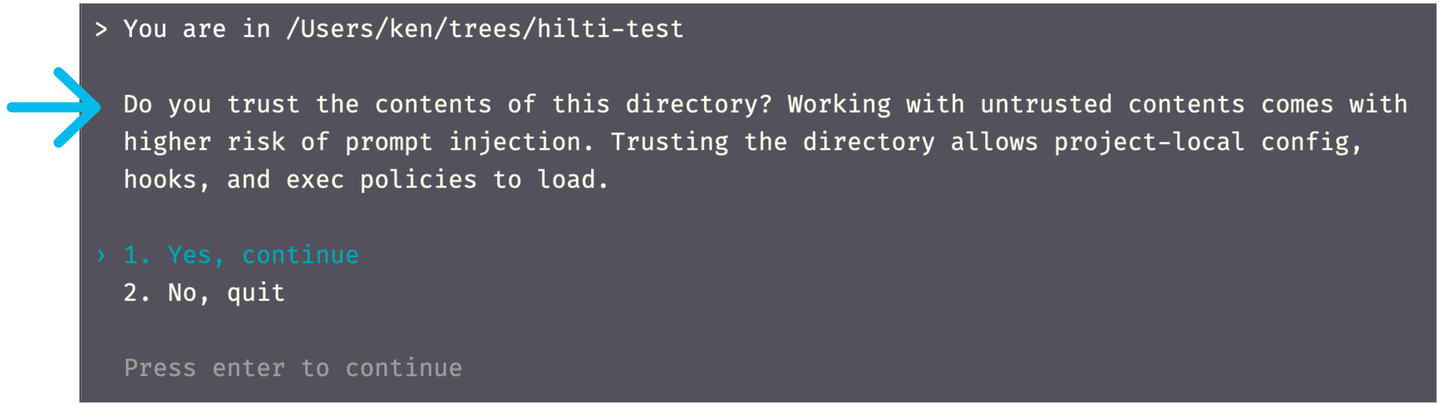
It reports evidence, not just confidence

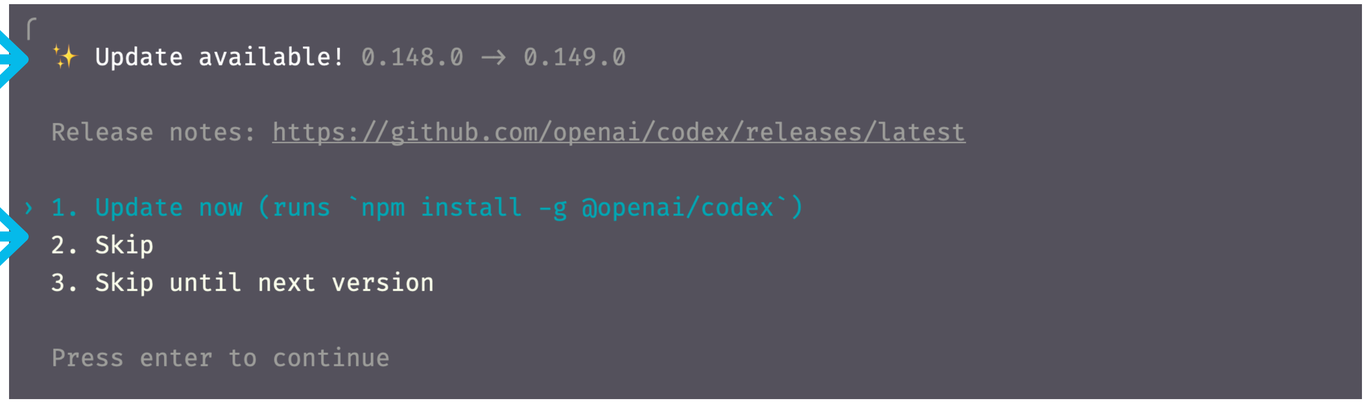
Autonomy with Human Control

Trust is explicit, not assumed

Local context changes agent behavior

Security starts at the boundary





The agent manages its own operating environment

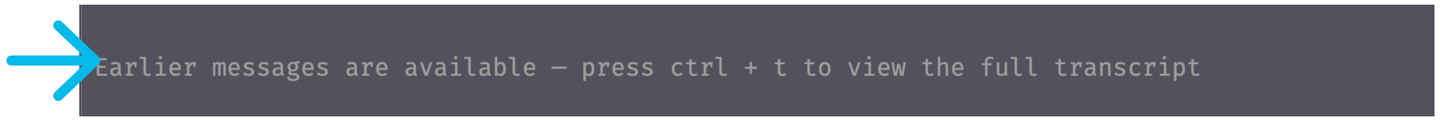
The human keeps control of change

Maintenance is part of the experience

The agent has continuity beyond the visible

Context is managed, not simply dumped into view

The human can inspect the memory trail



KenTabor

Coding Agent Case Study

A capable AI agent looks less like a chatbot and more like a well-equipped junior colleague.

Qualities of Useful AI Agents

Understands the situation

Forms a plan

Uses tools

Acts

Checks its work

Knows when to hand judgment back to a human

Why Did Coding Become So Agent-Friendly?

Accessible Context

The AI agent can see and understand much of the working environment.

The work is already highly digital

A great deal of relevant context can be inspected

Git reverses what changed

Executable Actions

The agent can move beyond advice and actually perform the work.

Tools are readily available

Actions can be executed directly

Many failed attempts are reversible

Tight Feedback

The environment continuously tells the agent whether its work is succeeding.

Linters complain

Compilers assert

Tests pass or fail

Programming gives AI agents something unusually valuable

A world they can observe, act upon, and learn from. They are enabled to read and write, test and learn, hypothesize and verify. Given a functional sandbox, coding agents can explore confidently while bounded permissions, observable behavior, and reversible actions enables safety.

See it → Do it → Check it

Choose It

See it → Do it → Check it ⤴

Approve It

Choose It

See it → Do it → Check it ⤴

Approve It

Human intention

Loop engineering

Read relevant context

Model plans

Tools are selected

Actions executed

Environment responds

Agent observes results

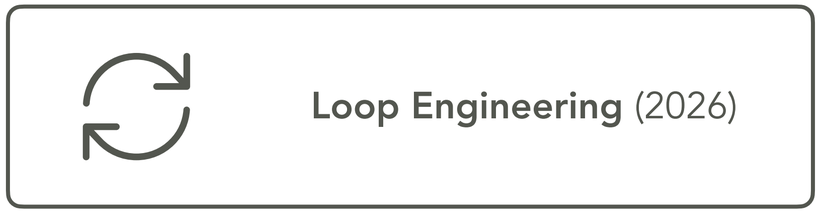
Continues or retries

Outcome evaluation

Human controls judgement

KenTabor

Surface Area Keeps Expanding

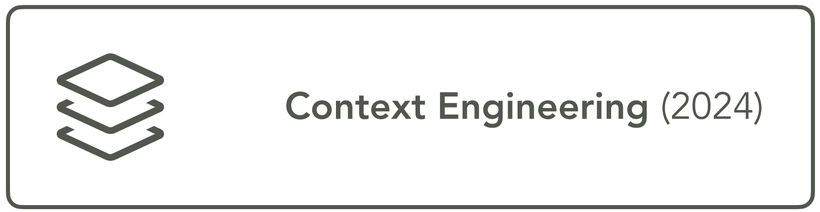


Loop Engineering (2026)

Iteration

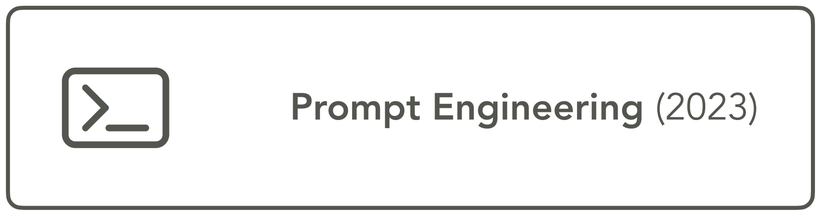
Harness Engineering (2025)

Execution



Context Engineering (2024)

Information



Prompt Engineering (2023)

Instruction

ChatGPT (Nov 2022)

Conversation

Nothing Went Away

Each new engineering discipline builds on the ones before it. The surface area of responsibility keeps expanding.

Builder's Burden

More AI capability brings more responsibility. Understand how the models work, and how to make their behavior useful.

Model is Not a Product

Useful AI emerges from the larger system we design around the LLM: context, tools, state, boundaries, and human control.

From Response to Work

We see the progression from Generative AI conversations towards iterative execution to pursue valuable work over time.

The Machinery Around the Model

A **harness** provides the reusable machinery that enables a model to pursue work across multiple steps over time.

Provides

Context

Tools

State

Loop

Permissions

Traces

Adopt (Reuse)

Choose a proven foundation from a known provider.

Invent (Build)

Domain knowledge, workflows, control surface and business judgement.

Approaches

OpenAI Agents SDK (Model-native)

Claude Agents SDK (Model-native)

Google ADK (ecosystem/cloud)

LangGraph (flexible orchestration)

Microsoft Agent Framework (enterprise)

Given our coding agent case-study

Codex shows many useful agentic patterns. A harness provides reusable machinery. Your domain determines the purpose as you apply the lessons learned.
NOTE: the same model + a different harness → produces different outcomes.

Copy the Pattern, Not the Domain

Do less of this

Don’t reproduce the coding agent’s job.

Don’t ask, “Where can we bolt AI onto the existing workflow?”

Do more of this

Reproduce the conditions that allow it to work well.

Ask, “Where does useful work already have context, tools, feedback, verification, and bounded risk?”

Define Success Before Autonomy

Can’t evaluate it ?

Don’t automate it !

Design Trustworthy Systems

Evaluation

How will we know whether the result is correct or sufficiently good?

Permissions

What can the agent read? What can it change?

What requires human approval?

Blast Radius

What happens when it's wrong?

Can the failure be contained?

Reversibility

How easily can the agent's actions be undone?

Observability

Can we inspect what happened? What's in the context? Tool telemetry? Time?

Human Control

What happens when it is wrong?

Can the failure be contained?

KenTabor

Give Humans a Control Surface

A useful conceptual architecture

Humans set intent and exercise judgment. The agent coordinates the work. Systems provide the facts and actions. Humans decide. → Agents coordinate. → Systems execute.

Control Surface

The human-centered presentation layer for the AI agent.

Frames the work

Shows progress

Authorization requests

Enables intervention

Displays the outcome

Trust is Earned Over Time

Users build from repeatedly seeing what your AI agent does, why it did it, and what happens when it gets something wrong.

Evidence

Show its work

Sources

Tool calls

Decisions

Verification

Control

Let humans approve

Interrupt

Redirect

Constrain

Inspect

Recovery

Contain failures

Undo actions

Retry safely

Learn from outcomes

Trust comes with predictability

Humans gain trust when they begin to understand: what the agent is good at, where it tends to struggle, when it will ask for help, and how it behaves when uncertain. An agent doesn't need to be perfect to become trusted. It needs to be legible enough so that a human learns "What kind of coworker is ai?"

Everyone Has a Role Building AI agents

Building an AI agent is a team sport. Each discipline owns a different part of marking autonomy useful and trustworthy.

Tribe lead

Value

Find work worth redesigning

Prioritize business impact

Resist AI novelty

Product owner

Outcome

Start with customer friction

Define success

Protect the job-to-be-done

Architect

System

Connect context & tools

Design permissions & boundaries

Define integration seams

UX

Agency

Make work visible

Enable intervention

Build calibrated trust

DevOps

Control

Observe every run

Govern deployment & access

Make failure recoverable

QA

Evaluation

Define “good”

Test expected & edge cases

Measure regressions

Developer

Action

Build the agent loop

Give it useful tools

Make actions reliable

KenTabor

Find the Loop (Not the Chatbot)

Your role is finding a job to automate.

Does a job have "agentability"?

Finding Promising Agentic Problems

1. Real Friction

Is there a problem worth solving?

2. Accessible Context

Can the agent understand enough of the world?

3. Useful Action

Can it actually improve the problem?

4. Clear Evaluation

Can we tell wether it did well?

5. Bounded Failure

Can we afford for it to be wrong?

Customer Pain

Who is already spending meaningful time, attention, or effort on work? What part of the experience is actually worth improving?

Job Before Agent

Are we solving a real recurring job, or starting with an impressive AI capability and then searching for someone to use it?

Changed Possibility

Does modern AI materially change how this job can be performed because it can now understand context, take action, or iterate with feedback?

Worth Building

If this worked well, would the result be valuable enough for a customer or teammate to notice, trust, and want to use it again?

Let's Think This Through

Hypothetical Agent → 5-point Assessment → What Needs Validation?

1. Luxury Property Follow-up Agent

Illustration only

Could an agent help turn open-house interest into qualified, well-timed, conversations?

Friction

Strong

Open-house follow-up

Prospect qualification

Personalized outreach

Timely responses

Meeting coordination

Context

Promising

Contact details

Showing notes

Buyer interests

Property information

Prior conversations

Action

Strong

Prioritize prospects

Draft personalized outreach

Share relevant details

Recommend next steps

Schedule conversations

Evaluation

Strong

Response rate

Qualified prospects

Follow-up speed

Meetings booked

Relationship progression

Boundary

Promising

Human-reviewed outreach

Sensitive qualification

Financial claims

Negotiation

High-stakes communication

What would we validate?

Can the AI agent increase timely, personalized follow-up without making a high-touch luxury relationship feel automated?

2. Customer Inbox Agent

Illustration only

Could an agent help a small business owner manage customer email without losing the human touch?

Friction

Strong

Repetitive questions

Account maintenance

Billing investigations

Cancellation requests

Daily inbox backlog

Context

Promising

Email history

Customer accounts

Billing records

Policies + FAQs

Schedule + pricing

Action

Strong

Triage messages

Answer routine questions

Draft personalized replies

Prepare account changes

Escalate exceptions

Evaluation

Strong

Response time

Resolution rate

Answer accuracy

Customer satisfaction

Retention outcomes

Boundary

Promising

Auto-answer low risk

Human-review billing

Approve refunds

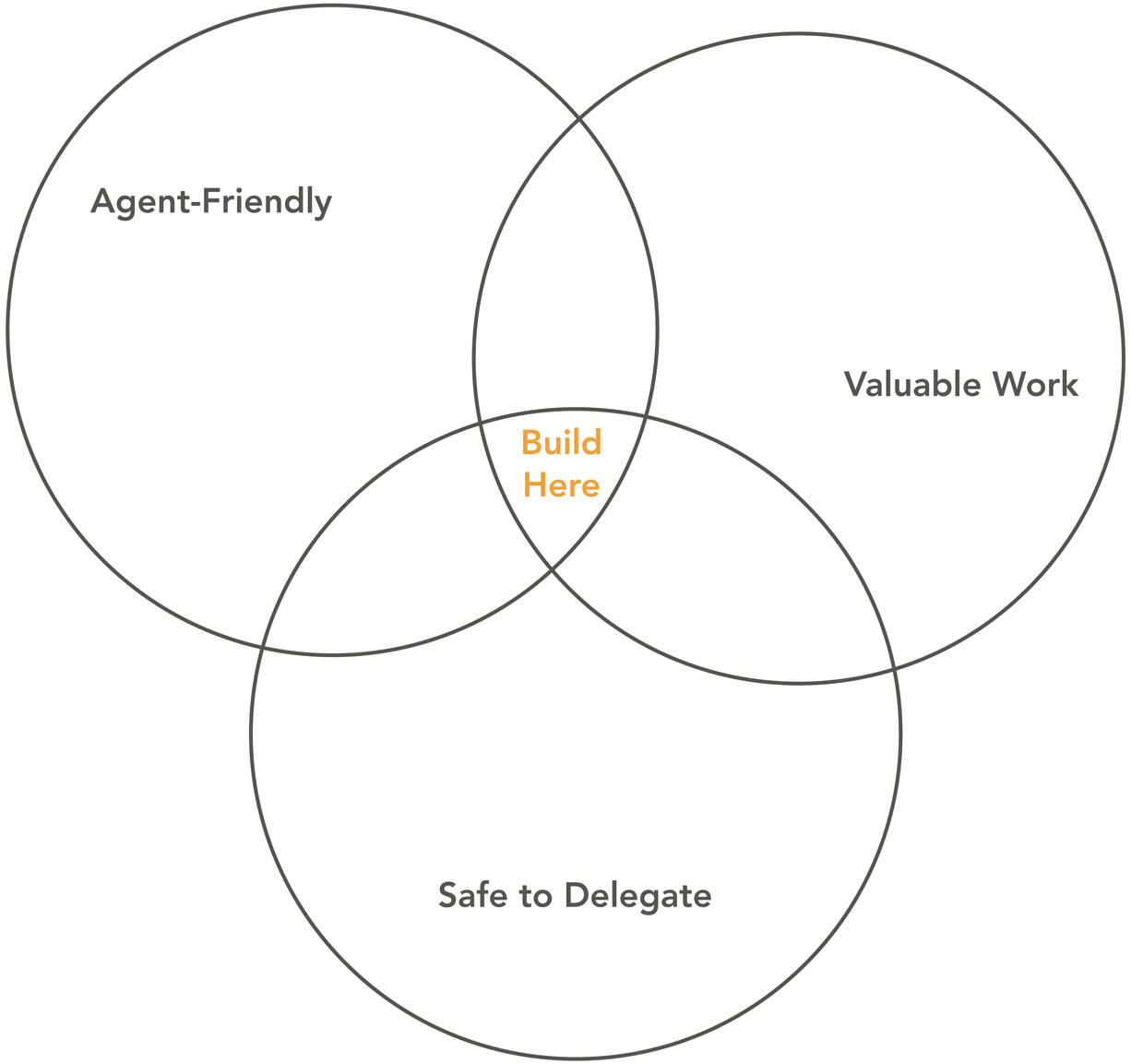
Review cancellations

Escalate sensitive cases

What would we validate?

Can the agent remove routine inbox work while preserving the responsiveness, judgment, and personal care customers expect from a small business?

Minimum Viable Agent



Beware Traps

You'll discover overlaps carry meaning. Decide if they're an anti-pattern or a pursuit.

An agent can do the work, but mistakes have unacceptable consequences. (High Potential)

A problem can be valuable but not agent-friendly. (Worth Exploring)

It can be agent-friendly but not valuable. (Easy Automation)

Spend the Gain on Ambition

What new things become possible when people have more leverage?

Don’t spend it simply doing more of what your did yesterday!

What can your team attempt now that would have seemed unreasonable before?

Let’s do something awesome together!

Ken Tabor

Code and Chats