AI 開発ハーネス — AI 駆動開発を実用にする

2026-09-21
SimpleModeling

AI 開発は、ソフトウェア開発の進め方を大きく変えつつあります。業務アプリケーションでもその力を活かすには、AI が生成したプログラムを業務規則や品質の条件に照らして検証し、人間が採用を判断できるようにする必要があります。

鍵となるのは、AI 開発ハーネスという考え方と、それを実現する技術です。

本稿を第 1 回とする全 10 回の「AI 開発ハーネス」シリーズでは、ソフトウェア工学の成果を AI が活用できる形にするさまざまなハーネスと、その組み合わせを考えます。第 1 回では、背景と課題、共通文脈、シリーズ全体の構成を示します。

1　AI が変えるソフトウェア開発

AI 開発の美点は、次のようなところにあります。

- 要求や仕様を自然言語で記述し、構想や業務知識を開発の出発点にできます。
- 簡潔な仕様から未記述の部分を推論・補完し、検討すべき前提や選択肢を明らかにできます。
- 既存システムや技術情報を探索し、必要な知識を設計と実装の候補に適用できます。
- プログラムを圧倒的なスピードと規模で生成できます。
- その結果、短期間に繰り返しプログラムを作り直し、異なる設計や実装を試せます。
- 曖昧な情報や状況の変化に応じる新しい応用へ踏み出せます。

AI 開発の美点は、画面中心で業務規則や状態管理が比較的単純な Web アプリケーションで、早くから活かせる可能性があります。

2　業務アプリケーションに求められる条件と課題

中核的な業務アプリケーションでは、個々の機能に加えて、システム全体の規則と品質を揃える必要があります。

業務アプリケーションは、業務規則と不変条件、認可、状態遷移、トランザクション、監査可能性を一貫して満たす必要があります。さらに、外部システムとの契約を守り、利用状況に見合う応答性能、信頼性、可用性、セキュリティ、可観測性を維持する必要があります。これらの条件はシステム全体と運用に及びます。

これらの条件を自然言語プロンプトと AI の注意力だけで継続的に満たすことは難しく、中核的な業務アプリケーションへの適用には大きな課題が残ります。

AI が簡潔な仕様を補いながら開発する場合、次の三つを区別して確認する必要があります。

- 開発効率：要求された変更を、手戻りを抑えて完成させられるか。

- 成果物の品質：AI が補った判断を含む変更が、業務上の正しさと必要な品質属性に適合しているか。
- AI 計算コスト：コンテキストの再構成、探索、再生成、再レビューをどこまで減らせるか。

候補生成だけを速くしても、不適切な前提や技術を適用した候補が増えれば、レビューと修正が増えます。品質をコンテキストの量だけで支えようとすると、毎回の再構成と推論に大きな計算が必要になります。

AI が補った判断と適用した技術をシステム全体の要求と照合するには、前提と検証条件を明示する必要があります。

AI が補った判断を仕様や設計として残さず、自然言語のやり取りと AI の注意力だけに依存すると、対話履歴や参照情報が変わったときに同じ要求から異なる実装が生まれ得ます。処理手順や例外時の振る舞いまで毎回 AI に組み立てさせる場合の設計上の問題は、後続回で具体的に扱います。

AI 開発は、候補の生成から、採用できる変更として承認するまでに要する時間、計算、判断、手戻りで評価します。ここでは、その総量を収束コストと呼びます。

解決すべき問題は、AI が補完した内容と検証条件を明示し、不適切な候補を見分け、修正後に必要な再確認を限定できるようにすることです。

3　AI 開発ハーネスによる対策

AI 開発ハーネスは、AI へ与える意味、探索範囲、成果物の形式、状態変更の境界、検証方法、エビデンス、承認を構造化します。AI が推論で補った判断を確認し、業務規則に適合する候補へ探索を導きます。

要求と AI が補った判断をモデルや型付き意図として残すと、候補を同じ基準で機械的に検証し、業務上の意味をレビューできます。修正後は変更箇所と影響を受ける箇所を再検証し、その結果を承認の判断材料として残します。判断基準の再利用はレビューの手戻りを抑え、再検証の範囲を明確にすることは不要な探索や再生成に使う AI 計算を抑えます。

ハーネスは、モデル、アーキテクチャ、型、DSL、プログラム、ランタイム、テスト、レビュー、承認など、ソフトウェア工学の成果を AI 駆動開発で活用できるタガです。その内側で AI を意図理解、探索、仮説、候補生成、説明などの得意分野と、AI でないと実現できない応用へ集中させます。

業務アプリケーション級の保証には、決定性、再現性、追跡可能性が必要です。連載で扱う各ハーネスは、ソフトウェア工学の成果を入力、境界、成果物、検証として具体化します。

生成 AI 向けのスキル、ツール、指示、ガードレールのセットもハーネスの一形態です。しかし、本シリーズで扱うハーネスはそれより広い概念です。ハーネスは、モデルを作る場所、アーキテクチャ上の責務を決める場所、プログラムを記述する場所、ランタイムで実行する場所、結果を検証する場所、知識を参照する場所に、それぞれ異なる形で現れます。

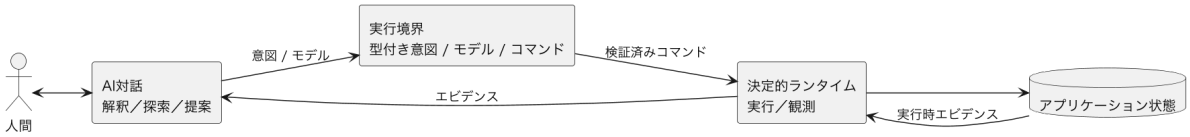
開発の各場所にあるハーネスを組み合わせ、AI の解釈・推論・探索を業務規則、決定的実行、エビデンス、人間の承認へ接続します。この組み合わせが、業務アプリケーション級のプログラムを AI とともに開発するための基盤になります。

本シリーズは全 10 回です。第 1 回である本稿は、メリット、適用を妨げる問題、広義のハーネスという対策、共通語彙、シリーズ全体の構成を示します。第 2 回以降の 9 回では、モデル、アーキテクチャ、プログラミング、ランタイム、検証、知識に現れるさまざまなハーネスを取り上げ、それらを検証・改善のフィードバックループとして組み合わせる方法を提案します。

中心原則は次の二つです。

- プロンプトは意図、モデルはプログラム。
- 非決定的に考え、決定的に実行する。

AI の非決定性は、意図の理解、質問、探索、仮説、候補生成、説明で活用します。一方、状態変更、業務規則、妥当性確認、認可、永続化、ワークフロー、再試行、冪等性、並行性、補償は、明示された意味を実行するプログラムとランタイムへ委ねます。



ここで区別するのは、AI が解釈・探索・提案する段階と、承認済みの意味に従って状態を変更する段階です。どの責務に非決定性を許し、どの境界から決定的な実行を要求するかを明確にします。

4　シリーズで用いる技術とプロダクト

本シリーズでは、ハーネスの働きを具体的に示すため、以下の技術とプロダクトを使用します。

- SimpleModeling：モデルを中心に設計・実装・実行をつなぐ開発方法論。
- CML：モデルを記述する言語。
- Cozy：モデルをプログラム・設定・文書へ変換するツール。
- Textus：ソフトウェアの実行基盤。

変換規則だけで定まらない実装には AI を活用します。

5　シリーズ全体の構成

シリーズは全 10 回、4 部で構成します。

- 第 I 部 問題設定（第 1 回）：なぜ AI 開発ハーネスが必要か。
- 第 II 部 モデル空間を制約する（第 2〜4 回）：何をモデル化し、どの構造と振る舞いを許すか。
- 第 III 部 実現空間を制約する（第 5〜8 回）：責務、知識、実装、実行をどう制約するか。
- 第 IV 部 実行から正本へ戻す（第 9〜10 回）：エビデンスをどう評価し、人間の承認へ戻すか。

本シリーズの次には、Essence フレームワークによる開発プロセス定義を扱います。

6　まとめ

- AI は自然言語の仕様を推論で補い、技術情報を探して適用できます。その判断が要求に合うかは別途確認が必要です。
- 業務アプリケーションの正しさと品質は、明示した規則、検証、エビデンス、人間の承認で支えます。
- 全 10 回で、ソフトウェア工学の成果を AI が活用できるさまざまなハーネスと、その組み合わせを提案します。

7　次回

第 2 回「モデル・ハーネス」では、自然言語の意図を、人間とプログラムが確認できるモデル、型付き意図、コマンドへ具体化します。