Course intro & what is agentic coding

Key takeaways

1. Agentic coding shifts work from typing toward framing, steering, and acceptance.
2. Use short loops: bound one turn, inspect its evidence, then decide.
3. Control blast radius with sandboxes, approvals, checkpoints, and scoped secrets.

Common misunderstandings of working with coding agents

Which claim would you defend right now? Choose before the evidence.

"A detailed prompt should be enough for the agent to finish a feature."

"If the agent's own tests pass, the change is ready to accept."

"Security becomes important only when the agent can access production."

Keep your choice. Look for the mechanism or counterexample that would make it fail.

The Winds of Change

The shift is already large enough that pretending it is temporary is a bad bet

When the winds of change blow, some build walls and others build windmills.

75 %

of new Google code is AI-generated

Pichai (Apr 2026) — 25 % in late 2024

20 M

GitHub Copilot users

GitHub Universe (2025)

8,000

Meta jobs cut as AI agents take the work

Entrepreneur (May 2026)

80 yr

Erdős unit-distance conjecture disproved

Sci. American (May 2026)

Agentic Coding

Software engineering when AI agents write code

This course is about becoming effective at directing coding agents.

You own the system

Correctness, performance, data, security, cost, and maintainability stay human responsibilities.

Say what correct means before you delegate.

The agent owns typing

It searches, edits, tests, refactors, explains, and iterates faster than you can type.

Delegate the keystrokes, not the decisions.

The job is management

Set goals, supply context, demand evidence, and decide when the work is actually accepted.

Manage: specify, context, validate, decide.

The future software engineer is less a typist and more a systems-aware manager of agents.

“Vibe coding” names a workflow, not a tool

Prototype-speed defaults are legitimate; the mistake is not noticing what they skip

- Vibe coding (Karpathy, 2025): accept the output, paste errors back, never read the code.
- The same agent serves prototypes and production — the workflow is what differs.

> ***“The key differentiator is not whether you use AI. It's how much structure, verification, and human judgment surrounds the AI's output.”***
>
> — Osmani, Saboo & Kartakis, The New SDLC With Vibe Coding (Google, May 2026)

This course teaches the surrounding structure: context, plans, isolation, gates, review, reset.

What I do all day to steer agents

Six controls that turn agent output into manageable engineering work

Context
AGENTS.md / CLAUDE.md, repo maps, commands, live gotchas

Plans
Ask for the approach before expensive edits; challenge file boundaries

Isolation
Use branches, worktrees, checkpoints, and narrow diffs

Gates
Run tests, type checks, linters, builds, audits, and benchmarks

Review
Read the diff; ask what changed, why, and what could break

Reset
Use /clear, handoff notes, and fresh reviewers when context gets stale

Every control converts uncertainty into context, constraints, evidence, or a smaller task.

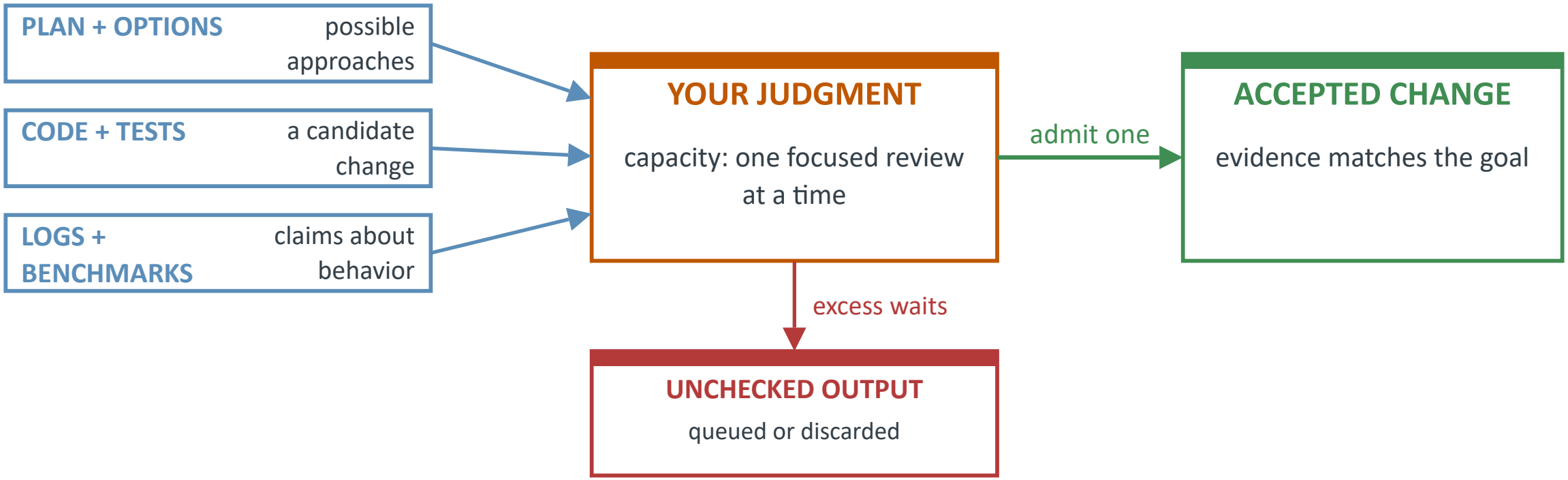
When steering fails, use the same six controls

Each symptom points back to one habit from the previous slide

Symptom	Control	Next move
The agent repeats an old assumption	Context	Restate the current goal and the facts that changed.
The plan touches surprising files	Plans	Stop before edits; ask for two narrower approaches.
The diff spills into unrelated code	Isolation	Split the task or revert the unrelated part.
More edits produce the same failing result	Gates	Run the smallest failing check; ask the agent to explain the mismatch.
The summary sounds right, but behavior feels wrong	Review	Trace one normal and one failure case through the changed code.
The conversation argues with itself	Reset	Save a short handoff and restart with fresh context.

Code is abundant; judgment is scarce

Agents multiply proposals; they do not multiply your verification bandwidth



More agent output increases the review queue unless you narrow scope or improve evidence.

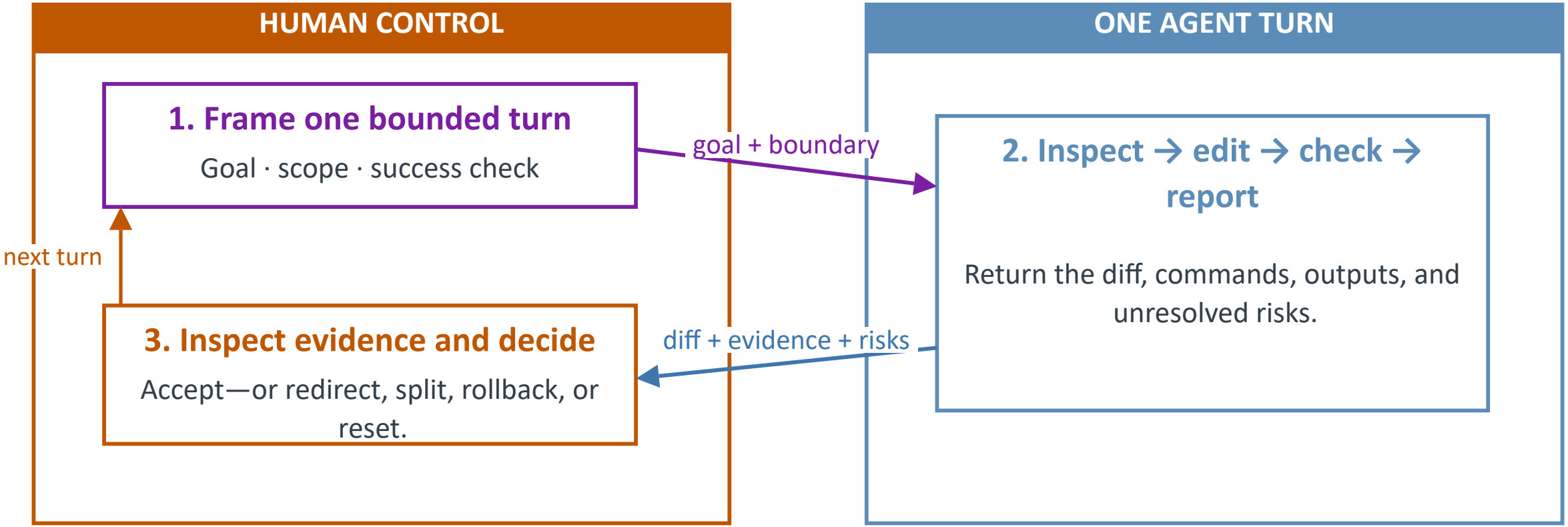
Systems judgment remains the hard part

Agents remove drudgery; they do not remove engineering tradeoffs

Systems issue	What the agent can do	What the human must own
Correctness	Generate tests and plausible fixes	Define what correct means
Concurrency	Patch locks, atomics, or queues	Choose the invariant and review interleavings
Persistence	Write storage code	Demand crash recovery and durable evidence
Performance	Try optimizations	Design honest measurements and reject noise
Security	Find obvious risks	Decide what capabilities and secrets are allowed

Manage agents one inspectable turn at a time

The human closes the loop: every agent turn returns evidence to a decision



Accept exits the loop; redirect, split, rollback, or reset starts the next bounded turn.

Good specifications make agents cheaper

A spec is a management artifact, not ceremonial writing

Weak instruction	Useful instruction
Make the search faster.	Median search under 10 ms on corpus X; preserve exact ranking; report benchmark command and raw output.
Fix the crash.	Reproduce the crash, name the failing invariant, add a regression test, then patch the narrowest cause.
Clean up the code.	Remove duplicate parsing paths without changing public behavior; keep generated files out of the diff.
Build the feature.	Implement the documented API, list touched files, include acceptance checks, and stop before broad refactors.

Three ways to check an agent's claim

Use evidence you can rerun, inspect, or try to break

Rerun the check

Run the exact command yourself from a clean state. Example: pytest tests/test_search.py -q passes after the change.

Ask: Can I reproduce it?

Inspect the artifact

Open the thing that changed. Example: read the diff and render the affected PDF page instead of trusting the summary.

Ask: Does the output match?

Challenge one case

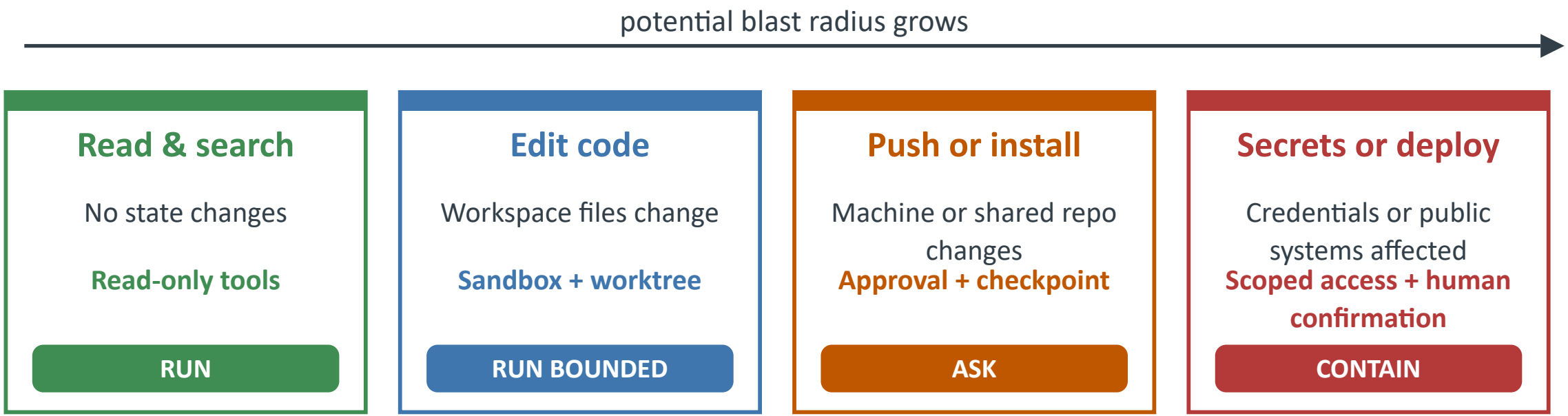
Pick the feature you know best. Ask the agent to trace one normal input and one failure through the exact changed functions.

Ask: Where could this break?

An explanation tells you where to look; evidence tells you whether to accept.

Security starts by limiting the blast radius

Sandboxes bound routine work; approvals guard deliberate escape hatches



Start with the boundary: sandbox routine work, approve exceptions, and scope secrets.

Every class ends with a short graded quiz

Starting next class (Thursday) — not today

- Three Canvas questions in the last minutes of every normal class, covering that day's two lessons. You get six minutes from the moment you open it.
- The quiz access code is projected in the room at quiz time — you take it in class.
- End-of-class quizzes are 12% of the course grade; your four lowest quiz days are dropped.
- Nothing to prepare: if you were present and engaged, you have what the quiz asks for.

Bring a laptop or phone from Thursday on — the quiz is taken in class.

Most classes also open discussion with a quick ungraded poll

UT Instapoll in Review mode — it never affects your grade

- When a poll slide is projected, open UT Instapoll from the course sidebar in Canvas — phone or laptop, it opens in a new tab.
- The matching poll goes live when I send it from my phone; answer on your device while the question stays on the screen.
- Commit to an answer before we discuss it — the poll exists to surface the room's thinking, not to test you.
- Polls are ungraded and anonymous to the room; the next slide shows the format, as an example only.

Canvas → UT Instapoll in the sidebar; answer on your device when the poll opens.

Instapoll — example only (not run today)

Instapoll · Review · Aug 25 A

Your agent works in one development environment but not another. Which fact would you compare first?

- A. The shell, executable path, and tool versions each environment resolves
- B. The operating system and release version each machine is running
- C. The model and agent version each of the two sessions used
- D. The editor or terminal application that launched the agent in each place

The agent proposes; you own acceptance

Production can be delegated; accountability cannot

Your Responsibility
Define observable success and non-goals
Choose behavior and tradeoffs callers depend on
Set scope, permissions, checkpoints, and rollback
Accept risk only when evidence is sufficient

Agent's Responsibility
Discover the repo and surface assumptions
Propose options and trace their consequences
Implement bounded changes and run checks
Produce rerunnable artifacts and unresolved risks

Shared: Keep uncertainty explicit; shrink, reset, or escalate when evidence stops improving.

Key Takeaways

1. Agentic coding shifts work from typing toward framing, steering, and acceptance.
2. Work one inspectable turn at a time: bound it, read the evidence, then decide.
3. Treat security as blast-radius control: sandbox routine work and gate escape hatches.

Compare your experience with coding agents

10 min

- Form a pair with the person on your right. At a row end, join the pair on your left.
- 0:00–4:00 — Each person names one coding agent they have tried — or says “none yet.”
- Discuss one success, concern, or barrier; choose one observation to share.
- 4:00–10:00 — Groups share observations; listen for experiences the room has in common.