

What changes a *coding agent's* behaviour

Isaac Correa

Models, written rules and runtime guards over
eight months of daily use of Claude Code



Abstract

Companies govern coding agents with written instructions and, increasingly, with hooks that block an action when it breaks a rule. This study measures what changed the behaviour of Claude Code over eight months of daily use by one operator with no training in programming who directed many sessions at once, using 46,464 operator messages, 1.3 million agent messages and 3,411 recorded blocks. A fixed instrument applied the rules of one guard to every surviving answer, including those written before the guard existed. In the week Claude Opus 5.5 was released, it broke the operator's writing rules in 18% of first drafts and Claude Opus 5 in 57%, a gap that held within the same long-running sessions. For one language rule, a written instruction was followed by a fall that could not be distinguished from chance, and a reinforced rule with a guard by a fall to a sixth of the starting rate. Blocks roughly halved the violations that reached the operator, with no detectable effect on later drafts in the same session. Supervision accounted for a third to a half of the operator's messages throughout, while its content shifted from corrections to nudges to keep working. Agents built into the operator's own applications, each with a defined role and sharing its data with the application's interface, received more new work and less supervision than development sessions, a difference within chance.

Keywords: coding agents; AI agent governance; runtime guardrails; instruction following; human oversight; agentic software engineering.

How to cite: Correa, I. (2026). What changes a coding agent's behaviour: models, written rules and runtime guards over eight months of daily use of Claude Code. Working paper, Zenodo. <https://doi.org/10.5281/zenodo.23510601>

Contents

Findings in brief	03
The record	04
The author's position	05
Sources and cleaning	06
The fixed instrument	07
Prior evidence	08
The operator's workload	09
Corrections and nudges	10
Parallel sessions	11
Attention per session	12
The model release	13
Same days, same rules, two models	14
Checks for selection bias	15
The weeks around the release	16
The older model before the release	17
Written rules and guards	18
Inventory of guards	19
One rule in three regimes	20
Filtering by the writing guard	21
Learning within a session	22
The guard and first drafts	23
Repeated instructions	24
Agents inside applications	25
Supervision of application agents	26
Implications for teams	27
Threats to validity	28
Replication material	29
References	30

Findings in brief

Three levers shape a coding agent's behaviour. They are the model Anthropic ships, the rules the operator writes into the agent's instructions, and guards, which are hooks that inspect an action or an answer and block it if it breaks a rule. This study separates their effects over eight months of daily work with Claude Code by a single operator.

Between February and October 2026 the operator sent 46,464 messages to Claude Code sessions, at times with nine or more running at once, and built up a 4,927-word instruction file, 1,701 memory notes and 69 hook scripts.

-
- 01** The model release changed behaviour as much as the strongest combination of rule and guard, with no change to the rules. Under the same writing rules, 18% of the newer model's first drafts in main sessions broke a rule, against 57% of the older model's, and the gap held in the three main sessions that used both models.

 - 02** For the language rule, the share of first drafts in the wrong language fell from 16% with no rule to 10% under a written rule, a fall not statistically distinguishable from no change, and to 3% once the rule had been reinforced and backed by a guard.

 - 03** The writing guard cut by about half the proportion of final answers containing a violation. Of the violations it read and failed to block, nine in ten lay in the turn's final message, which was not yet on disk when the guard ran. A block did not detectably reduce violations in the drafts that came after it.

 - 04** From February to October, corrections fell from about a third of the operator's messages to under a tenth, while nudges to keep working rose from one in twenty to one in five.

 - 05** Running more sessions at once had no detectable effect on the share of the operator's messages devoted to supervision. The operator's message rate levelled off at about 21 an hour from seven sessions upwards.

 - 06** In three applications the operator built around an agent with its own role, sharing its data with the screens, the operator's messages leaned more to new work and less to supervision than in development sessions, a difference not statistically significant.
-

The record

Most evidence on coding agents comes from standard test suites, controlled trials or pull requests on public repositories. None of these sources shows the conversation in which a person instructs an agent, corrects it and decides which instructions to codify as rules. This record does, and the study uses it to answer five questions.

- Q1** How much did a model release change compliance with the operator's rules?
- Q2** What did a guard add to a rule that was already written down?
- Q3** Did a block change the agent's later drafts in the same session?
- Q4** How did the operator's supervision change over time and with the number of parallel sessions?
- Q5** Did agents built into the operator's applications need less supervision?

Operator's messages 46,464 February to October 2026	Agent's messages 1.30M July to October 2026	Guard blocks 3,411 August to October 2026
--	--	--

The author's position

The operator is the author, a co-founder of a company that builds AI agents for businesses, who never trained as a programmer and, as described in Correa (2026b), built most of the software used in daily work. The record is therefore also a test of a position set out in two earlier essays.

Correa (2026a) holds that an agent needs memory, an objective and principles to decide by, and that as agents take over execution, the person's work moves to intention: deciding what is worth doing. Correa (2026b) holds that an application's interface should become the complement of an agent that acts on the same data, with the person supervising and correcting it, and that the agent should keep each correction as a rule.

The record supports part of this position and qualifies the rest. Corrections gave way to nudges to keep working, consistent with a move from fixing the work to directing it (section 02). A written rule, the memory and principles of the first essay, produced no detectable change for the one rule followed under three regimes, which fell clearly only once the rule was reinforced and a guard enforced it, while a model release changed compliance as much as that combination (sections 03 and 04). The agents inside the operator's own tools were given more new work and less supervision, by a margin within chance (section 05).

Sources and cleaning

The operator's messages are taken from Claude Code's own input log and cover 6 February to 20 October 2026. Complete agent transcripts are available from 25 July, when Claude Code's retention period was extended; earlier transcripts had been deleted automatically. Hook scripts and the instruction file have been kept in a git repository since 17 August, although there were no commits between 18 August and 25 September.

Table 1 Cleaning the operator's log

Three in ten logged entries were not messages typed by the operator

Layer	Entries	Rule
Logged entries	66,522	Every line of the input log up to the cut-off
Commands	−9,259	Slash commands such as model or session switches
Pastes with no typed text	−7,504	Pasted text or images with nothing typed around them
Other agents	−3,271	Messages relayed from other agent sessions
Scheduled messages	−24	Long messages repeated word for word by a timer, and empty entries
Operator's messages	46,464	The base for every operator-side measure

Cut-off: 20 October 2026, 09:57 Madrid time. Scheduled messages are texts repeated five or more times; two of the 24 entries in that row are empty.

The fixed instrument

A guard's log begins only when the guard is installed, so it cannot show how the agent behaved beforehand. The study therefore applies the writing guard's rules unchanged to every answer in every surviving transcript, including those from the nine weeks before the guard existed.

Table 2 The rules the instrument applies

Six writing rules and one language rule, all checked mechanically

Rule	What breaks it
Dash	An em or en dash used as punctuation
Emoji	Any emoji or pictographic symbol
Inverted exclamation	An inverted exclamation mark (¡)
Filler opening	A reply opening with a Spanish filler word such as "done" or "okay"
Colloquial register	A colloquial phrase from a fixed list of Spanish expressions
Asking permission	A reply ending by asking permission for what the agent can decide
Language	A reply written mainly in English; added to the guard on 6 October

As in the guard itself, quotations, code and links are stripped before the check.

On a random sample of 3,000 answers, the instrument and the guard flagged exactly the same 618 answers; a copy with the dash rule deliberately removed disagreed with the guard on 475. At run time the guard sometimes received less text (section 04). The unit of analysis is the first draft: the text the agent wrote in a turn before it first stopped, and so before any guard had read it. A turn runs from one operator message to the next.

Prior evidence

Studies of agent-written pull requests capture the output, not the conversation that produced it. Li et al. (2026) assembled a large collection of them from GitHub, Khelifi et al. (2026) classified the interventions developers make in them, and He et al. (2026) measured the trade-off between speed and quality after teams adopted an agentic editor.

Instruction files are now studied directly. Chatlatanagulchai et al. (2026) found that they evolve like configuration code; Gloaguen et al. (2026) found that they did not improve task success rates on a standard test suite and increased inference cost; McMillan (2026) found that their size and position mattered less than the length of the session. Jaroslawicz et al. (2025) showed that instruction following declines as instructions accumulate, Zhang et al. (2026) found prohibitions more effective than guidance, and Laban et al. (2025) documented a loss of performance over multi-turn conversations.

On the human side, Dhanorkar et al. (2026) drew on interviews to describe oversight work, Chou et al. (2026) observed vibe-coding sessions, Becker et al. (2025) ran a randomised controlled trial of developer productivity, and Chi (2027) analysed a sixteen-month activity trace from a single developer. Immediato (2025) names this human work cognitive verification and ethical oversight and argues that its cost is missing from estimates of the gains from AI. None of these works measures how a model release, a written rule and a guard each change what the agent writes over months of daily use.

The operator's workload

Supervision comprises the operator's messages that do not assign new work: those that correct the agent, urge it to continue, check a claim or set a lasting rule. Its volume and composition show how much effort it takes a person to direct agents, regardless of the quality of the resulting code.

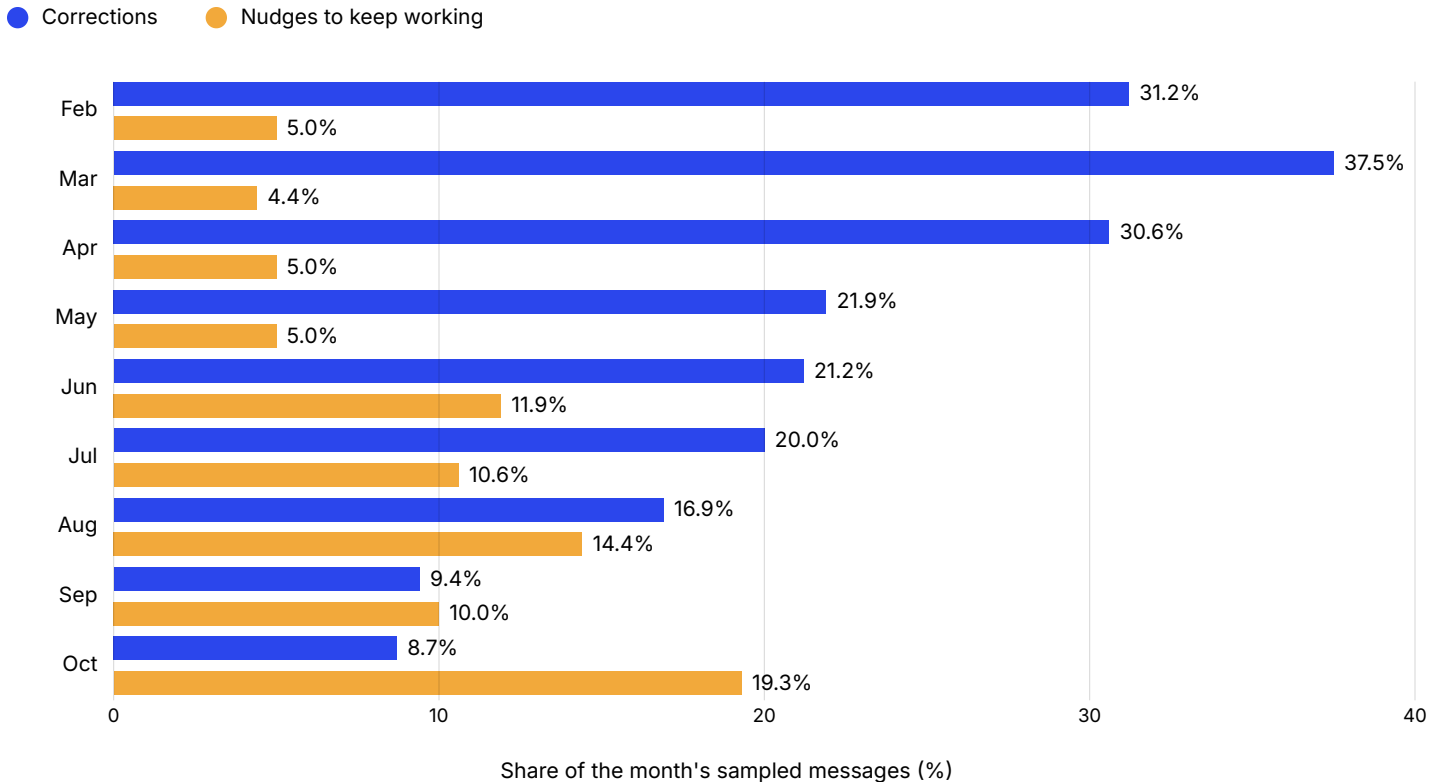
A language-model judge, blind to dates and projects, assigned 1,430 randomly drawn messages (160 per month, 150 in October) to seven categories defined in a fixed codebook. On a subsample of 317 messages, agreement with a second judge built on a different model was 0.79 (Cohen's kappa).

Corrections and nudges

Fig. 1 Share of the operator's messages, by month

Corrections fell by almost three quarters and nudges to keep working nearly quadrupled

160 messages drawn at random each month; 150 in October, when the classifier returned no valid answer for one batch of ten.



A correction points out an error, asks to undo or redo, or repeats an unmet request. A nudge asks the agent to carry on or asks whether it is stuck. Odds ratio per month from a logistic trend: corrections 0.81 (0.77 to 0.86); nudges 1.22 (1.13 to 1.31).

Supervision as a whole, which adds requests to verify and new standing rules to the two series shown, varied between 32% and 46% of messages, with no significant trend; the odds ratio per month was 0.97 (0.93 to 1.01). Its composition shifted: the operator devoted far fewer messages to errors and far more to keeping agents working.

Parallel sessions

Each operator message sent from 10 August onwards (the start of the first week with at least a hundred main-session drafts) was matched with the number of main sessions in which the agent had written something during the preceding fifteen minutes. A further 1,200 messages, 300 from each of four bands defined by the number of active sessions, were then classified by the same judge.

Table 3 Supervision by number of active sessions

The supervision share shows no clear relationship with the number of active sessions

Active sessions	Messages	Supervision	Corrections	Nudges
1 to 2	300	38.3%	15.0%	15.7%
3 to 5	300	34.7%	11.3%	13.0%
6 to 8	300	41.3%	17.3%	13.0%
9+	300	37.3%	9.7%	17.3%

Logistic regression with week fixed effects and standard errors clustered by day. Odds ratio per doubling of active sessions: supervision 0.96 (0.84 to 1.11); corrections 1.16 (0.95 to 1.43).

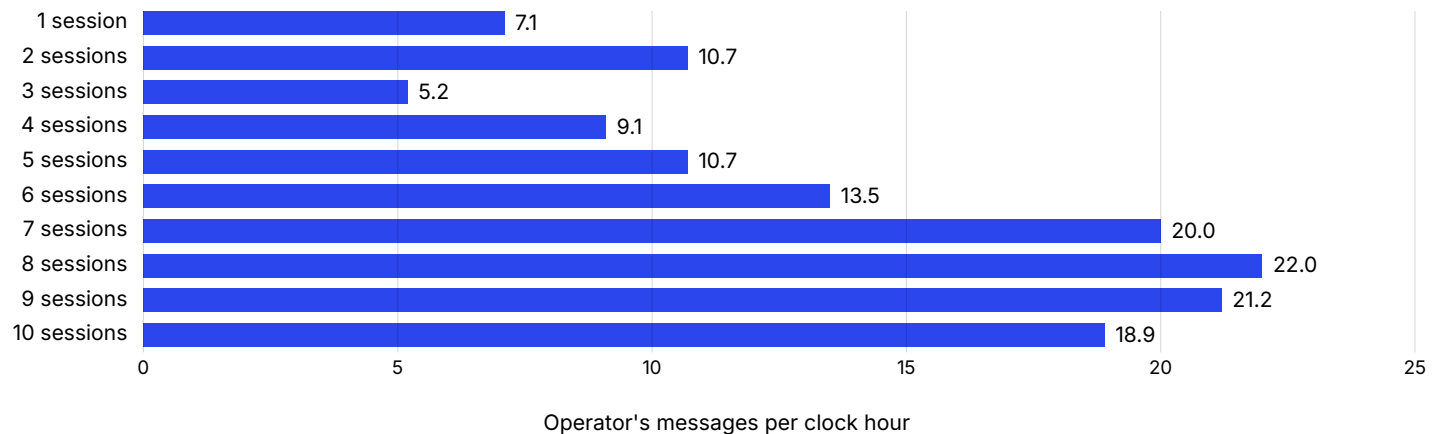
The interval for supervision is compatible with modest effects in either direction, so the defensible reading is that no change was detected, not that none exists.

Attention per session

Fig. 2 Operator's messages per hour, by number of active sessions

With seven or more active sessions the operator wrote about 21 messages per hour; with fewer, the rate was lower and irregular

Each clock hour from 10 August with at least one operator message, grouped by the number of main sessions active in it.



Hours per group: 240, 137, 76, 86, 111, 160, 167, 104, 71, 40. A session counts as active if the agent wrote in it during that hour, unlike Table 3, which looks at the preceding fifteen minutes; hours with more than ten active sessions are omitted. The agent mostly writes after the operator does, so the count of active sessions partly reflects the operator's own activity.

In hours with one active session the operator wrote about seven messages; with seven or more the rate stayed close to 21. Beyond seven sessions, therefore, each additional session divided the same number of messages more thinly. The rise below seven sessions is irregular, and the operator's own activity partly drives the session count.

The model release

Anthropic released Claude Opus 5.5 on 22 September 2026. From that day the new model was used alongside the older one, often in the same projects, until the older one was retired in early October. From 20 to 25 September the writing rules, the guards that check them and the operator were the same, which makes that window the closest the record comes to a controlled comparison, although the operator, not chance, decided which model handled each session. The two models compared, Claude Opus 5 (the older model) and Claude Opus 5.5 (the newer model), are Anthropic's flagship models; other models, used mostly by subagents, are left out.

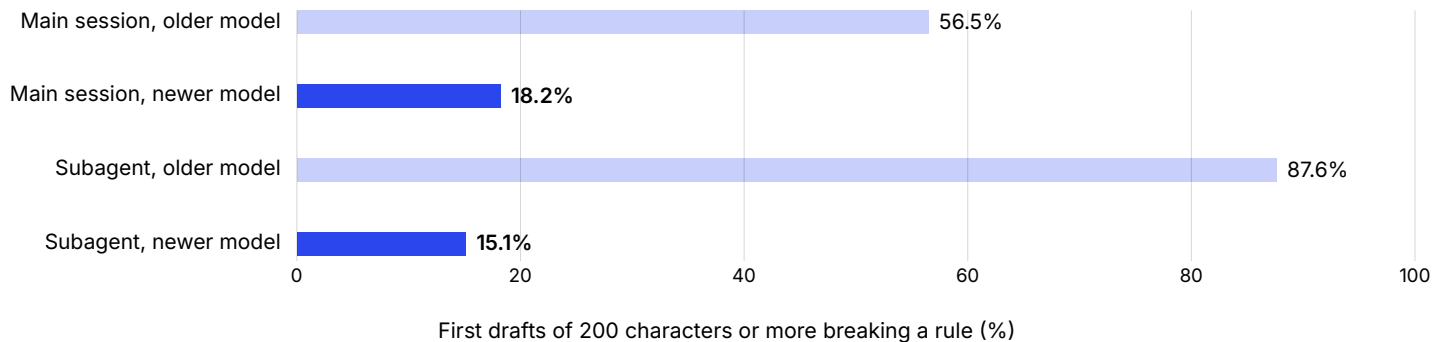
The comparison window closes on 25 September, the day before the writing guard was installed. The language rule written on 25 September is not among the writing rules compared here, and the result is the same from 22 to 24 September (Table 4).

Same days, same rules, two models

Fig. 3 First drafts breaking a writing rule, 20 to 25 September

The newer model broke the rules about a third as often in main sessions and a sixth as often in subagent reports

Older model: 423 main-session drafts and 524 subagent reports. Newer model: 466 and 305.



Odds ratio for the newer model, standard errors clustered by session: main sessions 0.17 (0.11 to 0.27); subagent reports 0.025 (0.014 to 0.047). For dashes alone, main-session rates were 38.3% (older model) and 1.1% (newer model).

Subagents are agents launched by a main session for a subtask. They load the same instruction file and memory index, their reports go to the main session rather than to the operator, and the writing guard never checked those reports. Their violation rate fell with the change of model just as the main sessions' rate did.

Checks for selection bias

Because the operator chose the model for each session, the older model may have been left with the long sessions opened before the release, which had more turns and a longer accumulated context. The comparison was therefore repeated, first with turn number and project as covariates, and then restricted to sessions opened before the release, in which both models worked on the same pool of long sessions.

Table 4 Odds ratio for the newer model, main sessions, 20 to 25 September

The gap between the models holds under every check

Specification	First drafts	Sessions	Odds ratio (95% interval)
As in Fig. 3	889	32	0.17 (0.11 to 0.27)
With turn number and project	889	32	0.16 (0.10 to 0.26)
Sessions opened before the release	585	12	0.12 (0.09 to 0.17)
22 to 24 September only	344	18	0.02 (0.00 to 0.18)
Sessions that used both models	379	3	54.8% against 16.7%

In sessions opened before the release, 56.5% of the older model's 423 drafts broke a rule, against 16.7% of the newer model's 162. No session opened after the release used the older model. Rows two to four add the logarithm of the turn number, and row two also the project; the last row gives the raw shares, older against newer model, because three sessions are too few for a clustered interval, and with twelve sessions or fewer the clustered intervals are likely too narrow. With the model labels shuffled, the first three odds ratios lie between 0.90 and 0.92.

The weeks around the release

Table 5 Weekly share of first drafts with a violation

Violations in subagent reports fell when the newer model took over; in main sessions they had begun to fall about three weeks earlier

Week starting	Main sessions	Subagent reports	Newer model's share
10 Aug	96.1%	90.8%	0%
17 Aug	98.4%		0%
24 Aug	97.0%	92.9%	0%
31 Aug	79.8%	88.0%	12%
7 Sep	66.4%	90.4%	10%
14 Sep	44.9%	74.8%	6%
21 Sep	29.5%	71.9%	36%
28 Sep	14.2%	15.0%	88%
5 Oct	18.9%	12.1%	100%
12 Oct	6.9%	13.8%	100%
19 Oct	3.5%		100%

First drafts of 200 characters or more; cells with fewer than 100 drafts are left blank. The column for the newer model's share covers the whole model family released on 22 September, one member of which was already in limited use from early September, and pools main sessions and subagent reports. The week of 19 October covers two days.

The older model before the release

Dashes, main folder, week of 24 Aug	Same folder, week of 14 Sep
98%	28%
999 first drafts of the older model	871 first drafts of the older model

The older model had begun to improve before the release, mainly in the principal project folder, which held most of the operator's work. Over the same weeks its dash rate fell from 81% to 72% in other folders and from 91% to 58% in subagent reports.

The instruction file has no recorded versions between 18 August and 25 September, so the early decline cannot be attributed to a specific edit. This early decline is why the effect of the release is measured within days rather than across weeks: on the same days, both models worked under the same instructions.

Written rules and guards

A written rule is a sentence, in the instruction file or in a memory note, that every session loads. A guard is a script that Claude Code runs at a fixed point, before a command is executed or when the agent ends its turn. It reads what the agent is about to do and, if this breaks a rule, blocks it and returns an explanation, after which the agent tries again.

Of the 69 hook scripts, only 12 ever blocked anything; the rest add context or keep logs. The writing and completion-claim guards, which read the agent's answers, together accounted for 49% of all blocks.

Inventory of guards

Table 6 The guards that blocked most often

Nine guards account for 99% of all blocks

Guard	What it blocks	First block	Blocks
Safety	Destructive or forbidden commands	17 Aug	1,018
Writing	Breaches of the rules in Table 2 in answers to the operator	26 Sep	882
Completion claims	Unverified claims that a task is complete	17 Aug	796
Workspace	Commands that break the rules of a shared workspace	2 Oct	172
Accounts	Actions that could put a connected account at risk	21 Aug	146
Needless questions	Stopping to ask when the agent can decide	18 Aug	128
Session close	Closing a session without a changelog entry	28 Aug	111
Finish open tasks	Stopping after one batch of an open-ended task	26 Sep	62
Answer every request	Closing a turn with requests left unanswered	12 Oct	62

Blocks recorded in the transcripts from 25 July to 20 October 2026: 3,411 in all. The date is each guard's first recorded block, which marks when it actually came into effect.

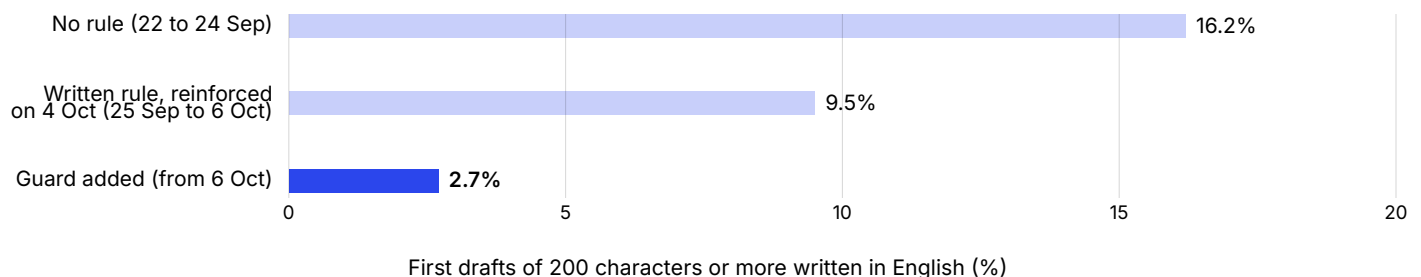
One rule in three regimes

For the newer model, the requirement to answer the operator in Spanish passed through three regimes: no rule until 25 September; a written rule from that date, reinforced in the memory index on 4 October; and a guard from 6 October. Only main-session answers are counted here, since subagents are permitted to report in English.

Fig. 4 First drafts written in English, newer model, main sessions

The share in English was lowest once the guard was in place

Drafts: 315, 2,386 and 2,736. The guard came into effect on 6 October at 19:49 Madrid time.



Odds ratios against the period with no rule, standard errors clustered by session: written rule 0.54 (95% interval 0.28 to 1.06); reinforced rule and guard 0.15 (0.07 to 0.30).

The interval for the written rule includes an odds ratio of one, that is, no effect; the interval for the third step does not. Because the rule was reinforced two days before the guard came into effect, the second period includes two days of the reinforced rule, and the third measures the combined effect of the reinforced rule and the guard. The comparison is before and after, with no control group.

Filtering by the writing guard

First drafts breaking a rule	Final answers breaking a rule
11.2%	6.1%
Main-session turns after the writing guard	The same turns after any retries

Across the 6,377 main-session turns after it came into effect, the writing guard blocked at least once in 9.1% of them: in 329 of them the first draft broke a rule, and in the rest the block fell on a later stop of the same turn, for instance after a message from another session arrived. The guard roughly halved the share of answers that reached the operator with a violation.

Of the 386 first drafts that broke a rule without being blocked, 134 never reached a stop because the operator wrote again before the agent finished, 36 came from installations where the guard was not running, and in 19 the guard read a truncated text, cut short either by a newly arrived message or by the length of very long transcripts.

In the remaining 197, replaying the transcript shows that the text the guard should have read did break a rule, and in every case the violation lay solely in the turn's final message. Claude Code writes its transcript asynchronously, and its public issue tracker documents that the final message may not yet be on disk when a stop hook runs (Anthropic (2026)); a guard that reads the transcript can therefore miss that message. The same replay finds the violation in 327 of the 329 blocked turns, which indicates that the replay reproduces what the guard read. In 34% of the blocked turns the violation was likewise confined to the final message, which must therefore have reached the disk in time; when the final message arrived late, the guard missed the violation.

Learning within a session

If blocks taught the agent anything, drafts following a block should be cleaner than drafts following an unblocked violation. Each session's first violating draft was taken as an anchor, and the next ten first drafts were checked. Once the guard was in place, some first violations were blocked and others were missed for the reasons given above, which allows a comparison within the same weeks and models. Misses are not random: they concentrate in final messages, interrupted turns and very long transcripts.

Table 7 First drafts breaking a rule in the ten turns after a session's first violation

Drafts after a block were not detectably cleaner than drafts after a missed violation

First violation	Sessions	Later drafts	Breaking a rule
Blocked by the guard	30	218	11.9%
Missed by the guard	27	214	16.8%
Before the guard existed	54	382	64.1%

Main sessions, drafts of 200 characters or more, installations where the guard ran. Blocked compared with missed: difference -4.9 percentage points; permutation test over sessions with 2,000 shuffles, $p = 0.19$. With random labels the test rejects at the 5% level in 4.8% of cases.

The point estimate suggests a small benefit from the block (12% against 17%), but a difference at least this large would arise by chance in about one permutation in five if blocks had no effect. The much higher rate before the guard reflects the older model and the earlier weeks, and so says nothing about the effect of the block itself.

The guard and first drafts

Whether the writing guard lowered the violation rate in first drafts can only be estimated against a group the guard never checked. Subagent reports share the instructions but not the guard; their trend before the guard was flatter than that of main sessions (Table 5), so this control is imperfect. Following Simonsohn et al. (2020), the study fitted all 108 combinations of the choices in Table 8 and reports the full set.

Table 8 Specification curve for the guard term

Across 108 specifications the median odds ratio is close to one, and it differs between the two models

Choice	Specifications	Median odds ratio
All specifications	108	1.09
Newer model only	54	0.52
All models, model fixed effects	54	1.61
Outcome: colloquial register	36	0.41
Outcome: dashes	36	2.15
Outcome: any rule	36	1.07

Logistic regressions of a first-draft violation on main session, post-guard period and their interaction; the odds ratio of the interaction is reported, with standard errors clustered by session. Choices: models (two), time trend (two), window (three), minimum length (three) and outcome (three). Permutation test with 1,000 shuffles of the outcome within model: median p = 0.52; 17 specifications significant downwards and 15 upwards, counts with permutation p = 0.098 and 0.14.

The median is compatible with no effect. Significant downward results are not significantly more frequent than chance predicts and are concentrated in the newer-model specifications and the colloquial-register outcome. The record gives no consistent evidence that the guard changed first drafts.

Repeated instructions

Table 9 Repetitions per thousand operator messages, 28 days either side
Repetitions fell by similar amounts for guarded and written rules, with two exceptions

Rule	Regime	Before	After	Ratio	Percentile	Precision
Show the change	Guard	1.0	0.3	0.33	0.15	15/15
Save lasting rules	Guard	1.3	0.4	0.34	0.10	9/15
Report progress	Guard	3.1	15.7	4.93	0.81	12/15
Keep approved work	Guard	0.4	0.2	0.59	0.44	14/14
Answer every request	Guard	2.7	0.0	0.08	0.00	14/15
Finish open tasks	Guard	5.5	7.3	1.31	0.86	15/15
Work without subagents	Written	0.1	0.0	0.33	0.17	5/5
Propose the next step	Written	7.1	3.5	0.50	0.44	15/15
Say where to see it	Written	0.6	0.4	0.73	0.28	15/15
Use what already exists	Written	1.3	1.5	1.19	0.76	14/15

After: from the guard's first block, or from the written rule if there is no guard; the newest guard has eight days. Ratios use a 0.5 continuity correction. Percentile: rank among 200 placebo dates. Precision: sampled matches confirmed as the operator repeating the rule.

The geometric mean of the after-to-before ratio was 0.58 for guarded rules and 0.61 for written ones; only the newest guard beat all 200 placebo dates, over eight days.

Agents inside applications

Besides developing software, the operator built applications in which an agent with its own role is the main way to work: a terminal for analysing equity options, an email client and a studio that writes the company's blog posts. In each one, the application's screens and the agent work on the same information, so the interface complements the agent rather than repeating it.

The same kind of application fits other work: a customer-service desk whose agent prepares the replies shown next to each ticket, a sales tracker whose agent proposes the next step for each opportunity, or a reporting tool whose agent writes the commentary beside each chart.

Supervision of application agents

Table 10 Operator's messages to application agents and to development sessions
Application agents received a larger share of new work and a smaller share of supervision

Sessions	Messages	New work	Supervision	Corrections
Options terminal	120	72.5%	20.8%	5.8%
Email client	120	60.0%	32.5%	9.2%
Blog studio	120	59.2%	32.5%	20.0%
Development sessions	326	46.3%	37.7%	10.4%

Messages drawn at random: 120 per application and, for development sessions in the principal project folder, the same number per month as the three applications together; the record has no development messages for July. Same judge and codebook as Fig. 1. Logistic regression with month fixed effects and standard errors clustered by session, applications against development: supervision 0.68 (0.44 to 1.04); any correction 0.69 (0.45 to 1.06). With the labels shuffled within month, the supervision odds ratio is 0.85.

The direction favours the application agents: the operator used a larger share of messages to assign them work and a smaller share to supervise them, while corrections as the main function of a message were equally frequent (odds ratio 1.03). The intervals include no difference, and an email client poses different tasks from a code base, so the comparison describes the record rather than isolating the design.

Implications for teams

Instruction files and, more recently, hooks are the means teams use to govern coding agents; Chatlatanagulchai et al. (2026) studied the first and Zhang et al. (2026) compared both. This record suggests where that effort pays off.

-
- 01** Test every model release against your own rules before rewriting the rules. Under the newer model the same instructions produced a third as many violations in main sessions over six days, and a team that had edited its rules that week would have attributed the improvement to the edit.

 - 02** Use guards where violations are costly. In this record they filtered output, and no teaching effect was detected. The writing guard removed about half of the violations from final answers at the cost of one retry per block. Check what the guard actually reads: nine in ten of the violations it failed to block despite running lay in the final message, which had not yet been written to the transcript when the guard read it.

 - 03** Keep a fixed instrument. Applying a guard's rules to earlier transcripts provides a point of comparison from before the guard existed, which the guard's own log cannot provide.

 - 04** Measure what supervision consists of, not only how much of it there is. In this record its share showed no significant trend over eight months while its composition changed markedly.

 - 05** Treat the operator's attention as a limiting resource. From about seven parallel sessions the operator's message rate stopped rising, so additional sessions shared a roughly constant number of messages.

 - 06** Build agents into the application where the work lives, give each one a defined role, and let the interface and the agent share one source of data. In this record those agents received a larger share of new work, although the difference in supervision was within chance.

Threats to validity

With a single operator, a single agent tool (Claude Code) and models from a single vendor (Anthropic), this is a case study. Whether the effect sizes generalise to other people, languages and tools is a question for replication.

-
- 01** The author is also the operator. Rules and guards were introduced after repeated violations, so their timing is not random, and part of any later fall may be regression to the mean.
-
- 02** Several changes overlapped: the model release, the writing guard and the reinforced language rule all came within the same two weeks. For that reason the design relies on within-day comparisons, on sessions that used both models and on subagents as an imperfect control.
-
- 03** The fixed instrument covers only rules that can be checked mechanically, and it measures compliance as the guard defines it, not whether the guard's patterns capture the intent of the rule.
-
- 04** The judge that classified messages is a language model; its agreement with a second model was substantial (Cohen's kappa 0.79), and the author, who wrote the messages, validated its labels.
-
- 05** Transcripts from before 25 July had been deleted under Claude Code's retention setting, so the agent-side measures begin only in late July.
-
- 06** Pastes with no typed text (7,504) were removed before the supervision analysis; some were error logs that acted as corrections, so the correction share is a lower bound.
-

Replication material

This report is accompanied by the analysis scripts, the codebook, the rule patterns, the aggregated tables behind every figure, and a design file that records the study design and every dated change made to it after the data had been seen. The operator's messages and the agent transcripts are not: they contain client work and private correspondence.

Every script opens the raw logs in read-only mode and is paired with a deliberately broken variant whose output must differ from the original's, as a check that the script responds to what it is meant to measure. Statistical models were fitted in Python 3.12 with statsmodels 0.15 and pandas 3.0. All intervals are 95% confidence intervals, with standard errors clustered by session or by day where a note says so; the monthly trends in Fig. 1 use ordinary standard errors.

References

-
- 01** Anthropic (2026). Claude Code, public issue tracker, issue 74340, on stop hooks that read the transcript before the final assistant message is written. <https://github.com/anthropics/claude-code/issues/74340>
-
- 02** Becker, J. et al. (2025). Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity. arXiv preprint. <https://doi.org/10.48550/arxiv.2507.09089>
-
- 03** Chatlatanagulchai, W. et al. (2026). Agent READMEs: An Empirical Study of Context Files for Agentic Coding. ACM Transactions on Software Engineering and Methodology. <https://doi.org/10.1145/3840295>
-
- 04** Chi, C. C. (2027). Trust, gating, delegation, and identity: Human-factor dynamics in a 16-month single-developer trace of generative-AI-assisted development. Information and Software Technology. <https://doi.org/10.1016/j.infsof.2026.108351>
-
- 05** Chou, Y. H. et al. (2026). Building Software by Rolling the Dice: A Qualitative Study of Vibe Coding. Proceedings of the ACM on Software Engineering. <https://doi.org/10.1145/3797105>
-
- 06** Correa, I. (2026a). The machine of the future: what the next tool is, and what it will always need from us. Working paper, Zenodo. <https://doi.org/10.5281/zenodo.22671771>
-
- 07** Correa, I. (2026b). The end of software: why code no longer matters and relationships are everything. Working paper, Zenodo. <https://doi.org/10.5281/zenodo.23468806>
-
- 08** Dhanorkar, S., Passi, S. and Vorvoreanu, M. (2026). Human oversight of agentic systems in practice: Examining the oversight work, challenges, and heuristics of developers using software agents. Proceedings of the 2026 ACM Conference on Fairness, Accountability, and Transparency. <https://doi.org/10.1145/3805689.3812402>
-
- 09** Gloaguen, T. et al. (2026). Evaluating AGENTS.md: Are Repository-Level Context Files Helpful for Coding Agents? arXiv preprint. <https://doi.org/10.48550/arxiv.2602.11988>
-

References

-
- 10** He, H. et al. (2026). Speed at the Cost of Quality. Proceedings of the 23rd International Conference on Mining Software Repositories. <https://doi.org/10.1145/3793302.3793349>
-
- 11** Immediato, G. (2025). Philosophy of the Machines: A Manifesto for Humans in the Age of Artificial Agents. Manuscript, first published on Medium on 1 May 2025. <https://medium.com/@generoso.immediato/philosophy-of-the-machine-0590bea0623e>
-
- 12** Jaroslawicz, D. et al. (2025). How Many Instructions Can LLMs Follow at Once? arXiv preprint. <https://doi.org/10.48550/arxiv.2507.11538>
-
- 13** Khelifi, S., Ouni, A. and Khemaja, M. (2026). Behind Agentic Pull Requests: An Empirical Study on Developer Interventions in AI Agent-Authored Pull Requests. Proceedings of the 23rd International Conference on Mining Software Repositories. <https://doi.org/10.1145/3793302.3793586>
-
- 14** Laban, P. et al. (2025). LLMs Get Lost In Multi-Turn Conversation. arXiv preprint. <https://doi.org/10.48550/arxiv.2505.06120>
-
- 15** Li, H., Zhang, H. and Hassan, A. E. (2026). AIDev: Studying AI Coding Agents on GitHub. Proceedings of the 23rd International Conference on Mining Software Repositories. <https://doi.org/10.1145/3793302.3797249>
-
- 16** McMillan, D. (2026). Instruction Adherence in Coding Agent Configuration Files: A Factorial Study of Four File-Structure Variables. arXiv preprint. <https://doi.org/10.48550/arxiv.2605.10039>
-
- 17** Simonsohn, U., Simmons, J. P. and Nelson, L. D. (2020). Specification curve analysis. Nature Human Behaviour. <https://doi.org/10.1038/s41562-020-0912-z>
-
- 18** Zhang, X. et al. (2026). Guardrails Beat Guidance: A Large-Scale Study of Rules, Skills, and Persistent Configuration for Coding Agents. arXiv preprint. <https://doi.org/10.48550/arxiv.2604.11088>
-