ランタイム・ハーネス

品質属性を支える実行基盤を、AIのハーネスとしても活用

逸脱を抑える

規則・契約の範囲で実行

正しい利用へ導く

モデルと部品の作法を使う

ハーネスの三つの役割

制約	規則・契約・権限に合わない操作を制限
誘導	使う操作と接続方法を明確にする
フィードバック	検証結果と理由を修正へ返す

組み合わせることで、AIが個別に判断する範囲を減らす。

モデル構築・プログラミング・実行

モデル・ハーネス	構造・振る舞い・規則を定義し検査
プログラミング・ハーネス	型・合成規則・APIで実装を制約
ランタイム・ハーネス	実行時の権限・状態・作用を管理

各層で扱う条件を分担し、制約の抜けを補う。

品質属性とハーネスの関係

元々の目的	セキュリティや信頼性を支える
実行基盤の仕組み	認可・更新・状態遷移を管理
AIにも適用	逸脱を抑え、正しい利用へ導く

通常の作法に沿う実装に、共通の制御が働く。

CMLから生成したモデルを使う

CMLのモデル

構造・操作・進行の定義

生成された型と操作

補完コードも同じ定義を使う

Textus/CNCF

モデルの意味に従って実行

モデルを別のデータ構造や手続きへ作り直す範囲を減らす。

Entityを基盤とする更新・参照

Entity

業務データの基盤

Aggregate：更新

業務操作と不変条件を扱う

View：参照

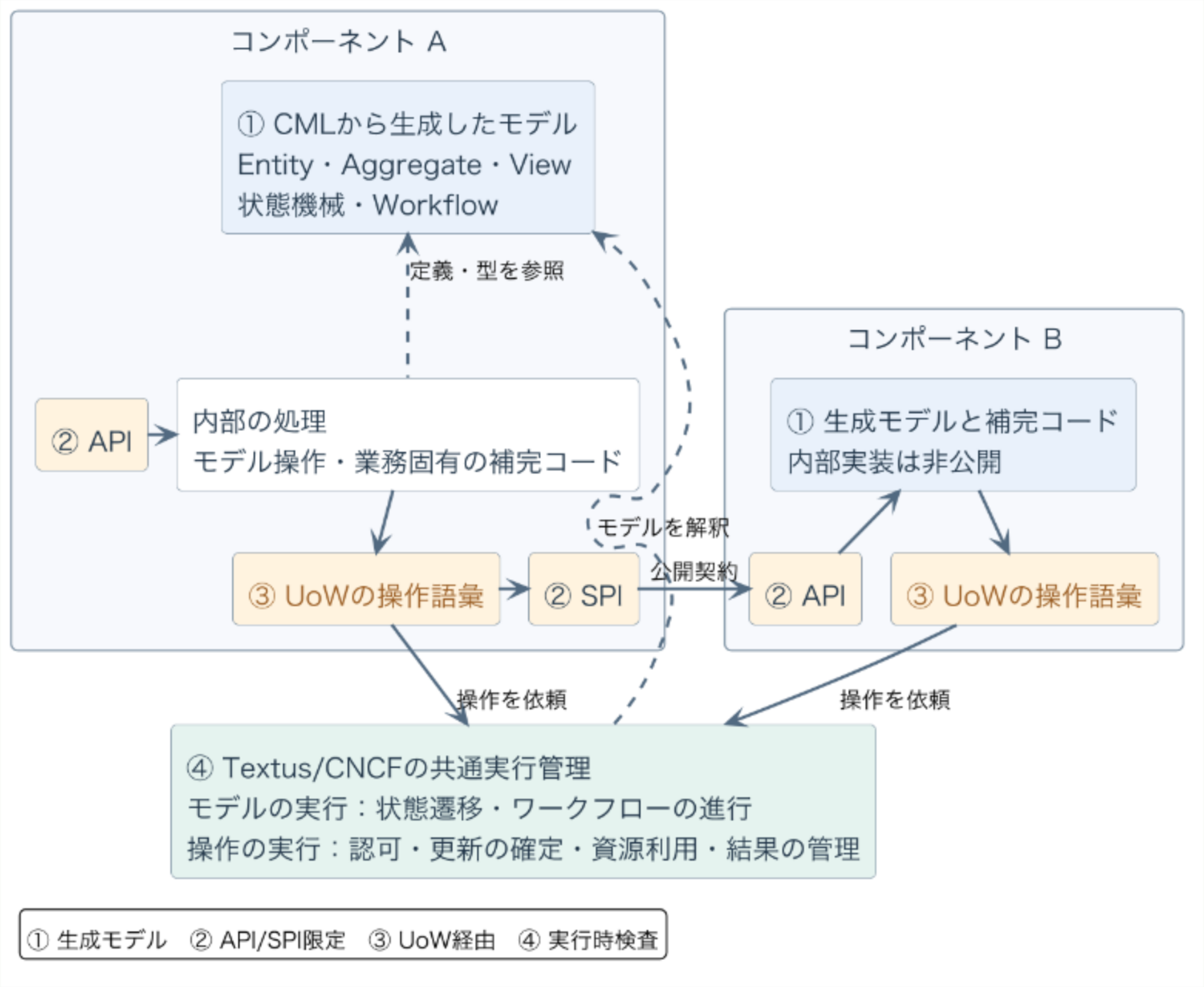
結果をDisplayで表示へ投影

更新・参照・表示の役割が、使う操作を明確にする。

状態機械を基盤とするワークフロー

状態機械	状態・イベント・遷移条件を明示
ワークフロー	業務の順序・分岐・進行を記述
コンポーネントの公開操作	AggregateやViewの操作へ接続

進行とデータ操作が、モデルと部品の契約に従って動く。


コンポーネント A
① CMLから生成したモデル
Entity・Aggregate・View
状態機械・Workflow
定義・型を参照
② API
内部の処理
モデル操作・業務固有の補完コード
③ UoWの操作語彙
② SPI
モデルを解釈
公開契約
コンポーネント B
① 生成モデルと補完コード
内部実装は非公開
② API
③ UoWの操作語彙
操作を依頼
操作を依頼
④ Textus/CNCFの共通実行管理
モデルの実行：状態遷移・ワークフローの進行
操作の実行：認可・更新の確定・資源利用・結果の管理
① 生成モデル ② API/SPI限定 ③ UoW経由 ④ 実行時検査

内部の作用と実行管理

UoWの操作語彙

保存・参照・外部呼出し

→

Free Monad

操作を合成し、実行を基盤へ渡す

→

共通の実行管理

認可・更新の確定・資源管理

使う操作を限定し、その経路に実行時の制御を適用する。

入力の検査と結果の集約

CNCFのプロパティ

パラメタの取得と検査

→

Consequenceを合成

独立した項目はアプリカティブに集約

→

共通の結果体系

有効な入力を業務処理へ渡す

独自の解析やエラー収集を部品ごとに作り込まない。

ConsequenceとConclusion

Consequence

処理の成功・失敗を共通の型で表す

成功

結果を次の処理へ渡す

失敗：Conclusion

入力・業務・認可・実行の問題を構造化

部品を組み合わせても、結果と失敗の扱いを揃えられる。

品質属性を支える仕組み

品質属性	実行基盤の仕組み
整合性・正確性・信頼性	モデルの規則、状態遷移、更新管理
セキュリティ・観測・監査	標準経路の保護と操作の記録
可用性・耐障害性・回復	部品の境界と進行管理による影響の限定

標準操作を使うことで、共通の制御の恩恵を受ける。

性能・変更・運用を支える分担

品質属性	実行基盤の仕組み
性能・応答性・弾力性	参照モデルと共通の実行・容量管理
保守・検証・再利用・接続	型付き契約と業務・実行制御の分離
導入・運用容易性・資源効率	標準設定と共通の資源管理

必要な処理量や可用性の水準は、配備と設定で調整する。

AIの作業をワークフロー内へ置く

作業の条件	入力・制約・完了条件を定義
AIの判断と生成	結果と根拠を返す
基盤の進行管理	全体の順序・継続・再開を扱う

AI自身が処理に参加する場合も、同じ管理の枠組みを使う。

実行結果を検証し、修正へ返す

実行結果と記録

Consequence・Conclusionと実行記録

→

検証

要求・モデル・完了条件と照合

→

ワークフローの判断

修正・再検証、停止・人による確認

結果型だけで正しさや収束を保証せず、検証と継続判断を組み合わせる。

AI開発ハーネス　第7回

ランタイム・ハーネス

モデルと部品の作法を、実行時の共通制御へつなぐ

モデル構築
構造・振る舞い・規則を定義し検査

▶

プログラミング
型・合成規則・APIで実装を制約

▶

実行
権限・状態・作用を基盤が管理

実行時の構造と制約点

コンポーネント A

① CMLから生成したモデル

モデル操作 ＋ 業務固有の補完コード

③ UoWの操作語彙　② API・SPI

契約 ⟶

コンポーネント B

① CMLから生成したモデル

モデル操作 ＋ 業務固有の補完コード

③ UoWの操作語彙　② API・SPI

④ Textus/CNCFの共通実行管理
モデル実行・認可・状態遷移・更新の確定・資源管理・結果と記録

品質属性を支える実行基盤の仕組みを、AIのハーネスとしても活用

制約｜逸脱を抑える
規則・契約・権限に合わない
操作を実行時に制限する

誘導｜適切な操作へ導く
モデル操作・API・SPI・UoWが
実装と接続の補助線になる

フィードバック｜修正へ返す
結果と記録を要求に照らして検証し、
問題の箇所と理由から修正範囲を絞る

実行結果・記録 → 要求・完了条件と照合 → 修正・再検証／停止・人による確認

通常の作法に沿う実装で、品質を支える共通制御の恩恵を受け、AIの判断・実装の負担を減らす。

検証と再試行・停止の条件をワークフローに定め、収束を支える。

SimpleModeling.org

まとめ

多重のハーネス	モデル・プログラム・実行で制約を補う
制約・誘導・フィードバック	逸脱を抑え、適切な実装と修正へ導く
通常の作法による恩恵	実行基盤の制御が業務処理に適用される

次回：知識ハーネス — AIの判断と生成の前提を揃える