Application Modeling

2026-09-14

第 8 回では、対象世界の事実、概念、規則をドメイン・モデルとして整理するドメイン・モデリングを扱いました。第 9 回は、その意味的な基盤を利用して、ステークホルダーのニーズや目標をシステムの振る舞いとして具体化するアプリケーション・モデリングです。

アプリケーション・モデルは、利用者の目的に沿ってドメインの機能を結びつける薄いオーケストレータになります。重要なデータ操作や業務状態の管理はドメイン・モデルが担い、アプリケーション・モデルは処理の組み合わせと流れを定義します。

1　ドメイン・モデルとアプリケーション・モデルの位置付け

SimpleModeling モデルは、オブジェクト・モデル、知識モデル、文芸モデルで構成されます。ドメイン・モデルとアプリケーション・モデルは、これらの構成要素から必要なモデル要素を選び、目的に沿って組織した適用モデルです。

観点	ドメイン・モデル	アプリケーション・モデル
関心	対象世界で成り立つこと	利用者が実現したいこと
出発点	事実と観察	ニーズ・目標・ユースケース
主な役割	意味・規則・状態を定義する	ドメインの機能を組み合わせる

ドメイン・モデリングでは、事実の観察から概念や規則を帰納的に整理します。アプリケーション・モデリングでは、その語彙、規則、制約を利用して、ニーズや目標を実現する振る舞いを定義します。

2　アプリケーションは薄いオーケストレータになる

アプリケーション・モデルは、入力を受け取り、必要なドメインのサービスやオペレーションを呼び出し、結果やイベントを次の処理や利用者への応答につなぎます。

業務上の判断、重要なデータ操作、不変条件の確認、業務状態とその遷移の管理はドメイン側の責務です。状態機械もドメイン・モデル側で定義し、アプリケーションはサービスやイベントを通じて利用します。

この役割分担によって、アプリケーションは利用者の目的に沿った調整に集中し、ドメインの意味と規則は共通の基盤として保たれます。

2.1　分けて定義する意義

ドメイン・モデルは対象世界の理解を蓄積し、比較的長いライフサイクルを持ちます。事実の発見、理解の深化、制度や業務の変化を契機として更新します。

アプリケーション・モデルは、ステークホルダーのニーズや利用環境の変化に応じて更新します。両者を分けて定義すると、対象世界の理解を共有しながら、ニーズに応じた振る舞いを継続的に具体化できます。

3　一つのアプリケーションが複数のドメインを束ねる

アプリケーション・モデルの意義の一つは、複数のドメインを束ねるオーケストレータとしての役割です。利用者の目的を達成するために、各ドメインの機能を組み合わせ、呼び出し順序、結果やイベントの受け渡し、失敗時の扱いを調整します。

たとえば注文、在庫、決済をそれぞれのドメインで管理する場合、アプリケーションは一連の手続きを組み立てます。各ドメインは自身の意味、規則、状態を管理し、アプリケーションはそれらの機能を目的に沿って結びつけます。

4　ニーズごとのアプリケーションから共通ドメインを利用する

同じドメインに対して、異なるステークホルダーが異なるニーズを持つことがあります。この場合は、ニーズごとのアプリケーション・モデルを受け皿として、共通のドメイン・モデルを利用します。

たとえば顧客向けの注文アプリケーションと業務担当者向けの注文管理アプリケーションは、目標やユースケースが異なります。両者で注文ドメインの意味や規則を共有し、それぞれに必要な振る舞いを定義できます。

共有の対象には、エンティティ、バリュー、関係、規則、イベント、業務状態などのドメイン知識があります。実装時には、必要に応じてコンポーネントや実行基盤も共有します。

個別のニーズの変化は各アプリケーションで受け止めます。共通ドメインを更新する場合は、それを利用する各アプリケーションへの影響を確認します。

5　ユースケースシナリオから実現モデルへ

ユースケースは、アクターの目標と、その目標を達成するシナリオを記述します。アプリケーション・モデリングでは、シナリオに現れる行動を、UI との境界、内部の参加者、処理、結果へ対応付けます。

文芸モデルのユースケースシナリオからユースケース実現モデルを構成し、その対応をレビューします。承認した実行可能モデルを CML で記述することで、要求から実装へ至る経路を追跡できます。

5.1　「注文を確定する」を具体化する

利用者が注文内容を確認し、確定を要求し、その結果を受け取るシナリオを考えます。

- UI は注文内容を提示し、確定要求をアプリケーションへ渡します。
- アプリケーションは要求を受け取り、注文ドメインの確定オペレーションを呼び出します。
- 注文ドメインは業務規則や不変条件を確認し、注文データと業務状態を更新して、注文確定イベントを発行します。

- アプリケーションは結果を受け取り、必要な後続処理につなぎ、UI へ応答します。

シナリオの各行動と、これらの処理や結果の対応を保持します。正常な流れと失敗時の扱いを確認する際にも、この対応が根拠になります。

5.2 協調と相互作用で対応を確認する

協調は、UI、アプリケーション、ドメインのサービスなどが、どの役割と責務を担って目的を実現するかを表します。相互作用は、具体的な呼び出し、結果、イベントの順序を実行例として表します。

協調で参加者の構造を整理し、相互作用でシナリオの実行例を確認します。複数のドメインを利用する場合も、アプリケーションによる調整と、各ドメインが担う判断や操作への対応を明らかにします。

6 プロパティによってオペレーションの性質を調整する

アプリケーション・モデルとドメイン・モデルでは、必要とされるオペレーションの性質も異なります。その性質をコンポーネント、サービス、オペレーションのプロパティとして設定し、デフォルトと個別の指定によって調整します。必要な性質を明確にすることで、実現時に適した実行形態を選び、最適化につなげられます。

コンポーネントには、アプリケーション用途、ドメイン用途、両方という用途を設定します。単一用途のコンポーネントでは、その用途がサービスのデフォルトになります。サービスには個別の用途を指定でき、オペレーションはサービスの性質を引き継ぎます。

アプリケーション用途のサービスはステートレス（stateless)、ドメイン用途のサービスはステートフル（stateful）をデフォルトとします。個別の要件に応じて、サービスやオペレーションで設定を変更できます。

ステートレスなオペレーションは、正しい実行に必要な状態を引数や外部の永続化機構から取得し、呼び出しをまたぐメモリ常駐状態に依存せずに処理します。ドメインの永続化データを更新する処理も、この性質を持てます。ステートフルの区別では、呼び出しをまたぐメモリ常駐状態への依存を扱います。

6.1 クラウド・ファンクションとして実現する例

ステートレスなオペレーションは、クラウド・ファンクションとしての実現候補になります。アプリケーション用途のコンポーネントやサービスでは、デフォルトの性質を引き継ぐオペレーションがその対象になります。

実装時には処理時間や資源などの実行基盤の条件を確認し、クラウド・ファンクションや通常のサーバー上のオペレーションとして実現します。利用量と料金体系の条件が合えば、クラウド・ファンクション化によって大幅なコスト削減につながります。これは、プロパティで定めた性質を実現時の最適化に生かす一例です。

7　生成 AI によるモデル管理と人間の承認

生成 AI が、ユースケースシナリオからサービス、オペレーション、イベントなどへの対応を提案し、モデルを生成・管理します。人間は、目的が達成されるか、重要な業務判断やデータ操作が適切なドメインに置かれているか、境界や失敗時の扱いが明確かをレビューして承認します。

CBD を支援する textus-cbd-support は、シナリオとモデルの対応、変更の差分、根拠、検証情報を可視化します。相互作用に記述した実行例への適合も、モデルの振る舞いを確認する材料になります。

承認した実行可能モデルを CML で記述し、Cozy によるプログラムの自動生成と Textus の実行基盤へ接続します。利用者からのフィードバックを次のニーズやモデルの更新へつなぐことで、ドメインの意味的な基盤を共有しながら、アプリケーションを継続的に進化させます。