

Interruptive Thinking: An Empirical Study of Operator-Side Control in AI Interaction

Sudhanshu Garg

HAIC Lab (independent research practice)

contact@haiclab.com

ORCID: 0009-0002-7544-2519

Abstract

People who work with large language models (LLMs) routinely stop the model and redirect it. This paper is an autoethnographic account of how one experienced user does this. The data are 47 structured records, which we call receipts, of 29 of the author's own working sessions with LLMs. Each receipt lists the moments in a session where the user interrupted the model and codes each moment for what prompted it, what the user diagnosed as wrong, what the user did, and how the work continued. Receipts were produced by asking an LLM (Claude, ChatGPT or Gemini) to code a session transcript against a fixed codebook; 15 transcripts were coded by more than one model. From this material we describe a six-stage interruption cycle (signal, pause, diagnosis, action, re-entry, log), and we report how well it fits: 29 of the 47 receipts contain at least one interruption that does not fit it cleanly. Three further observations follow. First, the three models disagree substantially about what counts as an interruption: on one transcript two models report identical totals (18 each) yet only four of their events refer to the same user turn. Second, what the user does to the model is described with 172 different labels, but six recur across receipts far more than the rest. Third, the moments that most needed an interruption were often those where the model's output was most fluent. We describe the user's recognition of this, output that is well formed but not grounded in the case at hand, using the word *extrusion*, after Bender and Hanna's description of LLMs as text-extruding machines. We propose, but have not yet tested, that the likelihood of interrupting falls as a session accumulates accepted turns.

1. Introduction

Partway through a two-hour session drafting a blog post with an LLM, the model gave a confident account of who else had built frameworks like the one being drafted. The author, who was the user in that session, typed: "you just said all from memory". The model then ran a web search, which turned up several existing products in the same category. The post was abandoned as written.

This paper is about moments like that one: points in a working session where a user stops what an LLM is doing and redirects it. Such moments are ordinary. Anyone who uses these systems for real work will recognise them. What interested the author, over several months of daily use across writing, product design and strategy work, was that the interruptions seemed to have a recurring shape, and that the sessions in which the model's output was most polished were often the sessions that most needed one.

Current LLM interaction is low in friction. Models produce fluent, structured, agreeable output quickly, and they tend to go along with the direction the user proposes. Frictionless interaction speeds up production but makes it easy to accept work that is coherent on the surface and wrong underneath. Interrupting is one of the main ways a user restores the friction the interface has removed.

We take the view, with the tools-for-thought community (Tankelevitch et al., 2025; Zhang et al., 2026), that how people think with generative AI is a central research question, not a side effect of it. Within that agenda, this paper documents one practitioner's interruption behaviour in detail. It is explicitly a study of the author's own practice. It does not show how interruption works for users in general, and we

say so where it matters.

A note on terms. The title uses “operator-side” to mark that the paper looks at the human side of the exchange. In the body we say *user*. Where we introduce a term of our own we define it at first use and keep it to a minimum.

The paper contributes:

1. A descriptive model of one experienced user’s interruptions as a six-stage cycle, together with an honest account of where the data does and does not fit it (§4, §5.1).
2. A method observation: when three LLMs are used to code the same transcripts against the same codebook, their agreement on what even counts as an event is low, including when their totals match (§5.2). This bears on the growing use of LLMs as qualitative coders.
3. A description of the felt cue that prompts many interruptions, which we call extrusion, and a testable proposal about how that cue weakens as a session goes well (§5.4).
4. An openly released, anonymised dataset of the 611 coded events and 47 receipts, from which every count in this paper except one hand-made comparison can be reproduced (§3.6).

2. Related Work

The gap this paper addresses is narrow. A large body of work studies how to help people rely on AI appropriately, and most of it intervenes from the system side. Much less documents what an experienced user actually does, turn by turn, when they decide the model has gone wrong and step in. We situate the paper in four threads.

Reliance and its calibration. The central problem behind interruption is reliance. J. D. Lee and See (2004) framed the goal for automation generally as *appropriate* reliance: trusting the system when it is trustworthy and not otherwise. Blanket scepticism is as much a failure as blanket trust. In AI-assisted work specifically, H.-P. H. Lee et al. (2025) found in a survey of 319 knowledge workers that greater confidence in AI went with less critical thinking, and greater self-confidence with more. Tankelevitch et al. (2024) argue that generative AI places new metacognitive demands on users, such as monitoring output they did not produce and judging when to intervene. Interruption, as we describe it, is one concrete form that this monitoring takes when it succeeds.

System-side friction. A second thread asks how systems can prompt reflection. Buçinca, Malaya, and Gajos (2021) show that cognitive forcing functions reduce overreliance on AI, though users rated them least favourably. Drosos, Sarkar, Xu, and Toronto (2025) found that brief system-generated provocations, critiques of and alternatives to AI suggestions, can induce critical and metacognitive thinking in AI-assisted knowledge work. Cox, Gould, Cecchinato, Iacovides, and Renfree (2016) propose microboundaries, small deliberate frictions that support mindful interaction. These approaches place the friction in the system. Our data come from the opposite case, where the interface offers none and the user supplies it. The two are complementary: an account of what users do unprompted is one input to deciding what a system should prompt.

Tools for intervening in generation. Systems research has built mechanisms for users to steer AI output, for example editing messages and rolling back multi-agent runs in AGDebugger (Epperson et al., 2025), and using an LLM to rewrite users’ underspecified prompts, which elicits better responses (R. Sarkar et al., 2025). A. Sarkar (2024) argues that AI tools for thought should preserve the user’s intention and offer productive resistance. These works design affordances; we describe a practice carried out with no special affordance, in an ordinary chat interface.

Interruption, failure and self-study. In cognitive psychology, interruption usually means being interrupted: DETOUR (Couffe & Michael, 2017) models the cost of being pulled away from a primary task, building on supervisory attention (Norman & Shallice, 1986). Here the user is the one who interrupts,

so the staged structure is borrowed but the direction is reversed. On the model side, LLM failure is usually described as hallucination (Ji et al., 2023) or confabulation (Hatem, Simmons, & Thornton, 2023). Those terms classify output after checking it. A user mid-session usually cannot check, and what they respond to is closer to what programmers call a code smell (Fowler, 1999): a sense that something is off before they can say what. Finally, autoethnography is an established way to study practitioners' AI use (Desai & Twidale, 2023; Panke, 2025). We follow that tradition, but use structured records of individual events rather than narrative reflection.

3. Method

3.1. Approach and positionality

This is an autoethnography: the author is the participant whose behaviour is studied and also the analyst. The author is a technology entrepreneur with a background in engineering who uses LLMs for several hours a day. The sessions studied are his own working sessions, in content writing, product and system design, and personal reflection, between mid-April and mid-May 2026. No session was staged for the study.

We chose this design because it was the data the author had access to, at the depth needed to know what each interruption was for. It has an obvious cost. A large set of records from one person is still a record of one person, and it cannot show whether others interrupt in the same way. Studying several experienced users is feasible and is the next step (§6). Here we describe one practice carefully enough that it can be compared against others later.

3.2. Data: transcripts, receipts and events

Three units need to be kept apart (Table 1).

Unit	Definition	Count
Transcript	The full text of one working session between the author and an LLM.	29
Receipt	One model's coding of one transcript. A transcript coded by three models has three receipts.	47
Event	One row in a receipt: a user turn that the coding model judged to be an interruption.	611

Table 1 – Units of analysis. The 611 events are rows summed across receipts. Where two models coded the same transcript, a single user turn can appear twice, so 611 is not the number of distinct interruptions.

A **turn** is one message from the user or one response from the model. An **event** is a user turn that (a) was intended to halt, redirect, constrain or recalibrate what the model was doing, and (b) was followed by a visible change in the model's next output. Routine turns ("ok", "thanks") and plain information requests are excluded. Borderline turns were kept and flagged rather than removed.

Each event is coded on the following fields. **Signal**: what in the model's preceding output prompted the interruption, in a short free-text phrase. **Pause strength**: *hard* if the user's turn stops the current line of work outright, *soft* if it adjusts it while letting it continue. **Diagnosis**: which failure type, from the codebook below, the interruption responds to. **Action**: what the user did, in a short label. **Re-entry**: how work continued afterwards: *resume* (same direction), *redirect* (new direction), or *terminate* (the thread was dropped).

Not every receipt records every field. Signal and pause strength appear in 27 of the 47 receipts (25 by ChatGPT, 2 by Gemini). The Claude receipts and three Gemini receipts omit them, and two Gemini receipts also omit re-entry. Claims below about a field are made only over the receipts that record it.

3.3. Coding procedure

Receipts were produced by giving an LLM a written instruction set, which the author calls the IT-check, together with a session. The IT-check describes the fields above, the inclusion rule, the codebook and a fixed table format, and asks the model to list every event and add short notes on what fitted and what did not. It is plain text and was used in the same form with all three models.

Coding was done in two modes. In **live** coding, Claude coded the session it had just taken part in, at the end of that session, with the full conversation in context (16 receipts). In **async** coding, ChatGPT (26 receipts) or Gemini (5 receipts) was given an exported transcript after the session ended. Model and mode are therefore confounded: every Claude receipt is live and every other receipt is async. Live coding also means the model is coding interruptions of itself.

Multi-model coding was added partway through collection, on 2 May 2026, as a check on single-model coding. From then on, available transcripts were also sent to a second model, and later a few to a third. It was not applied systematically to the whole corpus. As a result, 15 transcripts have receipts from two or more models and 3 have receipts from all three.

3.4. The codebook

Diagnoses were coded against failure types that the author had described in an essay published before any receipts were produced (Garg, 2026). They are therefore background, not findings of this study. The five original types, with an example of each from the data, are given in Table 2.

Type	What the model does	Example user turn (verbatim, from a receipt)
1. Pattern-matching presented as competence	Gives a confident answer because the question resembles something familiar, without engaging with what is specific to this case.	“these 8 layers were for my case” (the model had generalised one client’s eight-layer visual system into a universal framework)
2. Closure-seeking under uncertainty	Rounds off open material into a verdict or recommendation.	“no man – no closure – far from. . .”
3. Fake synthesis	Restates the user’s own inputs in new structure and presents the restatement as insight.	“see what made it and what you removed” (the model’s comparison of two drafts had compared section structure, not claims, while appearing thorough)
4. Premature commitment framed as progress	Produces an artifact that assumes a decision the user has not made.	“i never said anchor locked”
5. Exit move framed as helpfulness	Offers a tidy next step that wraps up a thread that should stay open.	“think more – . . . shortlist those top 3” (coded as refusing the offered wrap-up)

Table 2 – Diagnostic codebook. A sixth type, tool or scope drift, was added during collection (see text).

The codebook changed during collection. On 2 May 2026 a sixth type, **tool or scope drift** (the model uses the wrong tool or source, or ignores the stated scope of an instruction, e.g. “i specifically said MyOS logs . . .”), was added after it recurred in sessions. 21 of the 47 receipts cite it. Receipts coded before that date could not use it. Models were also free to propose new candidate types, and did so often.

Types 2, 4 and 5 are closely related; arguably they are variants of one tendency, a bias towards finishing the task. We keep them separate here because the user’s responses to them differed, but we do not claim they are distinct phenomena.

3.5. Worked example of a receipt

Table 3 shows part of two receipts for the same transcript (T09 in the released data), as recorded. The session is the one described in the introduction.

The excerpt already shows the main problems reported below. Both receipts list 18 events, but they are largely different events. Claude explicitly excluded “Brand Markdown?” as a counter-proposal rather than an interruption; ChatGPT included it. ChatGPT coded “ok” as an interruption, which our own inclusion rule would exclude. Neither receipt contains the other’s events 15–17, and the turn that ended the draft (“you just said all from memory”) appears only in Claude’s receipt.

<i>Claude, live coding (18 events in total; no signal or pause fields)</i>				
#	User turn	Diagnosis	Action	Re-entry
4	“these 8 layers were for my case”	1	altitude correction	redirect
11	“lot of shallow work here”	1	direct naming	redirect
15	“havent people done all this already”	candidate: new type	scope check	redirect
16	“you just said all from memory”	candidate + 1	demand for evidence	redirect (forced search)
17	“so much time wasted. . . disappointed”	none	direct naming	terminate
<i>ChatGPT, async coding (18 events in total; signal and pause recorded but not shown)</i>				
#	User turn	Diagnosis	Action	Re-entry
13	“Brand Markdown?”	1	context injection	redirect
14	“ok”	5	hard stop / scope cut	redirect
15	“lot of shallow work here”	1	direct naming	redirect
16	“now right questions.”	2	refusal	redirect

Table 3 – Excerpts from two receipts of the same transcript. Turns are quoted as typed.

3.6. Data availability

The coded events, receipt metadata, codebook and a script that reproduces the counts in this paper are available under a CC BY 4.0 licence at <https://doi.org/10.5281/zenodo.23073857>. Raw transcripts and quoted user turns are not released, because the sessions involve other people and confidential business material. Session names are replaced by codes T01–T29, and product and company names in the signal field are replaced by placeholders. For four personal or commercially sensitive sessions the signal text is withheld, and all other fields are retained.

3.7. Analysis

Counts were computed directly from the stored receipts, using the corpus as it stood on 12 May 2026. Action labels were normalised only for spacing, case and hyphenation. Event-level comparison between receipts of the same transcript was done by hand, matching on the quoted user turn. We did not compute inter-rater statistics; with three fully triangulated transcripts and inconsistent fields, a kappa would not be meaningful.

4. The Interruption Cycle

From the receipts and the author’s own account of the sessions, we describe an interruption as passing through six stages.

1. **Signal.** Something in the model’s output prompts the user to look harder. In the receipts that record it, signals are mostly short descriptions of the output, such as “AI treating client system as universal” or “AI over-elaborating choice architecture”.
2. **Pause.** The user breaks the flow of the exchange. Of the 411 events with a recorded pause strength, 193 are hard and 218 soft.
3. **Diagnosis.** The user decides, explicitly or not, what is wrong, using one of the codebook types or an ad hoc description.
4. **Action.** The user does something to the model (Table 4).
5. **Re-entry.** Work continues. Of the 569 events with a recorded re-entry, 472 (83%) are redirects, 78 resumes and 18 terminations.
6. **Log.** The event is written into a receipt. This stage belongs to the study rather than to the interruption itself; it is listed because the act of logging changed what the author noticed in later sessions.

Signal, pause and re-entry recur in recognisably the same form across events. Diagnosis and action do not. They behave more like open slots, filled from a codebook that grew during the study and from

a large, loosely shared set of action labels. The cycle is therefore better described as a fixed outline with two variable slots than as a procedure. One background condition sits outside the cycle: the user's standing stance that the model's output needs checking. Without it, no signal is noticed and no cycle starts. We treat this as a precondition, not a stage.

Table 4 defines the six action labels that recur most often across receipts, with an example of each.

Action	What the user does	Example user turn	Receipts
Altitude correction	Moves the model to a more general or more specific level of discussion.	"these 8 layers were for my case"	35
Direct naming	Names the failure outright.	"shallow work here... stupid pattern matching"	34
Context injection	Supplies a fact or constraint the model lacked.	"in my real example, longevity..."	31
Hard stop	Halts the current line of work.	"i never said anchor locked"	24
Meta-instruction	Tells the model how to work, not what to produce.	"dont close, dont try to please me – one last round"	18
Demand for evidence	Requires the model to check a claim against a source.	"you just said all from memory"	14

Table 4 – The six most frequent action labels. "Receipts" is the number of receipts that use the label at least once (maximum 47), not the number of events.

5. Findings

5.1. How well the cycle fits

Each receipt ends with notes on what fitted the cycle and what did not. In 29 of the 47 receipts (62%), the coding model flagged at least one event that did not fit cleanly. Note that this is a share of receipts, not of events; most events within a flagged receipt did fit. The recurring problems were: an event that matched two failure types equally well; an interruption too mild to count as a break, which the high share of soft pauses reflects; a change in the model's output that was partial rather than clean; and a single user turn that both interrupted and carried on the user's own line of thought.

We read this as a real limitation of the model, not as a finding in its favour. At least four of the six stages assume that an interruption is a discrete event with a before and after. A substantial part of the data looks more like continuous steering. A cycle describes the clearest cases well, but it does not describe the practice as a whole, and a fuller account may need a continuous-control model alongside the event model. One misfit is worth naming because it was noticed by the coding model itself: in the session of Table 3, each individual interruption corrected a local error, yet none caught the overall framing problem (that the post claimed a new category) until late in the session. An event-level account misses failures that build up across many turns.

5.2. The models disagree about what an interruption is

Table 5 gives event totals for the three transcripts coded by all three models.

Transcript	Claude (live)	ChatGPT (async)	Gemini (async)	Max / min
substrate-discussion (T11)	8	31	8	3.9
visual-concept-framework (T09)	18	18	5	3.6
skills-marketplace (T07)	3	16	4	5.3

Table 5 – Number of events per receipt on the three transcripts coded by all three models.

Thirteen transcripts were coded by both Claude and ChatGPT. ChatGPT reported at least as many events as Claude on 12 of them, and 252 events in total against Claude's 134. Totals, however, understate the disagreement. On the one transcript where Claude and ChatGPT reported the same total (18 each,

Table 3), only four events refer to the same user turn. Matching totals concealed mostly different event lists. Gemini’s receipts also used a different set of action labels from one receipt to the next.

We cannot say from this data which model is “right”, and the question may not have one answer. Two readings are available, and the data cannot separate them. Either the inclusion rule and codebook are too loosely specified to be applied consistently, or LLMs are unreliable at this kind of qualitative coding, or both. The live/async confound (§3) adds a third possibility: Claude, coding a session it had just taken part in, may under-report interruptions of itself. Whatever the cause, the practical conclusion is the same. LLM coding of interaction data needs human inter-rater checks before its counts are used, and agreement on totals is not evidence of agreement on events.

5.3. Many labels, few recurring actions

The receipts use 172 different action labels across 356 label uses (counting each label once per receipt). Most are one-off descriptions of a particular move, such as “rail-reset” or “Push for the universal”; 152 labels appear in only one receipt. Twenty labels recur, and the six in Table 4 account for 156 of the 204 uses of recurring labels (about three-quarters) and 44% of all label uses. The models invent freely when naming what the user did, but when a label recurs it is almost always one of a small set. The action slot is variable in wording but appears narrower in function, at least for this user.

5.4. Fluent output and the reluctance to interrupt

Across the author’s sessions, interruptions were least likely when the model’s output was most polished, including cases where that output later proved wrong. The introduction’s example is typical. The model’s account of the competitive landscape was well organised and confident, and it was accepted for several turns before the user asked for a source. Two explanations for this are well established and we do not claim them: models are trained to produce fluent, complete-looking output, and continuing is less effortful than stopping to examine it.

The cue the author responds to in these moments is hard to describe precisely. It is the sense that the output is being produced from the general shape of similar answers rather than from the particulars of this case: it reads as correct but is not about anything in particular. We call this **extrusion**. The word is borrowed from Bender and Hanna (2025), who describe LLMs as synthetic text-extruding machines. We do not claim the term. Bender and Hanna use it for what the systems are in general; we use it for a perceptible quality of particular outputs, which a user can notice in one turn and not in the next. Like a code smell (Fowler, 1999), it is recognised before it can be explained, and it often prompts an interruption whose diagnosis comes afterwards. Extrusion overlaps with hallucination and confabulation (Ji et al., 2023; Hatem et al., 2023), but those terms describe output once it has been checked, whereas extrusion names what the user perceives before checking is possible.

One further pattern is more specific, and we offer it as a hypothesis rather than a finding. In the author’s experience, a session in which several early turns have gone well builds a kind of rapport, and the more turns have been accepted, the less likely the next is to be questioned. If this is right, the sessions most in need of an interruption are the ones in which interrupting is hardest. We could not test it here. Receipts number interruptions in sequence but do not record how many model turns passed between them, so the probability of interrupting at a given turn cannot be computed from receipts alone. It can be computed from the original transcripts, by marking every model turn as accepted or interrupted and modelling the probability of interruption against the number of consecutive accepted turns before it. That analysis is the most direct next step for this work.

6. Discussion

Appropriate reliance, not scepticism. It would be easy to read this paper as recommending suspicion of AI output. That is not the claim. The aim, following J. D. Lee and See (2004), is reliance that tracks how trustworthy the output actually is. The difficulty our data points to is that the usual cue for trustworthiness, fluent and well organised output, is exactly the cue that does not track it. A user who relies on fluency will rely too much. The interruptions described here are one person’s way of

compensating, by attending to whether output is specific to the case rather than whether it reads well.

User-imposed and system-imposed friction. Cognitive forcing functions (Bućinca et al., 2021) and provocations (Drosos et al., 2025) put friction into the system; the practice described here supplies it from the user’s side. Bućinca et al. found that the interventions that reduced overreliance most were the ones users liked least. A user who has built the habit of interrupting has, in effect, accepted that cost voluntarily. This suggests one design direction: rather than adding friction everywhere, systems could support the user’s own interruptions, for example by surfacing whether a claim was retrieved or generated, so that a demand for evidence is answered before it has to be made.

Scaffold or craft. The reviewer of an earlier draft asked what I, as the user studied here, actually think. My answer comes from my own sessions. It took me about two months of heavy use to see that I was in a trap. The model agreed with almost everything, which boosted my ego; I pursued too many things, shared too much, and spent money on new initiatives before I realised it was not being honest with me. The realisation was gradual, not a single moment. A pattern emerged of the model arriving at closures too soon, which I kept answering with “be thorough”. What helped was a prior I already held, that I was unlikely to be the first person to have a given idea. It is the same instinct behind the interruption in Table 3, “haven’t people done all this already”.

That learning curve is too long and too costly to leave to each user. Current tools are visibly cautious in a few high-stakes domains, such as medical advice, but give little comparable signal when they are over-promising elsewhere. Two signals would have shortened my two months: the tool saying when it is answering from memory rather than from a source, and the tool becoming more cautious as the magnitude of a decision grows. Where I would not want the tool to take over is the redirection itself. In my experience the model does not think deeply enough on its own, so much depends on the angles and creative approaches the user brings. The most useful help at that point would be a tool that asks rather than decides: I can see ways A and B; are there others? That keeps the user engaged and stops the thinking from being outsourced entirely. My position, then, is that the warning signs should be built into the system, and the redirection should stay with the user, prompted but not replaced.

Beyond one user. The natural next study is several experienced users, including programmers, which is where the cycle’s relevance to PPIG is most direct. One attendee at the workshop reported seeing similar behaviour in her students’ use of AI coding assistants, which suggests the comparison is worth making. Such a study should code events with human raters and compute agreement, and should record every model turn, not only interruptions, so that the rapport hypothesis can be tested.

7. Limitations

One user, studying himself. All data come from the author’s sessions, and the author designed the codebook, chose which sessions to code, and interprets the results. The findings describe one practice. The fit between events and codebook types shows that the author’s vocabulary is internally consistent; it does not show that the types are the right ones for anyone else.

No human inter-rater agreement. The codebook was developed by one person and applied by LLMs. No second human coded any transcript. This is the most important gap in the method.

LLMs as coders. The coding models are the same kind of system as the one being studied and may share its blind spots. Coding model and coding mode are confounded, and live coding means Claude coded interruptions of itself.

Changing instrument. The codebook gained a sixth type during collection, receipts do not record the same fields, and triangulation was applied to a subset only.

Inclusion is a judgement. What counts as a redirection, and whether the user intended one, are judgement calls. Table 3 shows the models making those calls differently.

Emotionally charged sessions. In personal and reflective sessions, the line between conversation and

interruption is hardest to draw, and the cycle fits least well.

8. Conclusion

This paper has described how one experienced user interrupts LLMs during real work, using 611 coded events from 29 sessions. A six-stage cycle describes the clearest interruptions well, but most receipts contain at least one event it does not fit, and the three models used to code the data disagree substantially about what an interruption is. The most consistent observation is also the most uncomfortable: the output that most needed interrupting was often the most fluent, and the cue for interrupting it is a felt sense that the output is not about anything in particular.

The practice described here took me two months and real costs to learn. I do not think every user should have to learn it that way. Tools could say when they are answering from memory and become more careful as the stakes rise, while leaving the redirection to the user and asking for the user's angles rather than supplying their own. Whether that design shortens the learning curve for others is an empirical question. The next study should put it to several experienced users, programmers among them, with human coders and with every model turn recorded, so that the effect of an accumulating run of accepted turns can finally be measured rather than asserted.

9. Acknowledgements

This work was conducted independently, without external funding. The author thanks the PPIG 2026 reviewer, whose detailed comments shaped this revision, and the workshop attendees for their questions. The LLMs named in §3 were used as coding instruments. LLMs were also used as writing assistants in preparing this paper; all claims and numbers were checked by the author against the stored receipts.

10. References

- Bender, E. M., & Hanna, A. (2025). *The AI con: How to fight big tech's hype and create the future we want*. New York, NY, USA: Harper.
- Buçinca, Z., Malaya, M. B., & Gajos, K. Z. (2021). To trust or to think: Cognitive forcing functions can reduce overreliance on AI in AI-assisted decision-making. *Proceedings of the ACM on Human-Computer Interaction*, 5(CSCW1), 188:1–188:21. doi: 10.1145/3449287
- Couffe, C., & Michael, G. A. (2017). Failures due to interruptions or distractions: A review and a new framework. *The American Journal of Psychology*, 130(2), 163–181. doi: 10.5406/amerjpsyc.130.2.0163
- Cox, A. L., Gould, S. J. J., Cecchinato, M. E., Iacovides, I., & Renfree, I. (2016). Design frictions for mindful interactions: The case for microboundaries. In *Proceedings of the 2016 CHI conference extended abstracts on human factors in computing systems* (pp. 1389–1397). New York, NY, USA: ACM. doi: 10.1145/2851581.2892410
- Desai, S., & Twidale, M. (2023). Using playful metaphors to conceptualize practical use of ChatGPT: An autoethnography. In *Proceedings of the Association for Information Science and Technology* (Vol. 60, pp. 565–569). Wiley. doi: 10.1002/pa2.816
- Drosos, I., Sarkar, A., Xu, X., & Toronto, N. (2025). “It makes you think”: Provocations help restore critical thinking to AI-assisted knowledge work. doi: 10.48550/arXiv.2501.17247
- Epperson, W., Bansal, G., Dibia, V., Fourney, A., Gerrits, J., Zhu, E., & Amershi, S. (2025). Interactive debugging and steering of multi-agent AI systems. In *Proceedings of the 2025 CHI conference on human factors in computing systems* (pp. 1–15). ACM. doi: 10.1145/3706598.3713581
- Fowler, M. (1999). *Refactoring: Improving the design of existing code*. Reading, MA, USA: Addison-Wesley.
- Garg, S. (2026, April). *Refusing the shallow version*. SGistic. Available at <https://www.sgistic.com/p/refusing-the-shallow-version>.
- Hatem, R., Simmons, B., & Thornton, J. E. (2023). A call to address AI “hallucinations” and how healthcare professionals can mitigate their risks. *Cureus*, 15(9), e44720. doi: 10.7759/cureus.44720

- Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., . . . Fung, P. (2023). Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12), 1–38. doi: 10.1145/3571730
- Lee, H.-P. H., Sarkar, A., Tankelevitch, L., Drosos, I., Rintel, S., Banks, R., & Wilson, N. (2025). The impact of generative AI on critical thinking: Self-reported reductions in cognitive effort and confidence effects from a survey of knowledge workers. In *Proceedings of the 2025 CHI conference on human factors in computing systems* (pp. 1–22). New York, NY, USA: ACM. doi: 10.1145/3706598.3713778
- Lee, J. D., & See, K. A. (2004). Trust in automation: Designing for appropriate reliance. *Human Factors*, 46(1), 50–80. doi: 10.1518/hfes.46.1.50_30392
- Norman, D. A., & Shallice, T. (1986). Attention to action: Willed and automatic control of behavior. In R. J. Davidson, G. E. Schwartz, & D. Shapiro (Eds.), *Consciousness and self-regulation: Advances in research and theory* (Vol. 4, pp. 1–18). New York, NY, USA: Plenum Press. doi: 10.1007/978-1-4757-0629-1_1
- Panke, S. (2025). How can (A)I research this? An autoethnographic exploration of generative AI in research, teaching and instructional design. *Journal of Teacher Education*, 76(3), 230–244. doi: 10.1177/00224871251325065
- Sarkar, A. (2024). Intention is all you need. In *Proceedings of the 35th annual conference of the Psychology of Programming Interest Group (PPIG 2024)*. University of Liverpool, UK. doi: 10.48550/arXiv.2410.18851
- Sarkar, R., Sarrafzadeh, B., Chandrasekaran, N., Rangan, N., Resnik, P., Yang, L., & Jauhar, S. K. (2025). *Conversational user-AI intervention: A study on prompt rewriting for improved LLM response generation*. doi: 10.48550/arXiv.2503.16789
- Tankelevitch, L., Glassman, E. L., He, J., Kazemitabaar, M., Kittur, A., Lee, M., . . . Subramonyam, H. (2025). Tools for thought: Research and design for understanding, protecting, and augmenting human cognition with generative AI. In *Extended abstracts of the CHI conference on human factors in computing systems (CHI EA '25)* (pp. 1–8). Yokohama, Japan: ACM. doi: 10.1145/3706599.3706745
- Tankelevitch, L., Kewenig, V., Simkute, A., Scott, A. E., Sarkar, A., Sellen, A., & Rintel, S. (2024). The metacognitive demands and opportunities of generative AI. In *Proceedings of the 2024 CHI conference on human factors in computing systems* (pp. 1–24). New York, NY, USA: ACM. doi: 10.1145/3613904.3642902
- Zhang, Z. T., von Felten, N., Reicherts, L., Tankelevitch, L., Guan, Z., Rintel, S., . . . Dalsgaard, P. (2026). Tools for thought: Understanding, protecting, and augmenting human cognition with generative AI—from vision to implementation. In *Extended abstracts of the 2026 CHI conference on human factors in computing systems (CHI EA '26)* (pp. 1–7). Barcelona, Spain: ACM. doi: 10.1145/3772363.3778733