B-025

データモデル：方法論の理論と実際

中村　正治
Masaharu NAKAMURA

1．まえがき

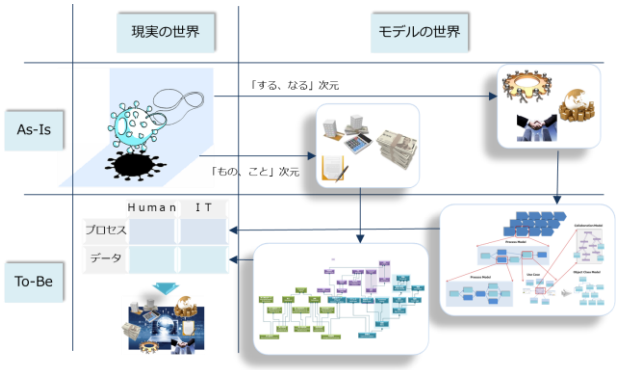
データモデルは、MDA(Model Driven Architectonics)の構成要素でもあり、その考え方や作成の方法論が 1980 年代半ばに確立した、いわば枯れたテーマである。

しかしながら、現実世界の「もの/こと」のメタアァーとしてエンティティ・クラスを定義するという最も重要な本質が理解されず、それに関して詳述してある師範所もないのが実情である。実際、システム開発現場では、古風で独善的な「データモデリングの専門家」が、単なるデータベースの定義手段として、理解できる範囲を都合よく使っているだけであるという状況を頻繁に目にする。残念ながら市販の良書の不在もあって、日本においてはデータモデリングがその価値を発揮できていないのである。

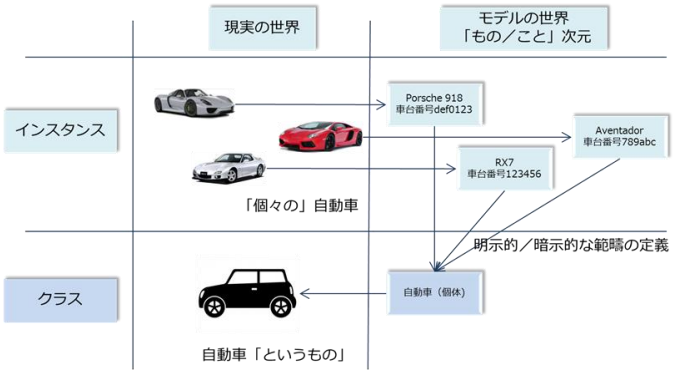
そこで、データモデリング全般のダイジェストとして、本論を起すこととした。本論は実際の部分に主眼を置いており、理論的な背景については多くを論じていない。必要に応じて、拙論「MDA:構想からプログラミングまでの継ぎ目のないアプローチの実際(FiT2014)」「MDA:その数学的定式化(FiT2015)」をご参照いただきたい。

2．データモデルの意義と目的

（1）データモデルは、ビジネ・スモデルの構成要素である。ビジネ・スモデルとは、「ビジネスを『もの／こと』ことへの『する／あらせる』作用として捉え、記述する」ものである。この「もの/こと」の次元で、その実体をエンティティ・リレーションシップ・アトリビュートによって Metaphorize したものがデータモデルである。



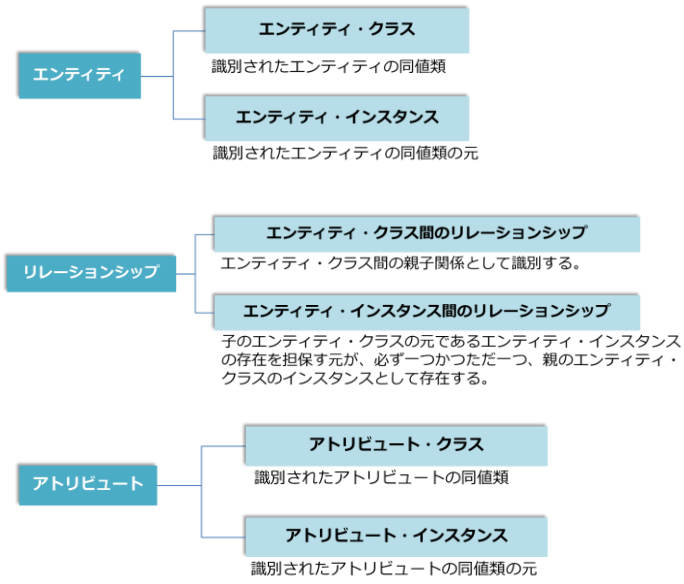
エンティティは、現実の「もの/こと」を単一の概念に還元し、それを隠喩(Metaphor)として命名したものである。現実の個々の実体は「エンティティ・インスタンス」としてモデルに写像され、共通に識別される範疇が「エンティティ・クラス」として定義される。実世界では、個々の実体を元とする範疇となる。



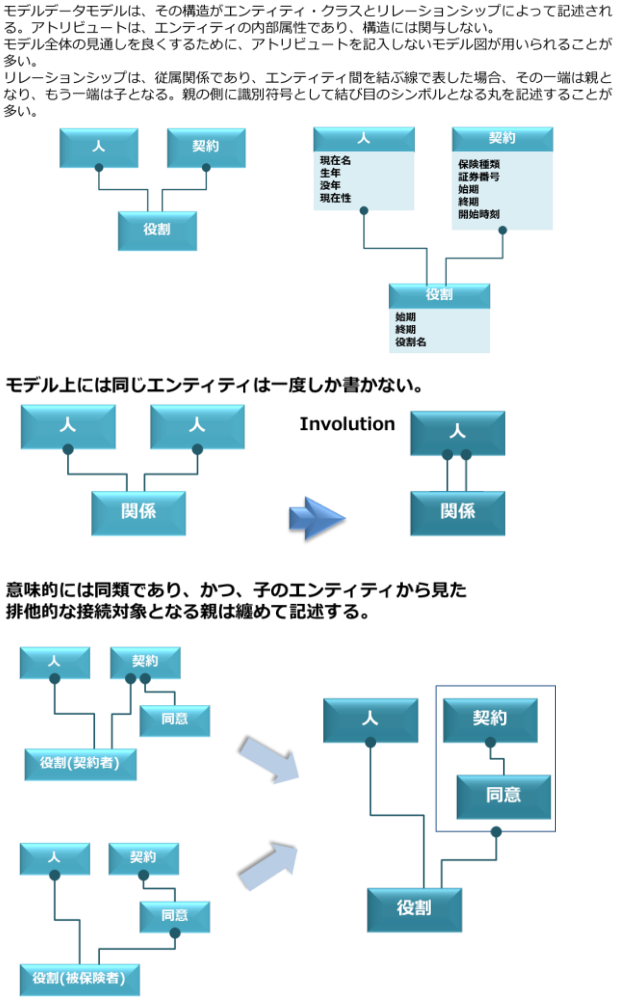
リレーションシップは、エンティティ・クラス間の従属関係(親子関係)として定義される。インスタンスレベルでは、あるインスタンスを子とする親のエンティティ・インスタンスが、一つかつただ一つ存在することを表す。このエンティティ・クラスとリレーションシップが、データモデルの構造を決定する。

一方、アトリビュートはエンティティの内部構造である。実際には、アトリビュート・クラスか従属エンティティ・クラスかの判断は、モデリングの文脈に依存する場合があり、注意が必要である。

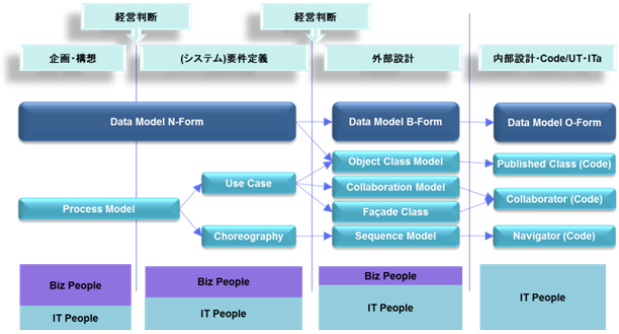
（2）データモデルは、ビジネスにおいて、企画・構想の段階からプロジェクト実施段階における、要件定義局面・外部設計局面・内部設計局面まで、形式を変えながら使用されるが、その目的は、段階・局面で大きく異なる。



（3）データモデルのクラス構造は、データモデル図によって記述される。



（4）企画・構想段階では、「エンティティ・クラス」と「リレーションシップ」が「もの/こと」のありようの隠喩として理解されるので、ビジネスの To-Be 像を描くのに非常に適している。データモデルと言われながら、IT 的側面ではなく、ビジネスそのものの記述に力を発揮する。システム化要件定義局面では、こうして定義されたビジネスの内容を IT サイドに理解させる、コミュニケーションツールとして利用価値がある。外部設計局面では、非正規化されたデータモデルがビジネス要件からシームレスに導かれるデータベース設計の記述として使用される。最後に内部設計局面で、データモデルは、データベース処理系に最適化された定義体となる。



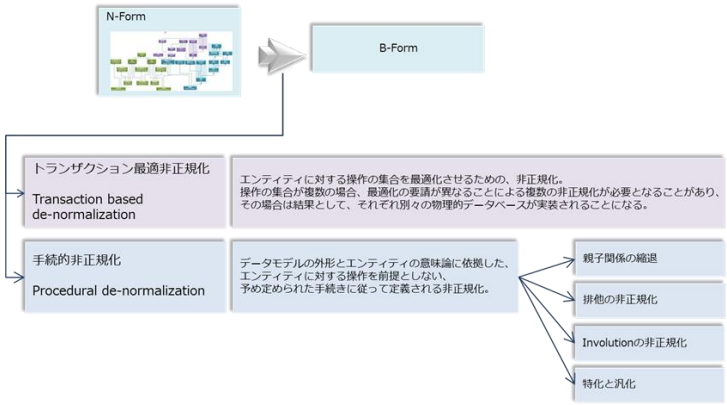
これらのモデルは、形式を変えていくが、シームレスであり、なおかつ、並行して、定義されたアクセスオブジェクトとセットにすることで、プロセスサイドとも矛盾のない設計が行えるのである。

3. データモデルのスコープと形式

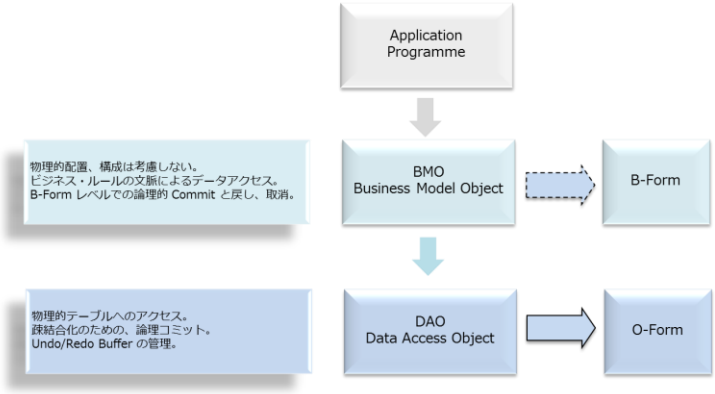
（1）データモデルは、エンティティ・クラスの単一概念性とリレーションシップの親子関係を前提とするので、必然的に正規形となる。これを **N-Form** と称する。N-Form はビジネス構造全体の Metaphor であり、そのスコープはエンタープライズレベルである必要がある。

（2）一方、要件定義以降のプロジェクト実施段階では、必要なスコープは、開発対象とその隣接系とのインターフェイスだけであり、対象はずっと小さくなる。

（3）N-Form のエンティティ・クラスはフォーマルなものであり、実装の定義時には整理が必要である。N-Form をこの観点から変形する作業が非正規化(De-Normalization)である。非正規化は、実際のトランザクションに最適化するとともに、親子関係の縮退・排他関係の非正規化・Involution の非正規化・特化と汎化という方法により、手続き的に実施できる。こうして非正規化されたデータモデルは **B-Form** と呼ばれる。

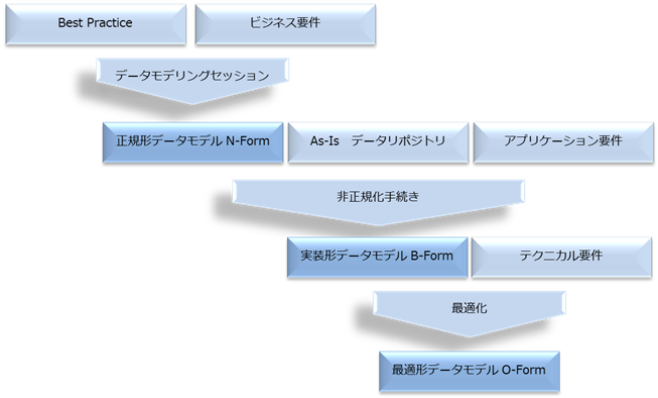


（4）B-Form は、データベースの論理的な形を決めるので、B-Form に基づいて、アプリケーションレベルでのアクセス・クラスである BMO(Business Model Object)を定義できる。BMO は、その中で、論理的なコミット範囲を定義しサービスを提供する。



（5）B-Form に基づいて、データベース処理系の仕様にあわせ、パラレリズム、バフォーマンス等の非機能要件に

より B-Form を再定義したものが、**O-Form** である。O-Form に基づいて、SQL を発行する DAO が定義され、BMO から呼び出されるのである。



4. データモデルとアーキテクチャー

（1）データモデルは、エンタープライズ・レベルでの大まかなプロセス群とデータ群の MECE マトリクスを識別する方法論を使用すると便利である。良く知られたものとしては BSP(Business Systems Planning)がある。N-Form 制作の前準備として、あまり手間暇をかけずに、エンタープライズ・レベルでのマトリクスを作っておくことは、データモデリング中の余計なディスカッションを縮減できる。データモデリングは、N-Form を定義することが肝であり、BDSM 等のセッション方法がモデラーによるファシリテーションによって用いられる。

（2）一方、データモデルを適用するフレームワークとしては、SOA が最適であり、矢張り全体設計への MDA の適用が現代的であろう。ただし、SOA はコリオグラフィによって統合されるサービスは、基本的にすべて疎結合となる。したがって、注意深い BMO,DAO の設計が必要である。DOA (データ中心設計)は、プロセスサイドの定義方法論の欠如が欠陥となり、オブジェクト指向プログラミングとの相性も良くないので、もはや採用すべき時代は過ぎたと思料する。

ただしかし、データモデルそのものは、プロセスサイドからの干渉を極めて受けにくい構造なので、フレームワークなしで設計に臨んだとしても、データサイドでは、さしたる問題は生じない。

5. モデル統治

（1）データモデルは、全てが連結されているので、全体の整合性を維持することに困難がある。作業としては、チェックイン・チェックアウトによりパーツ管理はできても、全体の整合性は少数の主任モデラーによる属人的な管理のもとで行わざるを得ない。

方法論・メカニズム共に今後の課題である。

（2）モデリングアプローチを全面的に採用している海外の大企業では、ビジネスと IT の中間に位置し、初期段階でのプロジェクト・コンテンツを管理する、モデリング組織を置いているところがある。

（3）モデリングツールは、B-Form 以降、個人レベルでの作業を支援するものは多いが、N-Form のディスカッション支援とセッション支援、及びアーティファクト作成支援のための最適なツールはない。実際、セッションでは数百のエンティティを俯瞰する必要があるので、ポストイットとマジックに代わる、手許の PC やタブレットとイメージが連動する壁面スクリーンのディスプレイなどができなければ、ツールとしての実用性は低いと言わざるを得ない。

以上