

AI高性能化で「ハーネス」は不要になるのか——動画『【転換点】AI高性能化でハーネスは不要に？研究が暴く真実』徹底検証

Executive Summary

本動画の中心命題は、かなり正確に言い直すと次のようになります。

「強いモデルでは、推論能力を補うために人間が手作業で組んだ複雑な“知能補助型ハーネス”の限界効用は下がっている。しかし、実行環境・コンテキスト管理・権限・検証・安全・永続状態を担う“システム型ハーネス”は不要になるどころか、エージェントが現実世界で自律的に動くほど重要になる。」

これは、2026年9月に相次いだ複数の実証研究とかなり整合しています。とりわけ動画の中心的根拠は、Kirill Brilliantov、Alejandro Hernández-Cano、Emmanuel Abbé による How Much of a Harness Does a Strong Agent Need for Autonomous ML Engineering? です。同研究は、同一のフロンティアLLM、同一ハードウェア、同一時間予算で比較すると、MLE (Machine Learning Engineering) タスクでは、検索木・複数エージェント・専用オーケストレーションを重ねた既存ハーネスが、単一の強力なコーディングエージェント「Malena」を統計的に上回らないという結果を報告しています。著者自身も、「**coding-agent post-training が、従来ハーネスが手作業で与えていた scaffolding を代替している可能性**」を仮説として提示しています。 ¹

しかし、この結果を「ハーネスはもう要らない」と一般化するのは誤りです。わずか4日前の別研究 Beyond the Model は、10モデル、3種類のソフトウェア工学ベンチマークで、SWE型の定型的なバグ修正では強いモデルほど複雑ハーネスの利益が縮小する一方、**ProgramBenchやGitTaskBenchのようなリポジトリ全体を扱う開放的タスクでは、強いモデルのほうが高度なハーネスを活用できる場合がある**と示しました。NanoHarnessはQwen3.7-Maxで+7.37ポイント、DeepSeek-V4-Proで+6.21ポイント改善しています。 ²

さらに、2026年10月の Finding the Right Fit は66のモデル×ハーネス構成を評価し、**モデル順位そのものがハーネスによって逆転すること**を示しました。Terminal-Bench 4では、ClaudeがGPTをOpenHands上で7.94ポイント上回る一方、PIハーネスでは逆に30.16ポイント下回りました。つまり性能は単純な

$$\text{Performance} / f(\text{Model}) =$$

ではなく、

$$\text{Performance} = f(\text{Model}, \text{Harness}, \text{Task}, \text{Budget}, \text{Environment})$$

と、相互作用項を含めて考える必要があります。 ³

安全性については、結論はさらに明確です。2026年のHarnessRiskでは、同じモデルでもハーネスを変えるだけで攻撃成功率が4倍以上変わる例があり、全14構成で何らかの安全上の失敗が確認されました。CapScope研究では、権限制御をモデルのプロンプトではなくホスト側ハーネスで強制すると、注入された悪意ある動作の実行が33-47/75回から3/75回まで低下しました。これは、**権限境界・サンドボックス・認可・監査をモデルの「賢さ」に吸収させてはいけない**ことを示す非常に重要な反証です。 ⁴

したがって、2026年10月26日時点の最も妥当な結論は次の通りです。

「ハーネス不要」は否。だが「ハーネスの役割が変わる」は是。
消えていく可能性が高いのは、強いモデルに「まず考えろ→計画を作れ→サブエージェントを呼べ→別のエージェントが批評しろ」と細かく教える**知能補助的 scaffolding**です。一方で残る、あるいは重要性が増すのは、**execution runtime、tool interface、context/state management、permissions、sandboxing、verification、logging、recovery、cost control**です。これは動画自身が後半で事実上認めている結論でもあります。fileciteurlturn0file0 5

動画の逐次要約と主張の構造

提供された動画トランスクリプトはScripsy.appによる自動書き起こしであり、「DEPV4Pro」「J. Di-A-Cats」「aceXAI」など明らかな音声認識誤りが含まれます。以下では逐語転載ではなく、**発言順を保持したタイムスタンプ付き逐次要約**として整理し、後段で原典との照合を行います。fileciteurlturn0file0

時刻	動画の内容・主張	原典照合後の評価
00:00–00:32	従来は「ハーネス設計でAIのスコアが大幅に上がる」と考えられてきたが、2026年秋になり逆方向の議論が登場したと導入。	妥当。2024年のSWE-agentなどではインターフェース設計が大きな性能差を生んでいた一方、2026年には強いモデルで限界効果が縮小する研究が複数登場。 6
00:32–01:01	強いAIでは、細かいステップ指定や複雑なframeworkが自然な探索を邪魔する場合がある。	条件付きで支持 。強いBash-capableモデルでは細粒度ツールよりBashだけのほうが低コストで同等以上の場合がある。ただし全タスクには一般化できない。 7
01:01–01:40	2026年9月30日、Alejandro Hernández-Canoらが自律MLEを研究。複雑ハーネスとread/write/bash中心のminimal agentを同じ強いLLMで比較。	特定済み。Brilliantov, Hernández-Cano, Abbé, EPFL/Appleの研究。 1
01:40–02:12	同じ時間予算では複雑なハーネスに有意な優位性がなく、モデル自身の能力が主要因だった。ただし「全AIでハーネス不要」とは証明していない。	ほぼ原論文どおり 。原論文も「current MLE benchmarks」に射程を限定し、統計検出力やベンチマーク汚染可能性などをlimitationsで明記。 8
02:12–02:50	9月26日の研究では10モデルを比較。定型的バグ修正ではモデルが強いほどハーネス効果が縮小するが、高自律なrepository-wide taskでは強いモデルでも恩恵が残る。	特定済み。Beyond the Model の主要結論を正確に要約している。 9
02:50–03:19	NanoHarnessが「3.7 model」で+7.37、「DEPV4Pro」で+6.21。	音声認識誤り。正しくは Qwen3.7-Max +7.37ポイント、DeepSeek-V4-Pro +6.21ポイント 。 10
03:19–03:48	「全ハーネスが不要」と一括りにせず、有効な部分と邪魔な部分を分離すべき。9月17日の論文が4モデル×176設定を比較。	特定済み。Fan et al. An Empirical Study of Harness Design for Coding Agents。 11

時刻	動画の内容・主張	原典照合後の評価
03:48– 04:24	ハーネスを「知能を補助する仕組み」と「安全に実行させる基盤」に分けるべき。Bashを使えるモデルには細かなtool定義より単純なcommand environmentが有効な場合がある。	前半は動画の整理、後半はFan et al.の結果と一致。Bash-capableモデルではbash-only interfaceが有効かつ低コスト。 ⁷
04:24– 04:48	強いモデルではplanningは精度向上よりコスト削減に効く。context managementはmemory/context limitによる失敗を防ぐため重要。	原論文の主要結論と一致。 ¹²
04:48– 05:32	元Cursorの「J. Di-A-Cats」が、今は「aceXAI」で、追加ハーネスはcoding intelligenceをほとんど上げなくなったと講演した、と紹介。	人物はJediah Katzと特定できる。AI Engineer 公式スケジュールでは役職は“Engineer, Agent Harness, SpaceX Super Intelligence”。Cursorは2026年8月14日にSpaceXに買収済み。 ¹³
05:32– 05:58	Katzは、それでもハーネスは「次の仕事のフロンティア」を開く手段だと主張。	講演要約と一致。coding-specific tricksはモデルに吸収されても、新しい実行領域を開くruntime、coordination、安全性は残るという主張。 ¹⁴
05:58– 06:25	「6か月以内にagent作業の90%がPC外で実行される」と予測。	Katz本人の予測として確認できるが、実証済みの事実ではない。 ¹⁵
06:25– 06:57	file操作、permission、long-term stateはモデルの知能だけでは解決できない。実世界で動かすには安全管理が必要。	現在のsecurity研究と強く整合。permission enforcementやstate persistenceは典型的なハーネス責務。 ¹⁶
06:57– 07:33	人間の役割は複雑workflowの手設計から、最小限のexecution environmentとverificationを与えることへ移行する。	2026年の実証研究から導ける合理的な設計指針。ただし全領域で証明された普遍則ではない。 ¹⁷
07:33– 07:54	知能を縛る過剰scaffoldingは消えるが、それはハーネス全体の終わりではない。	本動画の最も重要な限定。タイトルの「不要？」より本文ははるかに慎重。 filecite0turn0file0
07:54– 08:28	最適ハーネスはモデル能力・タスク・環境によって変わる。minimal harnessを安全に使うことが今後の課題。	最新研究の総体と最も一致する結論。モデル×ハーネス×タスクの相互作用が重要。 ¹⁸

動画の主張を三段階に分解すると

動画は表面的には「ハーネス不要論」を扱っていますが、実際には三つの異なる命題を混同しないことが重要です。

第一の命題は「強いモデルでは知能補助型scaffoldingの価値が下がる」です。これは2026年のMLEおよびcoding-agent研究でかなり強く支持されています。search tree、planner/executor分離、過剰なmulti-agent orchestration、非常に細かいtool abstractionなどは、モデル自身が探索・debug・tool useを行える場合、追加利益が小さかったり逆効果になったりします。 ¹⁹

第二の命題は「minimal interfaceのほうが常に良い」です。これは支持されません。repository-wide generationや高自律タスクでは、structured tools、task-specific subagents、外部task stateなどが依然として大きく効くことがあります。 ²⁰

第三の命題は「実行ハーネスそのものが不要」です。これは現在の証拠からは明確に否定されます。ツールの実行、権限、sandbox、state、audit、recoveryをモデルの重みだけで実現することはできず、現実世界への作用を仲介する外部システムは物理的・論理的に必要なからです。Harness-Benchもハーネスをcontext、tools、state、constraints、permissions、tracing、recoveryを管理するシステム層と定義しています。 ²¹

引用研究・人物の特定と原典

動画の三つの中核論文

9月30日の論文は、Kirill Brilliantov、Alejandro Hernández-Cano、Emmanuel Abbéによる How Much of a Harness Does a Strong Agent Need for Autonomous ML Engineering?、arXiv:2609.40303です。所属はEPFLとAppleで、論文には一部研究がAppleでのインターン中に行われたことも記されています。動画では「Alejandro Hernández-Cano's team」と表現されていますが、原典上は3名の共著研究です。 ¹

研究対象はMLE-benchを中心とする自律Machine Learning Engineeringです。OpenCode v1.15.6を土台とし、各agentにA100 80GB GPU×1、12 CPU cores、144GB RAMを与え、backbone、hardware、wall-clock budgetを揃えて比較しています。性能指標はhuman leaderboardに対する**mean percentile**、メダル獲得率、self-selected submissionとoracle selectionの差などで、macro averageには95% bootstrap confidence intervalを用いています。 ⁸

この論文のminimal baselineが**Malena**です。従来のMLEハーネスのようにsearch treeや専門subagentを多数組み込まず、単一のcoding-agent sessionにfilesystem、コード実行、submission/system/jobs等の少数のprimitiveを渡します。研究では、最も大きな改善は「chat completionを複雑にオーケストレーションすること」ではなく、**モデル自身にファイルと実行環境を直接触らせること**から生じました。 ⁸

一方、論文自身はかなり重要な留保を置いています。主実験のMLE-benchは公開Kaggle competition由来なのでpretraining contaminationの可能性があり、著者はcode similarity検査とNatureBenchで緩和しています。また24時間実験は高コストなためseed数が3-4程度で統計検出力に限界があり、multi-agent coordinationも限定的なablationに留まります。さらに自作Malenaに対するチューニング量の非対称性も潜在的limitationとして明記されています。 ⁸

9月26日の論文はHaichuan Hu、Quanjun Zhang、Shengcheng Yu、Zhifei Chen、Tianyu Luo、Chunrong Fang、Zhenyu Chen、Liang Xiaoによる Beyond the Model: Demystifying Harness Effects in Software Engineering Agents、arXiv:2609.32459です。公開メタデータではNanjing University of Science and Technology、Technical University of Munich、Nanjing Universityの所属が表示されています。 ²²

この研究はmini-SWE-agentとOpenCodeという対照的な二つのハーネスに、Qwen系5モデル+DeepSeek系5モデル、合計10モデルを組み合わせ、SWE-bench Pro、ProgramBench、GitTaskBenchの3領域で比較しました。さらにmini-SWE-agent上にNanoHarnessを構築し、**tool registry**、**context compression**、**explicit planning**、**subagents**、**lazy skills**を個別にablationしています。 ⁹

ここから得られる重要な結果は「複雑さの効用は単調ではない」ということです。SWE-bench Proのように比較的workflowが固定されたissue repairでは、強いモデルほどOpenCodeの追加利益が縮小します。ところがProgramBenchのようなゼロからのrepository generationやGitTaskBenchのような多目的repository operationでは、強いモデルが複雑なハーネスをむしろ上手に利用するケースがあります。NanoHarness全部入りはQwen3.7-Maxを+7.37、DeepSeek-V4-Proを+6.21ポイント改善しました。 ²

9月17日の論文はRun-Ze Fan、Zihao Zhang、Simin Ma、Yebowen Hu、Shouju Wang、Kaiqiang Song、Fei Liu、Hamed Zamani、Xiaoyang Wangによる An Empirical Study of Harness Design for Coding Agents、arXiv:2609.20804です。所属メタデータではUMass Amherst、Emory University、UNC Charlotte、Zoom Video Communications等が確認できます。 ²³

ここではexecution loopを固定し、planning、action space、context managementのみを変え、4モデル、SWE-Bench VerifiedとTerminal-Bench 2.1、計176のmatched settingsを調べています。結論はかなり示唆的で、**context managementはcontext windowが狭いほど重要、planningは弱いモデルではaccuracy scaffoldだが強いモデルでは主にcost saver、predefined toolsはBashが苦手なモデルには効くが、Bashに強いモデルではbash-only interfaceのほうが安価で十分でした。** ¹²

Jediah Katzは誰か

動画の「J. Di-A-Cats」は、**Jediah Katz**の音声認識ミスとほぼ確定できます。KatzはCursorでagent harnessに携わり、2026年4月のCursor公式記事 Continually improving our agent harness をStefan Heuleと執筆しました。そこでは、新しいfrontier modelごとにハーネスをモデルの強み・癖に合わせて調整し、**同じモデルでも専用ハーネス内ではより速く、賢く、効率的になると**明記されています。これは「ハーネスは無意味」という単純化に対する、本人自身による重要な反証です。 ²⁴

2026年8月14日にCursorは正式にSpaceXに買収されました。AI Engineer NYCの公式スケジュールではKatzの所属・役職は **Engineer, Agent Harness, SpaceX Super Intelligence** と表示されているため、動画トランスクリプトの「aceXAI」はSpaceXAI/SpaceX周辺の名称を誤認識したものとするのが自然です。 ²⁵

Katzの講演タイトルは Smarter Models Killed the Harness. Here's What's Left と確認できます。そこでの主張は動画よりニュアンスがあり、**「coding intelligenceを補う部分はモデルに吸収されているが、infrastructure、trust/security、efficiency、次の仕事領域を開くexecution layerとしてハーネスは残る」**というものです。また「6か月以内にagent workの90%がlaptop外で実行される」という発言も確認できますが、これはKatzの将来予測であって実証結果ではありません。 ²⁶

さらにKatz自身の2026年1月のCursor公式記事は、モデルが強くなるにつれて「最初から大量のcontextを詰め込む」のではなく、**必要なcontextをモデル自身に探索させるdynamic context discovery**が有効になったと説明しています。つまり「モデルが強くなるほど少なく与える」という動画の主張には、実運用上の裏付けがあります。ただしその仕組み自体——ファイル化、履歴参照、MCP toolの動的読み込み——はまさにハーネス機能です。 ²⁷

未特定・断定できない引用

動画冒頭の「2026年10月22日に以前紹介した、ハーネス設計でスコアが急上昇する話」がどの論文を指すのかは、この動画のトランスクリプトだけでは一意に特定できません。 ^{filecite} ^{turn0file0}

またKatz講演では、Pi、Codex、Claude Code等の比較研究とApple/EPFL研究に触れたと講演要約から確認できますが、動画中では前者の論文名が示されていないため、どのSWE-bench系研究を直接引用したかについては**未特定**とするのが安全です。 ¹⁴

「ハーネス不要」論の技術的根拠

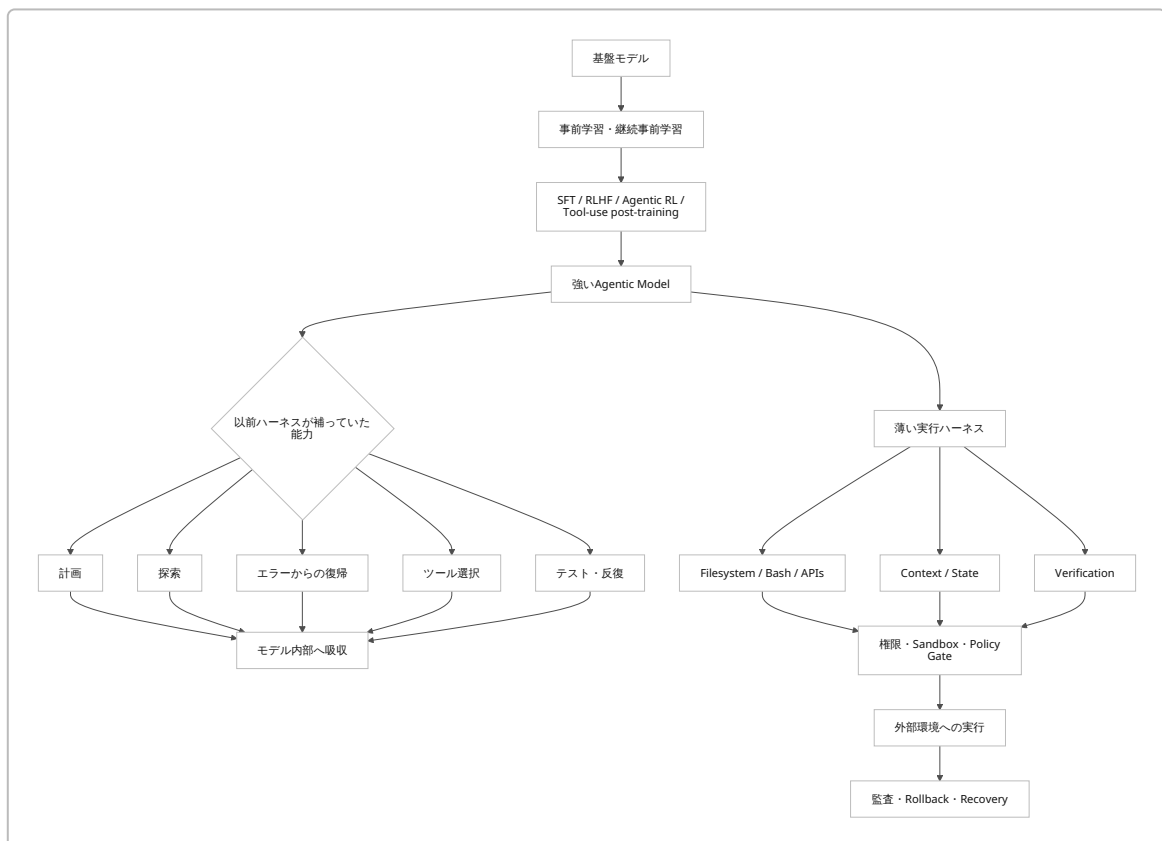
まず「ハーネス」を二種類に分ける必要がある

議論が混乱する最大の理由は、同じ「harness」という語で異なるものを指しているからです。Harness-Benchの定義ではハーネスはcontext、tools、state、constraints、permissions、tracing、recoveryなど、モデルと環境の間を仲介するシステム層全体です。 21

実務的には次の二層に分けると理解しやすくなります。

層	例	強いモデルで消えやすいか
知能補助型 scaffolding	手書きCoT、固定planner、search tree、reviewer agent、細かいstep decomposition、専用retrieval agent	かなり消え得る
実行・統制型 harness	filesystem、shell、tool dispatcher、context/state、sandbox、permission、verification、logging、rollback	消えにくい。 むしろ重要化

この違いを図にすると、動画の議論は次のようになります。



重要なのは、上図で上半分はモデルに吸収できても、下半分はモデル外のシステム責務のままだということです。 28

モデル性能向上はなぜscaffoldingを不要にするのか

第一の要因は単純なbase-model capabilityの向上です。古いモデルでは「探索」「計画」「コード編集」「実行結果の読み取り」「修正」を外側のプログラムが明示的に管理する必要がありました。SWE-agentの2024年研究では、モデル向けに設計されたAgent-Computer InterfaceがSWE-benchで12.5% pass@1、HumanEvalFixで87.7%を達成し、interface design自体が大きな性能要因であることを示していました。 29

ところがMLE研究では、2026年のフロンティアモデルにfilesystemとexecution primitivesを直接与えるだけで、モデル自身がデータを調べ、コードを書き、実行し、エラーを読み、修正するようになっています。論文ではchat型とcoding-agent環境を比較し、後者が最大の性能差を生み、その上にsearch machineryを重ねても有意な追加利益が得られませんでした。 8

検索アルゴリズムも比較されています。Chainは直前の解を改善し続け、Greedyはvalidation score最高のnodeを親に選び、UCB1はexploration/exploitationを調整し、Best-of-Nは独立試行を繰り返します。24時間のmatched budgetで比較するとconfidence intervalは大きく重なり、search topologyそのものの効果は小さいという結果でした。つまり強いagentは、**ハーネスが固定するsearch policyを自分のtrajectory内で実質的に再現している**と解釈できます。 8

データ量・データ効率の進歩

モデル能力向上は単なるparameter増加ではありません。HoffmannらのChinchilla研究は、同じtraining computeでもparameter数だけ増やすのではなくtraining token数を適切に増やすことで、70BのChinchillaが280BのGopherや175BのGPT-3等を広範なタスクで上回ることを示しました。これは「より大きいモデル」より「computeに対して適切な量のデータで学習したモデル」が重要であることを示す代表例です。 30

ただし、**Chinchillaが直接「ハーネスを不要にした」と示したわけではありません**。ここでの因果関係は間接的です。training efficiencyとdata scalingが基盤能力を高め、その後のtool-use post-trainingと組み合わせさせた結果、以前は外側のworkflowとして実装していた処理をモデル自身が行える範囲が広がった、と理解すべきです。MLE論文自身も「raw model sizeよりtool-use fluencyやcoding-agent post-trainingが重要かもしれない」と議論しています。 8

自己教師あり学習によるツール利用の内在化

Toolformerはこの流れを理解するうえで重要な先行研究です。Toolformerは少数のdemonstrationから、モデル自身がAPIを「いつ」「どのように」呼び出すべきかを自己教師あり的に学び、外部ツール使用をモデルのbehaviorに組み込みました。つまり本来ならharness側に「この場合calculatorを呼ぶ」と書いていたルールを、一部はモデル側のpolicyへ移せることを早くから示した研究です。 31

ReActも、reasoning traceとenvironment actionを交互に行うことで、単なる一回の生成ではなく「考える→行動→観察→考え直す」というagent loopの有効性を示しました。ALFWorldやWebShopでは従来のimitation/RLベースラインより大幅に高い成功率を報告しています。かつてはこうしたloop自体がprompt/harness engineeringの対象でしたが、現在は同様のinteraction patternをpost-trainingでモデルへ直接学習させる方向に移っています。 32

RLHF・instruction tuning・Agentic RL

InstructGPTは、人間のdemonstrationによるSFT、preference comparison、RLHFを組み合わせることで、1.3Bモデルでさえ人間評価上175B GPT-3より好まれるケースを示しました。重要なのはparameter数だけでなく、「**人間の指示にどう従うか**」を**post-trainingすること自体がbehaviorを大幅に変える**という点です。

33

これをagentに拡張すると、tool calls、長いtrajectory、error recovery、planning、environment feedbackを含む強化学習が可能になります。2026年のAgent Lightning v1.0は、deploy-time harnessとtrainerを分離してagentic RLを行い、報告例ではQwen3.5-9BのSWE-bench Verified成績を41.8から56.4へ14.6ポイント改善しています。これは「harnessに書いていた振る舞いを、trainingを通してモデルpolicyへ移す」という方向が実際に進んでいる証拠です。 ³⁴

したがって、動画の「モデルが補助輪を飲み込んだ」という比喻には技術的な根拠があります。ただしMLE論文も、**post-trainingがharnessを食べたという説明は実証済みの因果結論ではなくhypothesisだと明記**しています。DeepSeek V4系列でpost-trainingが増えた最終版ほどcoding-agent環境の利益が大きいという示唆的な観察はありますが、training recipeを完全にcontrolledした因果実験ではありません。 ⁸

「単純なBashが勝つ」の本当の意味

Fan et al.の結果は「toolは不要」という意味ではありません。正確には、**Bashを十分理解するモデルに、`read_file`、`grep`、`edit`、`run_test`のような大量の特殊tool schemaを渡すより、標準的なshell環境を渡し、モデル自身にコマンドを組み合わせさせるほうが安価かつ十分な場合がある**ということです。弱いモデルでは逆にpredefined toolsが有利でした。 ⁷

これはsoftware engineeringで特に成立しやすい性質です。Shellは既に非常に表現力が高く、grep、find、sed、git、compiler、test runner等を合成できます。したがって強いモデルにとって「100個の高水準toolを覚える」より「一つの汎用programming environmentを操作する」ほうが自然になる可能性があります。一方、決済・医療・クラウド管理のように許される操作を厳密に制限すべき環境では、同じ設計をそのまま適用するのは危険です。権限境界は表現力ではなく安全上の要請だからです。 ³⁵

コンテキスト管理は「知能」ではなく資源管理になる

Fan et al.ではcontext windowが狭くなるほどcontext managementの効果が大きくなり、主な利益は推論そのものを賢くすることより**context overflowによる失敗を避ける**ことでした。またrule-based elisionを先にを行い、その後必要ならLLM summarizationする方式が効率的でした。 ⁷

Cursorの実運用も同じ方向です。Jediah Katzらは、長時間agentではcontextが巨大になるため、必要な情報だけを動的にロードし、MCP tool definition等も必要時だけ読み込む設計を進めています。これは「強いモデルだからハーネス不要」ではなく、**強いモデルだからこそ、ハーネスは細かな思考誘導ではなくresource managementへ移る**ということです。 ³⁶

反論・限界・安全性

タスク分布が変わると結論も変わる

「強いモデルほどハーネス効果が小さい」という観察は、特定のbenchmark distributionに依存します。Beyond the Model はまさにこの点を示しており、localized issue repairではハーネス差が縮む一方、repository generationや高自律タスクではstructured tool useやtask-specific subagentsが有効でした。 ⁹

2026年10月のFinding the Right Fitはさらに強い反例です。4つのconfigurable harness、5モデル、3ベンチマークにnative Codex/Claude構成を加えた66 configurations、6,204 trajectoriesを調べると、モデル順位がハーネスごとに反転しました。これは「最強モデル+最小ハーネス」が普遍的最適解ではなく、**モデルとハーネスのcompatibility**が独立した設計変数であることを示します。 ³

またSet-shifting Behavioral Testでは、セッション途中で「どのtoolが信頼できるか」を密かに変更すると、agentは少数の慣れたroutineに固着する傾向を示しました。tool presentationの仕方だけでもrouting

behaviorが変化します。これはdistribution shift下では、強いモデルでも「以前うまくいった行動」に固着するという警告です。 ³⁷

Prompt Injectionはモデルの「賢さ」だけでは防げない

InjecAgentは1,054 test cases、17 user tools、62 attacker toolsを用いたindirect prompt injection benchmarkで、ReAct-prompted GPT-4でも24%のケースで攻撃に脆弱だったと報告しました。攻撃命令を強化すると成功率はさらに増加しました。 ³⁸

Agent Security Benchは10シナリオ、400超のtools、13 LLM backbones、約90,000 test casesを用いて、system prompt、user input、tool use、memory retrievalの各段階に脆弱性が存在し、評価した攻撃の最高平均attack success rateは84.30%に達したと報告しています。 ³⁹

AgentDojoも97の現実的user tasksと629 security test casesを使い、email、e-banking、travel booking等のtool-using agentでは、攻撃がなくても失敗があり、indirect prompt injectionへの完全な防御は難しいことを示しています。 ⁴⁰

これらが意味するのは、「モデルが十分賢くなれば、悪意ある外部データと正当なsystem authorityを自動的に見分ける」と期待する設計は危険だということです。

権限はpromptではなく実行層で強制すべき

この点を最も鮮明に示す2026年研究がAuthority Is Not a Stringです。CapScopeは「許可」を自然言語promptに書くのではなく、trusted inputからtask-wide authority ceilingを作り、typed capabilityをhost sideで保持し、各tool callを実行前に検査します。 ³⁵

5 Python tasks × 5 injection surfaces × 4 authorization conditions × 3 trials、合計300 runsで、通常のambient authority/global policyでは注入されたeffectが33-47/75回実行されたのに対し、CapScopeでは3/75でした。task successは68/75で、最も緩いbaselineの72/75と近い水準を維持しています。 ⁴¹

ここには非常に一般的な設計原則があります。

モデルは「行動を提案する」主体にし、行動を「許可する」主体にはしない。

モデルの知能が上がっても、least privilege、capability separation、sandbox、rate limits、transaction limits、approval boundariesは外部でenforceすべきです。CapScope自身も「付与済みcapabilityの悪用」までは防げず、policy designとsandboxingが別途必要だと明記しています。 ³⁵

HarnessRiskが示す「高性能≠安全」

HarnessRiskは、Configuration、Capability Extension、Runtime Operation、State Persistence、Action Control、Incident Recoveryの6段階に128 sandboxed casesを配置し、3 harnesses、6 language models、14 model-harness configurationsを調べました。指標はUtility、Attack Success Rate、Persistence、Detectionです。 ⁴²

結果は重要です。Utilityが75.0-97.6%と高いにもかかわらずASRは12.6-80.9%まで分布しました。同じGLM-5.2でもOpenClawではASR 54.7%、Nanobotでは12.6%で約4.3倍の差です。またrisk detectionが90%を超えていても実際のattack executionが十分抑制されない構成がありました。つまり、

「危険だと理解した」 / 「危険な操作をしなかった」

ということです。 43

したがって「より賢いモデルなら安全判断も任せられる」という推論は成立しません。認識とenforcementは別問題です。

長期状態・メモリは依然として難題

長時間agentでは、モデルの一回のcontext windowに全履歴を保持するのはコストと品質の両面で難しくなります。2026年のLongHorizon-Harnessは、task stateをexecution contextの外に明示的に保持し、Manager-Execute-Audit loopで環境から検証された事実だけをstateへ反映する方式を提案しました。Qwen3.7-PlusをWeaveBenchで51.8%から80.7%、Terminal-Bench 2.1で69.7%から77.2%へ改善し、Claude Opus 4.7でもOSWorld subsetを20.0%から34.3%へ改善したと報告しています。 44

これは動画の「複雑な知能補助を減らす」という主張に対する重要な限定で、**長期タスクではむしろstateをモデル外へexternalizeすることが有効な場合があります。**

2026年のreview Externalization in LLM Agents も、memoryはtemporal state、skillsはprocedural expertise、protocolsはinteraction structureを外部化し、harness engineeringがそれらを統合する層だと整理しています。つまり現在の研究潮流には、「全部モデル内部へ吸収する」方向と「信頼性のために外へ出す」方向の両方が存在し、最適点を探している段階です。 45

コスト・効率も「ハーネス不要」に対する反例

「正答率が変わらない＝ハーネスが無価値」とは限りません。planningが強いモデルのaccuracyをほとんど上げなくても、token/costを下げる場合があります。 7

さらにBetter Harnesses, Smaller Models は逆方向の結果を示しています。7つのbusiness-oriented agent tasks、3つのsmall-model familyで、failure trajectoriesからハーネスを自動適応させると21組中16組で有意に改善し、7組ではSLM-LLM gapをほぼ埋め、最良のSLM構成はfrontier LLM performanceの89.7%をわずか4%のコストで達成しました。つまりハーネスに**task regularity**を移すことで、**小さいモデルを実用化**するという経済的戦略も成立します。 46

したがって「最強モデルを買えばharness engineeringは不要」という判断は、推論コストを無視しています。

倫理・運用上の問題

エージェントがemail送信、deployment、payment、credential操作、データ削除など現実世界の不可逆操作を行うほど、誤作動の影響は単なる「間違った回答」から「実際の損害」に変わります。HarnessRiskがAction Control、Persistence、Incident Recoveryを独立したsafety phaseとして扱うのはこのためです。 42

人間とagentの協働を扱う2025年surveyも、完全自律agentにはhallucination、複雑taskでのreliability、安全・倫理上のriskが残るため、人間のfeedback、control、orchestrationを組み込むhuman-agent systemsを重要な方向性として整理しています。 47

したがって、高リスク領域での「minimal harness」は**minimal safety**を意味してはいけません。

最新研究の横断レビュー

以下は2026年10月26日時点で、この議論を判断するうえで特に重要な研究を「ハーネス不要寄り」「条件依存」「ハーネス必要寄り」を横断して比較したものです。信頼度は論文の真偽を採点したのではなく、この問いに対してどの程度直接的で、どの程度controlledな証拠かという本レポート上の評価です。

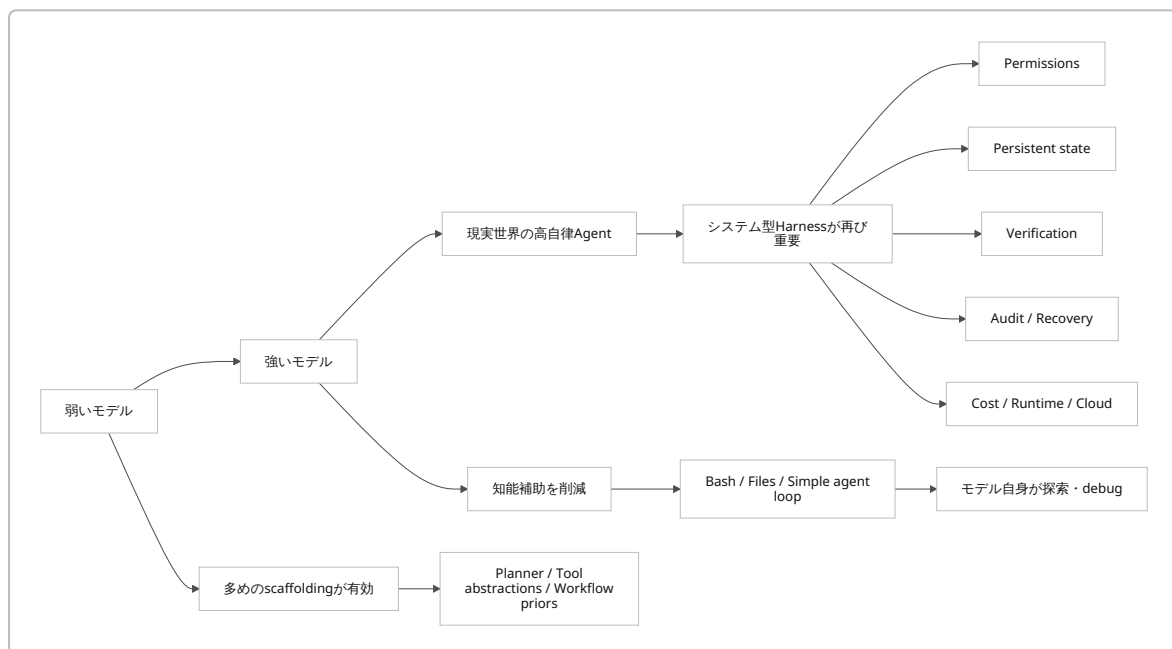
論文	著者・年	中心主張	手法・実験	主要結果	本議論での信頼度 / 限界
How Much of a Harness Does a Strong Agent Need for Autonomous ML Engineering?	Brilliantov, Hernández-Cano, Abbé, 2026	frontier coding agentでは複雑MLE harnessの追加価値が小さい	MLE-bench/ NatureBench、同一 backbone・hardware・time、search/multi-agent ablation	coding-agent environment が最大効果。その上の複雑機構に有意な優位なし	高。最も直接的。ただしMLE中心、preprint、seed数・contamination等に限界。 ¹
Beyond the Model	Hu et al., 2026	harness効果は model×task 依存	10 models × 2 harnesses × 3 SE benchmarks。 NanoHarness component ablation	SWE repairでは強いモデルほど差縮小。 open-ended taskでは強いモデルも恩恵。 +7.37/+6.21pp	高。動画の一般化を直接修正する重要研究。 ⁹
An Empirical Study of Harness Design for Coding Agents	Fan et al., 2026	component ごとの価値がモデル能力・context budgetで変わる	4 models、176 matched settings、SWE-Bench Verified / Terminal-Bench 2.1	planningは弱モデルで accuracy、強モデルで cost。 Bash-capable modelでは bash-only有効	高。 controlled ablation。ただし4モデル・2benchmark。 ¹¹
Finding the Right Fit	Li et al., 2026	普遍的best model/ harnessはなく pairingが重要	66 configurations、4 harnesses×5 models + native agents、3 benchmarks	harnessによりモデル順位が逆転。 Claude vs GPT 差が+7.94から -30.16へ	高。 モデル-ハーネス interactionへの非常に直接的証拠。 preprint。 ³
Harness-Bench	Yao et al., 2026	能力は model単体でなく model-harness構成として評価すべき	106 sandboxed tasks、5,194 trajectories	completion、process、efficiency、failureが harnessで大きく変動	高〜中。 広い workflowを扱うが個別機能の因果分解ではない。 ⁴⁸

論文	著者・年	中心主張	手法・実験	主要結果	本議論での信頼度 / 限界
Better Harnesses, Smaller Models	Yang et al., 2026	harnessへ task structureを移すと小型モデルを安価に使える	7 business tasks × 3 SLM families、自動 harness adaptation	16/21改善、best SLMがLLM性能の89.7%を4%コストで回収	高～中。経済性への強い反例。routine task中心。 ⁴⁶
Harness as a Language / JAZ	Li et al., 2026	少数の強力 primitiveだけでも複雑 workflowを表現できる	<code>invoke</code> のみを核とする minimal framework、StuLife/AppWorld	MemGPT比 +8%かつ半額、ACE比 +4%で低コスト	中～高。「少ない primitive」は支持するが「harnessゼロ」ではない。 ⁴⁹
LongHorizon-Harness	Ma et al., 2026	長期taskでは外部task state + audit が有効	Manager-Execute-Audit、複数benchmark/model	WeaveBench 51.8→80.7 等、大幅改善	高～中。長期taskへの強い反論。新しい preprint。 ⁴⁴
Authority Is Not a String / CapScope	Bouras, Dai, Mechtaev, 2026	security authorityを model contextではなく host側 capabilityで強制すべき	300 runs、prompt injection × authorization 条件	attack execution 33-47/75→3/75	高。ACM SIGPLAN LMPL '26。規模は小さいが security mechanismへの因果性が明確。 ⁵⁰
HarnessRisk	Bai et al., 2026	agent safety は modelではなく model-harness lifecycle全体の性質	128 cases、3 harnesses、6 models、14 configs	ASR 12.6-80.9%、high utilityでも unsafe	高～中。広範な安全評価。sandbox benchmarkで実運用そのものではない。 ⁵¹
InjecAgent	Zhan et al., 2024	tool-using agentsは indirect prompt injectionに弱い	1,054 cases、30 agents	ReAct GPT-4も 24%攻撃成功、強化 attackでさらに悪化	高。ハーネス安全性の古典的反例。モデル世代は 2024。 ³⁸

論文	著者・年	中心主張	手法・実験	主要結果	本議論での信頼度 / 限界
Externalization in LLM Agents	Zhou et al., 2026	agent進化は weightsだけでなく memory/skills/protocols/harnessへの外部化でも進む	systems-level review	parametric capabilityと externalized infrastructureのco-evolutionを整理	中〜高。レビューとして広いが個別因果実験ではない。 45

この比較から浮かぶのは、「ハーネス必要派」と「不要派」の研究が真正面から矛盾しているわけではないという点です。

研究結果はむしろ、次のような**U字型**ではなく**役割分化**を示しています。



つまりモデル性能が上がるほど「どう考えさせるか」のハーネスは薄くなる一方、agentが実世界へ出るほど「何を実行してよいか・何が実際に起きたか」を保証するハーネスは厚くなる、という二方向の変化です。

52

結論と実務者向け推奨・主要ソース

現時点で「ハーネス不要」と言えるか

総合判定は「いいえ」です。

より精密には、

「強力な、tool-use post-training済みモデルを、よく理解された低リスクのcoding/MLEタスクに使う場合、推論能力を補助する複雑なscaffoldingは大幅に削減できる」

まではかなり有力な実証結果があります。 53

しかし、

「agent harness全体を取り除ける」

という命題を支持する証拠はありません。むしろ2026年の研究群は、permissions、state、context、verification、security、recoveryなどを含むハーネスがagent systemの性能と安全性を大きく左右することを示しています。 54

したがって動画タイトルの「ハーネスは不要に？」への私の判定は、

「知能補助としての“重いハーネス”は部分的に不要になりつつある。しかし、実行・安全基盤としてのハーネスは不可欠であり、agentの自律性が高まるほどむしろ重要になる」

です。

Minimal harnessが合理的になる条件

次の条件がほぼ揃う場合には、「複雑なagent frameworkを削り、薄いagent loopへ戻す」戦略にはかなり合理性があります。

モデル側では、tool use・coding・multi-turn recoveryについて十分post-trainingされた強いモデルであること。**タスク側**ではworkflowが比較的既知で、local feedbackが速く、自動評価可能で、Bash/filesystemのような汎用interfaceで表現できること。**環境側**では十分なcontext/time budgetがあり、失敗してもsandbox内でやり直せること。そして**安全側**では外部への不可逆な副作用を与えず、最終成果をtest/verifierで機械的に検証できることが重要です。 19

典型例は、隔離されたrepository内でのコード修正、benchmark上のMLE、sandbox内のデータ解析です。

反対に、payment、credential、deployment、本番DB、メール送信、個人情報、医療判断などを含むタスク、untrusted webpages/documentsを読むタスク、数時間～数日に及ぶlong-horizon work、persistent memory、multi-agent delegation、未知のdistribution shiftを伴う環境では、execution/safety harnessを削る根拠はありません。 55

実務者向け推奨

- ・「featureを足す」より「ablationして消す」を標準にする。Planner、reviewer、subagent、memory、tool wrapperを一つずつ外し、同一model・同一budget・同一taskでaccuracy、cost、latencyを測定するべきです。モデルが強くなった後も古いscaffoldingを残すと、token costや探索制約だけが残る可能性があります。 56
- ・性能は必ず model × harness × task の組で評価する。「GPT系だから」「Claude系だから」とモデル名だけで評価せず、自社harness上でtask-specific evalを行うべきです。2026年研究では同じモデルの順位がハーネスで反転しています。 18

- **モデルには汎用的で表現力の高い実行環境を与え、認可は狭くする。** CodingではBash/filesystemのようなsimple primitiveが有効でも、OS権限まで無制限にすることを意味しません。「interfaceは単純、authorityはleast privilege」が望ましい設計です。 ⁵⁷
- **permission・sandbox・policy enforcementをpromptに任せない。** Tool callはモデルが提案しても、実行許可はhost側capability/policy engineで判定し、high-impact actionには追加承認を置くべきです。 ³⁵
- **長期タスクではcontext全部をモデルに詰め込まない。** Durable state、artifact、test result、decision logをモデル外に保持し、必要時だけcontextへ戻す設計のほうがtoken効率・監査性とも有利です。重要stateはモデルの自己申告ではなくenvironmentから検証して更新するのが安全です。 ⁵⁸
- **成功率だけでなく安全指標を別に測る。** Utility/successが95%でもattack successが高い構成は存在します。ASR、unauthorized side effects、persistence、risk detection、rollback success、permission violationsを独立metricsとして持つべきです。 ⁴³
- **「モデルが賢くなったから安全機構も削れる」と考えない。** intelligence scaffoldingとsecurity boundaryは別物です。前者はモデル内部へ吸収可能でも、後者はモデルが侵害された場合にも働く外部mechanismでなければ意味がありません。 ⁵⁹

本動画をどう評価するべきか

動画はタイトルだけ見るとセンセーショナルですが、本文は意外に慎重です。特に「すべてのハーネスが不要とは言っていない」「taskによって結果が変わる」「context managementやexecution foundationは残る」「権限・state・safetyはモデル知能だけでは解決できない」と繰り返しており、最終的な主張は主要論文の射程から大きく外れていません。 ^{filecite turn0file0}

一方で注意すべきなのは、「モデルが強くなった→ハーネス不要」という因果ストーリーが、研究結果より一段強く表現されている点です。Brilliantovらの研究が直接示したのは、**特定のMLE benchmarks、特定のfrontier backbones、matched budgetで、追加scaffoldingの統計的優位が見えない**ことです。「なぜそうなったか」についてのpost-training説は著者自身がhypothesisと位置づけています。 ⁸

また「モデルが補助輪を飲み込んだ」というJediah Katzのdata-flywheel論は、非常に説得力のある実務者仮説ですが、controlled causal experimentそのものではありません。Katz自身のCursor公式記事も同時に「frontier modelごとにharnessを調整すると同じモデルが速く・賢く・効率的になる」と述べているため、彼の主張を「harness engineering終了」と理解するのは明確な読み違いです。 ⁶⁰

最も正確な見方は、

Agent engineeringの中心が「モデルの弱さを人間のworkflowで補うこと」から、「モデルに自由度を与えつつ、環境・資源・権限・結果を厳密に管理すること」へ移行している

というものです。

これは「ハーネスの死」よりも、**ハーネスのレイヤー分離と役割転換**と呼ぶほうが、2026年10月時点の証拠に忠実です。 ⁶¹

主要ソース

- **Brilliantov, Hernández-Cano, Abbé — How Much of a Harness Does a Strong Agent Need for Autonomous ML Engineering?**。動画の最重要一次資料。EPFL/Apple、arXiv:2609.40303。 ¹
- **Hu et al. — Beyond the Model: Demystifying Harness Effects in Software Engineering Agents**。10モデル×3 benchmark、NanoHarness。arXiv:2609.32459。 ¹⁰
- **Fan et al. — An Empirical Study of Harness Design for Coding Agents**。176 matched settingsによるplanning/action/contextのablation。arXiv:2609.20804。 ¹¹
- **Li et al. — Finding the Right Fit: Model-Harness Interactions across Agent Tasks**。66 configurationsでmodel-harness interactionを直接測定。 ³
- **Yao et al. — Harness-Bench**。106 tasks、5,194 trajectories、model-harness単位での評価を提唱。 ⁶²
- **Bouras, Dai, Mechtaev — Authority Is Not a String**。capability-based authorizationを用いたprompt-injection対策。ACM SIGPLAN LMPL '26。 ⁵⁰
- **Bai et al. — HarnessRisk**。configurationからincident recoveryまでagent harnessのライフサイクル安全性を評価。 ⁴²
- **Cursor公式研究記事** — Jediah Katzらの「エージェントハーネスを継続的に改善する」「動的コンテキスト検出」「長時間のエージェント実行に向けたトークン効率の向上」。動画中のKatz発言を理解するうえで重要な日本語一次資料。 ⁶³
- **AI Engineer公式情報 / Katz講演記録** — Katzの所属および Smarter Models Killed the Harness. Here's What's Left の文脈確認。 ⁶⁴
- **Zhou et al. — Externalization in LLM Agents**。memory、skills、protocols、harnessの外部化を統一的に整理した2026年レビュー。 ⁴⁵

¹ ¹⁷ ⁵³ How Much of a Harness Does a Strong Agent Need for Autonomous ML Engineering?
https://arxiv.org/abs/2609.40303?utm_source=chatgpt.com

² ⁹ ²⁰ Beyond the Model: Demystifying Harness Effects in Software Engineering Agents
<https://arxiv.org/abs/2609.32459>

³ ¹⁸ Finding the Right Fit: Model-Harness Interactions across Agent Tasks
https://arxiv.org/abs/2610.00917?utm_source=chatgpt.com

⁴ ¹⁶ ³⁵ ⁴¹ ⁵⁰ ⁵⁹ Authority Is Not a String: A Capability-Scoped Harness for Prompt-Injection-Resistant Coding Agents
<https://arxiv.org/abs/2609.08371>

⁵ ²¹ ⁴⁸ ⁵⁴ Harness-Bench: Measuring Harness Effects across Models in Realistic Agent Workflows
<https://arxiv.org/abs/2605.27922>

⁶ ²⁹ SWE-agent: Agent-Computer Interfaces Enable Automated Software Engineering
https://arxiv.org/abs/2405.15793?utm_source=chatgpt.com

⁷ ¹² ⁵⁶ ⁵⁷ An Empirical Study of Harness Design for Coding Agents
<https://arxiv.org/abs/2609.20804>

⁸ ¹⁹ ²⁸ ⁵² How Much of a Harness Does a Strong Agent Need for Autonomous ML Engineering?
<https://arxiv.org/abs/2609.40303>

¹⁰ ²² Beyond the Model: Demystifying Harness Effects in Software Engineering Agents
https://arxiv.org/abs/2609.32459?utm_source=chatgpt.com

¹¹ ²³ An Empirical Study of Harness Design for Coding Agents
https://arxiv.org/abs/2609.20804?utm_source=chatgpt.com

- 13 64 Schedule | AI Engineer NYC 2026
https://ai.engineer/nyc/2026/schedule?utm_source=chatgpt.com
- 14 26 Smarter Models Killed the Harness. Here's What's Left | AIE Talks
https://aietalks.com/talks/smarter-models-killed-the-harness-heres-whats-left?utm_source=chatgpt.com
- 15 Smarter Models Killed the Harness. Here's What's Left — Jediah Katz, SpaceXAI — bash TV | QA Bash
https://www.qabash.com/bash-tv/57660?utm_source=chatgpt.com
- 24 60 63 エージェントハーネスを継続的に改善する · Cursor
https://cursor.com/ja/blog/continually-improving-agent-harness?utm_source=chatgpt.com
- 25 Cursorは現在、SpaceXの一員です · Cursor
https://cursor.com/ja/blog/joining-spacex?utm_source=chatgpt.com
- 27 58 動的コンテキスト検出 · Cursor
https://cursor.com/ja/blog/dynamic-context-discovery?utm_source=chatgpt.com
- 30 Training Compute-Optimal Large Language Models
https://arxiv.org/abs/2203.15556?utm_source=chatgpt.com
- 31 Toolformer: Language Models Can Teach Themselves to Use Tools
https://arxiv.org/abs/2302.04761?utm_source=chatgpt.com
- 32 ReAct: Synergizing Reasoning and Acting in Language Models
https://arxiv.org/abs/2210.03629?utm_source=chatgpt.com
- 33 Training language models to follow instructions with human feedback
https://arxiv.org/abs/2203.02155?utm_source=chatgpt.com
- 34 Agent Lightning v1.0: Towards Harnessed Agentic RL
https://arxiv.org/abs/2608.17528?utm_source=chatgpt.com
- 36 長時間のエージェント実行に向けたトークン効率の向上 · Cursor
https://cursor.com/ja/blog/improved-token-efficiency?utm_source=chatgpt.com
- 37 Set-shifting Behavioral Test for Harnessed Agents
https://arxiv.org/abs/2607.13396?utm_source=chatgpt.com
- 38 InjecAgent: Benchmarking Indirect Prompt Injections in Tool-Integrated Large Language Model Agents
https://arxiv.org/abs/2403.02691?utm_source=chatgpt.com
- 39 Agent Security Bench (ASB): Formalizing and Benchmarking Attacks and Defenses in LLM-based Agents
https://arxiv.org/abs/2410.02644?utm_source=chatgpt.com
- 40 AgentDojo: A Dynamic Environment to Evaluate Prompt Injection Attacks and Defenses for LLM Agents
https://arxiv.org/abs/2406.13352?utm_source=chatgpt.com
- 42 43 51 55 HarnessRisk: A Lifecycle-Oriented Benchmark for Agent Harness Safety
<https://arxiv.org/abs/2608.17597>
- 44 LongHorizon-Harness: Advancing Long-Horizon Agents for Real-World Tasks
https://arxiv.org/abs/2608.01964?utm_source=chatgpt.com
- 45 61 Externalization in LLM Agents: A Unified Review of Memory, Skills, Protocols and Harness Engineering
https://arxiv.org/abs/2604.08224?utm_source=chatgpt.com

46 **Better Harnesses, Smaller Models: Building 90% Cheaper Agents via Automated Harness Adaptation**
https://arxiv.org/abs/2607.08938?utm_source=chatgpt.com

47 **A Survey on Large Language Model based Human-Agent Systems**
https://arxiv.org/abs/2505.00753?utm_source=chatgpt.com

49 **Harness as a Language: A Minimalist Agent Framework With Maximal Expressivity**
https://arxiv.org/abs/2609.26891?utm_source=chatgpt.com

62 **Harness-Bench: Measuring Harness Effects across Models in Realistic Agent Workflows**
https://arxiv.org/abs/2605.27922?utm_source=chatgpt.com