

AI高性能化に伴うエージェント・ハーネス不要論の真相と次世代アーキテクチャへの転換

Gemini 4 Argon

議論の背景と核心: なぜ今「ハーネス不要論」が浮上したのか

2026年10月下旬、AIエンジニアリングおよび機械学習研究のコミュニティにおいて、自律型AIエージェントの設計思想を根底から揺るがす議論が急速に注目を集めている¹。従来、大規模言語モデル(LLM)を実用的な自律エージェントとして稼働させるためには、モデルの周囲に精巧なソフトウェア制御層、すなわち「エージェント・ハーネス (Agent Harness)」あるいは「スキャフォールディング (足場)」を構築することが不可欠とされてきた¹。ハーネスとは、本質的にステートレスなテキスト予測器に過ぎないLLMに対し、システムプロンプトや行動指針の付与、ツール群の定義と実行、コンテキスト履歴の管理、反復的な推論ループの制御、そしてセッション状態の永続化を提供する外部ソフトウェア機構の総称である²。Claude Code、Codex CLI、OpenCode、Cursor、OpenHands、SWE-agentといった主要なコーディング支援システムはすべてこのハーネスとしての性質を帯びており、長らく「モデルの選択と同等以上にハーネスの設計がベンチマークのスコアを左右する」という認識が業界の定説となっていた¹。

しかしながら、2026年9月から10月にかけて発表された一連の大規模な実証研究と、最前線の開発者による知見の共有を契機として、「基盤モデル (バックボーン LLM) の高性能化に伴い、人間が作り込んだ複雑なハーネスは不要になるばかりか、むしろAIの自律的な探索を阻害する足枷になっているのではないか」というパラダイムシフトが表面化した¹。かつての脆弱なモデルでは、人間が思考のステップを細分化して指示を与えたり、用途別に厳密に定義された専用ツール群を用意したりすることで、推論の脱線を防ぐ必要があった¹。ところが、最新のフロンティアモデルが事後学習 (Post-training) や強化学習を通じて高度なツール利用能力と自己修正能力をモデル内部の重み (Weights) として獲得した結果、人間が良かれと思って設けた過剰な誘導フレームワークが、モデル本来の柔軟な問題解決能力を制約する「取り外し忘れた補助輪」として機能し始めているのである¹。

注目を集めている映像コンテンツ『【転換点】AI高性能化でハーネスは不要に？研究が暴く真実』は、この「ハーネス不要論」の真偽を検証するため、2026年9月に相次いで公開された3本の重要論文と、業界のキープレイヤーであるJedaiah Katz氏 (元Cursor、現aceXAI) の知見を統合し、ハーネスが辿るべき進化の方向性を体系化したものである¹。これらの研究群の原典データ (arXivプレプリントおよび関連技術文献) を徹底的に照合・解剖し、映像内の音声認識による表記揺れ (モデル名や人名の誤変換等) を正確に補正すると、AIエージェント設計の現在地と今後の技術的展望が浮き彫りになる¹。

検証対象の情報要素	映像トランスクリプト内の表記・言及内容	原典調査に基づく正確な学術・技術情報	議論における核心的意義
-----------	---------------------	--------------------	-------------

論文1(2026年9月30日)	Alejandro Hernández-Canoらのチームによる自律型機械学習エンジニアリング研究 ¹	Kirill Brilliantov, Alejandro Hernández-Cano, Emmanuel Abbé(EPFL / Apple Machine Learning Research)『How Much of a Harness Does a Strong Agent Need for Autonomous ML Engineering?』(arXiv:2609.40303) ⁵	同一時間予算・同一フロンティアモデル条件下では、複雑なマルチエージェント・ハーネスは最小構成の単一セッションエージェント(Malena)に対して統計的な優位性を持たないことを実証 ⁵ 。
論文2(2026年9月26日)	10種類のモデル比較、nano-harnessesが「3.7- model」で+7.37点、「DEPV4Pro」で+6.21点改善 ¹	Haichuan Huら(ISE Research)『Beyond the Model: Demystifying Harness Effects in Software Engineering Agents』(arXiv:2609.32459)。モデル名は「Qwen3.7-Max」(+7.37ポイント)および「DeepSeek-V4-Pro」(+6.21ポイント) ⁴	定型的なバグ修正ではモデル高性能化に伴いハーネスの恩恵が逡減する一方、リポジトリ全体のコード生成等の高自律性タスクでは、軽量モジュラー型「NanoHarness」が強力なモデルの能力増幅器として機能することを実証 ⁴ 。
論文3(2026年9月17日)	4つのモデルにわたり176の設定を比較し、知能補助と実行基盤の2種類に分類 ¹	Run-Ze Fanら(UMass Amherst / Emory / UNC Charlotte / Zoom)『An Empirical Study of Harness Design for Coding Agents』(arXiv:2609.20804)。Nemotron-3(30B/120B/550B)とMistral-Medium-3.5-128Bの4モデル×2ベ	強力なモデルにおいて、計画機能は「精度向上」から「コスト削減」へと役割を変え、ツール空間は細かい定義より「Bashのみ」が効率的であり、コンテキスト管理はメモリ制約による失敗防止に不可欠であることを解明 ⁶ 。

		ンチマークで計176 構成を検証 ⁶	
業界識者の見解	元Cursor・現aceXAI の「J. Di-A-Cats」お よび「Jedaiah Katz」 氏による講演と予測 ¹	Jedaiah Katz氏（元 Cursor、現aceXAI） によるエージェント進 化論（前半の「J. Di-A-Cats」は「 Jedaiah Katz」の音 声認識上の表記揺 れ） ¹	モデルが「知能の補 助輪」を自身の能力 として吸収した現在、 ハーネスは知能補完 ツールから「未知の 領域へ踏み出す乗り 物（実行基盤）」へと 変貌し、6ヶ月以内に タスクの90%がPC外 で実行されると予測 ¹ 。

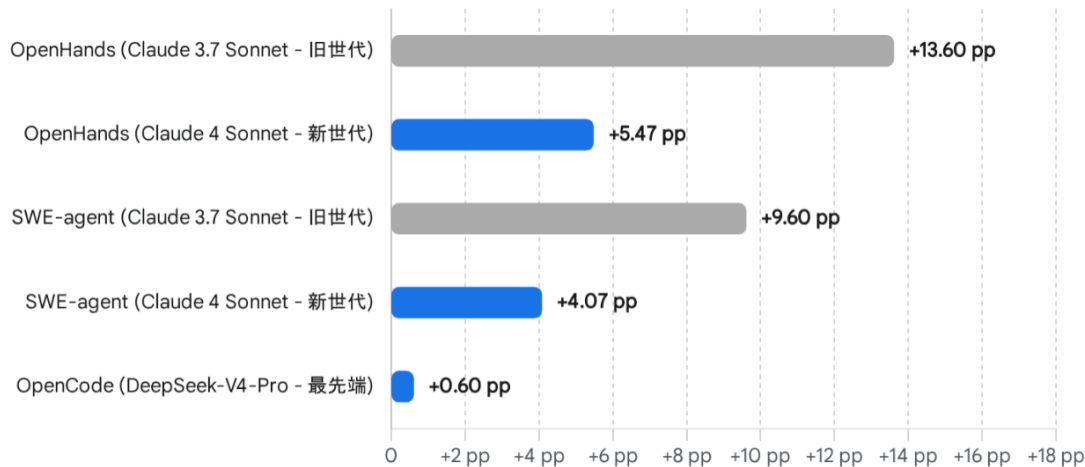
2026年9月の三大実証研究が暴いた「ハーネスの有効性」の境界線

「AIが賢くなればハーネスは不要になるのか」という問いに対する科学的な回答は、単一のイエス・ノーではなく、モデルの基礎能力、対象タスクの開放性（Open-endedness）、そして計算・コンテキスト予算の3変数が織りなす境界条件の中に存在する¹。2026年9月に発表された3つの大規模アブレーション研究（構成要素を一つずつ着脱して効果を分離する実験）は、従来のリーダーボード競争が見落としていた「モデルの進化」と「ハーネスの寄与」の交絡を解き明かし、定型的なバグ修正における恩恵の逡減、高自律性リポジトリ開発における能力増幅、および狭小コンテキスト環境下での壊滅的失敗の防止という3つの異なる挙動を定量的に実証した⁴。

ハーネスの性能向上効果はタスクの自律性とコンテキスト制約に強く依存する

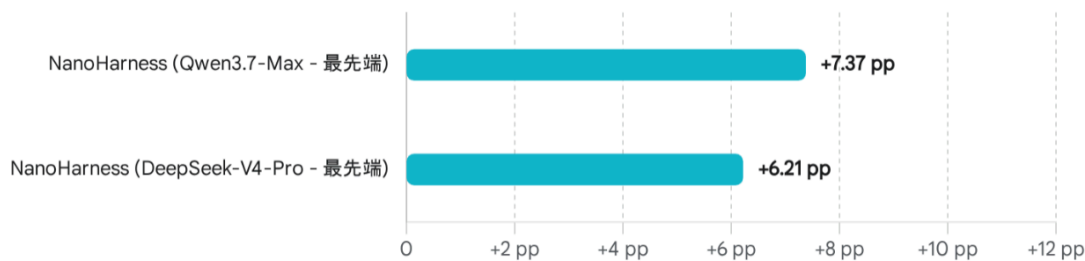
● 1. 定型バグ修正タスク（SWE-bench）：モデル高性能化でハーネス恩恵は縮小

■ 旧世代 ■ 新世代・最先端



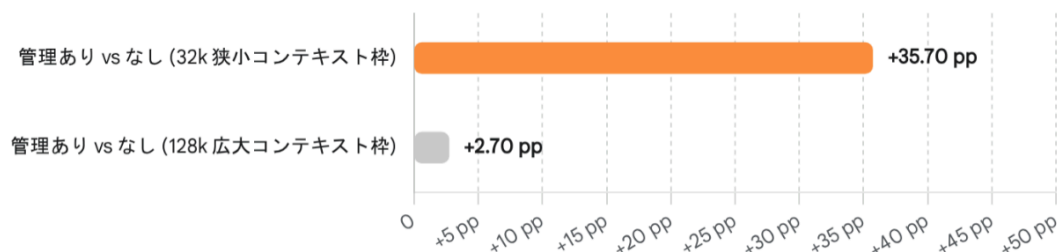
● 2. リポジトリ全体生成タスク（ProgramBench）：高自律タスクでは最先端でも有効

■ NanoHarness



● 3. コンテキスト管理機構：メモリ制約時の壊滅的失敗を防止（成功率ギャップ）

■ 32k 狭小枠 ■ 128k 広大枠



すべての数値はベースライン（軽量ハーネスまたはコンテキスト未管理状態）に対する成功率・スコアの改善幅（パーセンテージポイント：pp）を示す。上段は定型的なバグ修正タスク（SWE-bench Verified / Pro）におけるモデル高性能化に伴うハーネス恩恵の逓減、中段はリポジトリ全体のコード生成タスク（ProgramBench）においてNanoHarnessが最先端モデルにもたらす大幅な性能増幅、下段はコンテキスト長制約（32k対128k）に応じたコンテキスト管理機構の失敗防止効果を表している。

Data sources: [arXiv:2609.32459](https://arxiv.org/abs/2609.32459), [Parallax Research \(AI Agents in September 2026\)](#), [ResearchGate \(Demystifying Harness Effects\)](#), [ResearchGate \(CodeAgent\)](#).

論文1: 自律型機械学習エンジニアリングにおける複雑ハーネスの冗長性(EPFL・Apple共同研究)

2026年9月30日、スイス連邦工科大学ローザンヌ校(EPFL)およびApple Machine Learning Researchの研究チーム(Kirill Brilliantov、Alejandro Hernández-Cano、Emmanuel Abbé)は、『How Much of a Harness Does a Strong Agent Need for Autonomous ML Engineering?』(arXiv:2609.40303)を発表した¹。自律型機械学習エンジニアリング(MLE)の分野では、Kaggle等の実課題をベースにした標準テストベッド「MLE-bench」上で数々のエージェントがスコアを競ってきたが、従来の研究発表では「バックボーンとなるLLM自体の世代交代」と「ハーネス側のオーケストレーション機構の複雑化」が同時に行われており、純粋にハーネス設計がどれだけ性能に寄与しているのかが不明瞭であった⁵。また、ソフトウェア工学分野の先行研究(Sghaier et al., 2026)においても、SWE-bench向けにリリースされた35種類のハーネスを固定バックボーン上で比較した結果、統計的に有意な改善が見られないという兆候が報告されており、MLE領域での厳密な検証が求められていた⁵。

この問題に対し、Hernández-Canoらの研究チームは、計算資源、実行時間予算(Time budget)、およびバックボーンのフロンティアLLMを完全に同一に揃えた統制実験を実施した¹。実験では、単一のチャット補完を最小作業単位とする旧世代パラダイムの「MLEvolve」や、複数のエージェントセッションを階層的あるいはピア・ネットワーク状にオーケストレーションする最新鋭の複雑ハーネス「Arbor」「AiScientist」と対比させるため、「Malena」と名付けられた最小構成(ミニマル)の単一セッション・ハーネスを構築した⁵。Malenaに与えられたのは、ファイルの読み書き(read/write)、Bashコマンドの実行環境、そして提出ファイルをレジストリに登録する最小限のツール(submission tool)のみであり、探索や知識ベースの管理、エラーからの回復はすべて単一セッション内のモデルの自律性に委ねられた¹。さらに、マルチエージェント協調の有効性を分離検証するため、「委譲(Delegation)」を含む3つの調整介入(Coordination interventions)に関するアブレーション実験も併せて行われた⁵。

実験の結果は、従来の複雑なフレームワーク設計に対する強い警鐘となった¹。同一の時間予算と同一の強力なバックボーンモデルの下では、状態圧縮やパーティショニング、マルチエージェントによる並列探索を組み込んだ最先端のオープンソースMLEハーネス群は、read/writeとBashのみを備えた単一セッションのMalenaに対して、統計的に有意な優位性を一切示さなかったのである¹。近年のフロンティアLLMは、コードの実行、データの検査、エラーへの対応をツール経由で自律的に完遂できるよう事後学習されており、「LLMは単独では信頼できないため、外部のハーネスが思考や状態を細かく分割・管理しなければならない」という従来の設計前提がすでに陳腐化していることが明らかとなった⁵。同研究は、強力なモデルの周囲に手作業で精巧なハーネスを構築する労力は、現在のMLEベンチマークにおいては投資対効果が極めて低いと結論付けている⁵。ただし、あらゆるAIやあらゆるタスクにおいてハーネスが無価値になったとまでは断定されておらず、あくまで強力なコーディングエージェント環境における過剰なオーケストレーション層の冗長性が指摘されている点に留意が必要である¹。

論文2: タスク性質とモデル能力の相互作用およびNanoHarnessの実証(ISE Research)

Hernández-Canoらの研究が「単一課題の機械学習コンペティション」における複雑ハーネスの限界を示したのに対し、その4日前の2026年9月26日にISE ResearchのHaichuan Hu、QuanJun Zhang

、Liang Xiaoらの研究チームが発表した『Beyond the Model: Demystifying Harness Effects in Software Engineering Agents』(arXiv:2609.32459)は、ソフトウェア工学(SE)のライフサイクル全体にわたるタスクの性質の違いがハーネスの有効性をどう変化させるかを徹底的に解明した¹。同研究ではまず、1ステップあたり原則1回のBashツール呼び出しに制限された軽量ハーネス「mini-SWE-agent」と、複雑な製品レベルのワークフローを持つ「OpenCode」の2つを代表的なハーネスとして選定した⁴。そして、オープンウェイトの二大モデルファミリーである「Qwen」(Qwen3-MaxからQwen3.7-Maxまで)および「DeepSeek」(DeepSeek-R1、V3、V3.1、V3.2、V4-Pro)から計10種類のモデルを用意し、性格の異なる3つのベンチマーク、すなわち定型的な 이슈修正を評価する「SWE-bench Pro」(300インスタンスのランダムサンプル)、リポジトリレベルのゼロからのコード生成を評価する「ProgramBench」、および「GitTaskBench」上で交差評価を行った¹。また、SWE-bench Verifiedのリーダーボード全体のメタ分析(ハーネス間の性能差レンジが+19.4ポイント、モデル間の性能差レンジが+22.8ポイント)も併せて実施された⁴。

この大規模実験から導き出された第一の重要な発見は、「定型的なバグ修正(Issue repair)においては、モデルの能力が向上するにつれて複雑なハーネスがもたらす限界利益(Marginal benefit)が明確に縮小する」という法則である¹。SWE-bench Verifiedの同一モデル比較において、Claude 3.7 Sonnetからより強力なClaude 4 Sonnetへとモデルが進化すると、mini-SWE-agentに対するOpenHandsの性能上乗せ幅は+13.60パーセンテージポイント(pp)から+5.47 ppへと急減し、SWE-agentの上乗せ幅も+9.60 ppから+4.07 ppへと半減以下になった¹³。さらに興味深いことに、SWE-bench Proにおけるモデルファミリー別のスケーリング挙動には明確な構造差が確認された¹³。Qwenファミリーではモデル能力の向上に伴ってハーネス間の差が滑らかに縮小したのに対し、DeepSeekファミリーでは非単調な「閾値効果(Threshold effect)」が観測されたのである¹³。具体的には、ツール活用能力が適合しない初期のDeepSeek-R1では複雑なOpenCodeの恩恵をほとんど引き出せず、中堅モデルのDeepSeek-V3およびV3.1でOpenCodeによる探索ステップ数とプロンプトトークン数が急増して大きな性能向上が現れたものの、最先端のDeepSeek-V4-Proに達すると、軽量なmini-SWE-agent単体でもモデルが厳選された少数のステップで的確に課題を解決できるようになり、最終スコアはOpenCodeの34.3に対してmini-SWE-agentが33.7(差分わずか+0.6ポイント)とほぼ完全に収束した¹³。

しかし、第二の重要な発見として、タスクが「高い自律性を要するリポジトリ全体のオープンエンドな開発(ProgramBenchやGitTaskBench)」に移行すると、この関係性は劇的に逆転することが判明した¹。複数モジュールにわたるファイル群の検査、新規作成、反復的なビルド検証を伴う広大なタスク空間においては、どれほど優れた推論力を持つモデルであっても、俯瞰的な道標(Signposts)なしに生の探索(Raw exploration)を行えば迷走しやすい¹。Huらは、どのハーネス要素が真に有効かを特定するため、mini-SWE-agentをベースに5つの代表的なモジュールをプラグアンドプレイ方式で段階的に追加できる軽量モジュラー型ハーネス「NanoHarness」を独自に構築した⁴。

NanoHarnessの構成モジュール	実験における具体的な設計仕様・パラメータ	リポジトリ生成タスク(ProgramBench)における効果と挙動の分析
ツール・レジストリ(Tool	構造化されたリポジトリ操作・	生の探索をタスクに整合した

Registry)	編集・検索ツールの明示的な登録と公開 ⁴	構造化インタラクションへと変換し、安定した性能向上に寄与 ⁴ 。
コンテキスト圧縮 (Context Compression)	350Kトークン予算を設定し、圧縮発動時には直近20メッセージのみを保持して過去の履歴を要約・削減 ⁴	リポジトリ生成タスクにおいては、複数ファイル間の微細な実装文脈やインターフェース仕様が失われるため、かえって性能を悪化させる要因となることが判明 ⁴ 。
明示的プランニング (Explicit Planning)	最大10個のToDoアイテムに制限した進捗追跡と段階的な計画管理モジュール ¹³	長期的な作業における中間目標の逸脱を防ぎ、プロンプトの肥大化を抑えつつ着実なスコア改善に貢献 ⁴ 。
サブエージェント (Subagents)	最大50ラウンドに制限。汎用 (Generic) サブエージェントと、挙動分析・差分テスト・提出レビューに特化した専用プロファイルの2系統を検証 ¹³	役割が曖昧な汎用サブエージェントは調整オーバーヘッドにより性能を損なう一方、検証やレビューに特化した専用サブエージェントは有効に機能 ⁴ 。
遅延読み込みスキル (Lazy Skills)	手続き的知識 (Terminal Workflow、Minimal Implementation、Build Diagnostics、Test Triage、CLI Fuzzing、Submission Reviewの6種) を初期プロンプトに詰め込まず、必要時にのみオンデマンドでロード ⁴	初期プロンプトのトークン圧迫 (Prompt overhead) を回避しながら、ビルド診断やエッジケースのファジング等の高度な定型手順を再利用可能にし、高い有効性を発揮 ⁴ 。

ProgramBenchの全データセットを用いたアブレーション実験の結果、有効なモジュールを組み合わせたNanoHarnessは、mini-SWE-agentと比較して、最先端モデルである「Qwen3.7-Max」(映像内では「3.7- model」と言及)において+7.37ポイント、「DeepSeek-V4-Pro」(映像内では「DEPV4Pro」と言及)において+6.21ポイントという顕著なスコア向上を達成し、プロンプトの肥大化を最小限に抑えながら重厚な製品レベルハーネスとの性能差の大部分を埋めることに成功した¹。すなわち、ハーネスの真の価値は「むやみに足場を増やしてトークンを消費すること」にあるのではなく、「リポジトリ内でのエージェントの探索と検証の規律 (Regulation) を整えること」にあると結論付けられる⁴。

論文3: 176通りのアブレーション実験によるコンポーネント別の機能分化 (UMass・Zoom等)

さらに時間を遡る2026年9月17日、マサチューセッツ大学アマーست校 (UMass Amherst)、エモリー

大学、ノースカロライナ大学シャーロット校、およびZoom Video Communicationsの研究チーム(Run-Ze Fan, Zihao Zhang, Hamed Zamani, Xiaoyang Wangら)は、『An Empirical Study of Harness Design for Coding Agents』(arXiv:2609.20804)を発表した¹。従来の評価がハーネスを単一のモノリシック(一枚岩)なシステムとして比較していた欠点を克服するため、同研究はLangGraphとHarbor(コンテナおよび検証器を管理する基盤)上に構築された単一のReActループを固定し、4つのモデル(NVIDIAのNemotron-3ファミリーの30B、120B、550Bの3サイズ、および異なるモデル系統であるMistral-Medium-3.5-128B)と2つの長期コーディングベンチマーク(SWE-Bench Verifiedの500タスク、Terminal-Bench 2.1の89タスク)を用いて、合計**176**通り(モデル×ベンチマークあたり22設定)に及ぶ網羅的な実験を行った¹。

この研究において極めて重要なのは、ワークスペースへのアクセス制御ガード、編集後の構文診断(Post-edit diagnostics)、スタック検知(Stuck detection)といった「安全かつ確実に動作するための基盤コンポーネント」をベースラインとして全実験で固定した上で、知能や行動に直接介入する3つの主要要素——「プランニング(計画)」「アクション空間(ツール定義)」「コンテキスト管理」——を個別に変化させた点にある¹。対象の映像コンテンツは、この知見を「知能を補う仕組み(思考の補助輪)」と「安全に動かすための実行基盤(ワークスペースの整理)」という2つの類型に体系化している¹。

ハーネスの機能的二極化:「知能の補助輪」の消滅と「実行基盤」の台頭

Fanら(arXiv:2609.20804)の176構成の実験データと、同時期の関連研究を統合すると、AIモデルの高性能化に伴って「どの機能が不要になり、どの機能が形を変えて生き残るのか」という機能的二極化のメカニズムが明確に浮かび上がる¹。

モデルの「補助輪の飲み込み」と計画・ツール定義の変質

第一に、アクション空間(Action space)の設計において、人間が細かく定義した事前定義ツール群(Predefined tools)と、シンプルな「Bashのみ(Bash-only)」のインターフェースの優劣は、モデルのBash習熟度によって完全に逆転する¹。Bashの扱いに長けていない小規模・旧世代モデルにおいては、ファイル検索や部分置換などを個別のスキーマとして切り出した事前定義ツールが、コマンド構文エラーを防ぐ「補助輪(Training wheels)」として機能し、タスク成功率を引き上げる⁶。しかし、Bashを自在に操る能力を獲得した強力なモデルにおいては、細かいツール定義はむしろ1ステップあたりの操作を細切れにする制約となる¹。Bashのみの環境を与えられた高性能モデルは、パイプラインやスクリプトを用いて1回の呼び出しに複数の調査・変更操作を柔軟に統合して実行できるため、滑らかに動作するだけでなく、SWE-Bench Verifiedにおいてタスクあたりのコストを最大**53%**削減することが実証された¹。実務家の間でも、多機能なOpenCode環境より、拡張機能やカスタムツールを削ぎ落としてBash中心で動作させたミニマルなハーネス(Piなど)の方が、最新モデル上では高精度かつ安価に動作するという事例が報告されており、「かつてのモデルの弱点を補うために作られた複雑なツール群が、現在のモデルの足を引っ張っている」という事実が裏付けられている¹。

第二に、プランニング(Planning)機能の役割の変質である¹。Fanらの実験では、update_planツールを用いてモデルに明示的かつ永続的な計画(進捗状態)を維持させ、各ターンの入力に注入するスキヤフォールディングの効果が測定された⁶。その結果、30Bクラスの弱いモデルにおいては、プランニング機能はエージェントが途中で目的を見失わずにコード編集段階まで到達するための「精度向上の足場(Accuracy scaffold)」として機能し、追加のトークンコストと引き換えに成功率を高めた⁶。ところが、550Bクラスや最先端の強力なモデルにおいては、プランニングの有無によるタスク成功

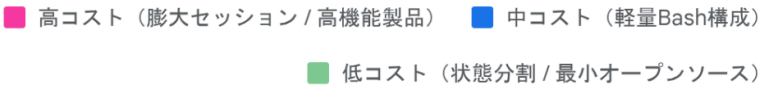
率(精度)の変化は極めて小さくなり、代わりに「コード編集後の冗長な確認・検証ループを抑制することで全体の実行コストを引き下げる効率化の補助具(Efficiency aid)」へとその主目的がシフトしていたのである¹。

第三に、コンテキスト管理(Context management)の不可欠性である¹。どれほどモデルの推論知能が向上しようとも、物理的なコンテキストウィンドウ(メモリ枠)の上限というハードウェア・アーキテクチャ上の制約は消滅しない¹。実験では、コンテキスト予算が32kトークンと狭小な環境下において、コンテキスト管理を行わない場合は78.7%もの試行がコンテキスト溢れ(Window overflow)によって途中終了したのに対し、管理機構を導入した場合はオーバーフロー率が128k環境下で8.7%にまで劇的に抑制され、32k環境下での成功率ギャップは実に35.7ポイント(128k環境下では2.7ポイント)に達した⁶。特に、5つのコンテキスト管理戦略の中でも、ソフト閾値(B_1)を超えた段階で中間領域の巨大なツール出力を安価なルール処理でスタブ(省略記号)に置換(Elision:M1)し、ハード閾値(B_2)を超えた場合にのみ古い履歴のLLM要約(Summarization:M3)を発動する段階的戦略「T4」が最も高いコスト効率を示した⁶。興味深いことに、省略したデータを外部ストアから再取得できる可逆的なリコール機構(recall_eventツール:M2)を用意しても、モデルはそれをほとんど呼び出さず、単純な切り捨て(Elision)単体と比べて精度は向上しなかった⁶。この事実は、ハーネスによるコンテキスト管理において重要なのは、複雑な記憶検索ツールをモデルに操作させることではなく、進行中の作業に不要なノイズ(大量の通過テストログやプログレスバー等)を自動的に掃除してワークスペースを整理することにあると示している¹。

トークン経済学とキャッシュコストが突きつける新たな設計課題

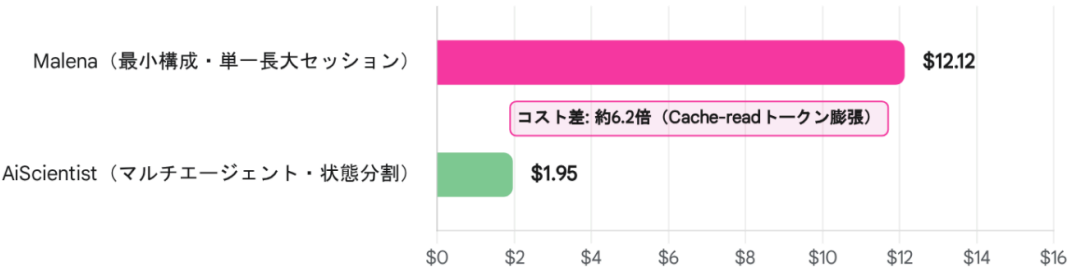
「知能を補うハーネスは不要になりつつある」という議論を深掘りする上で、極めて重要な構造的論点が「精度(Capability)と経済性(Cost)の乖離」である⁵。たとえ最小構成のハーネスと複雑なハーネスの間で最終的なタスク解決率が同等に収束したとしても、セッション設計やコンテキスト保持の方式次第で、実際の運用コストには数倍から6倍以上の格差が発生する⁵。

同等精度でもハーネス設計により最大6倍以上の実行コスト格差が生じる



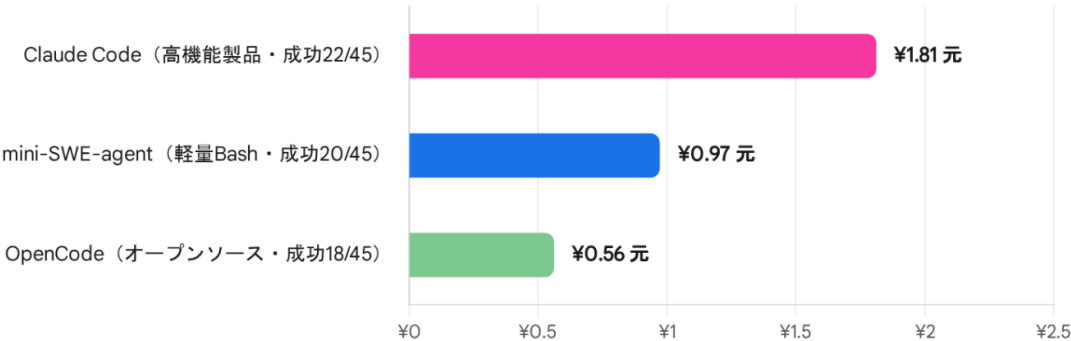
1. 自律型ML開発（MLE-bench / GLM 5.2）：推論コスト比較 単位：米ドル（USD / タスク）

Key Finding: Malenaは入出力トークン数が同等でも、反復キャッシュ読込の肥大化で約6.2倍のコストが発生



2. ソフトウェア課題解決（SWE-bench 45タスク）：平均試行コスト 単位：人民元（CNY / 試行）

Key Finding: 成功数（18～22件）は同水準ながら、プロンプト構造と制御設計により最大3.2倍の格差



上段は自律型MLエンジニアリング（MLE-bench、バックボーン：GLM 5.2）における1タスクあたりのモデル推論コスト（米ドル）を比較したもの。単一セッションで動作する最小構成のMalenaは、入出力トークン総量こそ複雑ハーネスと同等であるものの、長大な履歴の反復読み込みによるキャッシュ読込（Cache-read）トークンが一桁膨張するため、AiScientistの約6.2倍のコストを要する。下段はコーディング課題（45タスク検証）における主要ハーネス3種の1試行あたり平均コスト（人民元）を示しており、同一モデル上でもハーネスのプロンプト構造やステップ制御によって最大3倍以上のコスト差が発生する。

Data sources: [arXiv:2609.40303](#), [XenoSpectrum](#)

論文1(Brilliantov & Hernández-Cano et al., 2026)の詳細なトークン内訳データを精査すると、「ミニ

マルな単一セッション・エージェント(Malena)が複雑なハーネスと同等の精度を出した」という事実の裏に、重大なコスト構造の非対称性が隠されていることがわかる⁵。バックボーンにGLM 5.2を用いた際、Malenaの純粋な新規入力トークン量および出力トークン量は、複雑なハーネスであるArborやAiScientistとほぼ同等であったにもかかわらず、モデル推論の推定ドルコストにおいてMalenaは12.12ドルに達し、AiScientistの1.95ドルと比較して約6.2倍もの費用を消費していた⁵。この極端なコスト膨張の主因は、単一の長いセッションを維持し続けるMalenaにおいて、ターンが進行するたびに過去の全コンテキストを読み直す「キャッシュ読込トークン(Cache-read tokens)」の量が他のハーネスより約1桁(10倍規模)多く発生したためである⁵。

同様に、Claude Code、mini-SWE-agent、OpenCodeの3つのハーネスを同一のQwen3.6およびQwen3.8モデル上で比較した実証研究(YangzeLiu / XenoSpectrum、447タスクおよび45タスク検証)でも、タスク成功率の差が数パーセントの範囲で拮抗する一方、1試行あたりの平均コストはClaude Codeが1.81元(45タスク中22成功)、mini-SWE-agentが0.97元(同20成功)、OpenCodeが0.56元(同18成功)と、最大3倍以上の開きが生じることが確認されている²¹。つまり、モデルの高性能化によって「精度を引き上げるための複雑なワークフロー」は不要になったとしても、「コンテキストを適切に分割・圧縮してキャッシュ読込トークンの爆発を防ぎ、不要な推論ステップを刈り取るためのハーネス機構(T4コンテキスト管理やプランニング、Open SWE routerのようなタスクあたりコストの中央値を64%削減するルーティング層)」は、実運用における経済性を維持するために依然として極めて高い価値を持っているのである⁵。

産業界の視座と今後6ヶ月の展望:「PCの外」へ飛び出すエージェントのOSへ

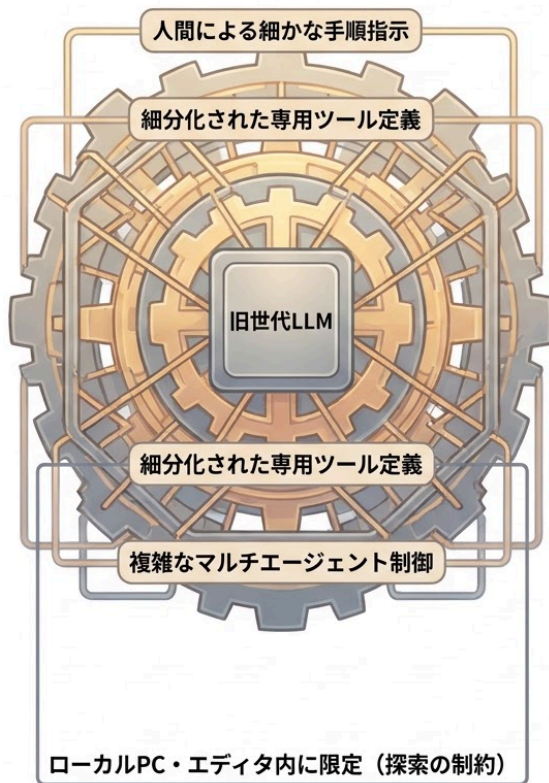
学術研究が暴いた「知能補助から実行基盤へのシフト」という結論は、産業界の最前線でAIコーディングツールを開発してきた実務家の視座とも完全に一致している¹。映像後半で取り上げられているのは、AIコードエディタ「Cursor」の元メンバーであり現在は「aceXAI」に所属するJedaiah Katz氏(映像前半の字幕では音声認識の揺れにより「J. Di-A-Cats」と表記)による講演での洞察である¹。

Jedaiah Katzが提唱する「未知への乗り物」としてのハーネスと脱PC化予測

Jedaiah Katz氏は、現代のAI開発において「ハーネスを付け足してもコーディングの知能自体はたいして上がらない」と明言し、その理由を「モデル自身がかつてハーネス側で担っていたスキルを、自らの基礎能力の一部として吸収してしまったからだ」と指摘している¹。この現象は、業界内で「AIが知能の補助輪を飲み込んだ(Swallowed the training wheels of intelligence)」と形容されている¹。実際、2025年から2026年にかけて、Subconscious社のような事後学習(Post-training)特化企業や大手モデルプロバイダーは、従来フレームワーク層で実装していたReActループ、ツール選択、自己検証、ファイルシステムをメモリとして扱う長期記憶管理(Coding Agent Memory Post-training等)を、強化学習(SHARPO等のセグメント単位のクレジット割り当て手法)を通じてモデル自身の重みに直接蒸留・統合するアプローチを加速させてきた¹⁷。

しかし、Katz氏は「補助輪が不要になったからといって、ハーネスそのものが消滅するわけではない」と強調する¹。同氏によれば、ハーネスの真の価値は「不足している知能を補う道具」から、「エージェントが仕事の次のフロンティア(未知の領域)を開拓するための乗り物(Vehicle)」へと完全に変貌を遂げたのである¹。そしてKatz氏は、極めて重要な未来予測として「今後6ヶ月以内に、エージェントのタスクの90%はローカルPCの外で実行されるようになる」との見通しを提示している¹。

旧世代：知能補完型ハーネス（思考の補助輪）



次世代：自律実行基盤型ハーネス（未知への乗り物）



これまで多くの開発者にとって、コーディングエージェントとは「自分のPC上のVS CodeやCursorの画面内で動き、人間の開発者が横で監視しながらチャットで指示を出すアシスタント」であった¹。しかし、エージェントが真の自律性を獲得するにつれ、タスクの実行場所は個人のPCという閉じた環境から脱却し、クラウド上の無数の仮想マシン（VM）、CI/CDパイプライン、企業の本番サーバー群、セキュリティオペレーションセンター（SOC）、さらにはエッジデバイスが連携するネットワーク空間へと急速に移行している¹。その時、人間がステップバイステップでAIの手を引くことは物理的に不可能となり、同時に「いかに賢いモデルであっても、モデルの知能だけでは絶対に解決できない問題」が立ち現れる¹。それこそが、実世界のシステムに対するファイル操作の境界設定、厳格なアクセス権限（Permission）と認可の管理、長期にわたるセッション状態の維持（State persistence）、そして暴走を防ぐ安全管理（Safety management）という「実行基盤としてのハーネス」の領域である¹。

次世代ハーネスに求められるセキュリティ境界と状態管理アーキテクチャ

エージェントがPCの外へと飛び出し、自律的にサーバーや外部ツールと接続する時代において、ハーネスが担うべき「実行基盤」としての技術的要件は、2026年9月から10月にかけて発表された最新のシステム研究および産業界の動向によって具体化されている²。

第一に、ステートフルな権限管理とセキュリティ境界の構築である¹。エージェントに自由度の高いBash環境を与えることは、生産性とコスト効率を劇的に高める反面、セキュリティ上の重大なリスクを伴う³。2026年9月の研究『Is Sandboxing Sufficient to Contain Rogue Agents?』は、個々のエージェントをサンドボックスに隔離しただけでは、共有通信チャネルや共有ツールを介して汚染が伝播

し、ワーム型のネットワークを形成してしまう危険性を警告している³⁰。また、エージェントの記憶システムを検証した『Revoked but Still Authoritative』では、失効したはずの古いセキュリティポリシーがメモリ内に残存していた場合、既存の5つのエージェント記憶システムのすべてがデフォルトで失効を強制できず、1,620回の試行のうち43.1%が危険なアクションを実行してしまったことが報告されている¹⁷。このような事態を防ぐため、次世代のハーネスには、単なる静的な許可リスト(Allow-list)ではなく、ツール呼び出しのシーケンス全体を監視するステートフルなポリシーエンジン(例: Toolathlonにおいて静的な許可リストの32.4%に対し64.6%、AgentDojoで93.2~95.2%の攻撃を阻止した「Sapient」)や、アクションの影響度に応じて複数の承認プリンシパル(主体)を要求する権限制御機構(「AuthorityLens」)、そして高リスクな判断を人間へ確実にエスカレーションする型付き意思決定ルーティングモデル(「Clef」)の実装が不可欠となっている¹⁷。Microsoftが2026年9月23日にDefenderへ統合した「Integrated Security Operations Center(ISOC)」や、月額15ドルで展開するエージェント管制塔「Agent 365」は、まさに「どのモデルが一番賢いか」ではなく「エージェントが何に触れることを許され、暴走時に誰が止め、監査記録がどこに残るか」を制御するOSレイヤーとしてハーネスを位置づける事業戦略の表れである²⁴。

第二に、長期セッションの永続化と実行環境の一体化(VM内常駐とFork機構)である¹。研究機関Earendilやhranessの分析が指摘するように、本番環境で稼働するハーネスには「指示(Instructions)」「ツール(Tools)」「エージェントループ(Loop)」「プロバイダー間のAPI翻訳層(Translation)」に加えて、5つ目の決定的な責務として「セッション(Session)」の管理が課される²。単一のAPI呼び出しや単一のプロセスを超えて作業を継続するためには、トランスクリプトの保持、チェックポイントの作成、クラッシュからの復旧、さらにはホスト間での実行状態の移送(RustカーネルとBunランタイム間で同じダイジェスト付きマニフェストを一時停止・再開する仕組みなど)がハーネス側に求められる²。さらに、クラウド実行基盤を手掛けるboxd.shなどが提唱する最新アーキテクチャでは、「エージェントの脳(プロセス)を外部に置き、ネットワーク越しにツール呼び出しをシリアルライズしてサンドボックスを操作する」という従来の分離型モデル自体が、推論速度の高速化に伴ってレイテンシのボトルネック(鍵穴を通す通信税)になりつつあると指摘されている²⁹。エージェント自体を仮想マシン(VM)内部のOSプロセスとして直接常駐させれば、ディスク速度でのファイル読み書きやローカルホスト経由でのサービス通信が可能になるだけでなく、環境・エージェントの思考状態・作業途中のファイルを数ミリ秒で丸ごと複製して複数の解決アプローチを並列試行する「Fork(フォーク)」という強力なプリミティブが実現する²⁹。

第三に、外部検証(Verification)による完了判定の厳格化である¹。AIエージェント開発の主役が「人間による細かい手順の指示」から「モデル自身の問題解決能力への委任」へと移行する際、人間の役割は「最小限の実行環境と検証の仕組みをそっと添えること」へと集約される¹。約970行のコードでTerminal-Bench 2.0において59.6%のスコアを記録した最小構成実装(TroyJLorents-GH版 nano-harness)の設計思想が示すように、優れたミニマル・ハーネスは「モデルが言葉で『完了した(Done)』と主張しただけでは決して完了を認めず、直前のツール呼び出しによるテスト成功や検証のエビデンスが伴わない限りループを差し戻す("Done" is earned, not claimed)」という厳格な契約(Contract)を実行層に組み込んでいる³⁰。ウェブ仕様書とクロスブラウザ・テストスイートの検証を通過したコードのみを受理してブラウザエンジンを生成する「Bez」の事例と同様に、思考の過程はモデルの自由な探索に任せつつ、出口の検証ゲートだけをハーネス側で機械的かつ厳格に縛るアプローチこそが、実世界での信頼性を担保する鍵となる¹。

ハーネスの機能領域	旧世代の設計思想 (知能補完・補助輪型)	次世代の設計思想 (自律実行基盤・OS型)	根拠となる実証研究・技術動向
推論・ワークフロー制御	人間が設計したステップバイステップの プロンプト連鎖や複雑なマルチエージェント階層による細かな誘導 ¹	モデル自身の自律的推論に委任。計画ツール(update_plan)は精度向上ではなく冗長な確認を省くコスト削減用に限定活用 ¹	Brilliantov & Hernández-Cano et al. (arXiv:2609.40303) / Fan et al. (arXiv:2609.20804) ⁵
ツール・アクション空間	ファイル閲覧や検索等を細分化した多数の専用ツールスキーマ(Predefined tools)の提供 ¹	永続的な「Bash」と最小限のファイル編集(read/edit)に集約し、モデルによる複合コマンド実行でコストを最大53%削減。必要に応じて遅延スキル(Lazy Skills)をオンデマンド読込 ⁶	Fan et al. (arXiv:2609.20804) / Hu et al. (arXiv:2609.32459) ⁶
コンテキスト・記憶管理	無制限な履歴の蓄積、または全履歴に対する頻繁なLLM要約と複雑な再検索(Recall)ツールの付与 ⁵	段階的戦略(T4)による中間ツール出力のルールベース省略(Elision)でコンテキスト溢れとキャッシュ読込コスト膨張を防止。ただしリポジトリ全体生成時は過度な圧縮を避ける ⁴	Fan et al. (arXiv:2609.20804) / Hu et al. (arXiv:2609.32459) / Brilliantov et al. (arXiv:2609.40303) ⁴
実行環境・状態永続化	ローカルPCのエディタ内での一時的なチャットセッション(メモリ内保持・外部からのシリアル化通信) ¹	タスクの90%がPC外(クラウドVM・サーバー・外部デバイス)へ移行。VM内常駐プロセス化、高速な状態Fork、クラッシュ復旧とホスト間移送が可能なセッション管理 ¹	Jedaiah Katz予測 / Earendil・hraness分析 / boxd.shアーキテクチャ ¹

セキュリティ・完了検証	エディタ画面上での人間による逐次目視確認と手動承認 ¹	ワークスペース境界ガード、ツール呼び出し列のステートフルなポリシー制御（Sapient）、権限昇格ルーティング（Clef）、テスト通過エビデンスに基づく厳格な完了受理ゲート ⁸	Sapient / AuthorityLens / Clef / Microsoft Agent 365 ¹⁷
-------------	--	---	--

結論：AIエージェント開発における人間とハーネスの新たな役割分担

「AIの高性能化によってハーネスは不要になるのか」という問いに対し、2026年秋の最先端研究群と産業界の実践が導き出した答えは極めて明快である¹。モデルの知能が未熟だった時代に、推論の弱点を補うために人間が張り巡らせた「思考の補助輪としてのハーネス（過剰な手順指示、細分化された専用ツール群、硬直的なマルチエージェント分割）」は、フロンティアモデルがそのスキルを自身の重みへと飲み込んだことで、確実にその歴史的役割を終えつつある¹。定型的なバグ修正や単一課題の機械学習エンジニアリングにおいては、そうした複雑な足場を取り払い、シンプルな Bash 環境とファイル操作のみを与えてモデル本来の探索能力を解放する方が、開発労力の面でも実行効率の面でも合理的である¹。

しかし、それはハーネスという存在の終焉ではなく、むしろ「実世界の未知の領域へエージェントを送り出すための実行基盤（OS・乗り物）」としての新たな始まりを意味している¹。リポジトリ全体を見渡すような高自律・長期間のオープンエンドな開発タスクにおいては、探索の規律を整える軽量な道標（NanoHarnessにおけるツールレジストリ、明示的計画、専用検証サブエージェント、遅延読み込みスキル）が最先端モデルの能力をさらに増幅させる¹。さらに、今後6ヶ月でエージェントの作業の90%がローカルPCの外へと拡大していく中、コンテキスト溢れやキャッシュ読込コストの爆発を防ぐワークスペースの整理機構（T4戦略等のコンテキスト管理）、ファイル操作やネットワーク越しの権限を制御するステートフルなセキュリティ境界、長期セッションの状態永続化、そして成果物の正しさを担保する自動検証メカニズムは、どれほどモデルの知能が進化しようとも外部基盤として不可欠であり続ける¹。

これからのAIエージェント開発において、エンジニアの役割は「AIの手足を縛って細かく思考手順を教え込むこと」から、「AIの知能を信頼して思考の足場を撤去し、代わりにAIが安全かつ自由に力を発揮できる最小限の実行環境と検証の境界線を設計すること」へと決定的にシフトしている¹。新しい世代のモデルが登場するたびに、既存のハーネスから不要になった「知能の補助輪」を勇気を持って削ぎ落とし、浮いたリソースを「PC外の広大な実世界と安全に接続するための実行基盤」の強化へと振り向けることこそが、この転換期における最も的確なアーキテクチャ戦略である¹。

引用文献

1. transcript (54).txt
2. What is an agent harness? - Hraness,

- <https://hraness.com/writing/what-is-an-agent-harness>
3. It's not the model, it's the harness | Arjun Mahishi,
<https://arjunmahishi.com/posts/its-not-the-model-its-the-harness/>
 4. Demystifying Harness Effects in Software Engineering Agents,
https://www.researchgate.net/publication/414925514_Beyond_the_Model_Demystifying_Harness_Effects_in_Software_Engineering_Agents
 5. How Much of a Harness Does a Strong Agent Need for Autonomous,
<https://arxiv.org/pdf/2609.40303>
 6. An Empirical Study of Harness Design for Coding Agents - arXiv,
<https://arxiv.org/html/2609.20804v1>
 7. The coding agent harness paper finally ran component ablations,
<https://dev.to/reidmarlow/the-coding-agent-harness-paper-finally-ran-component-ablations-1n39>
 8. An Empirical Study of Harness Design for Coding Agents - arXiv,
<https://arxiv.org/pdf/2609.20804>
 9. [2609.40303] How Much of a Harness Does a Strong Agent Need for,
<https://arxiv.org/abs/2609.40303>
 10. How Much of a Harness Does a Strong Agent Need for Autonomous,
<https://machinelearning.apple.com/research/harness-autonomous-ml-engineering>
 11. How Much of a Harness Does a Strong Agent Need for Autonomous,
<https://yadneshsalvi.com/newsletter/story/cffce8d944db881c>
 12. CodeAgent: Enhancing Code Generation with Tool-Integrated Agent,
https://www.researchgate.net/publication/384210491_CodeAgent_Enhancing_Code_Generation_with_Tool-Integrated_Agent_Systems_for_Real-World_Repo-level_Coding_Challenges
 13. Demystifying Harness Effects in Software Engineering Agents - arXiv,
<https://arxiv.org/html/2609.32459v1>
 14. Demystifying Harness Effects in Software Engineering Agents - arXiv,
<https://arxiv.org/pdf/2609.32459>
 15. [2609.20804] An Empirical Study of Harness Design for Coding Agents,
<https://arxiv.org/abs/2609.20804>
 16. Harness Overhead Benchmark: 11 Coding Agents Measured (2026),
<https://testdino.com/blog/harness-overhead-benchmark>
 17. AI Agents in September 2026: Instruments, Control and ... - Parallax,
<https://parallax.ink/research/en/ai-agents-in-september-2026>
 18. Alejandro Hernández-Cano's research works - ResearchGate,
<https://www.researchgate.net/scientific-contributions/Alejandro-Hernandez-Cano-2371018825>
 19. How Much of a Harness Does a Strong Agent Need for... - daily.dev,
<https://daily.dev/posts/how-much-of-a-harness-does-a-strong-agent-need-for-autonomous-ml-engineering--2f9zlu5zq>
 20. Context Mode, RTK, Headroom, Caveman and Claude-Mem,
<https://ai.dosa.dev/blog/context-compression-tools-for-ai-coding-agents>
 21. Study Compares Claude Code, OpenCode and mini-SWE-agent,
<https://xenospectrum.com/en/coding-agent-harness-cost-comparison/>

22. Summary of AI papers from the last 24 hours as of 9/20. Read daily,
https://note.com/tender_peony902/n/nb58989c36b47?hl=en
23. Paper page - An Empirical Study of Harness Design for Coding Agents,
<https://huggingface.co/papers/2609.20804>
24. [Analysis] Microsoft's Real AI Bet Is the Harness - Inseok,
<https://inseok.medium.com/analysis-microsofts-real-ai-bet-is-the-harness-ffbc01c7ea72>
25. What if your AI Harness could improve itself? | by Guido Russo,
<https://medium.com/@guidorusso95/what-if-your-ai-harness-could-improve-itself-3c06a3c2d270>
26. Agents Without Training Wheels: The Model Agent - Subconscious.dev,
<https://www.subconscious.dev/blog/agents-without-training-wheels>
27. For those using Cursor right now: is it actually worth it? From 0 to 10,
https://www.reddit.com/r/cursor/comments/1vi79ib/for_those_using_cursor_right_now_is_it_actually/
28. Aetox — the best harness for the AI you already use. A local ... - GitHub,
<https://github.com/Mikedev115/Aetox-Harness>
29. The agent lives inside the machine - boxd,
<https://boxd.sh/blog/the-agent-lives-inside-the-machine/>
30. Build agents around memory, proof, and controlled decisions - o16g,
<https://o16g.com/updates/2026-10-02-0002/>
31. GitHub - TroyJLorents-GH/nano-harness: Smallest readable coding,
<https://github.com/TroyJLorents-GH/nano-harness>
32. joacod/nano-harness: Personal, opinionated and local first AI Harness,
<https://github.com/joacod/nano-harness>
33. Anthropic's Harness Design for Long-Running AI Development (with,
<https://gist.github.com/0xK8oX/0292e8da944ceb9226a7c2500b47124e>