07: Context engineering and the buzzwords

Prompt → context → harness → loop → graph

Why this matters

- In agentic work, your typed prompt is a small part of the input.
- What fills the window decides the result, and the bill.
- A new "___ engineering" term arrives every few months.
- The same questions matter after the names change.

Part 1

The buzzword ladder

The ladder: each rung widens what you design

Rung	Took off	The question it asks	Substance or relabel?
Prompt engineering	2020–2023	What words do I send?	Substance; now one piece of a whole
Context engineering	Jun 2025	What's in the window at each step?	Mostly substance
Harness engineering	Feb 2026	What does the environment prevent, check, correct?	Substance; developer-experience and CI work underneath
Loop engineering	Jun 2026	What prompts the agent, and when does it stop?	Partly; also cron plus a while-loop
Graph engineering	Jul 2026	How do many agents and steps connect?	Mostly relabel

Prompting basics still matter

```
fix the login bug
```

```
Users whose email contains "+" can't log in (issue #482).
First reproduce it with a failing test in tests/auth/test_login.py.
Then fix the root cause. Don't change the public API of
auth/session.py. Done means: the new test passes, `make test`
passes, and you explain the root cause in two sentences.
If you can't reproduce it, stop and tell me.
```

The second has an outcome, a reason, a definition of done, and a check.

Context engineering vs harness engineering

	Context engineering	Harness engineering
Named by	Tobi Lütke, Jun 19, 2025	Mitchell Hashimoto, Feb 5, 2026
Asks	What should the agent see?	What should the environment prevent, measure, correct?
In their words	"the art of providing all the context for the task to be plausibly solvable by the LLM"	"anytime you find an agent makes a mistake, you take the time to engineer a solution"
Looks like	Choosing instructions, tools, files, history, memory	Docs, linters, tests, hooks, helper scripts

Loop engineering replaces typing "continue"

```
while :; do cat PROMPT.md | claude-code ; done
```

Geoffrey Huntley's "Ralph" loop (July 2025): a fresh context each pass, a plan file as memory.

> "You shouldn't be prompting coding agents anymore. You should be designing loops that prompt your agents."
> Peter Steinberger, quoted by Addy Osmani, June 2026

Osmani warns that "done" from a loop is "a claim and not a proof."

A loop worth building has a checkable end state

```
read state → do one item → verify: tests, linter, another model
   fail → do the item again
   pass → update state, commit → goal or budget reached?
             no  → read the state again
             yes → stop and report to a human
```

- An independent check, because a model grading its own work is optimistic.
- A budget of turns, wall-clock time, or tokens.
- A human reads the diff. "Done" is a claim.

Graph engineering is an old idea with a new name

July 18, 2026, Steinberger on X: "Are we still talking loops or did we shift to graphs yet?"

The shapes, as Anthropic described them in December 2024:

Pattern	Coding example
Prompt chaining	Plan → implement → write tests → summarize
Routing	Triage issues to bug, docs, or feature handlers
Parallelization	Three reviewers on one pull request
Orchestrator-workers	A lead splits a migration across worktrees
Evaluator-optimizer	Generate, critique, revise until tests pass

Substance or relabel?

- You're designing systems around a component whose output varies.
- "X is dead" is always wrong. Graphs contain loops.
- Measured claims have held up. Viral ones mostly haven't.
- New names often mark new product features.
- Expect another name within months for these same questions.

Part 2

What the model actually sees

Your prompt is often the smallest part

Component	Typical size	Loaded
Harness system prompt	Thousands of tokens	Every call
Tool definitions, built-in and MCP	A few thousand to 50K+ tokens	Every call, unless loaded lazily
Instruction files	Tens to thousands of lines	Session start, or per directory
Skill descriptions	One line each	Every call; the body only when used
Your messages	Small	As you type
Tool results: files, tests, searches	Often the largest share	As the agent works

Check `/context` in Claude Code, or your harness's equivalent.

Bigger windows, same attention

Study	Finding
Lost in the Middle, 2023	Best when key facts sit at the start or end; worst in the middle
RULER, 2024	Of 17 models claiming 32K+ contexts, only half held up at 32K
NoLiMa, 2025	Without shared keywords, 11 of 13 models fell below half their short-context score at 32K
Context Rot (Chroma), Jul 2025	Across 18 models, scores fell as input grew, even on simple tasks

By 2026 the limit is attention and cost, more than capacity.

Write, select, compress, isolate

Move	Means	Examples
Write	Save it outside the window	Plans and progress files; issues; memory; commits
Select	Bring in only what's relevant	`grep` on demand; skills; path-scoped rules; tool search
Compress	Keep only the tokens you need	Compaction; trimmed tool output; `jq` and `head`
Isolate	Separate work, separate windows	Subagents; a session per task; a fresh reviewer

The two common mistakes are preloading "just in case" and running one marathon session.

Part 3

The rise and fall of giant CLAUDE.md files

One file, loaded every time (2023–2025)

When	What happened
2023–2024	`.cursorrules`: one rules file added to every Cursor request
Meanwhile	Every tool wants its own file; teams keep near-duplicates
Feb 2025	`CLAUDE.md` loads into every Claude Code session; `/init` writes one
Apr 2025	Official advice: add "IMPORTANT" or "YOU MUST" to improve adherence
Aug 2025	`AGENTS.md`: one vendor-neutral name; the nearest file wins

The thousand-line file (2025)

```
# CLAUDE.md: READ THIS ENTIRE FILE BEFORE DOING ANYTHING!!!

You are a world-class principal engineer with 30 years of experience

## CRITICAL RULES (NEVER BREAK THESE!!!)
2. NEVER introduce bugs.
31. IMPORTANT: Use 2-space indentation.  <- the formatter does this
32. Use 4-space indentation in all files. <- contradicts rule 31

## Tech stack
- React 17                                  <- the repo is on React 19
```

Each line felt free when added. Nobody deleted anything.

The fall (late 2025–2026)

- Every line costs attention, on every task.
- More instructions, worse following; the best models hit 68% at 500.
- ETH Zurich (2026) found no significant gain in success, costs up about 20%.
- Skills and path-scoped rules load only when needed.
- Hooks replaced shouting in capital letters.
- Vendors set budgets, like "target under 200 lines per CLAUDE.md file."

Ask one question of every line

"Could the agent figure this out by reading the repository?" If yes, delete it.

Kind of instruction	Where it belongs
Must always happen	A hook
Must never happen	A deny rule, hook, or sandbox
Style a tool can enforce	Formatter and linter config
A procedure needed sometimes	A skill
Applies to one part of the repo	A path-scoped rule or nested `AGENTS.md`
Your personal preference	Your user-level file, or a gitignored local one

Keep commands, traps in the code, and unwritten decisions, well under 200 lines.

Bryan's take

> **Bryan's take:** Big `CLAUDE.md` and `AGENTS.md` files were popular, and now they're not.

Bryan's take

> **Bryan's take:** I like loop engineering and setting goals, but the goals can't be too ambitious or the agent goes off the rails. Loops with a hard, checkable goal and a human in the loop reading the result work for me. Open-ended loops that run unattended and ship to production don't, yet. (My broader skepticism about fully autonomous workflows is in module 08.)

Bryan's take

> **Bryan's take:** Context management is important. Once context gets stale, that's bad. Don't keep the same thread around for a long time. Models get dumber the larger the context gets and the longer a session has been going. Start fresh sessions often.

Try it

Individual (60–90 minutes)

- Take a repo's `CLAUDE.md` or `AGENTS.md`; count its lines and tokens.
- Sort every line into the "where it belongs" table, then trim.
- Run three tasks with each version in fresh sessions; compare.

Class or team (one session)

- Break 10 tests in a toy project.
- Groups build a Ralph loop, a `/goal`, and a Stop hook.
- Compare: did it finish, at what cost, and did any weaken a test?

Discussion

1. Which rung adds the most substance? Which is mostly a new name? Show evidence.
2. Hand-written context files helped a little; generated ones didn't; both cost more. Team policy?
3. A loop says "done." What would let you merge its pull request unread?
4. As models improve, which parts of context engineering disappear, and which remain?

Read more

- Module: 07 From prompt engineering to context, loop, and graph engineering
- Effective context engineering for AI agents (Anthropic, September 2025)
- Evaluating AGENTS.md (Gloaguen et al., 2026)
- Context Rot (Chroma, July 2025)