

アーキテクチャ・ハーネス

SimpleModelingのリファレンス・アーキテクチャ

仕組みの組合せ

品質・開発速度・AIコストを全体で考える

多重のハーネス

一つずつでは残る制約の抜けを補う

リファレンス・アーキテクチャの全体像

CMLによるモデル

構造・規則・振る舞い



Cozy + AI · Scala

接続の生成と契約に沿う補完



Textusの実行基盤

モデルの直接実行と実行条件



知識・観測・検証

用語・根拠・実例・実行の証拠



モデル、補完実装、実行基盤、検証の役割と接続を定める。

独立した技術を全体の設計へ結び付ける

単独でも利用可能

モデリング手法・Cozy・Textus



各技術もハーネス

モデル・生成規則・型・実行規則



全体最適を提示

品質・開発速度・AIコスト

単独利用できる各技術を組み合わせ、開発全体の最適化を図る。

モデルを生成・補完・実行の共通基準にする

CMLとTextus

モデルを記述し、直接実行



Cozy

モデルから接続構造を生成



AIとScala

契約の下で不足する処理を補完

モデルと契約を共有し、生成・補完・実行の間のずれを抑える。

部品の境界と3層の役割を組み合わせる

プレゼンテーション

画面・対話・入力の変換



アプリケーション

ユースケースを起点とするワークフロー



ドメイン

概念・構造・規則のモデル

責務の配置と公開契約を揃え、AIが変更する範囲を明確にする。

高い抽象度のモデルを検査・実行に使う

DDDレベルの構成要素

エンティティ・値・集約・操作



動的な規則

ワークフロー・状態機械



直接実行の効果

コードへの翻訳と照合を削減

ドメイン知識に近い記述を、そのまま検査と実行の基盤にする。

要求との対応と実行結果を両方確認する

ユースケース

利用者の目的と実例



モデルのレビュー

操作・ワークフロー・状態機械と照合



要求・モデルと照合



動く仕様書

同じ実例の実行結果を確認

モデル間の整合と、要求に対する結果を異なる検査で確認する。

規則の所有場所と評価方法を揃える

ドメイン側で所有

業務規則を集約



ルール・エンジン

共通の方法で規則を評価

設計課題

第一級の規則モデルと実行系の対応

規則の所在と評価方法を揃え、別々の実装への分散を抑える。

変更範囲を絞り、その内側を型で検査する

CBDの公開契約

部品間の接続と利用条件

Scalaの型

値・入力・結果の整合



契約を保つ変更

部品と接続を中心に確認

公開契約を保つ変更では、確認範囲を部品とその接続へ絞りやすい。

AIレビューへ持ち込む問題を減らす

型・代数的データ型

機械的に判定できる不整合を検出

不変データ・純粋関数

追跡する状態と作用を限定

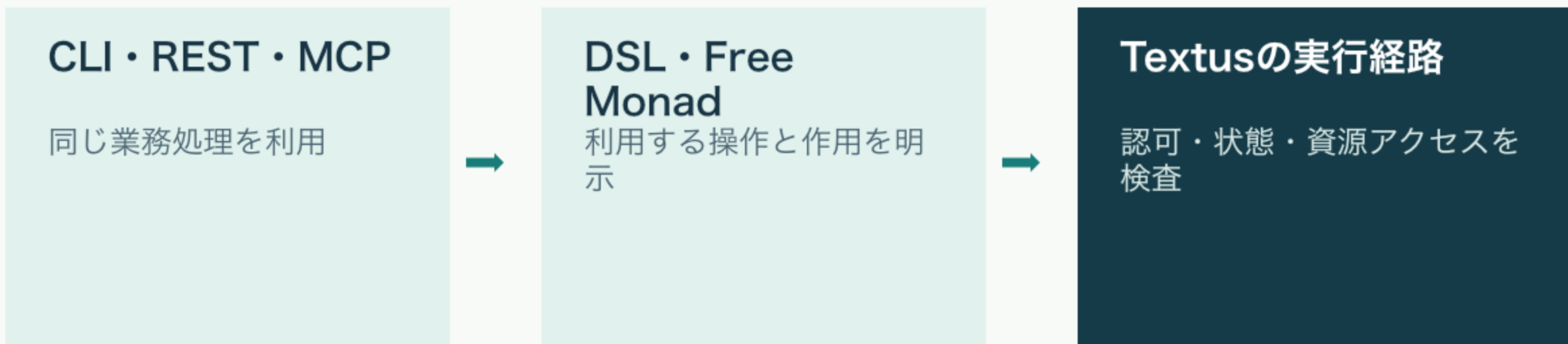
レビューと修正

対象範囲と反復回数を抑制

機械的な検査と状態の限定を組み合わせ、レビューと修正の負担を抑える。

設計上の効果であり、削減率の実測ではない。

許された操作を共通の実行経路へ接続する



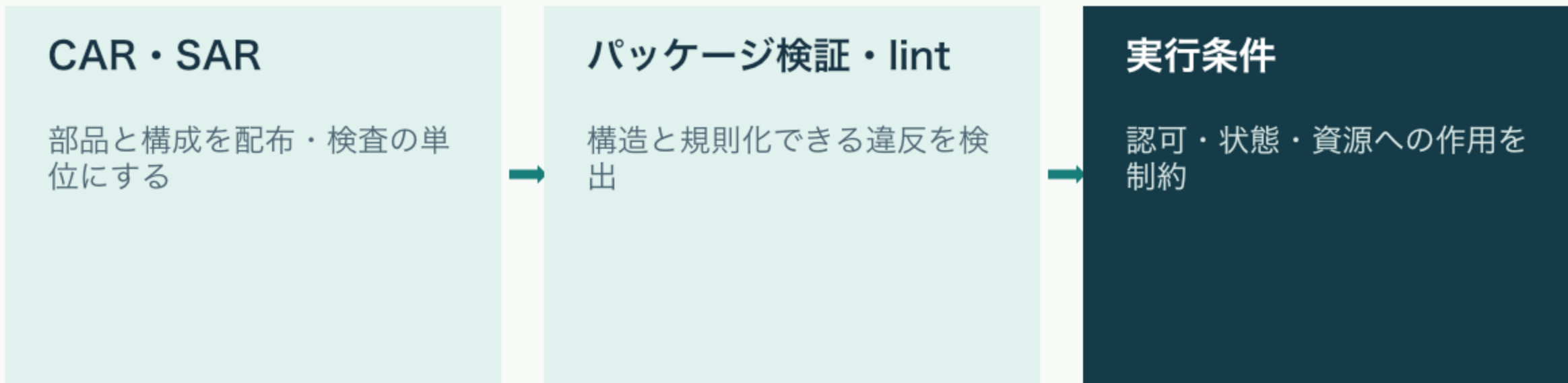
入口が異なっても、同じ業務処理と実行時の制約を利用する。

複数のハーネスで制約の抜けを補う

通過した条件	残る確認
型が正しい実装	業務規則への適合は別途確認
モデルに合う操作	現在の状態と利用者の権限を確認
要求との照合	実例に対する結果を検証

一つの検査を通過しても残る条件を、別のハーネスで確認する。

部品の検査と実行時の制約を対応付ける



配布する部品の検査と、利用時の条件を組み合わせる。

適用範囲・実装状況は第7回。悪意あるコードには別途隔離が必要。

知識と契約を揃え、判断の前提を限定する

BoKの用語集

文芸モデルとオブジェクト・モデルを関連付け



RDF・MCP

出典・版・適用範囲を持つ知識を提供



スキル

変更範囲と検査結果への対応を指定

モデル・契約・知識の対応を、AIの判断の前提にする。

対象に合う知識と契約を揃えることを目指す。

実行の証拠をモデルへ戻して修正に使う

観測モデル

操作・遷移・失敗をモデルの語彙へ対応



動く仕様書

要求の実例と実行結果を照合



要求・モデルと照合



AIレビュー・人間の判断

意味と要求への適合を確認

証拠をモデルへ対応付け、修正対象と判断が必要な箇所を明確にする。

注文処理でモデルと実装を接続する

ユースケース

購入者の目的・在庫不足・支払い失敗



静的モデルと規則

注文・明細・金額・価格の規則



ワークフローと状態機械

在庫確保 → 支払い承認 → 確定

要求・構造・振る舞いを接続し、注文処理をモデル上で検査する。

同じ注文処理に異なる条件を重ねる

状況	重ねる制約・検証
型検査を通過	価格・税額の規則を実例と照合
注文確定を呼出し	支払い承認と利用者権限を確認
支払いに失敗	確定を止め、定義した補償へ進む

同じ処理を異なる条件で制約し、個々の検査の抜けを補う。

連載の位置付け：第1～5回

第1回：AI開発の条件

検証・修正・承認までの収束コスト

第2～4回：モデル

CMLとモデル・アップ・ダウン
静的モデル、動的モデル

第5回：アーキテクチャ

各機構の担当と組合せによる全体最適

モデルをプログラムとして扱う基盤を、アーキテクチャへ接続する。

連載の位置付け：第6～10回

第6～7回：Scalaと実行基盤

型・不変データ・DSL・Free Monadと実行時の制約

第8回：知識

BoK・RDF・MCPで前提と適用範囲を揃える

第9～10回：観測と検証

実行の証拠、動く仕様書、AIレビュー、人間の判断

各技術の詳細は担当回で扱い、第5回では組合せを捉える。

アーキテクチャ・ハーネス

SimpleModelingのリファレンス・アーキテクチャ

モデルを共通の基準にする

CML：概念・関係・規則
要求と実現モデルを照合

→

生成と補完を組み合わせる

Cozy：構造を生成
CBD+Scala：補完範囲と型

→

同じ規則で実行する

Textus：モデルを直接実行
認可・状態・外部作用を制約

知識・観測・検証を同じモデルと契約へ対応付ける

BoK・RDF・MCPで前提を揃える / 実行の証拠を要求の実例と照合する

単独の検査に残る条件を、別のハーネスで確認する

型が合う → 業務規則も確認 操作が合う → 権限・現在の状態も確認

個別実装・確認範囲・AIレビューの反復を減らす

必要な品質を確保し、開発速度の向上とAIコストの低減を図る