

株式会社NTTデータ 技術開発本部
梅村晃広、滝本雅之、風戸広史 (mda-info-ml@rd.nttdata.co.jp)

1 はじめに

本連載では、これまで5回にわたりUMLの最新仕様「UML2.0」の入門と題して、記法の変更点とシステム開発における活用事例を紹介してきた。

最終回の今回は、まずUMLのメジャーバージョンアップのきっかけとなり、最近注目を集めているモデル駆動アーキテクチャ (Model Driven Architecture : MDA) の概要をのべる。次にMDAとUML2.0との関係を述べ、さらにMDAを実現するための関連標準についても説明する。そして最後に本連載を総括する。

なお、本稿はUML1.Xについてのある程度の知識を前提としている。UML1.Xの詳細に興味がある読者は、本誌で過去に連載されたUML 紹介記事 [1] 等を参照していただきたい。

2 MDAの概要

2.1 システム開発の課題

システム開発の分野は、その実装技術自体は日進月歩の世界で先進的である。一方で、システムを開発す

る現場に目を移すと、その開発手法には課題も多い。

たとえば、既存システム (レガシーシステム) に対し、新しい技術を採用する場合、対応したアプリケーションを一から開発しなおす必要がある。

また、仕様変更時には、対象となるドキュメントや設計書 (または設計図)、ソースコードをすべて人手で修正しなくてはならない。

現状は、他の業界と比べて発展途上にある。ちょうど、産業革命前の工場制手工業 (マニュファクチュア) の段階といえる。

では、システム開発における、産業革命は、いつ、何をきっかけに起こるのであろうか?

2.2 MDAとは

最近、「MDA」という言葉を耳にしている読者も多いだろう。MDAとは、UMLの標準化も行っているObject Management Group (OMG) が提唱する、いくつかの標準仕様 (後述) を体系化したアーキテクチャである (図1)。

MDAを用いた開発では、実装技術 (How) より実現する業務仕様 (What) を重視してモデリングを行う。

Whatだけ、つまりビジネスルールと制約で構成されるモデルを情報処理独立モデル (Computation Independent Model: CIM) と呼ぶ。また、CIMに業務ロジックを付加したモデルをプラットフォーム独立モデル (Platform Independent Model: PIM)、さらにPIMに、各実装に依

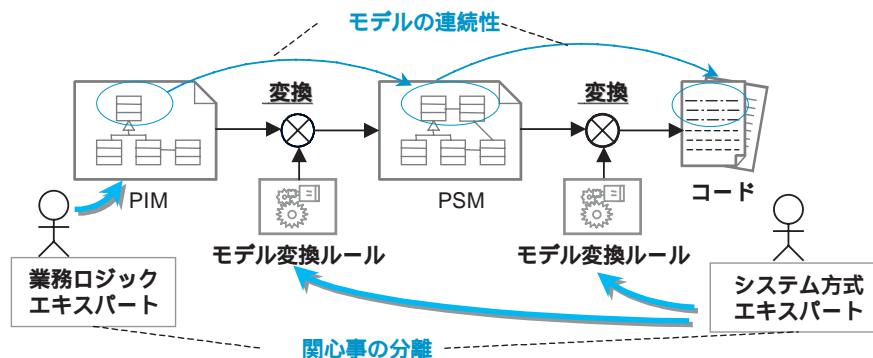


図1 MDAの概念図

存した (How) 情報を付与したモデルをプラットフォーム特化モデル (Platform Specific Model : PSM) と呼ぶ。

これらの主要なモデルに基づき、要件定義、分析、設計、実装、テスト、デプロイ、保守、システム更改などシステム開発のすべてのライフサイクル (工程) において、モデル中心に開発を進めていく手法が MDA の基本となる。このとき、上流工程のモデルから下流工程のモデルへの変換技術の確立が MDA による開発の実現に向けた主要な技術課題である。本連載では、これ以上技術の解説には立ち入らないが、詳しく知りたい読者は参考文献 [2] などを参照していただきたい。

2.3 MDAは4GLか?

MDAは、いわばコンセプトであり、その実態は非常につかみにくい。

一般には、MDAによる開発手法では、「プログラムの自動生成を実現する」という部分に焦点が当たっている。このため、過去のCASEツールや第4世代言語 (4GL) の再来ではないのかと揶揄する声もある。

確かに、これまでの4GLと同様、UMLのモデルを厳密に定義することでプログラムの自動生成が可能となる可能性は高い。しかし、これはあくまでMDAによる開発手法を採用した場合の1つのメリットでしかない。

では、MDAの意義とはいったいどこにあるのであろうか?

2.4 MDAの意義

MDAの意義は、各工程間におけるモデルの連続性の向上と、開発者毎の関心事の分離 (separation of concerns^[3]) にあると筆者らは考える。

モデルの連続性の向上とは、モデルの追跡可能性 (トレーサビリティ) の向上を意味する。現在、OMGで各工程における特定のドメインに特化したモデルの標準化が活発になされている。これらの標準化が進めば、モデル間のマッピング情報を規定することで、ある工程のあるモデル要素に変更を加えた場合、その変更が与える影響範囲の追跡 (インパクト分析) が可能となる (図 2)。

関心事の分離とは、ある役割の開発者が他の役割を持った開発者からの影響を最小限にとどめることを指

す。これにより、作業の並行化・独立性を確保する。

たとえば、業務ロジックのエキスパートがPIMを作成し、またそれぞれの要素技術に特化したシステム方式エキスパートが変換ルールを規定することで、PIMからPSMへの変換を実現することができる。

2.5 MDAによる開発手法が開発者に与える影響

次に、MDAによる開発手法が実用化された場合のメリットを、開発者の視点から、より具体的にのべる。

(1) 再利用性の高い設計

PIMは前述のとおり、特定の实装に依存しないモデルとなる。

たとえば、J2EE環境で動作しているシステムがあるとする。このときこのPIMを利用して.NET環境へ

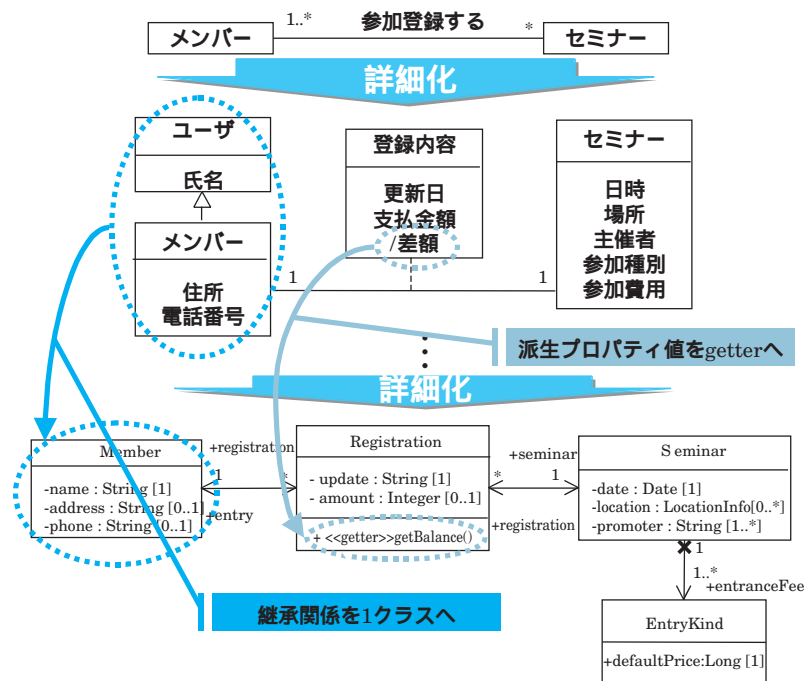


図 2 モデルの詳細化の例

の移行が簡単に可能となる。

(2) システム開発のコスト(開発費用・期間)の低減

前述の関心事の分離により、作業効率が向上し、開発コストの低減につながる。

(3) システムの品質の向上

MDAによる開発手法を採用した場合、開発者はこれまで以上に厳密なモデリングを要求される。より厳密に定義されたモデルは計算機上で実行可能である。もし、モデルを実行可能なレベルで定義することができれば、そのモデルを用いた機能テストが設計段階で可能となる。これは、バグの早期発見、品質の向上に非常に有効であるといえる。

以上のようにMDAによる開発が実現すれば、システム開発現場における産業革命の大きな起爆剤となる可能性を秘めている。広い範囲で実用化されれば、システムの開発方法論に新たなパラダイムシフトを起こすであろう。

3 MDA とUML2.0

3.1 OMG標準のモデリング層

UML2.0とMDAの関係についての議論を深めるためには、OMG標準で定義されているモデリング層についてその概要を理解する必要がある。

OMG標準が採用しているモデリング層は以下の4階層がある。(図3に例を示す)。

・M0層:インスタンス

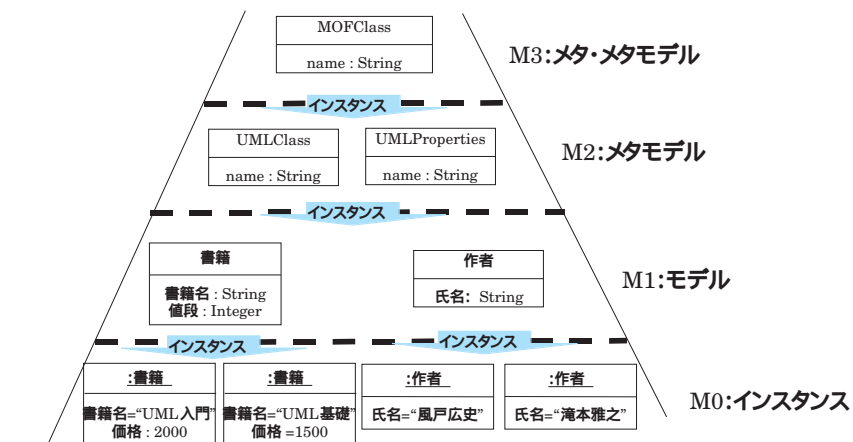


図3 モデリング層の一例

- ・M1層:モデル
- ・M2層:メタモデル
- ・M3層:メタメタモデル

M0層とM1層は、一般の開発者にもわかりやすい。通常のオブジェクト指向プログラミング言語でいうクラスがM1層であり、そのインスタンスがM0層と対応する。

M2層とはメタモデルと呼ばれ、モデルの文法を定義している。つまり「クラスには、クラス名とプロパティ(属性)と操作が定義できる」ということなどを規定している。

M3層は、メタメタモデルと呼ばれ、メタモデル同様、M2層モデルの文法ルールを規定している。つまり、OMG仕様の記法を規定しているものと理解すればよい。OMG標準では、このM3層を、後述する「MOF(Meta Object Facility)」という仕様で統一することを推し進めている。この統一によって各仕様間の整合性向上を目指している。

3.2 UML2.0がメジャーバージョンアップである訳

本連載で解説を行ってきたUML2.0へのメジャーバージョンアップは、第1回でも述べたが、MDA対応という意味合いが強い。このためUML2.0の最大の目標は、モデルの定義を厳格にすることにあった。つまり、UMLのM2層(メタモデル)の整合性・厳密性を高めることに主眼がおかれている。このため、UML2.0は、これまでのバージョンに比べ、より正確なモデルを作成できるようになった(詳しく知りたい方は参考文献[4]と[5]を見比べていただきたい)。

M2層の変更自体、つまり詳細な文法規則の変更は、必ずしも通常のシステム開発者が知る必要はない。しかし、より厳密なUMLを書くことによって、MDAの恩恵を得ることができるため、UML記述にはUMLエディタを活用して正しいモデルを記述するよう心がけたい。

4 MDAに関連する標準

MDAは単体で用いる仕様ではなく、UMLをはじめとする多くのOMG標準を連携させた上位概念である。OMGでは現在、UML 2.0以外にもMDA対応を意識した標準化・仕様改訂が活発に行われている。

ここでは、MDAと深く関連するUMLプロファイル、MOF、XMI、CWMなどの概要とその標準化動向について述べる。

4.1 UML プロファイル

第1回の連載でも述べたように、UMLは多種多様な方法論をもとにモデルの記法（モデリング言語）のみを標準化したものである。

つまり、UMLは方法論ごとの共通部分を取った最低限の合意であり、すべてのニーズを満たすようには設計されていない。そのかわり、UMLには方言を定義するための拡張メカニズムが用意されている。これがUMLプロファイルである。

UML 1.Xではステレオタイプによるラベル付け、タグ付き値によるブ

ロパティ値を組み合わせた簡易的な拡張方式であったが、UML 2.0のプロファイルはメタモデルと同等の構造を持つため表現力が高い。また、プロファイルを定義するための記法も新しく定義されている（図4）。

MDAではCIM、PIM、PSMといった抽象度や意味の異なるモデルを記述するためにUMLプロファイルを利用する。代表的なUMLプロファイルには以下のものがある。

- ・ UML Profile for CORBA
- ・ UML Profile for EDOC
- ・ UML Profile for EAI

4.2 MOF^[6]

前述のUMLプロファイルはメタモデルを「書き足す」ことでUMLの適用範囲を拡張する。このように、M2層をデータとして扱える能力を与えているのがM3層のMOFである。

MOFではUMLのクラス図と同じ構成要素を用いて「クラスは属性と操作を持ち、関連で結ばれる」といった構文を記述する（図5）。

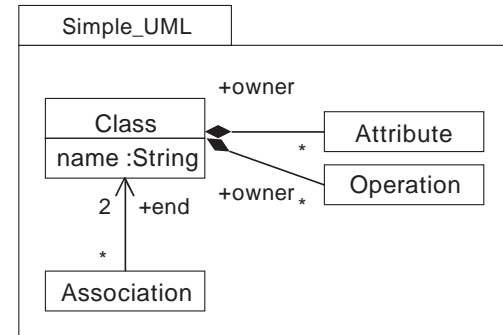


図5 MOFで記述した「クラス」の簡略版メタモデル

MOFを用いれば、M2層に独自のモデリング言語を作り出すことすら可能だが、現在市販されているUMLモデリングツールでは、独自に追加したモデルの構築ができない。このため、専用のツールの開発が必要になる。可能な限りUMLとプロファイルの範囲でモデリングできることが望ましい。

UML 1.XとMOF 1.Xの仕様では概念の重複や不整合があったが、MOF 2.0では仕様のモジュール化が行われ、UML 2.0 Infrastructureを再利用しているためにUML 2.0と一貫性のある仕様になっている。

4.3 XMI^[7]

MOFでメタモデルが記述されている場合、モデリングツールはどのようにモデルを操作し、どのような形式で保存すべきであろうか？

たとえば、クラス図をXML文書に保存したいとする。XMLは相互運用性に優れた記述形式であり、ツール間でモデル情報を交換する場合にも適した形式だろう。しかし、ツール実装者がXML文書の形式（スキーマ）を独自に決めてしまうと、

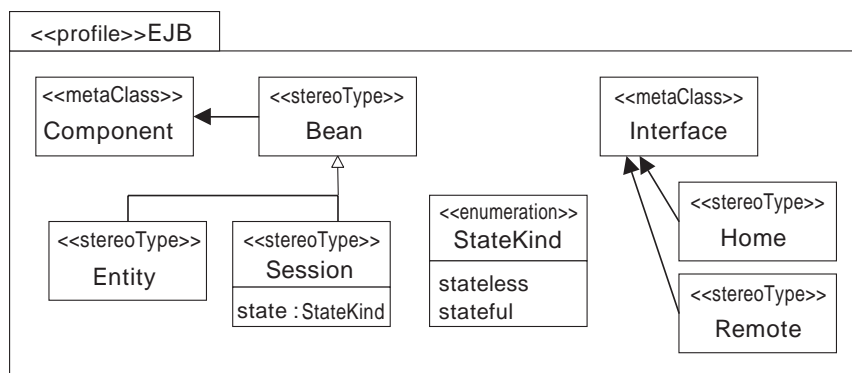


図4 クラス図を用いたUMLプロファイルの定義

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xmi="http://www.omg.org/XMI"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:Simple_UML="http://www.nttdata.co.jp/Simple_UML"
  targetNamespace="http://www.nttdata.co.jp/Simple_UML">
<xsd:import schemaLocation="XMI.xsd" namespace="http://www.omg.org/XMI"/>
<xsd:complexType name="Class">
  <xsd:choice minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="association" type="Simple_UML:Association"/>
    <xsd:element name="attribute" type="Simple_UML:Attribute"/>
    <xsd:element name="operation" type="Simple_UML:Operation"/>
    <xsd:element ref="xmi:Extension"/>
  </xsd:choice>

```

図6 XMIによって生成されたXML Schema (一部)

同じクラス図を書いても他のツールと互換性のないXML文書で保存される恐れがある。そこで、スキーマの決め方に何らかの標準が必要になる。

OMGではXMI (XML Metadata Interchange) という仕様によってM2層 (メタモデル) をXMLスキーマ、M1層 (モデル) をXML文書へと変換する方法を標準化している。

XMI 1.Xではスキーマ文書の形式にDTDを用いるが、XMI 2.0はより表現力の豊かなW3C XML Schemaを用いるように変更された。また、自動生成されたXML文書の可読性も向上している。図5のメタモデルをXMI 2.0を用いてXML Schemaに変換すると図6のようになる。

UMLメタモデルからXMIを用いて生成されたスキーマは特にUML DTDと呼ばれており、多くのUMLモデリングツールがこのDTDを用いたモデルのインポート/エクスポート機能を備えている。

XML以外にもMOF to CORBAやMOF to Javaへのマッピングも標準化が行われており、一度MOFでメタモデルを記述すれば、様々な技術を利用してモデルにアクセスす

ることができる。

4.4 CWM^[8]

UMLはシステムをオブジェクト指向アプローチでモデル化するためには適した言語であるが、データのモデリングやデータ処理の記述には不向きである。OMGではUMLとは別にCWM (Common Warehouse Metamodel) と呼ばれるモデリング言語の標準化を行っており、以下の用途に向けたメタモデルを定義している。

- ・データウェアハウスの構築
- ・オンライン分析処理 (OLAP)
- ・データマイニング
- ・データ変換

MDAでは既存資産のシステムと新規開発するシステムの統合、さらには将来開発するシステムとの統合を目指している。その文脈において、プラットフォームに依存しないメタデータの記述やその交換形式としてCWMはMDAの重要な一角を担う標準仕様である。

UMLと同様にCWMもMOFを用いて記述されている。CWM 1.XはUML 1.XやMOF 1.Xと概念上の不整合があったが、CWM 2.0ではこの点

の改善を目指しており、UMLのモデルとCWMのモデルを相互に連携するようなツールの出現も期待できる。

5 おわりに

5.1 MDAの今後について

第2章で述べたWhat (仕様) とHow (実装) の分離ということは、実際にはMDAということばが出来る以前からソフトウェア工学の分野では重要視されている (例えば、参考文献 [11] の第6章でもこの指摘がされている)。

また、WhatとHowに限らない一般的な関心事の分離ということであれば、MDAとは必ずしも関係のないアスペクト指向という文脈の中でも議論されている。

Whatの記述を厳密に行うことで品質の高いソフトウェアを開発しようという考え方は、昔から形式的手法という分野で行われてきた研究と方向性としては同一である。

開発生産物の追跡可能性も新規の概念というわけではない。

このように、個別の考え方としては、MDAは新規の提案をしているわけではない。

しかし、上に述べたようなさまざまな提案は、学会などの限られた範囲で議論されるにとどまり、なかなか一般の開発場面には取り入れられてこなかった。

この原因は、上記の項目のそれぞれによっても異なるが、筆者らが詳しい形式的手法の分野について言え

ば、それぞれの研究者が自身の学問的な興味の追及に熱中するあまり、一般の開発場面に伝える努力を怠ってきたことも挙げられるであろう。その結果として、それぞれの提唱者ごとに違う方法論や記法を使っていたり、高度な数学的概念や数式の利用を要求したり、実際の開発現場の実状と乖離した前提を置いたりすることが起こっていた。

これに対して、OMGの提唱するMDAは、UMLという開発現場に広まりつつある記法をベースに用いていることで、上述のような概念を開発現場に受け入れられる形に構成し直している点が新規性と言えよう。

本連載^[9]の第1回で述べたUMLの特徴である、(1)視覚的にわかりやすい、(2)標準として定義されている、ということもMDAは利用することで、過去のように一般の知らないところでの議論ではなく、現実の開発現場を巻き込んだ手法として定着していく可能性を持っているのである。

なお、過去の学会等の議論と現在のMDAの関係について、例えば、参考文献[10]の101ページあたりのMDAと形式的手法の関係の説明も御参照いただきたい。

5.2 本連載終了にあたり

従来のUML1.Xでは、上記の概念と組み合わせる上で厳密さの点などで問題が生じ、その問題を解決すべく提案されたのが本連載で説明してきたUML2.0である。このあたりの経緯については、本連載の第1回

に述べた通りである。また、具体的なUML2.0の変更点については、本連載の第2回、第3回、第4回に述べられている。第5回は、直接MDAによる開発を説明しているわけではないが、UML2.0によって従来に比べてより厳密にモデリングできる部分などが説明されている。

それでも、UML2.0ですべてが済むというわけではない。それだからこそ、今回の第3章で述べたさまざまな標準が必要となってくる。UMLプロファイルやMOFは、UMLを自分で拡張したり、独自のものを定義したりできる機構であるから、これによって表現力は向上するが、その反面、再びそれぞれの人が独自の記法や方法論をバラバラに使い出すことになり得る危険性もはらんでいる。

また、これとは別に、従来のUMLの利用者からは、UML2.0になることでUMLが使いにくくなるという批判も起こっている。これについては連載の第4回の最後にも述べた通りである。

本連載の立場も、UML2.0があれば他に何も必要ない、というものはなく、また、MDAで何でもできるということを主張するものでもない。まさに参考文献[11]が言うようにソフトウェア開発に銀の弾丸など存在しない。

しかし、これまで学問の世界に閉じていたようなさまざまな概念がこれを契機に一般の開発場面に広まり、それによってソフトウェア開発の効率化や高品質化がいくらかでも進めば、それに越したことはない。

本連載はこれで最終回である。最後に、UML2.0がそれぞれの開発現場でうまく利用され、開発の効率化や高品質化がすすむことを、筆者らも願って本連載を締めくくりたい。

参考文献

- [1] 山本修一郎,UMLの基礎と応用(連載記事)ビジネスコミュニケーション,Vol.38, No.9, 2001- Vol.40, No.3, 2003.
- [2] OMG, MDA Guide, <http://www.omg.org/docs/omg/03-06-01.pdf>, 2003
- [3] 千葉 滋,アスペクト指向ソフトウェア開発とそのツール,情報処理 45巻1号,pp.28-33(2004年1月号)
- [4] OMG, UML1.5 Specification, <http://www.omg.org/technology/documents/formal/uml.htm>, 2003
- [5] OMG, UML2.0 Superstructure Final Adopted specification, <http://www.omg.org/UML>, 2003.
- [6] OMG, Meta Object Facility (MOF) 2.0 Core Specification, <http://www.omg.org/cgi-bin/doc?ptc/2003-10-04>, 2003
- [7] OMG, XML Metadata Interchange (XMI) Specification, <http://www.omg.org/technology/documents/formal/xmi.htm>, 2002-2003
- [8] OMG, Common Warehouse Metamodel (CWM) Specification, <http://www.omg.org/technology/documents/formal/cwm.htm>, 2003
- [9] UML2.0入門(本連載) 第1回~第5回, ビジネスコミュニケーション, Vol.41, No.4-7, 2004.
- [10] 黒川利明,ソフトウェア入門,岩波新書, 2004.
- [11] ブルックス,(滝沢他訳),人月の神話(原著発行20周年記念増訂版),アジソン・ウェスレイ,1996.

本連載に関するお問い合わせ先

株式会社 NTT データ
技術開発本部 開発担当
TEL: 03-3523-8142
E-mail: mda-info-ml@rd.nttdata.co.jp
(担当: 滝本・風戸)