ChatGPT

Recent research and insights on coding-agent harnesses (late 2025 – Feb 2026)

1 Anthropic / Claude Code

finding	sources
Long-running agent harness – Anthropic engineers explain that long-running agents must split work across multiple context windows. Their harness uses two agents: an *initializer* to set up the environment/ logs and a *coding agent* to perform incremental work and leave clear artefacts such as `claude-progress.txt`. The post notes that compaction alone isn't enough; agents must leave the environment in a clean state, mark failing features and avoid trying to one-shot tasks [1]. By iterating and leaving signals for the next call, the harness maintains state across sessions and prevents premature termination [2] [3].	Anthropic blog "Effective harnesses for long-running agents" (Nov 26 2025)
Claude Agent SDK & filesystem-based context – Niels Rogge's summary of the AI Engineer Summit notes that the Claude Agent SDK exposes the same harness used in Claude Code. It treats the filesystem as a dynamic memory: skills, tool descriptions, conversations and terminal outputs are stored as files. Agents only load these files when needed, preserving context space. Anthropic's "skills" are markdown files with names and short descriptions; the agent uses a tool to search the filesystem for the full content, enabling "dynamic context discovery" and dramatically reducing token usage [4] [5]. The SDK encourages a three-step loop—gather context, take action and verify work—and emphasises verifying results through linting, compiling or other tests [6].	ML6 recap "Inside the Claude Agents SDK" (Feb 2026)
Agent skills & dynamic context discovery – Cursor's January 2026 case study (summarised by ZenML) details how Anthropic's Agent Skills standard and the Claude harness influence Cursor's design. Skills are YAML/markdown files containing instructions and executables; only their names and brief descriptions are included in the prompt, and agents use grep/semantic search to pull the full skill file when needed [7]. This dynamic retrieval approach, combined with writing long tool outputs and chat histories to files and selectively loading tools, reduced token usage by ~47 % while improving response quality [8] [9].	Cursor blog "Dynamic Context Discovery for Production Coding Agents" (Jan 2026)

finding	sources
Best practices for coding with agents – Cursor's own blog defines an agent harness as instructions + tools + user messages. They emphasise starting tasks with a plan (using Plan Mode), designing the harness per model, and starting a new conversation when context becomes noisy [10] [11] . They also encourage using static rules and skills to persist instructions across sessions.	Cursor blog "Best practices for coding with agents" (Jan 2026)

2 Cognition / Devin

finding	sources
Cascade agent harness & SWE-1.5 model – Cognition built the frontier-sized model **SWE-1.5** for software engineering. They co-developed the model and its agent harness using a *Cascade* architecture. The harness integrates RL training, tool calling and prompts, and the team emphasises that different harnesses have a larger impact on code-generation performance than model changes alone [12] .	Cognition blog "Introducing SWE-1.5" (Oct 29 2025)
SWE-grep and fast context retrieval – To address the speed–intelligence trade-off, Cognition released **SWE-grep**, a context-retrieval agent that uses reinforcement learning and parallel *grep* operations. A subagent called "Fast Context" performs code search in parallel, returning relevant files quickly and reducing context pollution [13] . This harness design shows how subagents and specialized tools can mitigate context-window limitations.	Cognition blog "Introducing SWE-grep and SWE-grep-mini" (Oct 16 2025)
Agent Trace – context graphs – Cognition proposed *Agent Trace*, an open specification for capturing a "context graph" of code changes. They argue that Git commits only show line diffs; context graphs link conversations, reasoning and code edits to create a living record of decisions. The harness attaches this context to code changes and allows both engineers and agents to recover reasoning later [14] [15] . The concept, supported by multiple companies, positions context as a new resource for AI coding.	Cognition blog "Agent Trace: Capturing the Context Graph of Code" (Jan 29 2026)
Devin Review & Auto-fix – Cognition's harness integrates **Devin Review**, an AI-powered code-review agent. The blog notes that code review—not generation—is the bottleneck; the harness allows a reviewer to leave comments, and Devin automatically fixes mechanical issues and integrates the fixes back into the pull request, closing the agent loop [16] . This cycle of write→review→auto-fix reduces human workload.	Cognition blog "Closing the Agent Loop: Devin Auto-fixes Review Comments" (Feb 10 2026)

finding	sources
Devin Review for quality assurance – A prior post, "Devin Review: AI to Stop Slop," emphasises that quality-of-life improvements—organizing PRs, interactive chat and bug detection—are now the bottleneck in code-assist systems [17]. The harness surfaces potential bugs and suggestions for improvements rather than just generating code.	Cognition blog "Devin Review: AI to Stop Slop" (Jan 21 2026)

3 OpenAI / Codex / Composer

finding	sources
Harness engineering & agent-first development – OpenAI's engineers describe building an entire product using Codex agents. Instead of writing code directly, engineers design the environment, specify intent and build feedback loops; agents generate code, evaluation harnesses and review comments [18]. They emphasise *harness engineering*: scaffolding, feedback loops, tool wrappers and control systems that maintain code consistency. The post lists milestones where agents validate state, reproduce bugs, implement fixes and merge changes with minimal human intervention [19]. It also suggests encoding "golden principles" and running continuous clean-up agents to counter drift [20].	OpenAI blog "Harness engineering: leveraging Codex in an agent-first world" (Feb 11 2026)
Composer 1.5 and self-summarisation – Cursor's release notes and blog posts note that their Composer 1.5 model (built on RL from OpenAI's base models) can *self-summarize* when context runs out. Self-summarisation is trained via RL; the model produces a useful summary when the context window is full and continues exploring. This ability, combined with improved RL scaling, allows the model to tackle long-running coding tasks with fewer context resets [21].	Cursor/Releasebot update (Feb 9 2026)

4 LangChain / LangGraph / Deep Agents

finding	sources
Deep Agents harness – LangChain's **Deep Agents** library builds an agent harness on LangGraph. It provides planning tools, a virtual file system for context management, subagent spawning and long-term memory [22]. Built-in tools like `ls`, `read_file`, `write_file` and `edit_file` allow agents to manage files as a form of persistent memory. Subagents isolate context and handle parallel tasks, while logs and state are stored on disk for retrieval.	LangChain documentation "Deep Agents overview" (Aug 2025)

finding	sources
Skills for deep agents – LangChain's blog introduces *skills* as YAML packages containing instructions and code. Skills are only loaded when required; the harness reveals the name/description and only loads full content when the agent decides to use the skill. This progressive disclosure reduces token usage and cognitive load [23].	LangChain blog "Using skills with Deep Agents" (Nov 25 2025)
Inside the Claude Agents SDK – A recap of the AI Engineer Summit highlights that LangChain founder Harrison Chase views harnesses and context engineering as equally important as models. In long-horizon agents, traces and memory become the source of truth, and future innovations will focus on persistent memory and asynchronous agent management [24].	Sequoia capital / ML6 summary (Oct 2025)
Unified agent calculus – A Dev.to article proposes an "agent calculus" that models an agent as an LLM plus a harness. All inputs (tools, skills, memory, user inputs, tool results) are treated as **entities** that flow through the harness. The harness manages two phases: loading entities into the limited context and executing actions to produce new entities. This formalism unifies disparate concepts like skills, tools, memory, sub-agents and dynamic context loading [25] [26].	Dev.to "Agent Calculus: A Unified Framework for AI Agent Design" (Jan 15 2026)

5 Other research & commentary

finding	sources
Recursive Language Models & context folding – Prime Intellect advocates the **Recursive Language Model (RLM)** as a way to manage extremely long contexts. Instead of loading all data into the context window, the model uses a persistent Python REPL to inspect and transform inputs and can spawn sub-LLMs to handle parts of the task. This approach allows *context folding*—branching into sub-tasks and returning only a summary to the main context—minimising context rot [27] [28]. The authors argue that training models to manage their own context via reinforcement learning will enable tasks spanning weeks or months [28].	Prime Intellect blog "Recursive Language Models: the paradigm of 2026" (Jan 1 2026)

finding	sources
ARES RL infrastructure – Martian's **ARES** is an open-source infrastructure for training coding agents with *online reinforcement learning*. It treats the LLM as the agent (policy) and the harness as the environment. Observations are LLM requests and actions are LLM responses; the harness (environment) handles planning, tool routing and context management. This separation allows researchers to swap harnesses without changing the training loop, focus RL on the model policy and run massively parallel rollouts [29]. ARES is asynchronous (asyncio-native) and includes tens of thousands of standardized tasks, enabling fast evaluation of harnesses and models [30].	Martian blog "ARES: Open-Source Infrastructure for Online RL on Coding Agents" (Jan 30 2026)
Guides to choosing a harness – Lemma's guide argues there is no single standard harness. Different frameworks specialise for different needs: Vercel AI SDK is a lightweight starting point; the Claude Agent SDK excels at coding agents needing file system access; LangGraph offers flexible but complex orchestration. The guide advises choosing based on your workflow rather than hype [31] [32].	Lemma blog "How to choose an agent harness in 2026" (Feb 2 2026)
Dynamic context discovery – Cursor's dynamic context discovery pattern emphasises writing large tool outputs, chat histories and terminal sessions to files and allowing the agent to fetch them when needed. This reduces token usage and prevents context bloat [8]. Techniques include converting long tool responses to files, storing chat history for lossless retrieval, supporting the Agent Skills standard, selective loading of MCP tools (reducing tokens by 46.9 %) and using terminal session files [33].	Cursor blog via ZenML summarisation (Jan 2026)
Importance of the harness – Independent commentators such as Philipp Schmid argue that an agent harness is like an operating system for an LLM; it manages context, tools and state. Benchmarks focusing on single-turn performance fail to capture reliability over long tasks. The article calls harness design essential for validating progress, improving user experience and enabling iterative improvement [34] [35]. Zed's blog similarly notes that LLMs automate typing, not thinking, and developers must remain in the loop with careful prompting and harnesses [36].	Blogs by Philipp Schmid and Zed (Jan 2026)
Agentic context engineering – A foundation capital article emphasises storing decision traces as *context graphs* to link each agent decision to its provenance. These context graphs become a new system of record and empower both humans and AI to recover context and rationale [37].	Foundation Capital "Context graphs: a new resource for AI" (2025)

finding	sources
Multi-agent architectures – Manus's **Wide Research** documentation describes using parallel sub-agents to overcome context-window limitations. Each agent receives a fresh context and processes its own subset of the problem, with a parent agent synthesizing the results. This prevents quality degradation over long tasks and ensures consistent depth of analysis [38] .	Manus documentation "Wide research" (2025)
Subagent harnesses in Hightouch – Hightouch's long-running agent harness separates planning and execution, uses dynamic subagents to isolate context and buffers context to files for later reference. Smaller models are used as subagents to offload tasks, improving performance [39] [40] .	Hightouch blog "How Hightouch built their long-running agent harness" (Jan 20 2026)
Agent harness selection & fragmentation – Lemma and other blogs note that the ecosystem is fragmenting rather than converging. Some frameworks optimise for short, tool-calling loops while others support multi-agent hierarchies or durable long-running tasks. Teams must match the harness to their problem instead of waiting for a universal standard [41] .	Lemma blog (Feb 2 2026)
Observations from practitioners – Posts by developers (e.g., "Eight more months of agents" on crawshaw.io) highlight that harnesses have lagged behind model improvements. Some early agents can still outperform current harnesses, and much innovation remains to be done [42] .	David Crawshaw's blog (Feb 8 2026)
Formal analysis & entity abstractions – The agent calculus formalism emphasises modelling the harness as the manager of limited context and world interactions. Inputs (entities) can have different loading strategies (static/dynamic, full/summary/reference), and the harness controls when and how they enter the LLM's context [43] [44] .	Dev.to "Agent Calculus" (Jan 2026)

6 Key insights and recommendations for building coding-agent harnesses

- **Harness design matters as much as model choice.** Several labs report that the same model performs significantly better when paired with a sophisticated harness [19] [45] . Invest in designing harnesses that manage context, tools and verification rather than relying solely on model upgrades.
- **Treat context as a scarce resource.** Many harnesses convert large tool outputs and chat histories into files and use dynamic context discovery to pull only what is needed [8] [4] . Strategies include summarization with history backups, file-based tool descriptions and selective loading of external tools.
- **Use subagents and skills to handle complexity.** Harnesses often spawn subagents for context isolation (SWE-grep, Wide Research, Hightouch) [13] [38] . Skills and dynamic modules allow agents to load capabilities on demand without overwhelming the prompt [23] [7] .

- **Plan, act and verify.** Effective harnesses encourage a planning stage, iterative execution and explicit verification of outputs. Anthropic's harness emphasises gather→act→verify [6], and OpenAI's harness lists milestones for autonomy and bug reproduction [19]. Developers should remain in the loop to review and guide agents [36].
- **Explore RL-first and context-folding approaches.** Emerging research suggests training models to manage their own context (Recursive Language Models) and using online RL frameworks like ARES to co-optimize models and harnesses [29] [28].

---

[1] [2] [3] Effective harnesses for long-running agents \ Anthropic
https://www.anthropic.com/engineering/effective-harnesses-for-long-running-agents

[4] [5] [6] [45] Inside the Claude Agents SDK: Lessons from the AI Engineer Summit
https://www.ml6.eu/en/blog/inside-the-claude-agents-sdk-lessons-from-the-ai-engineer-summit

[7] [8] [9] [33] Cursor: Dynamic Context Discovery for Production Coding Agents - ZenML LLMOps Database
https://www.zenml.io/llmops-database/dynamic-context-discovery-for-production-coding-agents

[10] [11] Best practices for coding with agents · Cursor
https://cursor.com/blog/agent-best-practices

[12] Cognition | Introducing SWE-1.5: Our Fast Agent Model
https://cognition.ai/blog/swe-1-5

[13] Cognition | Introducing SWE-grep and SWE-grep-mini: RL for Multi-Turn, Fast Context Retrieval
https://cognition.ai/blog/swe-grep

[14] [15] Cognition | Agent Trace: Capturing the Context Graph of Code
https://cognition.ai/blog/agent-trace

[16] Cognition | Closing the Agent Loop: Devin Autofixes Review Comments
https://cognition.ai/blog/closing-the-agent-loop-devin-autofixes-review-comments

[17] Cognition | Devin Review: AI to Stop Slop
https://cognition.ai/blog/devin-review

[18] [19] [20] Harness engineering: leveraging Codex in an agent-first world | OpenAI
https://openai.com/index/harness-engineering/

[21] Cursor Release Notes - February 2026 Latest Updates - Releasebot
https://releasebot.io/updates/cursor

[22] Deep Agents overview - Docs by LangChain
https://docs.langchain.com/oss/python/deepagents/overview

[23] Using skills with Deep Agents
https://blog.langchain.com/using-skills-with-deep-agents/

[24] LangChain's Harrison Chase: Context Engineering Long-Horizon Agents | Sequoia Capital
https://sequoiacap.com/podcast/context-engineering-our-way-to-long-horizon-agents-langchains-harrison-chase/

[25] [26] [43] [44] Agent Calculus: A Unified Framework for AI Agent Design - DEV Community
https://dev.to/herrington_darkholme/agent-calculus-a-unified-framework-for-ai-agent-design-32d8

27 28 Recursive Language Models: the paradigm of 2026
https://www.primeintellect.ai/blog/rlm

29 30 ARES: Open-Source Infrastructure for Online RL on Coding Agents
https://withmartian.com/post/ares-open-source-infrastructure-for-online-rl-on-coding-agents

31 32 41 How to Choose an Agent Harness in 2026 | Lemma Blog
https://www.uselemma.ai/blog/how-to-choose-an-agent-harness-in-2026

34 35 The importance of Agent Harness in 2026
https://www.philschmid.de/agent-harness-2026

36 On Programming with Agents — Zed's Blog
https://zed.dev/blog/on-programming-with-agents

37 AI’s trillion-dollar opportunity: Context graphs - Foundation Capital
https://foundationcapital.com/context-graphs-ais-trillion-dollar-opportunity/

38 Wide Research - Manus Documentation
https://manus.im/docs/features/wide-research

39 40 How Hightouch built their long-running agent harness | Amplify Partners
https://www.amplifypartners.com/blog-posts/how-hightouch-built-their-long-running-agent-harness

42 crawshaw - 2026-02-08
https://crawshaw.io/blog/eight-more-months-of-agents