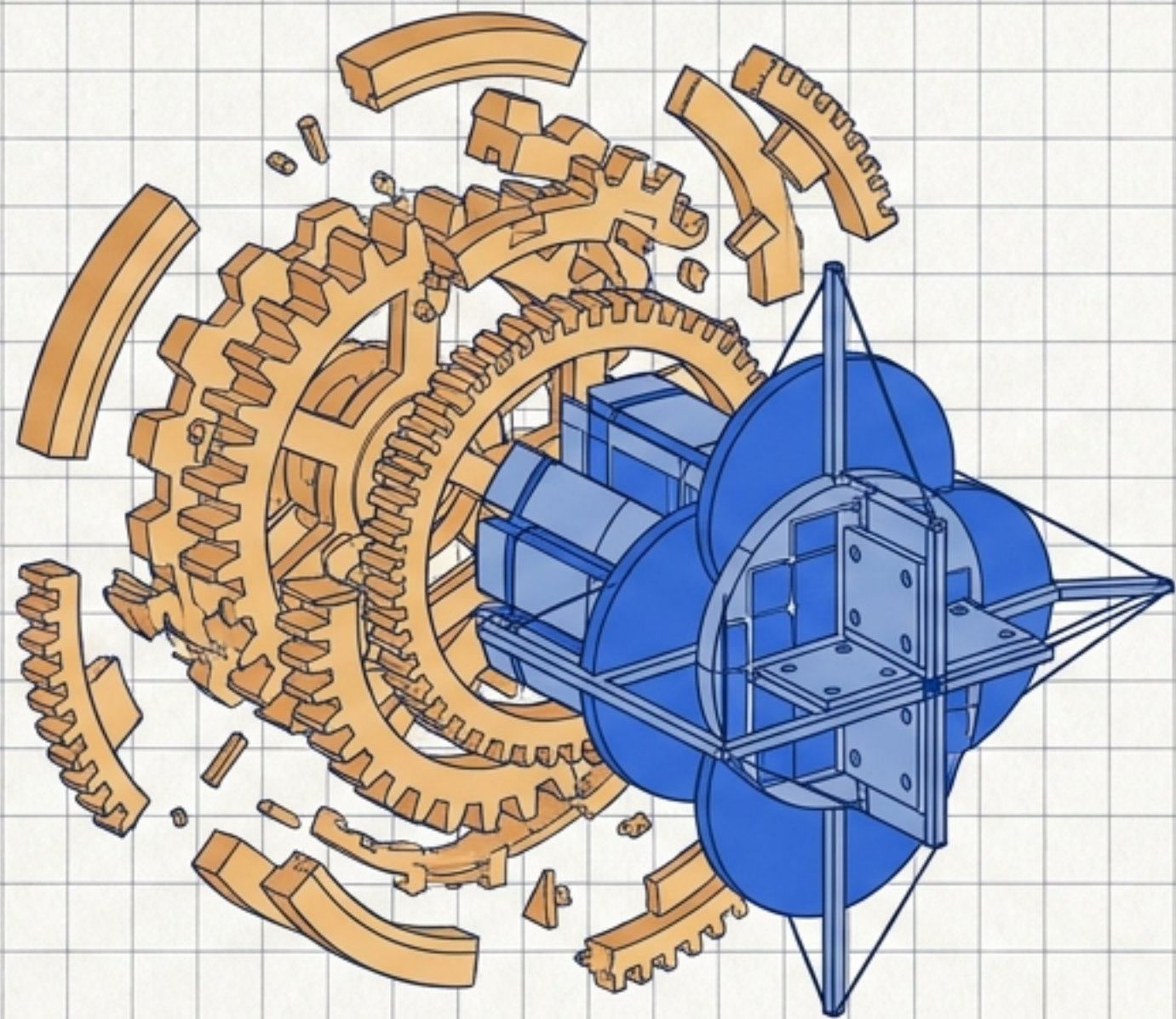
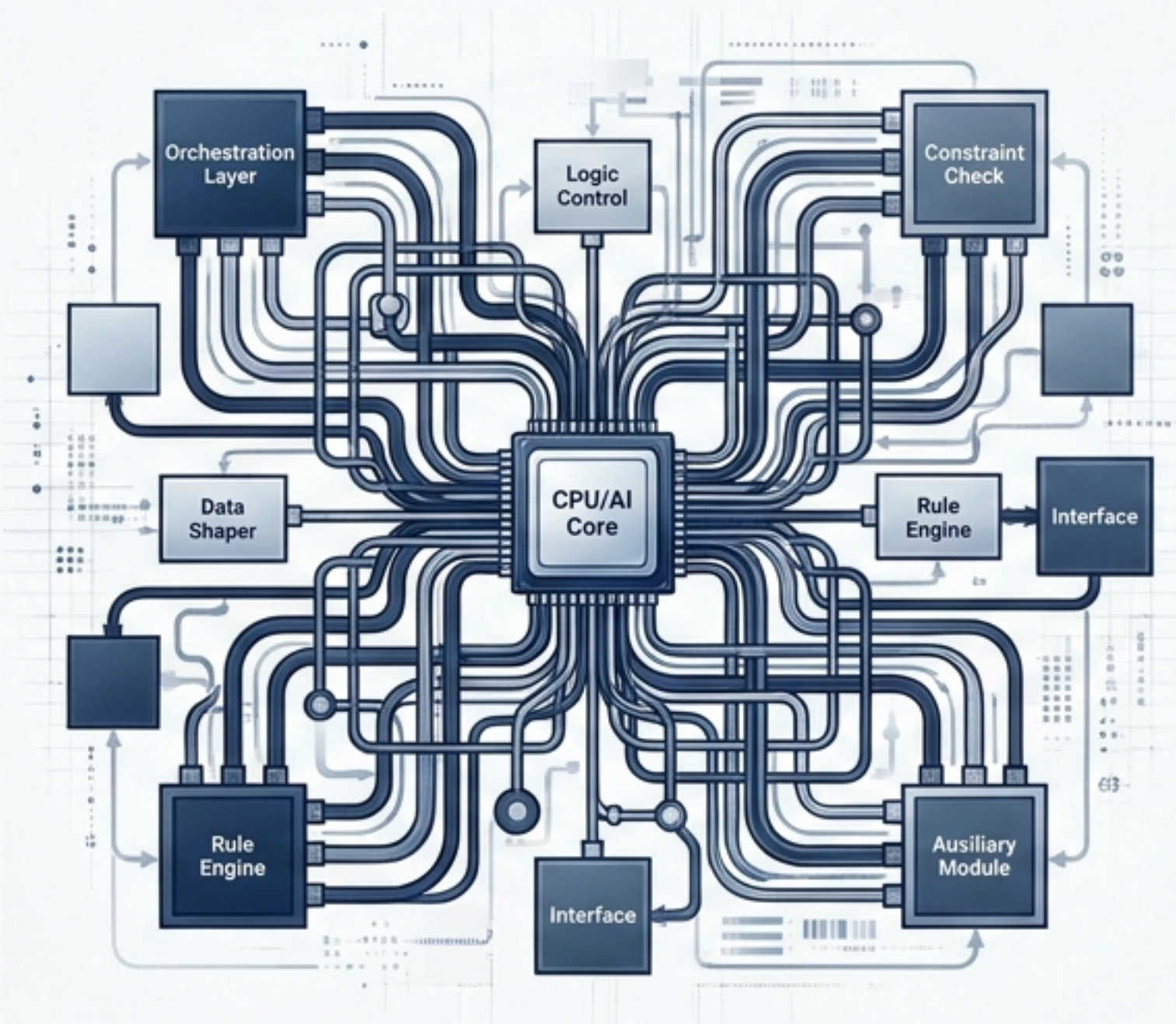




AIエージェント・ハーネスの死と新生

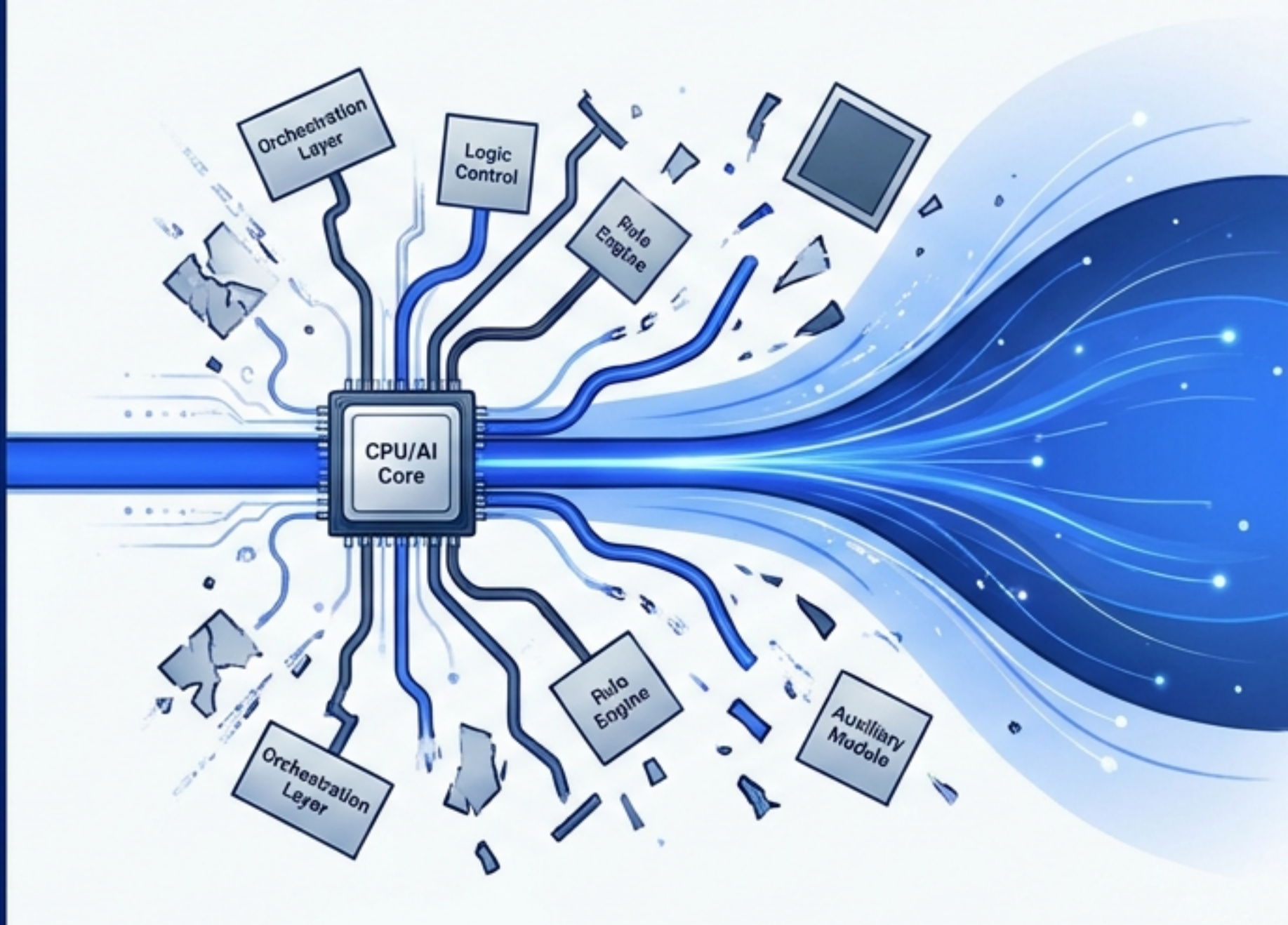
2026年秋の3大実証研究が暴く「補助輪」の陳腐化と、次世代OSアーキテクチャへの転換

ハーネスの設計が知能を決める



モデルの選択と同等以上に、複雑なオーケストレーション層の構築がベンチマークスコアを左右する

複雑なハーネスは『足枷』である



モデル自身の高性能化により、良かれと思って設けたフレームワークが、AIの自律的探索を阻害する「取り外し忘れた補助輪」と化している

旧世代：知能補完型ハーネス（思考の補助輪）

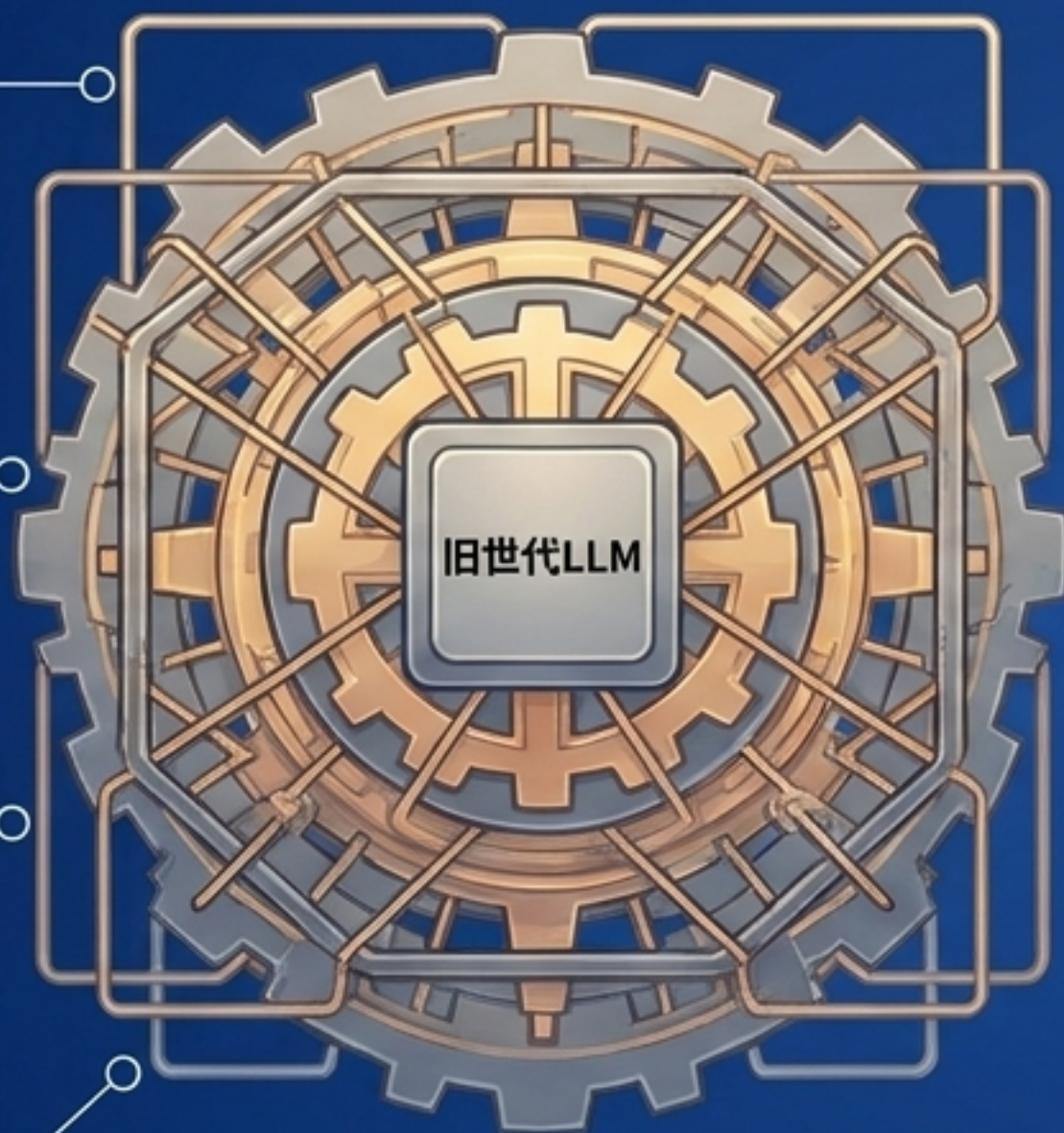
次世代：自律実行基盤型ハーネス（未知への乗り物）

人間による細かな手順指示

細分化された専用ツール

複雑なマルチエージェント制御

ローカルPC内に限定



シンプル環境 (Bash)

コンテキスト整理・権限管理

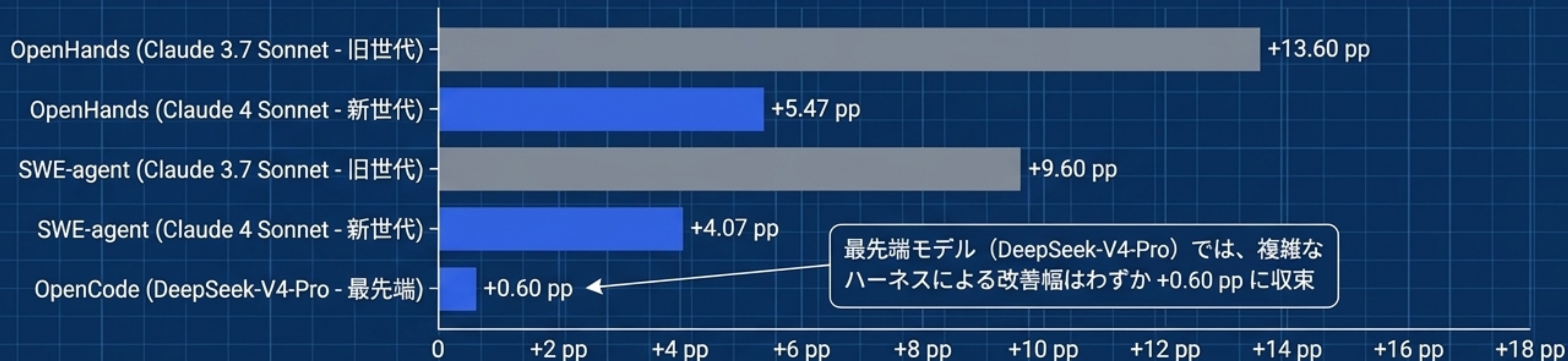
PC外の広大な領域 (クラウド/外部デバイス)



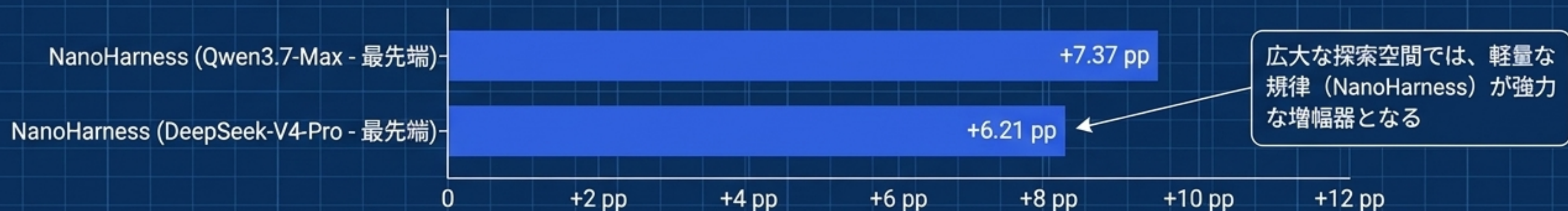
「AIは自らの能力として、知能の補助輪を飲み込んだ」
— Jedaiah Katz (aceXAI)

タスクの自律性とハーネス効果のパラドックス

1. 定型バグ修正タスク（SWE-bench）：限界利益の逓減



2. リポジトリ全体生成タスク（ProgramBench）：高自律空間での増幅

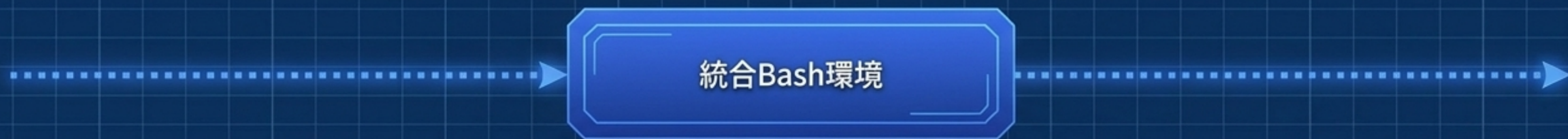


176通りの実験が暴く「アクション空間の断捨離」

旧来のパイプライン：細分化された専用ツール群



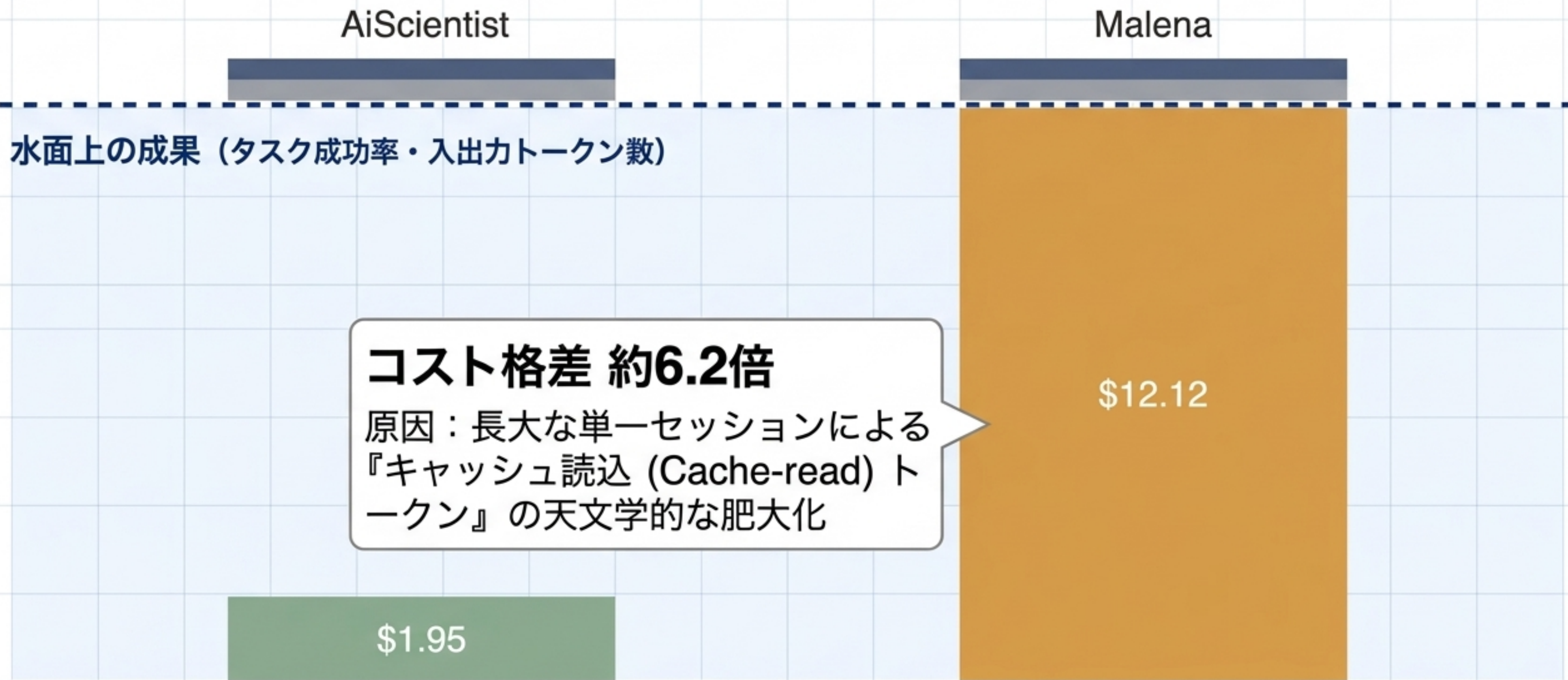
最先端のパイプライン：「Bashのみ」環境

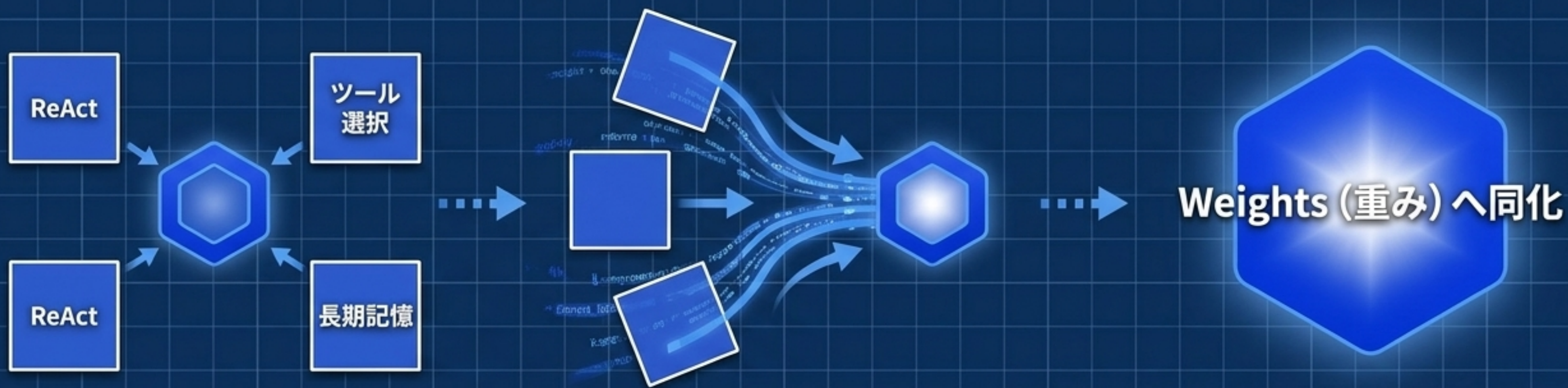


高性能モデルが自律的にスクリプトを書き、
複合操作を1回の呼び出しで完遂

**SWE-Benchにおけるタスクあたり
最大53%のコスト削減を達成**

トークン経済学の罫：同等精度に潜むコスト爆発

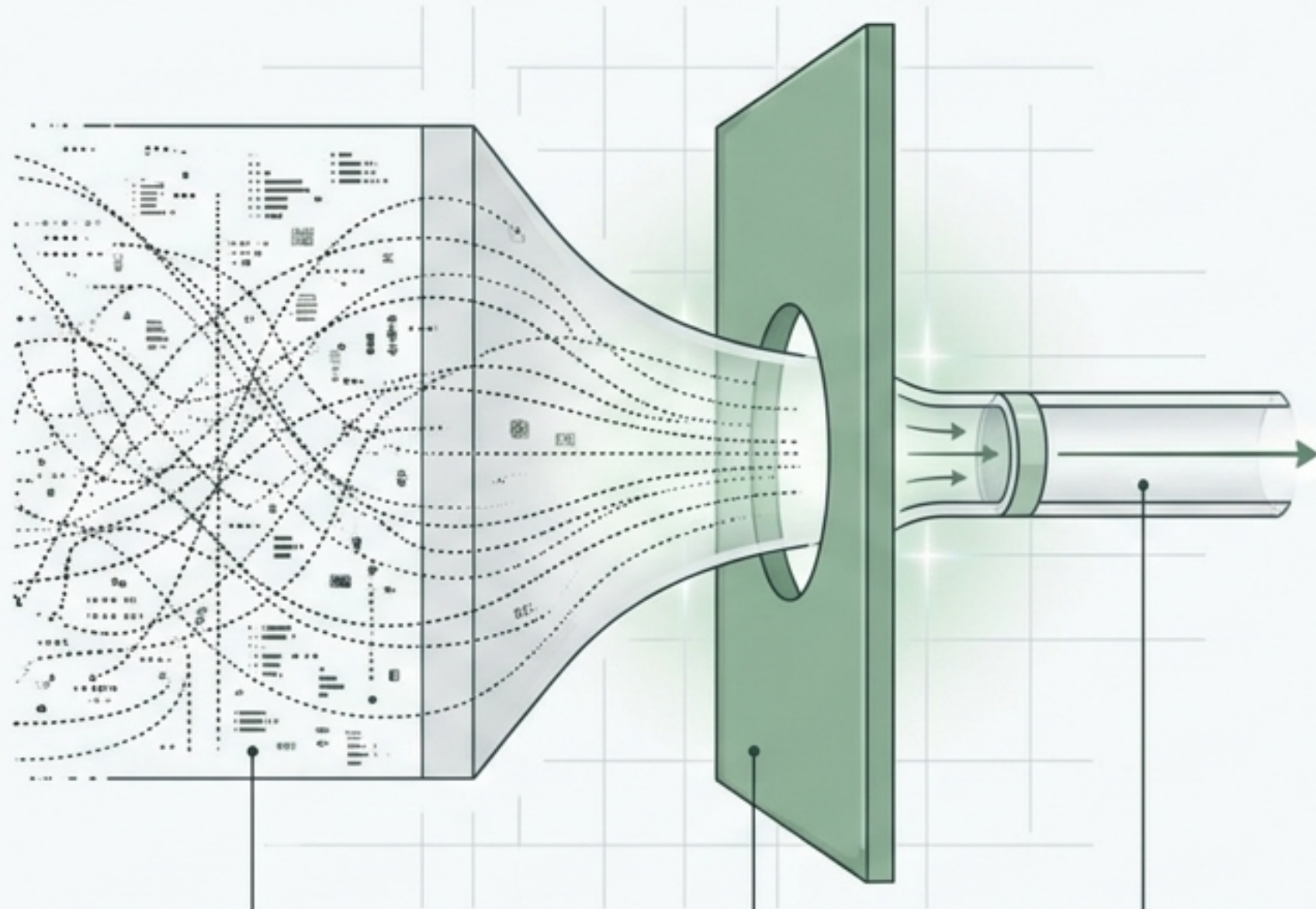




「知能を補うハーネスは不要になった。
しかし、それはハーネスの死ではない。」

今後6ヶ月以内に、AIタスクの90%はPC外(クラウド・実世界)へ移行する。
次の戦場は『安全で自由な実行基盤』の構築である。

要件1：コンテキスト管理（無制限の蓄積から T4 Elision 戦略へ）

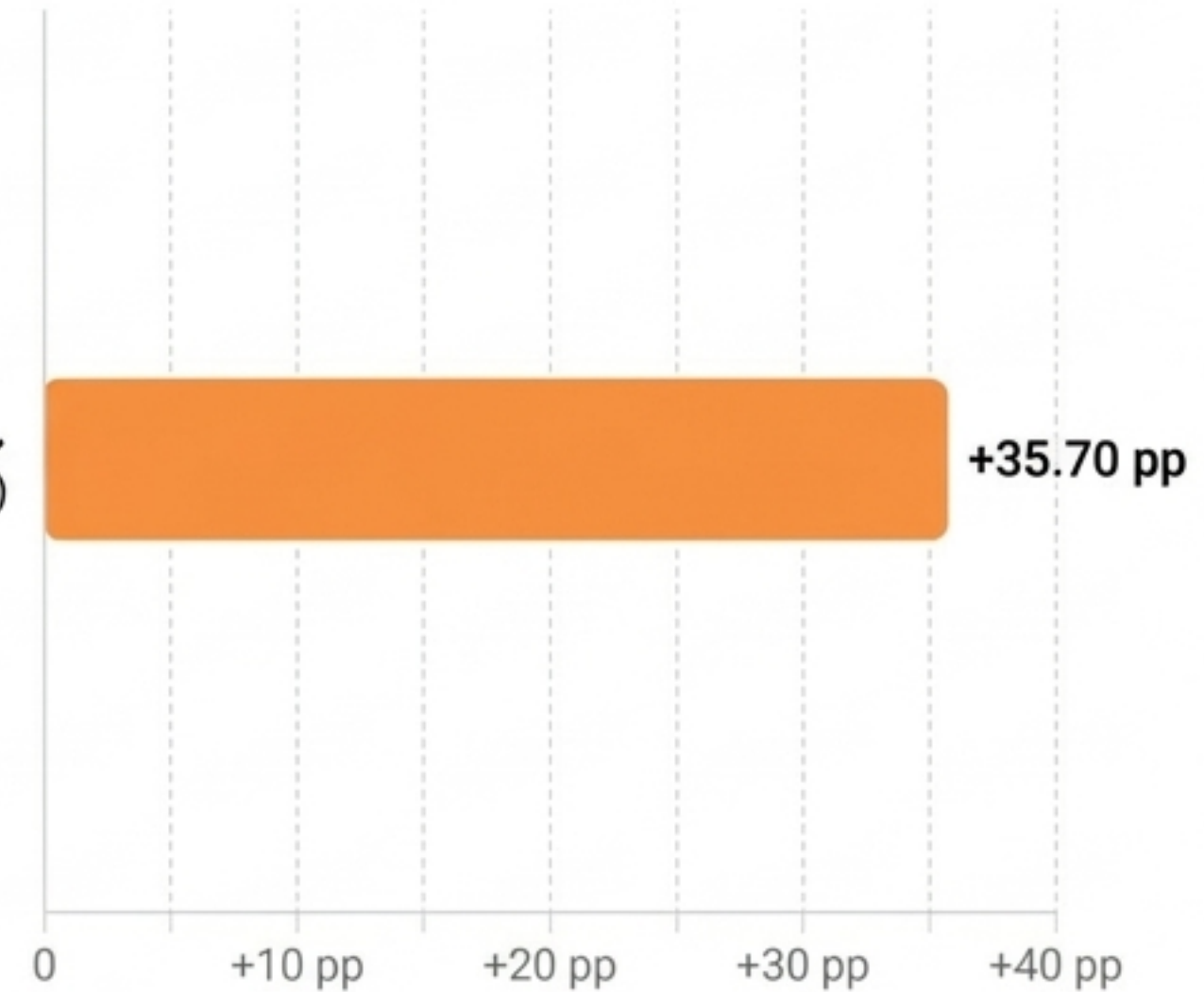


未管理の巨大ログ（78.7%が
コンテキスト溢れで崩壊）

T4戦略：Elision
（省略）

中間出力のスタブ置
換によるノイズ排除

管理あり vs なし
(32k 狭小コンテキスト枠)



インサイト：記憶管理とは、精巧に思い出させることでは
なく、不要なノイズを自動的に『捨てる』ことである。

要件2：ステートフルな権限とVM常駐化

本番環境 (Cloud / Server)

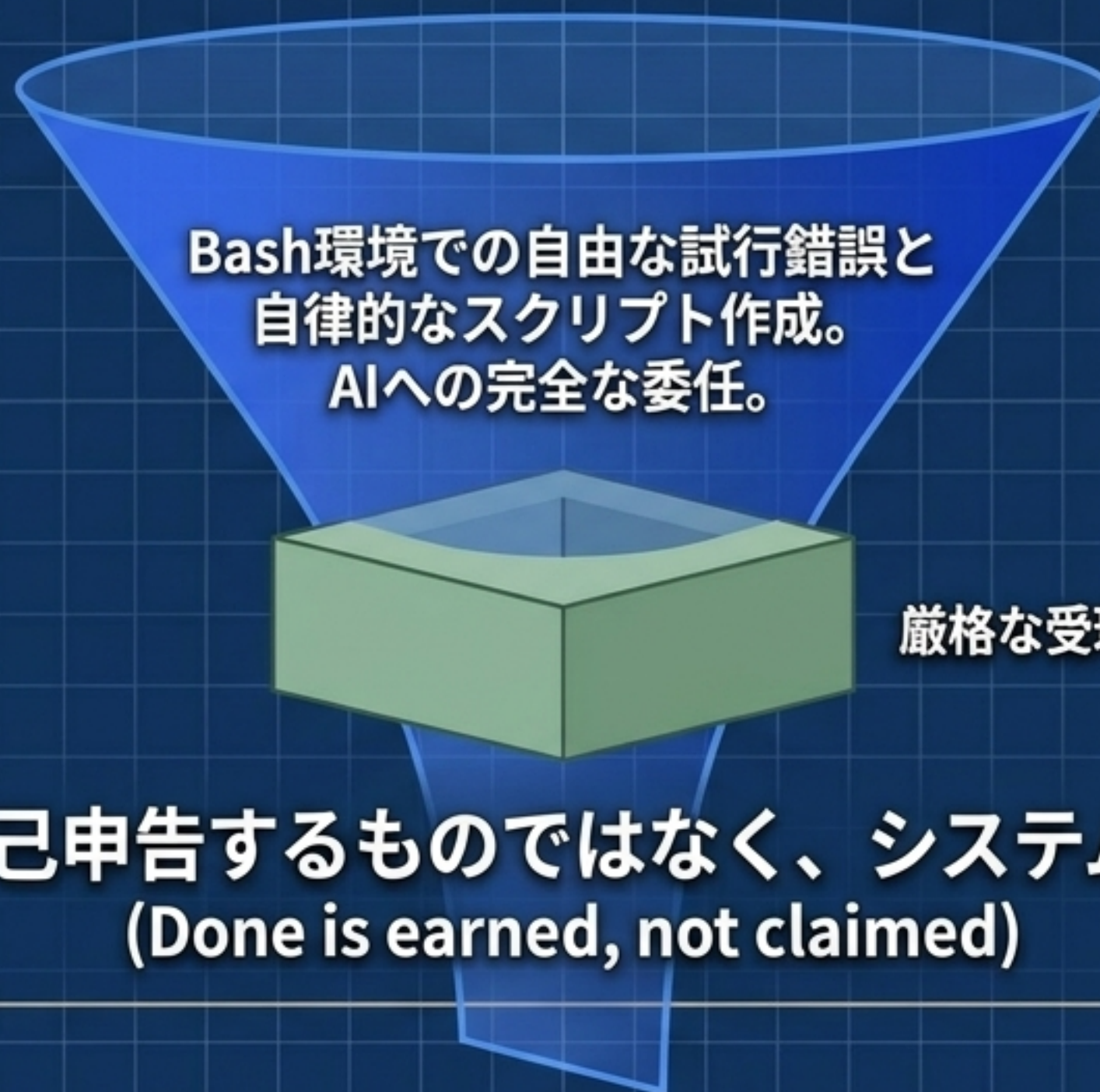
Sapien / Clef : ステートフル・ポリシーエンジン

VM Resident (OSプロセス常駐)



単なる静的許可リストではなく、
文脈に沿ってツール呼び出し
シーケンス全体を監視し、
ポリシー失効後の操作を遮断する

要件3：出口の厳格な検証（Verification）



入り口（探索・思考空間）

Bash環境での自由な試行錯誤と
自律的なスクリプト作成。
AIへの完全な委任。

厳格な受理ゲート（Bez / nano-harness 事例）

『完了』はAIが自己申告するものではなく、システムが証明するもの。
(Done is earned, not claimed)



テスト通過エビデンスに基づく受理

パラダイムシフトの全貌：旧世代 vs 次世代アーキテクチャ

	旧世代（知能補完・補助輪型）	次世代（自律実行基盤・OS型）
推論・ワークフロー	人間が設計した複雑なプロンプト連鎖とマルチエージェント階層	モデルの自律推論に完全委任 / 計画機能はコスト削減に限定
アクション空間	細分化された多数の専用ツールスキーマ	統合Bashと最小限のファイル操作への集約 / 遅延読込スキル
記憶管理	無制限の蓄積と頻繁なLLM要約・再検索	T4戦略（Elision）による中間出力の自動スタブ化とノイズ排除
状態・実行環境	ローカルPC・エディタ内での一時的なチャットセッション	クラウドVM常駐 / 高速Fork / クラッシュ復旧とセッション永続化
セキュリティ・検証	人間による逐次目視確認と手動承認	ステートフルなポリシー監視（Sapien）とテスト証拠に基づく厳格な受理

Developer Playbook：明日からのアクション指針

[STOP] やめるべきこと

1. 人間による細かい推論ステップのプロンプト・エンジニアリング
2. フロントティアモデルに対する『専用ツール群』の過剰な作り込み
3. ローカルPCでの実行を前提としたエディタ拡張機能中心の設計

[START] 始めるべきこと

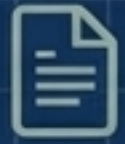
1. アクション空間を『Bash』に集約し、AI自身の複合コマンド実行力を解放する
2. 無駄なトークンを自動で捨てるコンテキスト省略（Elision）戦略の導入
3. 本番稼働を見据えたVM常駐アーキテクチャと動的権限制御の実装

次の戦場は、エンジンの馬力ではない。

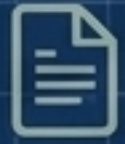
フロンティアLLM
(推論エンジン)

AIの知能を信頼して『思考の足場』を撤去せよ。
そして浮いたリソースのすべてを、
AIが実世界と安全に接続するための
『強靱な実行環境 (OS)』の構築へと投資せよ。

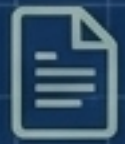
Appendix / References



Paper 1: Brilliantov, Hernández-Cano, et al. (EPFL / Apple, Sep 2026) - How Much of a Harness Does a Strong Agent Need for Autonomous ML Engineering? (arXiv:2609.40303)



Paper 2: Hu, Zhang, et al. (ISE Research, Sep 2026) - Beyond the Model: Demystifying Harness Effects in Software Engineering Agents (arXiv:2609.32459)



Paper 3: Fan, Zhang, et al. (UMass / Zoom, Sep 2026) - An Empirical Study of Harness Design for Coding Agents (arXiv:2609.20804)



Industry Insights & Architectures

Jedaiah Katz (aceXAI) on Agent Evolution

Microsoft Agent 365 & Integrated Security Operations Center (ISOC)

Parallax Research: AI Agents in September 2026

boxd.sh Cloud VM Architectures