The 10 Most Common Agentic Coding Mistakes

(and how to fix them)

Slides at scottsauber.com

scottsauber

Audience

- Anyone doing building with AI
- Whether AI writes 1% of your code, or 100%
- Doesn’t matter if using Claude Code, Copilot, or others (I use Claude primarily)
- If you’re at 1% - some of this may be overwhelming fair warning

Poll

- How many people using Claude Code?
- Copilot?
- Codex?
- Cursor?
- Devin/Windsurf?
- Something else?

Agenda

- Auto Mode
- Agent Verification
- Context Management
- Right-sizing Instructions
- Right-sizing work
- Don’t Repeat Past Mistakes
- Probabilistic enforcement vs deterministic enforcement
- The Right Guardrails
- 🌶️ Avoid reading every line of every PR 🌶️
- Live demo

Goals

- Things you should **start** doing on Monday
- Things you should **stop** doing on Monday

Who am I?

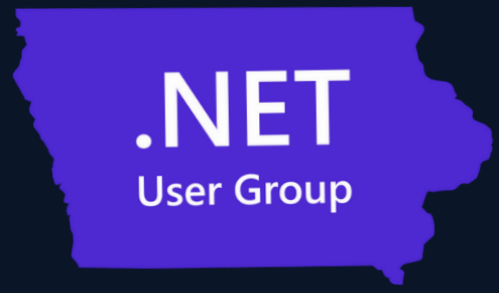
- Director of Engineering at Lean TECHniques
- Microsoft MVP
- Dometrain Author
- Redgate Community Ambassador
- Co-organizer of Iowa .NET User Group and Iowa Code Camp

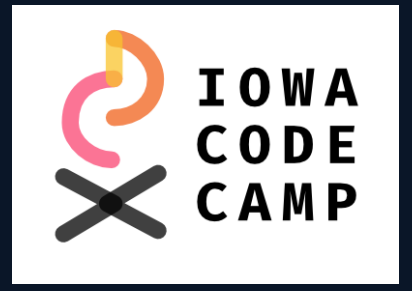












Why this talk?



What am I doing now?

- Agents write 99.9% of my code
- I’m easily 10x faster
- Quality is also up
  - “Before you guys came, I found 3-4 things every single story, now I might nitpick one out of every 10 stories” – QA at a client
- Enforce quality in pipeline – tests, cyclomatic complexity, etc
- Moving into Software Factories
- If you’re on a $50 Copilot budget – you can’t do this for very long
- I train internally and externally on AI – both Copilot and Claude

Auto Mode



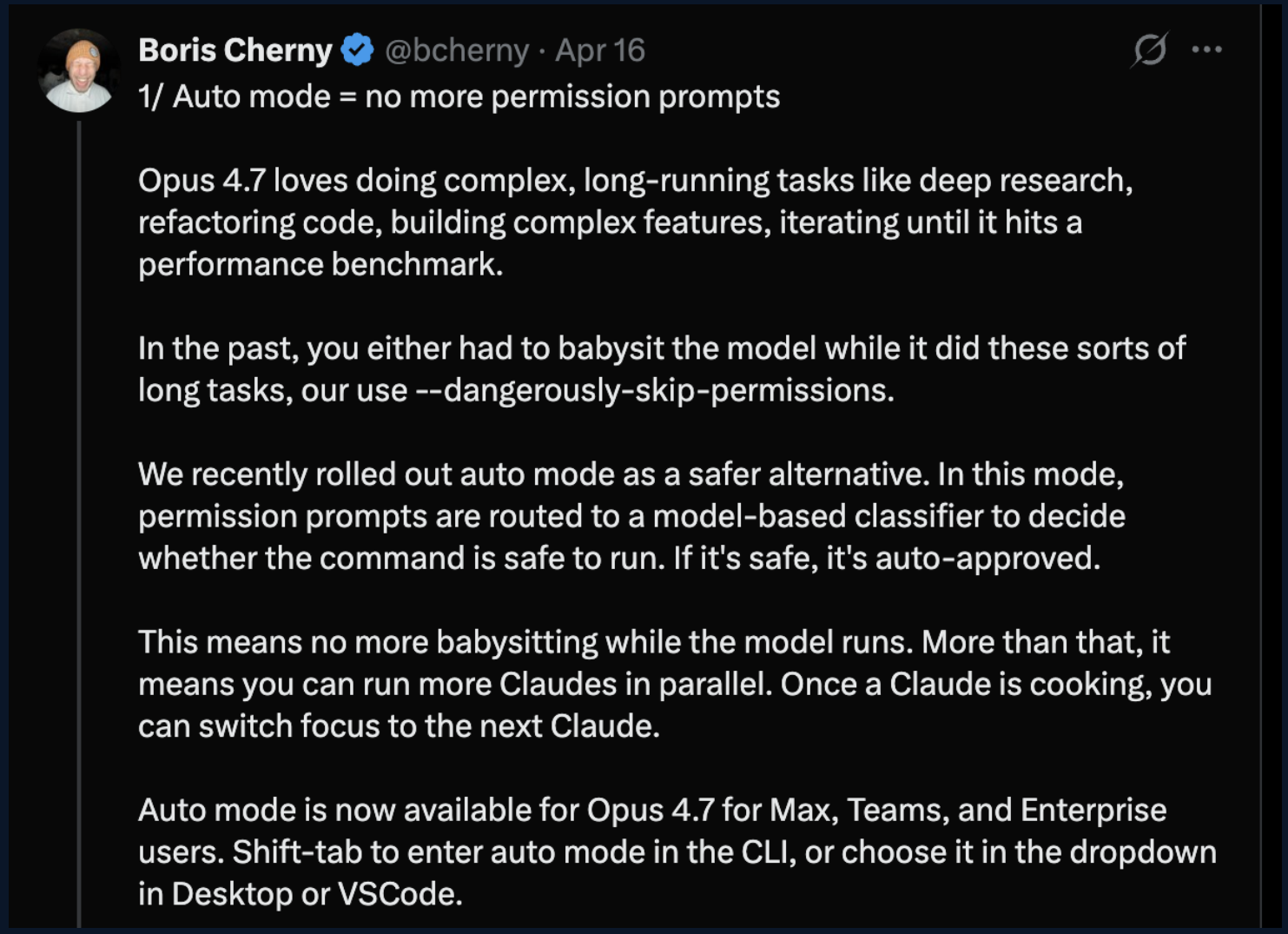
Problem: Continuously Allowing

Edit file
src/app/public/views/home/home.component.html

```
1 -<div
2 -  class="h-screen bg-cover relative"
3 -  style="background-image:url(assets/images/background-splash.png)"
4 ->
1 +<div class="h-screen bg-pink-500 relative">
2    <div class="absolute top-0 left-0 right-0 z-mg">
3      <app-attendees-banner></app-attendees-banner>
4    </div>
```

Do you want to make this edit to **home.component.html**?
❯ 1. Yes
  2. Yes, and switch to **accept edits (auto-approve file edits and common file commands)** for this session **(shift+tab)**
  3. No

Solution: Auto Mode



Boris Cherny @bcherny · Apr 16

1/ Auto mode = no more permission prompts

Opus 4.7 loves doing complex, long-running tasks like deep research, refactoring code, building complex features, iterating until it hits a performance benchmark.

In the past, you either had to babysit the model while it did these sorts of long tasks, our use --dangerously-skip-permissions.

We recently rolled out auto mode as a safer alternative. In this mode, permission prompts are routed to a model-based classifier to decide whether the command is safe to run. If it's safe, it's auto-approved.

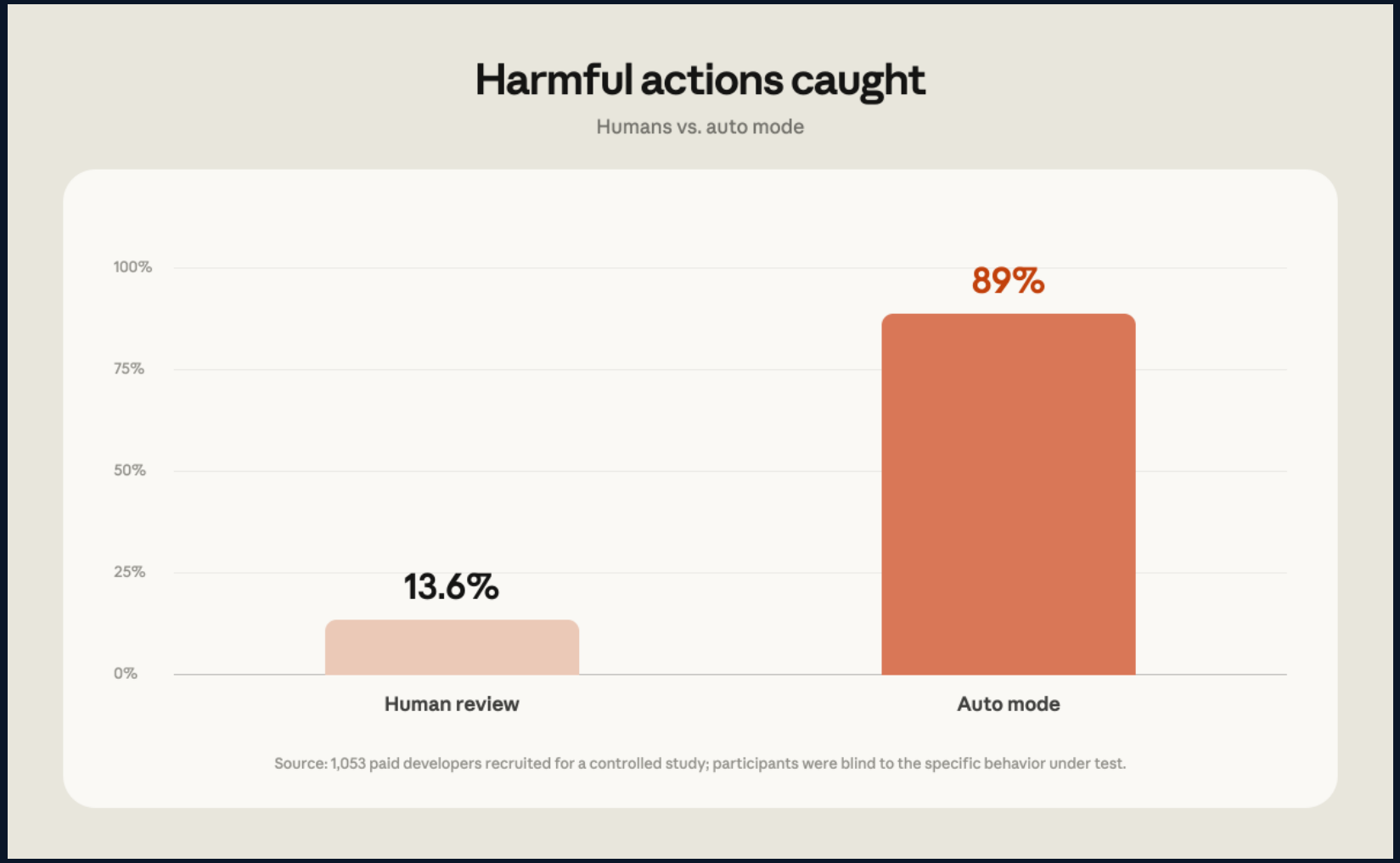
This means no more babysitting while the model runs. More than that, it means you can run more Claudes in parallel. Once a Claude is cooking, you can switch focus to the next Claude.

Auto mode is now available for Opus 4.7 for Max, Teams, and Enterprise users. Shift-tab to enter auto mode in the CLI, or choose it in the dropdown in Desktop or VSCode.

How Auto Mode works

- Auto mode evaluates is what it’s trying to do safe
- Uses another fast, capable model (Sonnet by default)
- Commit? Push a branch? Dropping a database? It’ll ask first
- But once you say yes in a session, it may not ask you again
- Copilot it’s called Autopilot but doesn’t use a model to check first

Auto Mode is safer than you



Consider: Your Environment

- Do you have access to Production cloud? DB?
- Consider Sandboxing

Agent Verification



Problem: Agents will just do things

- Agents will make assumptions if you let it
- Agents don’t know when they’re done unless you tell it
- Btw – humans didn’t either – they need Acceptance Criteria



Boris Cherny 
@bcherny

13/ A final tip: probably the most important thing to get great results out of Claude Code -- give Claude a way to verify its work. If Claude has that feedback loop, it will 2-3x the quality of the final result.

Claude tests every single change I land to claude.ai/code using the Claude Chrome extension. It opens a browser, tests the UI, and iterates until the code works and the UX feels good.

Verification looks different for each domain. It might be as simple as running a bash command, or running a test suite, or testing the app in a browser or phone simulator. Make sure to invest in making this rock-solid.

code.claude.com/docs/en/chrome

Solution: Give Agents a way to verify

- Automated Tests
- Browser access
- Claude / Copilot will do this by default now

Context Management

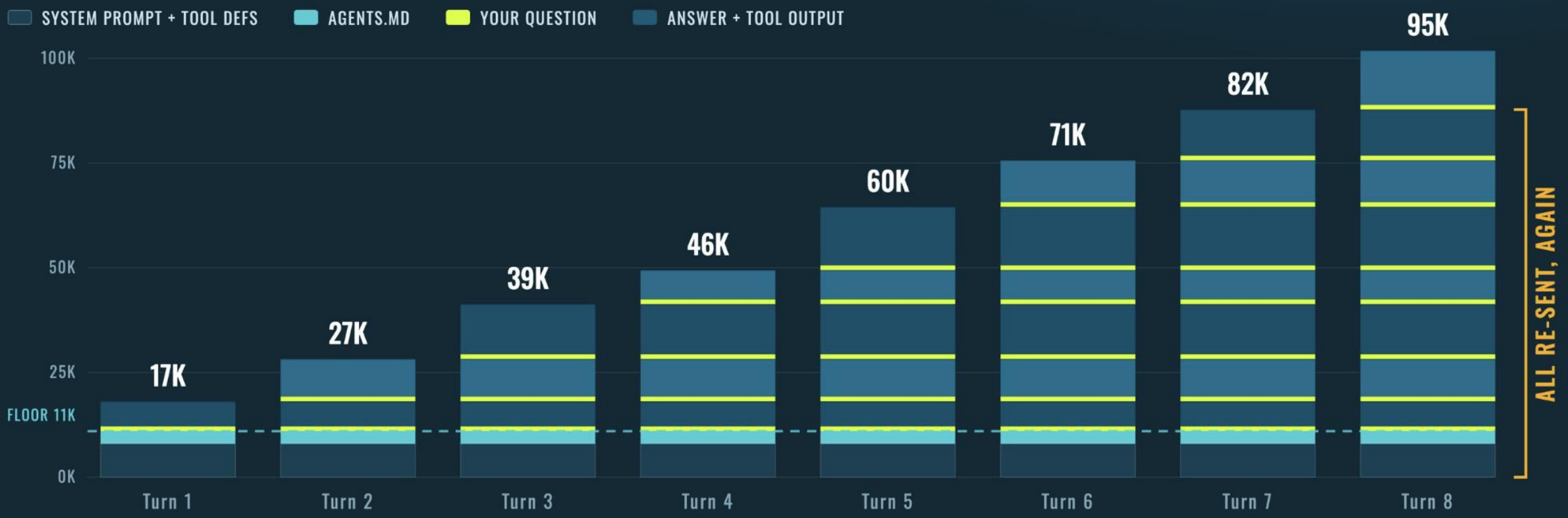


What are tokens?

- Roughly 1 token = 4 characters (depends on the model)
- Models have “context windows” – Opus can have 1M tokens
- Every message you send, sends all your old messages and answers, your AGENTS.md/CLAUDE.md/Copilot Instructions/etc

EVERY TURN RE-SENDS EVERY TURN BEFORE IT

The conversation is not a stream — it's a stack. Each prompt re-reads the whole thing from the top.



95K

FINAL CONTEXT SIZE

What **/statusline** shows you after 8 turns. This is the number people watch.

354K

INPUT TOKENS ACTUALLY BILLED

3.7× the final size. Nobody watches this number — it's the one you pay for.

8×

TIMES YOUR AGENTS.MD WAS SENT

Same for every tool schema. Caching is what makes this survivable — **until you break it.**

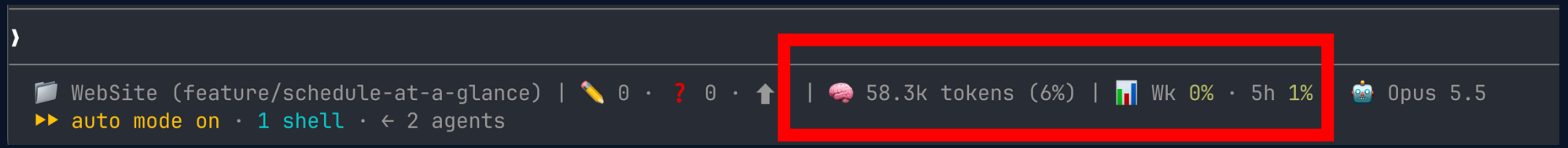
Problem: Agents need Context

- Agents need context
- Humans do too
- But if you give agents too much to do, you'll get worse results
- Humans too
- If you're autocompacting, you're doing it wrong

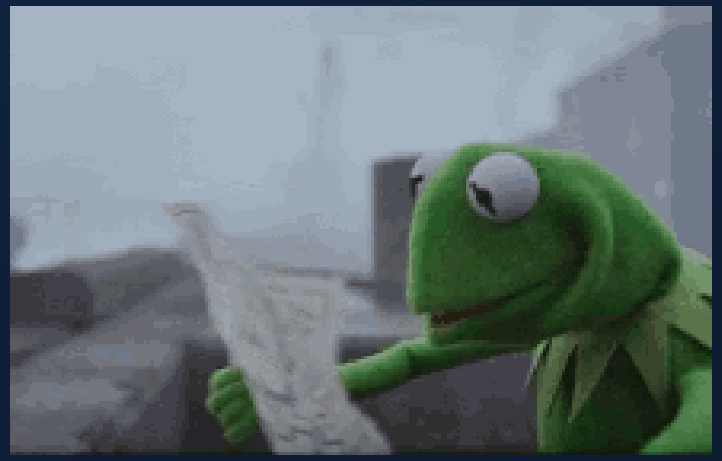
Solution: Manage your Context

- Once you get >250K tokens – you’re entering The Dumb Zone
- I /clear or new session all the time
- Change for a feature already shipped? New session, don’t resume old

Make it front and center



Right Sizing Instructions



Problem: You have instructions

- Instructions get sent on every message
- AGENTS.md / CLAUDE.md

Solution: Audit your instructions

- AGENTS.md, CLAUDE.md, copilot-instructions, .cursor/rules
- Use AGENTS.md
- 6+ months ago you needed lots, now you don’t, models are so good
- 80% chance your instructions are out of date
- Your Instructions should be a lot of “Don’ts” based on experience
- “Instructions are a last resort” – Cory House, 2 hours ago



Boris Cherny 
@bcherny

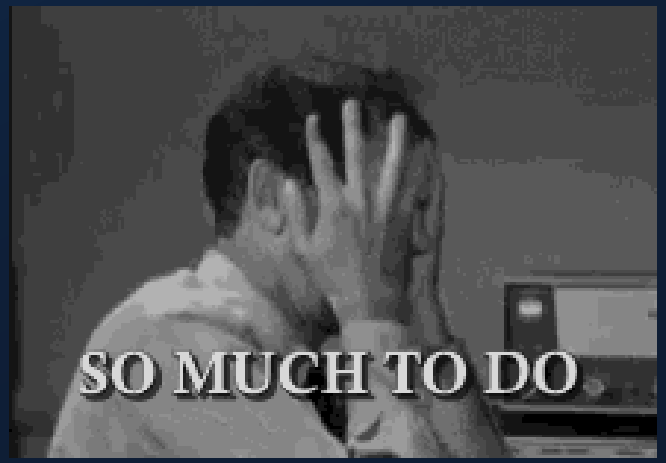
Just whittle it down to the bare minimum claude needs to know. A good way to do that: delete your CLAUDE.md, then add it back an instruction at a time as you see Claude making mistakes. Do that every few months



From code.claude.com

3:49 PM · Jan 3, 2026 · **3,829** Views

Right Sizing Work



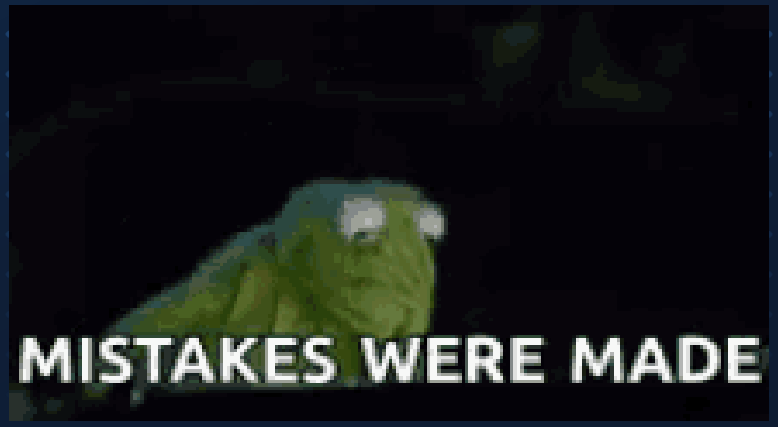
Problem: AI getting confused

- Most likely in the Dumb Zone, stay in the Smart Zone (<250K tokens)

Solution: Break work down

- Break work down
- If you have 3+ acceptance criteria, rip it out
- If your story says “And”, rip it out

Don’t repeat past mistakes



Getting bad results with AI? Fix it

- Getting bad results with AI? Make it never happen again.
- Find deterministic solutions (ie linters, formatters, etc)
- Change your model to a better one
- Only use instructions as a last resort
- /insights (Claude) or /chronicle (Copilot)

Claude Code Insights

142 messages across 18 sessions (21 total) | 2026-08-02 to 2026-11-07

At a Glance

What's working: You hold Claude to a real definition of done: features ship on their own branches with tests, ADRs, docs and verification scripts, and your Log Analytics auto-PR demo added simplify, security and review gates so work arrives already reviewed. You also use Claude as a coordinator for work that spans many places, such as rolling doc updates and merges across a set of interview repos, or keeping it in the loop while you reshaped the Iowa Code Camp schedule live in Sessionize. A steady flow of conceptual questions about agent security, auto-mode and Azure PIM shows you're thinking carefully about how far to trust agents. Impressive Things You Did →

What's hindering you: On Claude's side, it sometimes answered a broader or different question than the one you asked, such as a full model comparison when you wanted a tier mapping, or a macOS permission when you meant a Claude Code prompt. It also answered from memory before searching, and its first passes had small bugs like broken list numbering, mislabeled image files or moving a session you needed pinned. On your side, sessions often stalled on expired logins, the wrong GitHub account, branch rulesets or auto-mode refusing risky actions, so a quick auth and permissions check before multi-repo or deploy work would keep things moving. Where Things Go Wrong →

Quick wins to try: Turn your recurring patterns into Custom Skills: a /sourced-answer skill that always searches the web and cites its sources, and a /multi-repo-rollout skill that checks auth and rulesets before making any change. Add Hooks that run markdown lint and your verify script after edits, so problems like '1,1,1,1' numbering never reach a commit. Headless mode is a natural fit for your Jidoka and CI work, since Claude can run as a step in a pipeline rather than only in a chat. Features to Try →

Ambitious workflows: Your Log Analytics demo is the seed of an unattended loop: an agent reads new exceptions, writes a failing test that reproduces each one, iterates until your gates pass, and opens a reviewed PR, with Jidoka's 'stop the line' rule deciding when to escalate. Multi-repo sweeps can fan out into parallel subagents, one per worktree, each checking rulesets and rendering before it opens a PR. Your schedule reviews could become a standing agent that compares Sessionize with the published site and checks pinned constraints from a file, so you no longer have to remember them. On the Horizon →

Where Things Go Wrong

Your friction mostly comes from Claude reading your request differently than you meant it, from auth and guardrail walls in GitHub and auto-mode, and from small first-pass bugs that took an extra round to fix.

Misread intent and answers from memory

Claude sometimes answered a broader or different question than the one you asked, or answered from memory when you wanted current, sourced facts. Saying the exact output you want up front ("just a mapping table," "look it up online," "a Claude Code prompt, not a macOS one") would save a round.

- You asked Claude to trigger a Claude Code permission prompt for a screenshot. It took that to mean the macOS Screen Recording permission, its screencapture attempts failed, and you took the screenshot yourself, so the session achieved nothing.
- You asked where the auto-mode safety check runs. Claude answered from memory with hedges until you told it to check the internet. Similarly, it gave a broad GPT vs. Claude comparison when you only wanted a simple tier mapping, and you had to clarify.

Probabilistic vs Deterministic Enforcement



Probabilistic

- Models are trained on data
- Models are predicting the next word based on its training data
- Models are probabilistic – same prompt two times, different results
- ❌ “Follow our format” “Follow my naming conventions” “Use repositories for data access”
- ✅ “Does this PR introduce security issues” /security-review
- Your CLAUDE.md should have a lot of Don’ts or “Follow the patterns in this part of the codebase, not that part”

Deterministic

- Same thing multiple times, same results
- Formatters
- Linters
- Code Coverage
- Cyclomatic Complexity
- CRAP

Probabilistic vs Deterministic

- Prefer Deterministic
- Default to Probabilistic as a last resort
- No models won’t comment out things anymore

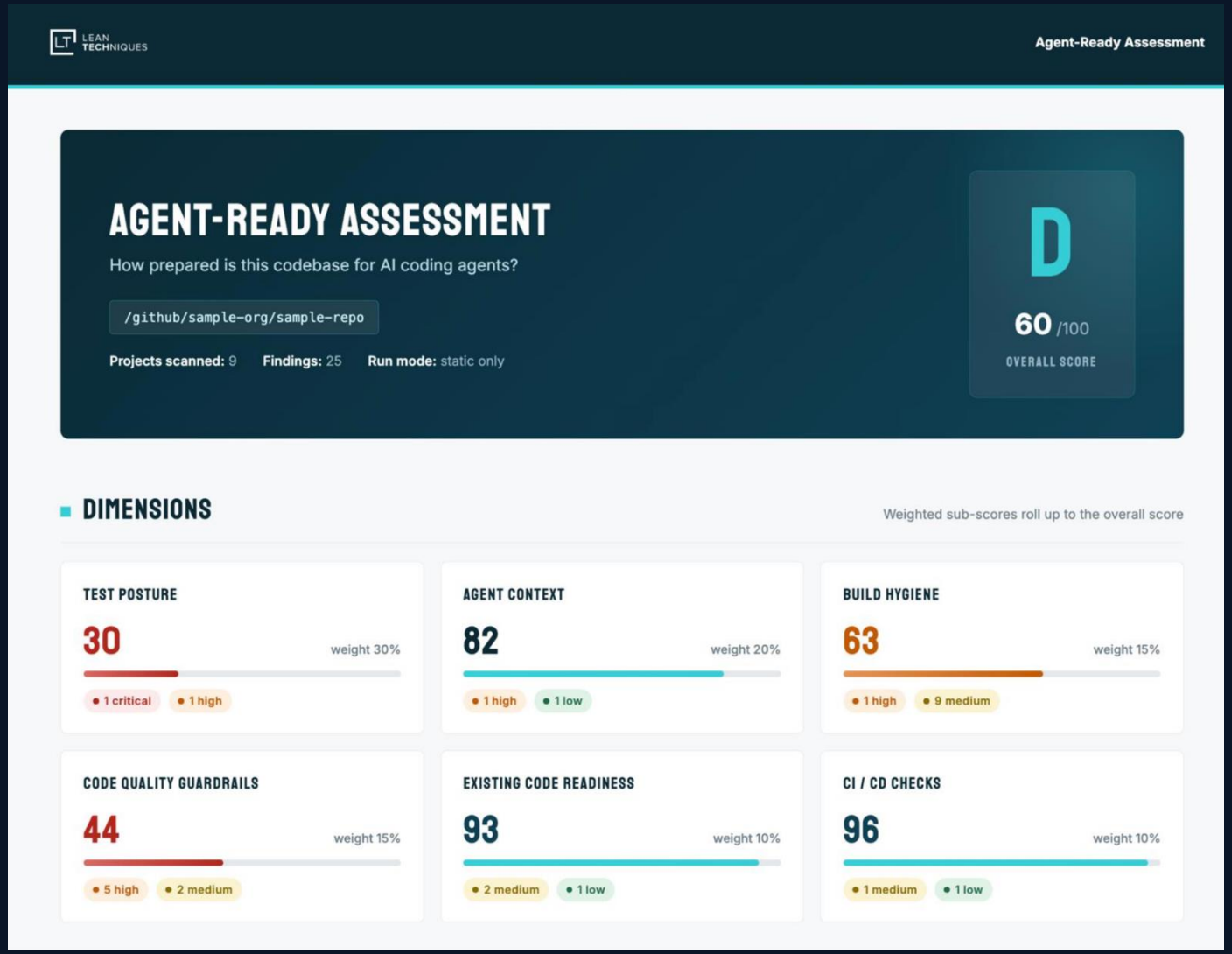
The Right Guardrails



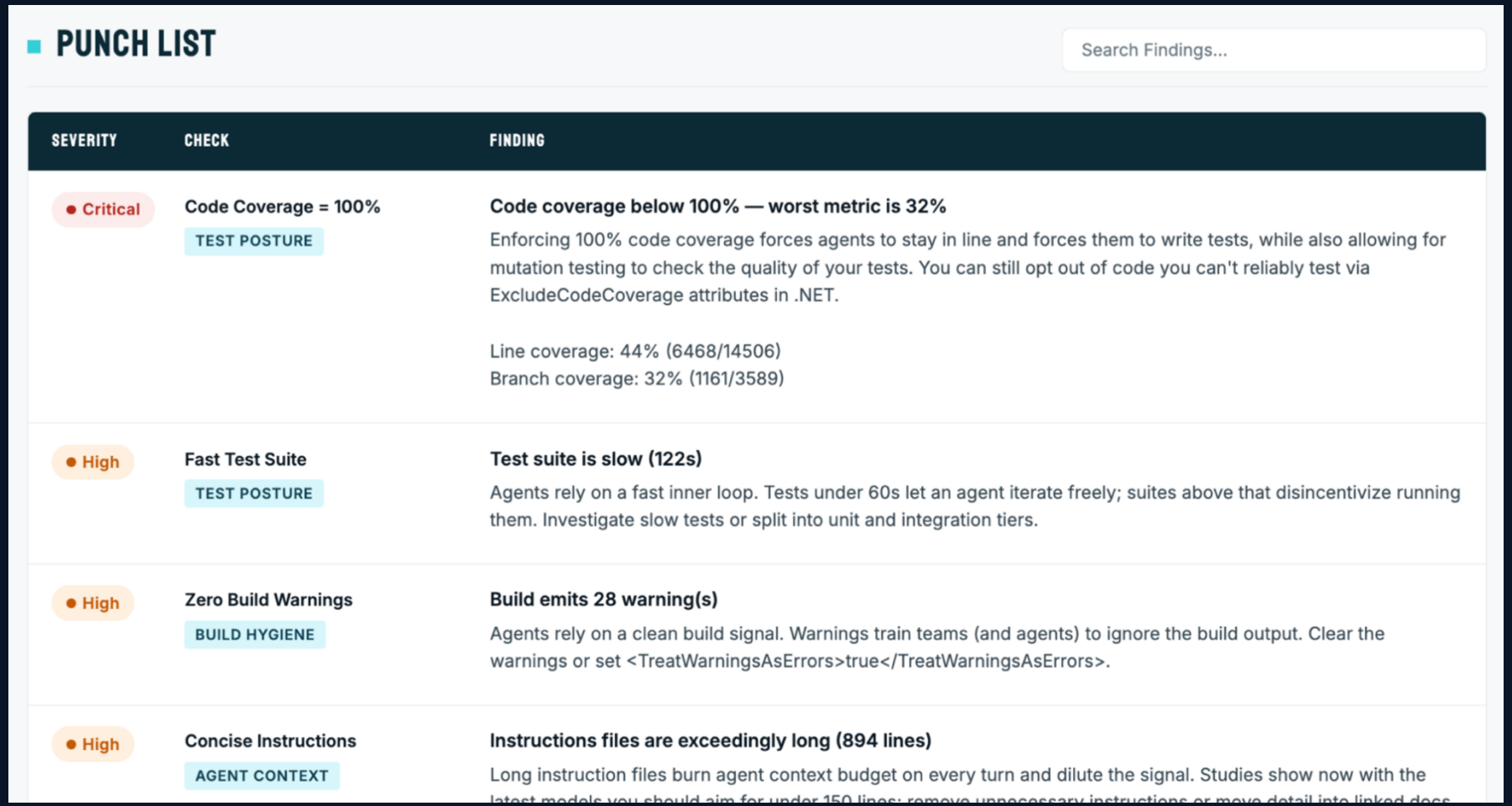
The Right Guardrails

- Formatters – dotnet format, prettier, oxfmt
- Linters – dotnet format, eslint, oxlint
- Code coverage – 100%
- Cyclomatic complexity - <10
- Treat Warnings As Errors
- CRAP score - $CRAP(m) = comp(m)^2 \times (1 - cov(m))^3 + comp(m)$
  - comp = cyclomatic complexity
  - cov = code coverage

Agent Ready Assessment



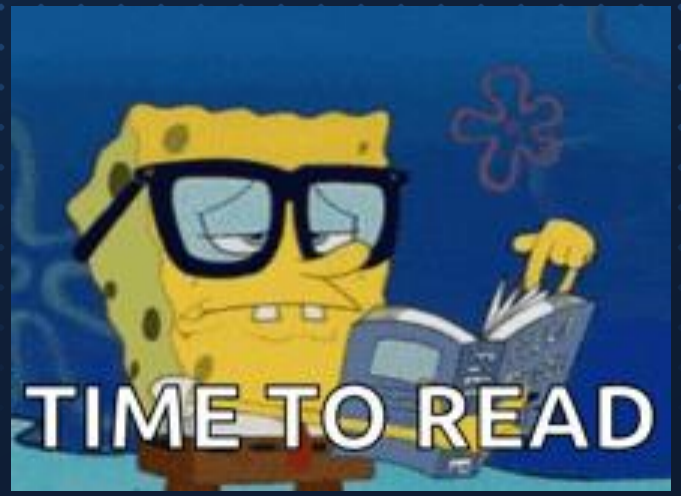
Agent Ready Assessment



Agent Ready Assessment

- Evals to prove a codebase with a score of 100 has 5x less bugs than a score of 50 = higher quality
- Evals to prove 100 score you can use Sonnet instead of Opus for the same quality = saves money

Avoid Reading Every Line of Every PR



The Problem: Writing Faster, More to Review

- Writing code 10x more code than before
- Bottleneck is now reviewing code

CHALLENGE

- “If you had to review every PR no matter the size in under 5 minutes, what would you do?”
- Most likely – enforce quality in the pipeline
- Most likely - increase signal to noise
- Good ideas – even before AI!

Solution: Stop reading every single line of code

- 
- You didn’t read every line of code with the same scrutiny before AI
- Button color change? LGTM!
- Changing the security model? Pulling branch, change things to verify, reviewing deeply, etc

Solution: Create a PR Classifier to Codify This

- AI reviews every PR as a first pass
- Classifier – use AI to classify a PR based on risk, impact, error detection, rollback cost
- Anthropic does a Classifier system of 1-3
- 1 means AI can approve (Copilot Reviews can approve PRs as of 9/1)
- 2 means a human needs to approve
- 3means 2 humans need to approve of a certain caliber
- You'll be surprise how few 3s you get
- Also: gh CLI now supports screenshots + gifs

Live Demo



Basic workflow

1. 👨 /grill-me on a GH issue, Jira issue, or just free text
2. 👨 Approve plan and go on auto/autopilot mode
3. 🤖 Create a new worktree
4. 🤖 Implement the plan
5. 🤖 Run the verify script til it passes (formatting, linting, tests, etc)

Intermediate Workflow /ship-it abc-123

1. 🧑 /grill-me on a GH issue, Jira issue, or just free text
2. 🧑 Approve plan and go on auto/autopilot mode
3. 🤖 Create a new worktree
4. 🤖 Implement the plan
5. 🤖 Run the verify script til it passes (formatting, linting, tests, etc)
6. 🤖 **Run /code-review and /simplify fix anything you have a lot of confidence in. If it's a code style, back it up with reduced cyclomatic complexity or another metric that proves the code is better**
7. ❓🧑 **Present anything you're unsure about for approval (rare maybe 2% of the time)**
8. 🤖 **Run /security-review and fix anything you have a lot of confidence in. If you're unsure then write a test to prove there's an issue**
9. ❓🧑 **Present anything you're unsure about for approval (rare maybe 1% of the time)**
10. 🤖 **Run the verify script til it passes (formatting, linting, tests, etc)**
11. 🤖 **Smoke test the app to make sure it works**
12. 🤖 **Open a Draft PR in my particular format using the /concise-pr skill**
13. 🤖 **Watch CI and fix any build errors. If a test is flaky, root cause and fix it**
14. 🤖 **Watch for the Copilot Code Review and fix any comments**
15. ❓🧑 **Present any comments you're unsure about for my review (uncommon, maybe 25% of the time).**
16. 🤖 **Notify me when PR is green with no comments and ready for me to review**
17. 🤖 **Mark PR ready for review (move out of Draft)**
18. 🧑 **Approve or request changes**

Advanced Lights On Factory

1. 🤖 **Feed product ideas from existing backlog, errors, feedback from stakeholder meetings, telemetry**
2. 😎 **Approves ideas**
3. 🤖 **Creates Plan**
4. 😎 **Approve plan**
5. 🤖 Create a new worktree
6. 🤖 Implement the plan
7. 🤖 Run the verify script til it passes (formatting, linting, tests, etc)
8. 🤖 Run /code-review and /simplify fix anything you have a lot of confidence in. If it's a code style, back it up with reduced cyclomatic complexity or another metric that proves the code is better
9. ❓🧑 Present anything you're unsure about for approval (rare maybe 2% of the time)
10. 🤖 Run /security-review and fix anything you have a lot of confidence in. If you're unsure then write a test to prove there's an issue
11. ❓🧑 Present anything you're unsure about for approval (rare maybe 1% of the time)
12. 🤖 Run the verify script til it passes (formatting, linting, tests, etc)
13. 🤖 Open a Draft PR in my particular format using the /concise-pr skill
14. 🤖 Watch CI and fix any build errors. If a test is flaky, root cause and fix it
15. 🤖 Watch for the Copilot Code Review and fix any comments
16. ❓🧑 Present any comments you're unsure about for my review (uncommon, maybe 25% of the time).
17. 🤖 Notify me when PR is green with no comments and ready for me to review
18. 🤖 Mark PR ready for review (move out of Draft)
19. 😎 Approve or request changes
20. 🤖 **Attaches evidence to PR of satisfying AC**
21. 🤖 **Metrics of how long each stop took fed into self-improvement loop**
22. 🤖 **Kaizen self-improvement of proposing new fixes to the factory**
23. 😎 **Approve any changes to the factory**

Takeaways

- Hopefully you got at least one idea today you can take back to work
- This slide deck is your resource – up at scottsauber.com

Questions?

ssauber@leantechniques.com
@scottsauber on Twitter
LinkedIn 👉

Add me on LinkedIn:



Select
Services Partner
Claude Partner Network

GitHub Verified Partner

Slides at scottsauber.com

scottsauber

Thanks!

Slides at scottsauber.com

scottsauber