Why AI Struggles to Pick Up a Half-Finished Task

Originally published at https://www.unite.ai/why-ai-struggles-to-pick-up-a-half-finished-task/

Though AI agents can solve complex tasks, a new study indicates that they struggle to continue work started by another, leading to duplicated effort, slower progress, and greater costs.

One of the most exhausting yet essential tasks in dealing with AI agents and interfaces is that the AI needs 'bringing up to speed' at the outset of an exchange, in almost every case.

While popular language models such as ChatGPT do offer some access to 'persistent' custom memories, the implementation is usually a hit-and-miss affair; in the end, it's normally safer to accept the effort of contextualizing* the task for the AI – at least, to stop it 'guessing' a wrong context from its trained latent space.

Picking up Real-World Slack

The challenge predates AI, of course; many companies already require staff to maintain documentation regarding processes that they develop or refine (partly for smoother onboarding, but arguably also to avoid employees gaining leverage).

However, in practice, it is often only larger and better-funded organizations that honor a commitment to creating, updating and maintaining documentation. Very often, instead, employees required to pick up the work of others are handed a 'detective'-style task that requires them to painstakingly unpick the timeline that led to the abandoned work they have now been given.

Needless to say, immaculate documentation would save days, weeks, or even months of work – if only it was a financially rational proposition.

However, where AI agents are the operatives in question, there may be greater scope to potentially resolve the problem.

Hand it Over

This burden of 'undocumentation' is quantified in a new research paper from the US, which calls the problem *handoff debt*.

If *technical debt* is the syndrome where quick-and-dirty (and cheap) tech solutions today lead to brittle or hard-to-maintain solutions in the future, then *handoff debt* defines the cost of *rediscovery* – the forensic retracing of the steps of a worker or entity who is either not available to advise (hostile firing, too busy, dead, etc.) or otherwise unable to advise (for instance, an LLM that has long-since dumped the context that led to the current state of the work).

The new paper[†] – a collaboration between independent and Georgia State University-affiliated researchers – deals with handoff debt as it applies to coding agents that are tasked with picking up where another session, person or entity left off in a codebase.

One of the aims of the work is to establish exactly how much documentation is necessary to reduce handoff debt, and what procedures and protocols might be recommended to adopt as standard practice in the future, to minimize the issue.

Budgetary Concerns

In an ideal world, one could set logging to *verbose* and just feed the neophyte agent (the one picking up the task) the logs related to the incomplete task.

However, parsing such a volume of data into useful data would be both time-consuming, and would also eat into the token budget – as well as bringing storage-space constraints into play.

This *is* a budgetary problem, because using raw dumps is draining, while using curated logs is less confusing, but requires prior commitment of resources.

Proper, dedicated notes would be very effective in getting a 'pickup artist' up to speed, but at the cost of an even greater commitment of effort – effort that may never be needed, if the work's logic should ultimately prove self-evident, or if the work is abandoned, or never revised again.

The authors of the new work, titled *Handoff Debt: The Rediscovery Cost When Coding Agents Take Over Interrupted Tasks*, have considered all these scenarios, and have adapted existing task models to new ways of quantifying and addressing handoff debt. Though the work deals specifically with coding agents, it may nonetheless indicate useful routes forward in wider AI contexts, and in the logistics of documentation policies.

The authors state:

> *‘Handoff debt arises when an agent makes visible progress but leaves state that a successor cannot readily continue from, such as unexplained edits, scratch files, hidden assumptions, or missing validation evidence.*
>
> *‘A metric based solely on final resolution cannot distinguish between costly rediscovery and efficient continuation.*
>
> *‘Two predecessor agents may leave the same checkpointed repository, yet their successors can face very different continuation costs: one may continue immediately, while another must spend many tool interactions rediscovering intent from scratch files and incomplete command history.’*

Method

The authors define *predecessor* as the prior agent (the one who originated or last undertook the work) and *successor* as the current agent (the one tasked with picking up the work),

In support of a benchmark designed to measure the cost of transferring unfinished software-engineering tasks across agents, 75 tasks from SWE-bench Verified were converted into 181 *handoff* scenarios, each representing a point where work had been interrupted and passed to a successor agent. Three different successor models were then tested across 2,172 takeover attempts.

The model families used, and variously mixed in these handoff tests, were Qwen, Gemma, and Devstral.

The experiments examined four levels of inherited information: in the most restrictive setting, the successor received only the state of the repository (effectively, walking into an undocumented ‘disaster area’). Other settings provided increasingly detailed context, from activity traces and command histories, to compact summaries describing what had already been attempted and learned:

Repository only The successor receives only the repository and task description, with no record of earlier actions, decisions, or failed attempts.	**Raw trace** The successor receives the predecessor’s complete history, exposing every command, observation, edit, success, and failure.
Summary notes The successor receives a natural-language summary generated from the predecessor’s activity history, condensing key information into prose.	**Structured notes** The successor receives a compact handoff document containing standardized fields describing task status, changes made, and validation results.

Rather than focusing solely on whether a task was eventually solved, the study was designed to measure the *cost of continuation itself*, with attention paid to tool use, token consumption, and the amount of effort required to reconstruct the reasoning behind earlier work.

Three *handoff point detection* definitions and three *handoff states* were defined for the experiments:

Handoff Point Detection	Handoff States
After first source edit. After first code change. The first agent has started working but has not yet checked whether the change actually works.	***Needs completion.*** The task is unfinished, and the successor must continue working to reach a correct solution.
After first validation result. The first agent has already run a test or validation step, providing some evidence about progress.	***Already solved and preserved***. The task has effectively been completed, and the successor's job is to avoid breaking it.
After first post-failure edit. A test has failed and the first agent has already tried to respond by making another change.	***Existing behavior broken.*** Something that worked before is now broken.

Data and Tests

To create realistic handoff scenarios, the authors' benchmark was built from 75 software-engineering tasks drawn from SWE-Bench Verified, with an emphasis on problems that typically take between 15 minutes and 4 hours to solve.

Rather than evaluating only completed tasks, the researchers captured multiple intermediate checkpoints during the work, creating situations where one AI agent had to take over from another:

Stage	Breakdown	Count
Source tasks	SWE-bench Verified	75
Predecessor runs	Qwen	75
Handoff points	*After first source edit*	75
	After first validation result	75
	After first post-failure edit	31
	Total handoff points	**181**
Handoff states	*Needs completion*	110
	Already solved; preserve	61
	Existing behavior broken	10
	Total state-labeled handoff points	**181**
Handoff views	*Repository only*	1
	Raw trace	1
	Summary notes	1
	Structured notes	1
	Views per handoff	**4**
Takeover runs	181 handoffs × 4 views	724/model
Total takeover runs	**724/model × 3 successors**	**2,172**

Construction of the takeover benchmark. Seventy-five SWE-bench Verified tasks were expanded into 181 handoff points spanning three stages of work, labeled according to repository state at takeover time, and evaluated under four information-sharing conditions, producing 2,172 total successor-agent takeover runs. Source

Because each task could generate several handoff points, and each handoff was tested using four different forms of transferred information, the benchmark expanded rapidly, with the final dataset comprising 181 distinct handoff tasks, and 724 takeover evaluations for each successor model, producing 2,172 takeover runs across the three AI systems tested.

An OpenHands-style coding agent environment was used for the tests, featuring terminal actions, repository freezing at handoff points, file-editing, and official validation from the SWE-Bench benchmark.

In the primary study, the handoff points all issue from Qwen-based predecessor runs, in order to provide a fixed starting point to evaluate the difference between various agent combinations and the diverse scenarios.

Takeover pairs tested were Qwen-to-Qwen; Qwen-to-Gemma; and Qwen-to-Devstral.

Raw trace produced the largest reductions in successor effort, cutting agent events by 57-59%, while *Summary notes* and *Structured notes* reduced events by 20-46%. Prompt-token usage also fell across all three approaches, with reductions ranging from 42-63%:

View	Runs	Solved rate (Δ pp)	Agent events (Δ%)	Prompt tokens (Δ%)
Qwen → Qwen				
Repository only	181	46.4%	99	1.63M
Raw trace	181	52.5% (+6.1 pp)	41 (-59%)	811k (-50%)
Summary notes	181	51.4% (+5.0 pp)	53 (-46%)	602k (-63%)
Structured notes	181	50.8% (+4.4 pp)	55 (-44%)	660k (-60%)
Qwen → Gemma				
Repository only	181	42.5%	49	738k
Raw trace	181	49.2% (+6.6 pp)	21 (-57%)	300k (-59%)
Summary notes	181	44.2% (+1.7 pp)	33 (-33%)	319k (-57%)
Structured notes	181	43.6% (+1.1 pp)	39 (-20%)	317k (-57%)
Qwen → Devstral				
Repository only	181	34.3%	175	3.94M
Raw trace	181	49.2% (+14.9 pp)	73 (-58%)	1.66M (-58%)
Summary notes	181	43.6% (+9.4 pp)	123 (-30%)	2.30M (-42%)
Structured notes	181	44.8% (+10.5 pp)	125 (-29%)	2.30M (-42%)

Under *Repository only* handoffs, successor agents had to spend additional interactions reconstructing predecessor intent, previous evidence, and failed approaches. *Raw trace*, *Summary notes*, and *Structured notes* transferred part of that information directly, reducing the amount of rediscovery required, though at the cost of larger initial prompts.

To test whether the gains were genuine, each context-rich handoff was matched against a repository-only handoff starting from the same point. Across all model pairings, richer handoffs consistently reduced the work required from successor agents.

Full event traces produced the largest reductions, while summary and structured notes also delivered substantial savings. The effect appeared across the benchmark rather than being driven by a small number of outliers:

View	Matched Runs	Repo-Only Agent Events	Agent Events (Δ%)	95% CI for Δ Events	Prompt Tokens (Δ%)
Qwen → Qwen					
Raw Trace	181	99	41 (-59%)	[-50%, -42%]	798k (-51%)
Summary Notes	181	99	53 (-46%)	[-38%, -28%]	572k (-65%)
Structured Notes	181	99	55 (-44%)	[-34%, -24%]	646k (-60%)
Qwen → Gemma					
Raw Trace	181	49	21 (-57%)	[-47%, -33%]	300k (-59%)
Summary Notes	181	49	33 (-33%)	[-25%, -8%]	319k (-57%)
Structured Notes	181	49	39 (-20%)	[-18%, -1%]	317k (-57%)
Qwen → Devstral					
Raw Trace	181	175	73 (-58%)	[-45%, -22%]	1.65M (-58%)
Summary Notes	181	175	123 (-30%)	[-28%, -15%]	2.28M (-42%)
Structured Notes	181	175	125 (-29%)	[-28%, -17%]	2.29M (-42%)

To confirm that the effect was not driven by a handful of unusual cases, the researchers compared each handoff against an equivalent *repository-only* handoff starting from the same point. The reductions remained consistent across all model pairings, indicating that the benefits reflect a meaningful pattern, rather than a few exceptional examples.

Take it Away…

In short†, the authors found that when one AI hands a task to another, even simple notes help the second AI continue more efficiently.

Full records of what happened work best, but any handoff information is better than leaving the successor to reconstruct everything from the code alone; and the results above illustrate that the ‘full fat’ raw log approach inevitably has a higher token cost.

Conclusion

Though the paper itself is aimed strictly at peer researchers, with limited appeal for the casual reader, the new work nonetheless addresses one of the most interesting and pressing problems in regard to the current state of the art in human>AI interfaces and protocols.

One would hope that the paradigms developed and insights gained in this kind of exploration might eventually extend to a wider context of AI usage than just agentic coding.

One additional avenue of exploration might be for future projects to consider ways to evaluate what level of documentation might be considered the minimum for a particular project, based on its characteristics and use case. However, even this functionality, which would help to rationalize expenditure of time and money, itself costs time and money; and so the budgetary conundrum involved in documentation scenarios remains hard to escape.

* *Personally, for ChatGPT sessions which become burdened with lag and excessive context, I have lately taken to exporting (with some difficulty) a clean PDF of the chat and using it as a starting point for a new session, which becomes 'part 2'.*

† *Unfortunately this is not the most approachable paper I have read this year, and for this reason I cannot recommend the reader to the source work, though the digested results remain of interest.*

First published Wednesday, June 3, 2026

Writer on machine learning, domain specialist in human image synthesis. Former head of research content at Metaphysic.ai, until its dissolution into DNEG's Brahma.ai.
Portfolio site: martinanderson.ai
Contact: martin@martinanderson.ai

Discover More