LLM更新に負けない！「陳腐化しない」AIエージェントの運用設計：プロンプトは『文章』から『契約』へ

プロンプトが「壊れる」5つの技術的要因

推論・指示追従性の変化
モデルが賢くなることで、以前は有助だった「Step by Stepで考えて」という指示が冗長になり、逆効果になる、あるいは指示を「より文字通り」に解釈しすぎて融通が利かなくなるケースがあります。

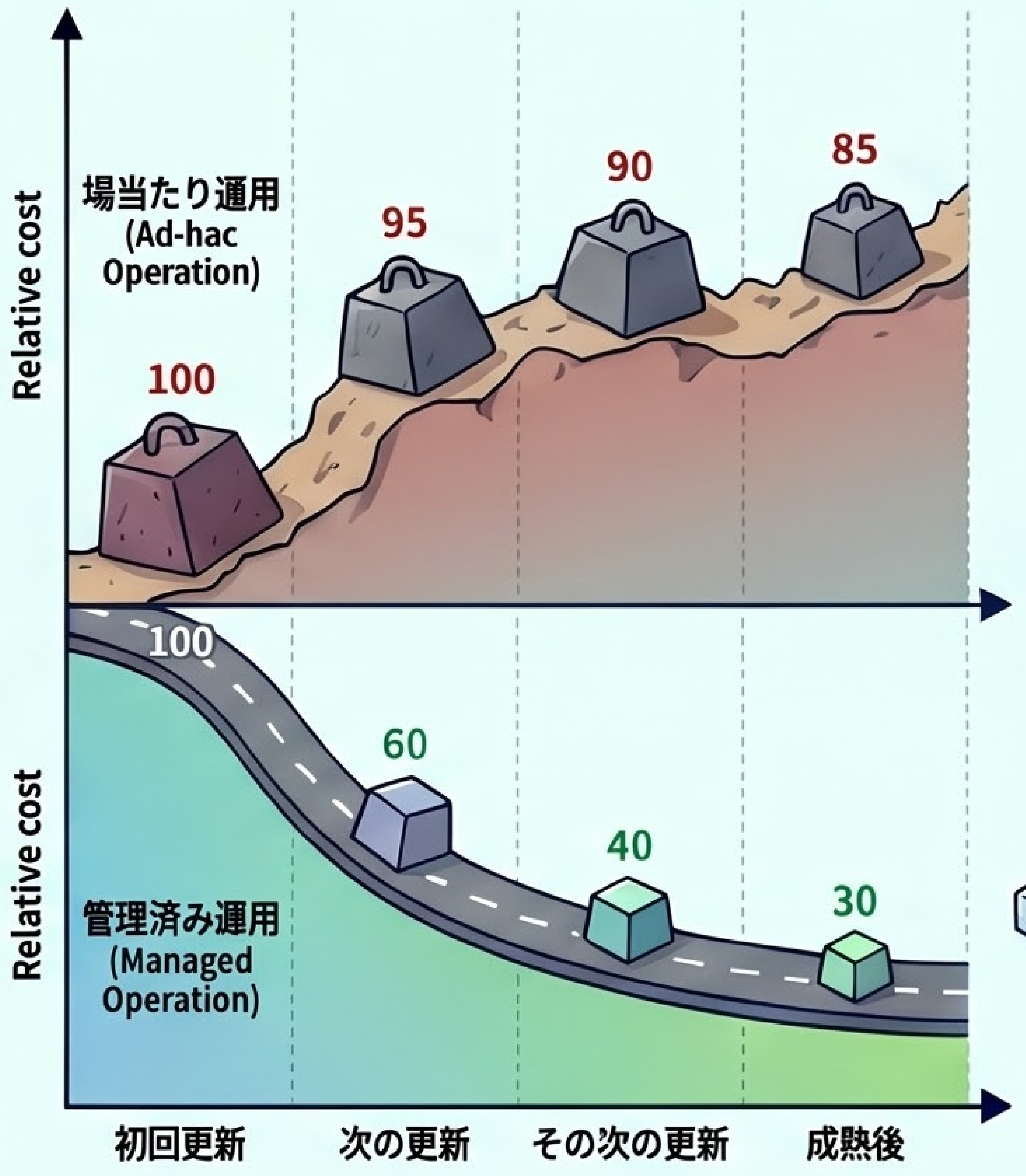
ポストトレーニングによる「好み」の更新
RLHF (人間によるフィードバック) の重み付けが変わることで、回答のの口調、相否率、ユーザーへの過度な同調 (Sycophoncy) などの「性格」が変化し、既存プロンプトとの相性が變化します。

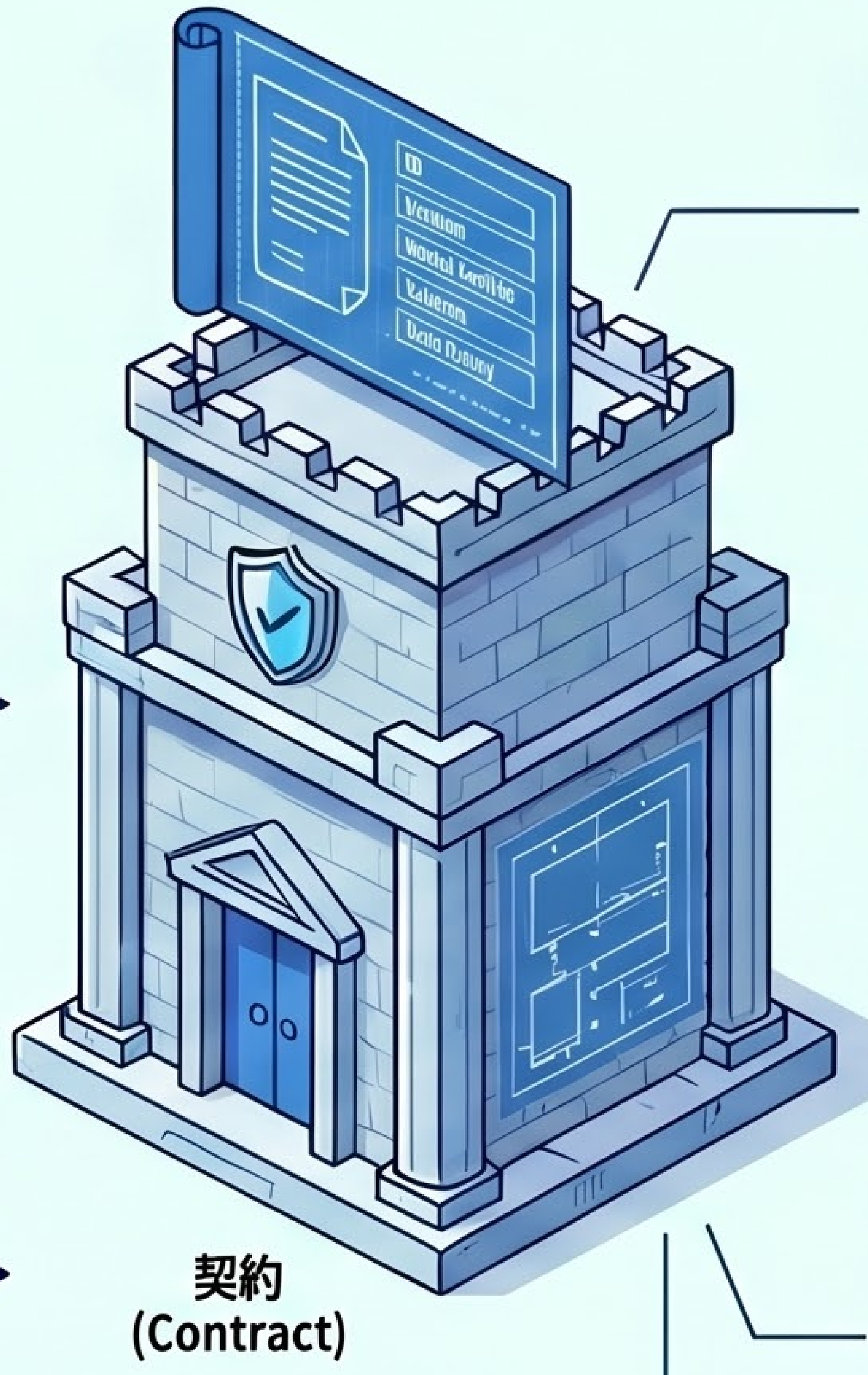
システム/安全層の隠れたポリシー
アプリUIとAPIでは適用されるシステムプロンプトや安全ガードレルが異なり、プラットフォーム側の暗黙的なポリシー変更が予期せぬ拒否や挙動の変化を引き起こします。

トークナイザーと構造化インターフェース
モデル更新によりトークン計算式が変わることでコストやコンテキスト制限が変動するほか、JSON Schema等の構造化出力の構造や例示の仕方が極度に僵話するようになっています。

ツール (Function Calling) の挙動変化
モデルがツールを呼ぶタイミングや引数の構造方法が変化されるため、自然言語で「必ずツールを使って」と懇願するよりも、厳密なスキーマ定義とツールポリシーによる制御が不可欠です。

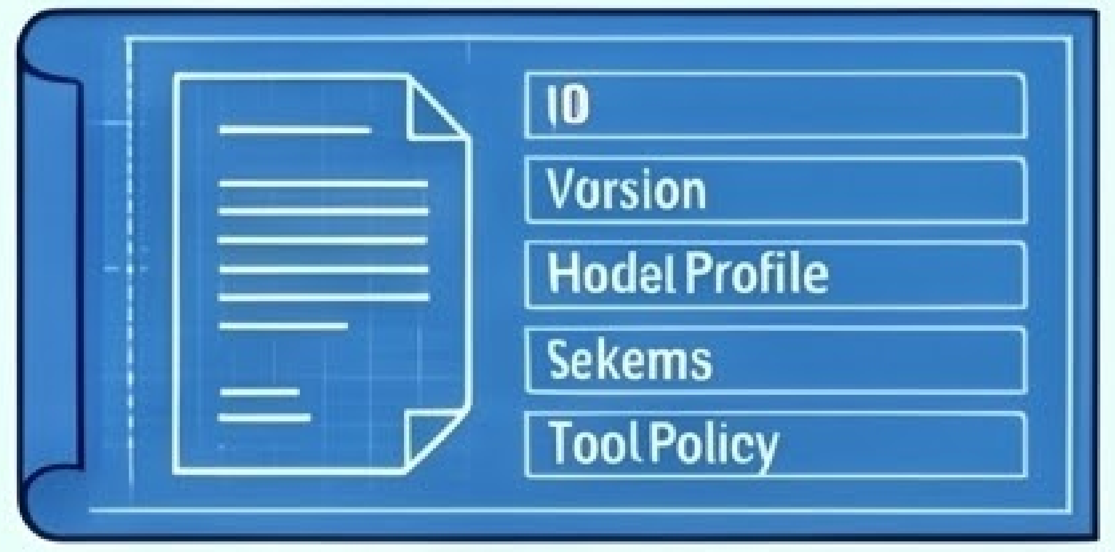
運用モデルの比較：場当たり運用 vs. 管理済み運用





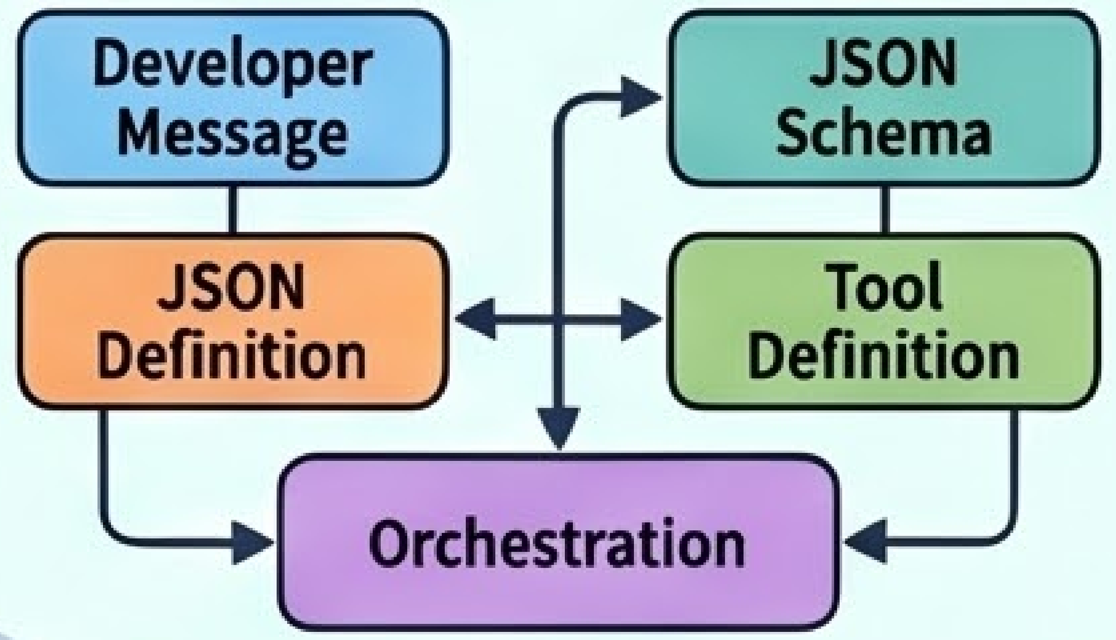
「Prompt as Contract（契約としてのプロンプト）」設計原則

自然言語から「構造化された契約」へ



プロンプトを単なる文章ではなく、ID・バージョン・モデルプロファイル・スキーマ・ツールポリシーを含む一つの「成果物」として扱います。

責務の分離（Decoupling）



役割はDeveloper Message、出力形式はJSON Schema、ツール許可はTool Definition、リトライ処理はOrchestration層へと、役割ごとに機能を分離します。

評価スイート (Canonical Evals) の構築
正常系（Golden set)、過去の障害 (Breakage set)、攻撃的入力 (Adversa-rial set)、本番ログ（Replay set）の4群で、更新時の回帰テストを自動化します。

実務で効く移行チェックリスト

変更は「一度に一箇所」
モデルだけを先に更新してベースラインを測定し、プロンプト調整はその後、評価指標 (Evals) を見ながら一箇所ずつ行います。

形式崩れをプロンプトで直さない
出力形式の崩壊は、プロンプトの変更をこねくり回すのではなく、Schema定義、Validater、または構造化出力オプション (Structured Outputs) 側で解決します。

ロールバック戦略の分離
「プロンプトのバージョン」「モデルのスナップショット」「ツールスキーマ」の3軸で、個別に以前の状態へ戻せるRunbookを整備しておきます。