モノ・コト分析実践のためのパターン言語
A pattern language for “mono - koto” (entity - event) analysis practice

河合 昭男†
Akio Kawai†
†(有)オブジェクトデザイン研究所
†Object Design Laboratory, Inc.

要旨
前回発表論文（C-01）「独立エンティティと関連クラスによるドメイン・モデル」の内容を基に、ドメイン・モデリングの実践方法をモノ・コト分析の視点でパターン言語としてまとめた。イベントが発生するひとつの具体的シーンに注目し、そのシーンに参加するオブジェクトをエンティティ系のオブジェクト（ステレオタイプ《mono》）、イベント系のオブジェクト（ステレオタイプ《koto》）に分離して捉えることにより、作成と理解の容易なドメイン・モデリングの実践方法を示す。

1. はじめに

オブジェクト指向の特徴のひとつは、現実世界をそのままモデリングできることである。そうは言っても現実問題、特に初心者には容易ではない。筆者は前回発表論文にて、一般的にも行われているが、オブジェクトをエンティティ系とイベント系に、言い換えればモノとコトを分離して捉えるドメイン・モデリングについて考察した。本稿ではそれぞれステレオタイプ《mono》《koto》で表し、図１のようなアイコンで表現する。エンティティ系《mono》はそれ自体で存在し得るが、イベント系《koto》は関連クラスとして捉えると分かりやすく、モデルの作成と理解の容易性を高める。

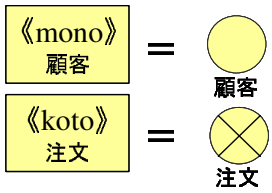

図１ モノとコト

今回本稿では、モノ・コト分析によるドメイン・モデリング実践方法をパターン言語の手法でまとめた。パターン言語は、実践の基本原則を記述するためのひとつの優れた方法である。

2. 責務モデルと意味モデル

オブジェクト指向の代表的なモデルのひとつはコラボレーション・モデルである。これは、個々のオブジェクトに責務を割り振って、メッセージによるコラボレーションにより１群のオブジェクトとしてひとつのサービスを提供するという自律分散協調のモデルである。このモデルは責務に注目するという意味で責務モデルと言える。

本稿で対象としているドメイン・モデルは抽象概念の意味を明確にするのが目的であり、責務は含めない。責務には内部データ管理的な基本的なものもあるが、コラボレーション・モデルに重要な責務は外部に対するサービス提供のためのものである。ドメイン・モデルは抽象概念の意味を明確にするためのものであり、意味モデルと言える。意味を明確にするためには、クラス汎化関係による分類構造と他クラスとの意味的関係で表すことができ、ドメイン・モデルはオントロジーの表現と見ることもできる。本稿では汎化関係は対象外で、関連の発見にのみ注目する。

責務モデルはビジネスなど利用側の影響を受け易いが、ビジネスなど対象領域で使用される情報の意味を表すオントロジーは変更を受けにくい。責務モデルは動的、意味モデルは静的とも言える。

3. シナリオとシーン

ドメインモデリングはどこから始めるべきか。ビジネスで使用されているデータに注目する DOA 的手法、抽象概念に注目するオブジェクト指向の手法がある。ここではユースケースを取っ掛かりとし、そのシナリオの中の１シーンに注目する。シーンとはアクターが関与する何らかのイベントが発生する

最小単位である。これらシーン単位のクラス図を少しずつ脹らまして行ってドメイン・モデルを完成させる。

ひとつのシーンで発生するイベントにより、そのシーンに参加するオブジェクトは生成・消滅あるいは状態変化を行う。ここで発生するオブジェクトをモノまたはコトとして識別して捉える。

例えば顧客が注文を行うシーンでは、シーンが始まる前の状態では顧客オブジェクトと商品オブジェクトが別々に存在するが、注文するというイベントが終了した時点では顧客オブジェクトと商品オブジェクトがリンクし、リンク属性として注文オブジェクトが生成された状態となる。このオブジェクト図を基にこのシーンのクラス図を起こすと図２のようになる。

クラス図で関連クラスにしなかったのは、UMLの定義では「両側のインスタンスを固定すれば関連クラスのインスタンスも一意に決まる」とあり厳密には事後状態のオブジェクト図でも同一顧客が同一商品を２回注文できないことになる。ここでは理解し易さのための中間モデルとしてリンク属性を用いたが、最終的なクラス図では関連クラスを用いなかった。

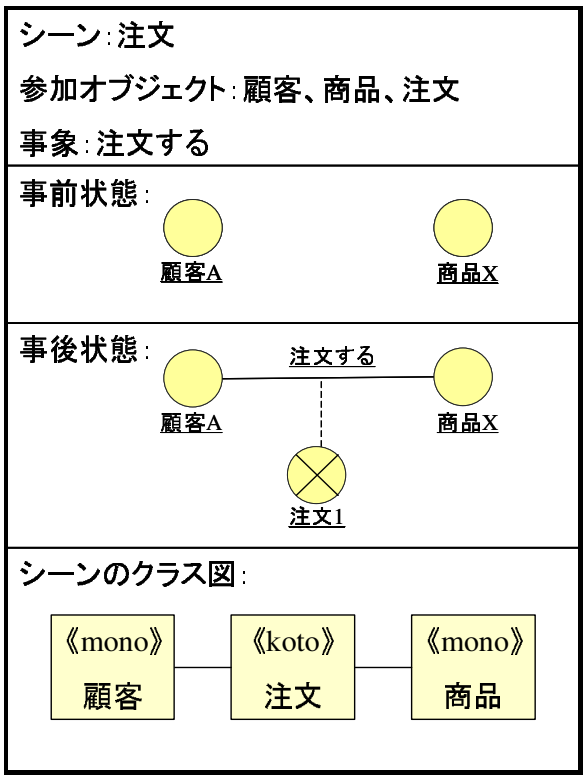

図２　シーン

4. パターン言語

モノ・コト分析実践パターン言語として７つのパターンを抽出した。パターンの名前は、次に示すように簡単で分かり易く、かつインパクトのある名前を工夫した。

表１　モノ・コト分析実践パターン言語まとめ

パターン名	概要
(1) 物事をわきまえよ	モノとコトを分離することがオブジェクト抽出の基本戦略
(2) 具体例から始めよ	オブジェクト図から描き始めることがモデリングの基本戦術
(3) 物事は単純に	アクターから見た、ひとつの具体的なシーンをイメージする
(4) 始めにモノありき	オブジェクト抽出の第一歩はモノから始める
(5) コトは物事を変える	変化した状態はどこに持つべきか
(6) コトは物事をつなぐ	コトはモノ（または他のコト）の間のリンクとして捉える
(7) コトの詳細は下に	コトの内容はリンク属性で表す

アレグザンダーの提唱になるパターン言語では、パターンとは繰り返し起きる問題とその解決策の組であり、そこに問題が発生する状況、解決策の選択肢を規定するフォース（制約）がある。それらを「状況－問題―制約－解法」という形式で以下にまとめた。

(1) 物事をわきまえよ

状況：

本来取り扱いの異なるリソース系のオブジェクトとイベント系のオブジェクトが共にオブジェクトという概念で統一的に扱われ、便利な反面ドメイン・モデルの作成と理解を困難にしている。

オブジェクト指向初心者の阻害要因のひとつ。

問題：

リソース系のオブジェクトとイベント系のオブジェクトをモデル上で容易に区別できない。

制約：

オブジェクト指向では、識別可能なものはほぼ何でもオブジェクトとできてしまう。UML ではモノもコトも同じクラス。

解法：

モノとコトをステレオタイプで分離する（基本戦略）。

例：《mono》、《koto》

(2) 具体例から始めよ

状況：

クラス図は抽象的で作成も理解も容易でない。本当は正しく理解できていないのに、分かった気持ちにさせてしまう危険性を含む。

問題：

クラス図の作成・理解が容易でない。正しくないクラス図や間違った解釈がないように確認できる方法が必要。

制約：

本来人間には抽象概念の理解は容易ではない。これはコミュニケーションの難しさの原因でもある。

解法：

まず、オブジェクト図を描く（基本戦術）。クラス図を描く前に、またクラス図を正しく理解するためにもその１つの具体例としてオブジェクト図を描く。

(3) 物事は単純に

状況：

ドメイン・モデリングはどこから取り掛かるべきか。どのように対象領域からオブジェクトを抽出すればよいか。

問題：

オブジェクト抽出は容易ではない。

制約：

人間の認識能力。

解法：

ユースケースを取っ掛かりとし、シナリオの１つの具体的なシーンをイメージし、そこに参加するオブジェクトとそこで起きるひとつのイベントに注目する。

(4) 始めにモノありき

状況：

対象領域に混在しているモノとコトを抽出したい（物事をわきまえよ(1)）。しかしながら目に見えるモノは抽出し易いが、目に見えないコトは抽出が容易ではない．．．

問題：

モノとコトを同列のオブジェクトと考えると混乱する。

制約：

本来人間には抽象概念の理解は容易ではない。

解法：

モノの抽出から始める。

(5) コトは物事を変える

状況：

モノ・コトの状態はイベントにより時々刻々変化していく。あるとき生成し、また消滅する。状

態はオブジェクトの属性の値で表される場合と、他オブジェクトとのリンクで表される場合（コトは物事をつなぐ(6)）がある．．．

問題：

イベントによるオブジェクトの変化をどのように表現すればよいか。

制約：

主要なオブジェクトは抽出済み。

解法：

様々なシーンを想定し（物事は単純に(3)）モノ・コトの状態を変化させるイベントを発見する。そのイベントによりオブジェクトの属性値が変化するのか、リンクが生成・消滅する（コトは物事をつなぐ(6)）のか検討する。

(6) コトは物事をつなぐ

状況：

シーンをイメージしてそこで発生するイベントから状態変化を発見した（物事は単純に(3)）。次にその変化した状態をモデリングしたい．．．

問題：

イベントの結果はどのように表現すべきか。

制約：

イベントに関係する主要なモノは抽出済み。

解法：

イベントはオブジェクトの属性値の変化になる場合とオブジェクト間のリンクとして現れる場合がある（コトは物事を変える(5)）。単独のオブジェクトの属性変化として表現できない場合は複数オブジェクト間のリンクとして表現する。

(7) コトの詳細は下に

状況：

イベントによりモノまたはコトの間にリンクが生成された（コトは物事をつなぐ(6)）。イベントにより発生する属性はどのオブジェクトが持つべきか．．．

問題：

イベントにより発生する属性はどのオブジェクトが持つべきか。

制約：

イベントはリンクとして抽出されている。

イベントはまだ独立したオブジェクトになっていない。

解法：

イベントの属性はリンク属性（関連クラスのインスタンス）としてモデリングする。

5. まとめ

本稿で述べたパターン言語は、ドメイン・モデリングのための実践方法のほんの一部を成すものである。他の知見もパターンとして取り込んでパターン言語としてより洗練化して行きたい。本稿のアイデアによりドメイン・モデルの作成と理解がより一層容易になれば幸いである。

参考文献

[1] 河合昭男、“独立エンティティと関連クラスによるドメイン・モデル”、情報システム学会、2005/11