

# UML 2.0 概要

株式会社NTTデータ 技術開発本部

神谷慎吾、岡秀樹、梅村晃広、明関恵美、我妻智之、滝本雅之

E-mail:mda-info-ml@rd.nttdata.co.jp

## 1 本連載について

UML ( Unified Modeling Language ) は、オブジェクト指向開発のためのダイアグラム記法体系である。CORBA ( Common Object Request Broker Architecture ) 標準の策定団体として知られるOMG ( Object Management Group、<http://www.omg.org/> ) が、1997年に最初の業界標準であるUML Ver.1.1を発表して以来、世界的に普及が進んでいる。そしてUMLに対する多様な要求に対応するため、2003年夏にはメジャーバージョンアップの草案が一般公開された。これが、本稿の主題となるUML 2.0である。

本稿では、6回の連載形式で、UML 2.0を中心としたUML 最新動向を紹介していく。あわせて、2001年からOMGのスローガンとなっているMDA ( Model Driven Architecture ) についても、UMLとの関連において紹介する予定である。実は両者には密接な関係があり、UML 2.0改訂のかなりの部分は、MDAを実現するための意味論の厳密化だと言っても過言ではない。

なお、読者は従来のUML 1.Xについては既知であると想定し、記事のスタイルとしては、連載第1回目

に、UML 自体の一般的な解説とUML 2.0の概要を示し、第2回目以降は最新動向の話題に集中していく。そのため、UML 最新動向には興味があるものの、従来のUMLにあまり詳しくないという方は、UML 自体の基本的説明がないため読みにくいであろう。そのような方は、恐縮だが、本誌に過去に連載されたUML 紹介記事<sup>[1]</sup>等をまず参照していただきたい。

## 2 UMLの変遷と最新動向

### (1) UMLまでの流れ

UMLはオブジェクト指向開発のためのダイアグラム記法体系であり、Rational Software社(注:現在はIBM社に買収されている)に結集した3人の著名なオブジェクト指向開発方法論提唱者であるG. Booch (Booch法)、J. Rumbaugh (OMT法)、I. Jacobson (Objectory/OOSE法)が、乱立するオブジェクト指向開発方法論を収束させるための統一記法体系として1995年に発表した。(彼らはUMLの生みの親として、しばしばThree Amigosというニックネームで呼ばれる。)

したがって、UMLは、もともとRational Software社の記法体系な

のだが、IT業界に広く普及させて業界全体の生産性/品質向上に寄与するためには業界標準にするべきであるとの動きがあり、OMGがUMLの標準化推進を行うこととなった。

### (2) メジャーバージョンアップに至るまで

UML利用者の増加とともに、OMGにはUMLに関する多数の意見や改善要望が寄せられ、OMGはマイナーバージョンアップを繰り返してきた。執筆時点での最新版はUML Ver.1.5であるが、OMG自体、数年前からマイナーバージョンアップの積み重ねでは多様な要望に対処し切れないと認識しており、2000年にメジャーバージョンアップ活動をスタートさせた。

### (3) UML 2.0の最新情報

UMLに対する多様な要求に抜本的な対応を行うため、長い間議論が積み重ねられたが、2003年夏によりやく収束し、ほぼ安定したUML 2.0の草案となった。OMGの標準化手続きに従い、正式版の公表は2004年の夏または秋になる見込みだが、技術的にはほぼ確定したと考えられる。また草案そのものも一般公開されている。

### 3 UML の特長

では、ここで、(UML1.Xについて既知である読者にとっては当たり前のことかもしれないが)オブジェクト指向でのシステム開発における、UMLのメリットについて説明する。

#### (1) 視覚的にわかりやすい

オブジェクト指向開発では、システムの内容をモデルで記述する。そのため、複雑なシステムでも視覚的に全体を把握することができる。もしこれを自然言語(例えば日本語)で書いてあったとすると、短時間で全体を把握することは難しいであろう。

たとえば、システムの持つ各機能がそれぞれ数行ずつ日本語で書かれている場合、それら一つ一つを理解することは難しくなく、システム全体でどんなことができるかを把握

するのは難しくはないであろう。では、ある2つの機能は似た機能を持っている、または、ある機能は別の機能を必ず必要とする等、機能同士の構成を日本語で表そうと思った場合はどうだろう?誰がその機能を使うかを表す場合はどうだろう?そのような場合、短いページでは表すことができずシステム全体を理解するには時間がかかるかもしれない。UMLはこれを解決する。UMLのダイアグラムを用いてシステムを記述することで、システムの機能一覧、機能同士の関係、誰がその機能を使うことが許されているか等を少ないスペースに記述でき、なおかつ一目瞭然になるのである。

#### (2) 標準として定義

「視覚的にわかりやすい」というメリットは実はUMLを用いなくても享受することができる。システム全体をわかりやく自由な記述方法で

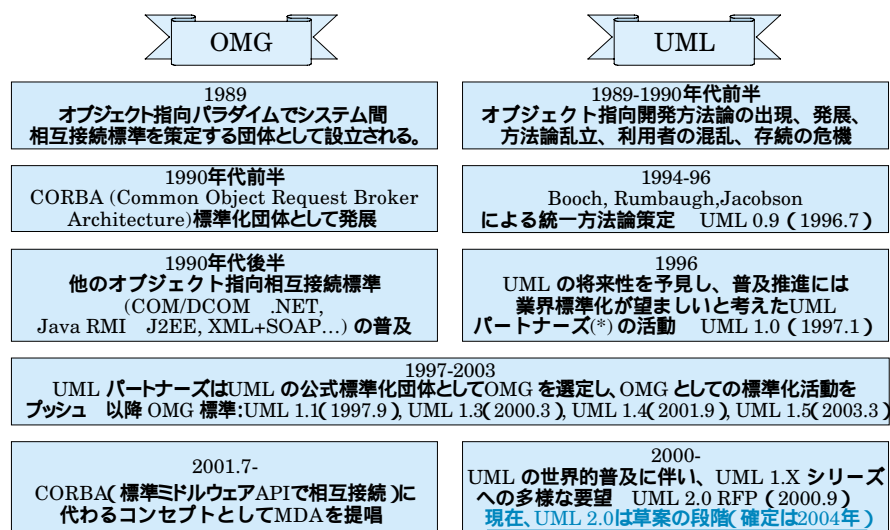
図を作成すればよいのだ。では、それらの図をUMLを用いて記述するメリットは何であろうか?

自由な記述方法でシステムを表す図を作成した場合、作成した本人にはよくわかって他の人にはわかりにくい、ということはよくある。また、同じグループ内や社内ではある程度標準化されたわかりやすい図であっても、社外に対しては口頭で説明を添えないとわかりにくい、ということもあり得る。それに対して、UMLでは、ダイアグラム内のアイコンや、アイコンからアイコンへの矢印はどのような意味を持っているか、といった表記方法が標準として定義されている。そのため、UMLを使ってシステムを表した場合、書いた本人以外がその図を見ても、作成者の意図しない解釈をされることはないのである。

また、表記方法が標準で定義されているUMLのメリットは、作成された図を見るときだけでなく、図を作成する場合にも現れる。UMLを記述するためのツールがたくさんベンダから提供されているが、どのツールを用いてもほぼ同じ図を作成することができる。このことは、一つのツールやそのバージョンに依存せずにUMLで記述を行うことができることを表している。

#### (3) 開発プロセスから独立

UMLはオブジェクト指向開発のためのモデル記法を定めたものである。ただし、オブジェクト指向開発と一口に言っても、XPやRUPなど



\* DEC, HP, I-Logix, Intellicorp, IBM, Icon Computing, MCI Systemhouse, Microsoft, Oracle, Rational Software, TI, Unisys

図1 UMLの変遷

様々な開発プロセスがある。それぞれのプロセスでは工程毎に用いるモデルや、それらの細かい記述方法が異なるのが普通である。UMLでは、開発に用いるプロセスに関係なく記法だけを定めているため、どの開発プロセスでも利用することができる。また、UMLでは、ユースケース図、クラス図、相互作用図など、システムの複数側面を表すことのできるダイアグラムがいくつか用意されている。それぞれの開発プロセスの工程で、どのダイアグラムを利用するかを自由に選択することができる。

たとえば、RUPでは、1回のイテレーション（繰返し）内に要求、分析、設計、実装、テストワークフローが存在する。要求ワークフローでは、ユースケース図やアクティビティ図、クラス図を、分析ワークフローでは、コラボレーション図、クラス図、パッケージ図等を用いる。

#### 4 UML 1.Xの課題と要因

まず図3にUMLを使った開発の概要を示す。UML1.Xでは、モデルを作成するエンジニアがUMLの規格（メタモデル及び記法）に応じてモデルを作成する。また、必要があれば、モデル情報をツール間で交換することも想定している。

UML1.Xの目的は、3で述べたように、多数乱立していた方法論を統一して、記法を標準化することで、要員間のコミュニケーションを向上させることにあった。

このような経緯で作られた

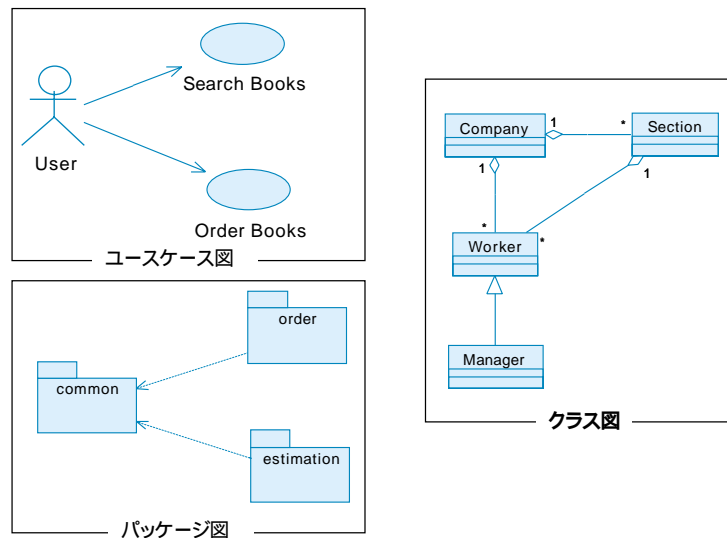


図2 代表的なUMLダイアグラム

UML1.Xだが、いくつかの問題を含んでいることが分かってきた。ここでは、執筆時点で最新の規格であるUML 1.5における課題とその原因を紹介する。

##### (1) モデルが複雑で巨大になる

3で示したように、現在のUML1.Xの規格は、様々な方法論で用いられていた記法の寄せ集めに過ぎなかった。そのため、それぞれのモデルが独立ではなく、重複するもの、依存関係にあるものが多数できてしまった。さらに、モデルの役割やモデル間の関係について明確に定義していなかったため、利用者が混乱してしまうことがあった。

たとえば、複雑なモデルを構造化して記述する方法が定義されていなかったため、複雑で巨大な処理フローをシーケンス図で記述する場合に画面や紙面に入りきれない程大きくなるといったことがあった。

##### (2) 厳密さにかける

UML1.Xでは図3に示すように記述するモデルの意味を規定したメタモデルを提供している。例えば、「クラスとオペレーションはどういった関係にあるのか」とか「クラス間の関係はどうなるのか」といったものを規定したものである。

UML1.Xの場合、このメタモデルに厳密さがなかった。このため、膨大なメタモデルを定義しているにも関わらず、モデルの記述方法（例えばシーケンス図のループや分岐の記述方法）を利用者が決めなければならず、結果的にモデルが曖昧になってしまうことがあった。

##### (3) 拡張性に乏しい

UML1.Xでは、メタモデルの拡張方法が十分に提供されていなかった。例えば、UML1.Xでは、拡張記法として、ステレオタイプとタグ値が利用できた。しかし、この記述能

力および厳密性には限界があった。このため、例えば、業務プロセスモデルの構築であるとか、特定のプラットフォームに向けた（EJBや.NETなど）モデルの構築をする場合には、メタモデル自体を特定ドメイン向けに変形することで対応することが多かった。その結果として、モデルの互換性や整合性が失われることとなった。

#### (4) ツールの実装に依存している

(2)で述べたように、厳密さに欠けるため、UML1.Xの仕様に準拠したモデルを構築する場合、あるいはツールを提供する場合は、厳密に定義されていない部分については、モデルの作成者、またはツールの実装者に依存してしまうことが多かった。これによって、エンジニア間でのコミュニケーションが難しい、あるいはツール間でのデータの交換が難しいといった問題が生じていた。

#### (5) ダイアグラム情報の交換が規定されていない

UML1.Xは要員間のコミュニケーションの向上を図ることを目的としていた。この点でUML1.Xは、設計情報を視覚的に分かりやすい形で記述しているため、成功したといえる。

また、異なるツール間で設計データを交換するための規格として、XMIが定義されている。これによって、ツール間のコミュニケーションの向上を図っていた。しかし、現在のXMIの定義では、モデルデータの交換のみの規定であり、ダイアグラム情報の交換方法は規定していない。このため、ツール提供者が独自の方法でダイアグラム情報の受け渡し方法を実装していた。結果として、異なるツール間のデータの受け渡しが困難になるという問題が生じていた。

たとえば、モデル情報のみしか交換していない場合には、図形情報は受け手の側で作成しなければならない。モデル要素の配置を自動的におこなうと可読性の悪いモデルになっ

てしまうことが多かった。

#### (6) 記述能力が低い

先に述べたように、UMLは様々な方法論を融合して記法を統一化した。そのため、様々なモデルを記述することができた。しかし、上流工程や下流工程では依然として記述能力が低いことが指摘されている。

特に、上流では業務モデルを記述する場合、下流ではソフトウェアの動作を記述する場合にUML1.Xでは記述能力に限界がある。

## 5 UML 2.0の概要

UML 2.0への改訂は4で述べた課題の解決（少なくとも改善）を目指したものである。ここでは、2003年に一般に公開された草案に基づきその概要を紹介する。

なお、本連載では、読者の興味が最も高いと予想されるUML2.0の記法（SuperstructureとOCL）に焦点を絞り解説する予定だが、初回である今回は残りの仕様についての概要にも一部触れている。

#### (1) UML2.0の4仕様による解決

UML2.0は4つの仕様で構成されている。はじめに、これらの仕様が先に述べた課題の解決にどのように寄与しているかについて述べる。

##### (a) Superstructure

Superstructureでは、図の記法が定義されており、一般的な開発者にとって最も関わりの深い仕様といえる。今回の改訂では、新しいダイ

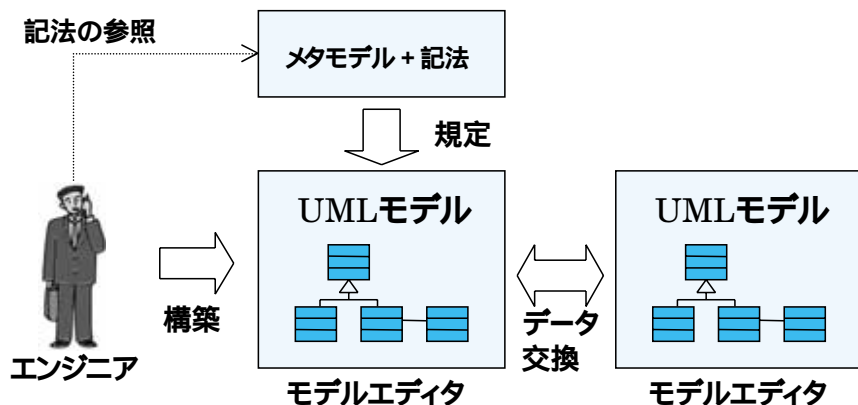


図3 UMLを使った開発概要

アグラムが追加されたほか、既存の図にも多くの記法がサポートされた。特に階層化の概念をサポートすることで、同一ダイアグラムであっても概念レベルから、より詳細なレベルまで様々な粒度で（互いに関連をもって）表現が可能となった。またメタモデルが大幅に整理され、これまで以上に厳密な定義が可能となった。

#### (b) Infrastructure

InfrastructureはUML2.0の基盤部分といえる仕様であり、メタモデルを定義するのに必要な各要素を定義している。このようなメタモデル自体の階層化は、主にUMLメタモデルを組み込んだツール群に対して恩恵を与える。また、InfrastructureはUMLのみならずあらゆるメタモデルの基盤となる部分を抽出しており、これを用いてメタモデルを独自に定義する拡張性を有する。

#### (c) OCL

オブジェクト指向による開発思想

の一つである“契約に基づく設計<sup>[2]</sup>”では、事前、事後条件などを形式的に表明することが必要である。このような制約（条件）を記述するための記法として、UMLでは、オブジェクト制約言語（OCL、テキストベースの記法）を用意している。これまでOCLにはメタモデルが定義されていなかったが、UML2.0への改訂にともない定義された。これにより、条件記述の部分においても、厳密な仕様が規定されたといえる。

#### (d) Diagram Interchange

Diagram Interchangeはツール間の互換性確保のため策定された。これにより、異なったツール間でもダイアグラム自体の情報（色、配置など）が引き継げるようになった。

以上のように、UML2.0では課題の解決を中心に、特にその意味論の拡張と厳密化がなされている（図4）。

## (2) ダイアグラム体系

以降はSuperstructure<sup>[3]</sup>について論じる。ここでは、全体ダイアグラム体系について述べる。

UML2.0では、大きく構造モデルと振る舞いモデルの2つに分類された13のダイアグラムが規定されている（図5参照）。そのほかにも、記法が定義されていないアクションやダイアグラムと等価なテーブル（表）も定義されている。

## (3) 各図の概要

次にUML各図の概要を説明する。

なお、各図の詳細は次回以降に説明する。

### ・クラス図（既存）

主にシステム内部の構造とその関係を表す。

### ・パッケージ図（既存）

クラスのグループをグループ化し、その関係と各要素の一意名称の範囲（ネームスペース）を明確にする。

### ・オブジェクト図（既存）

オブジェクト（クラスのインスタンス）の関係を表す。

### ・コンポーネント図（既存）

複数クラスで1機能を実現するためのクラスの集合と内部構成要素を定義する。

### ・配置図（既存）

現実世界の物理的配置を表現する。

### ・ユースケース図（既存）

要求仕様やシステム境界を明確にする。

### ・シーケンス図（既存）

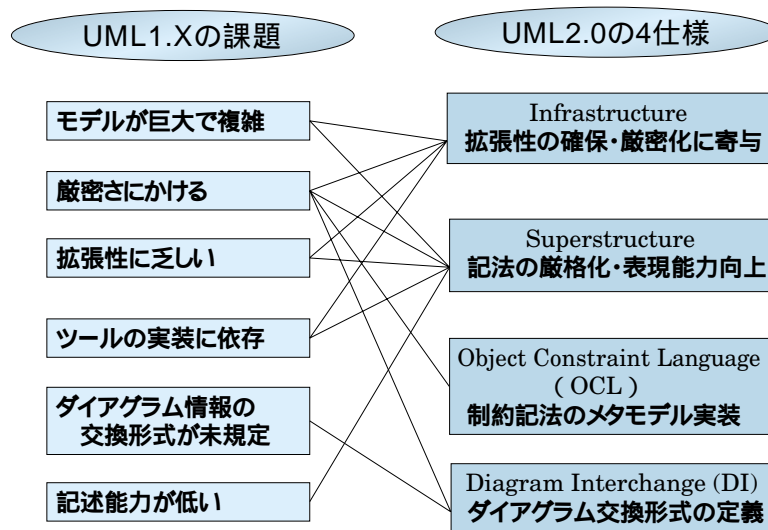


図4 UML1.Xの課題とUML2.0の4仕様の関係

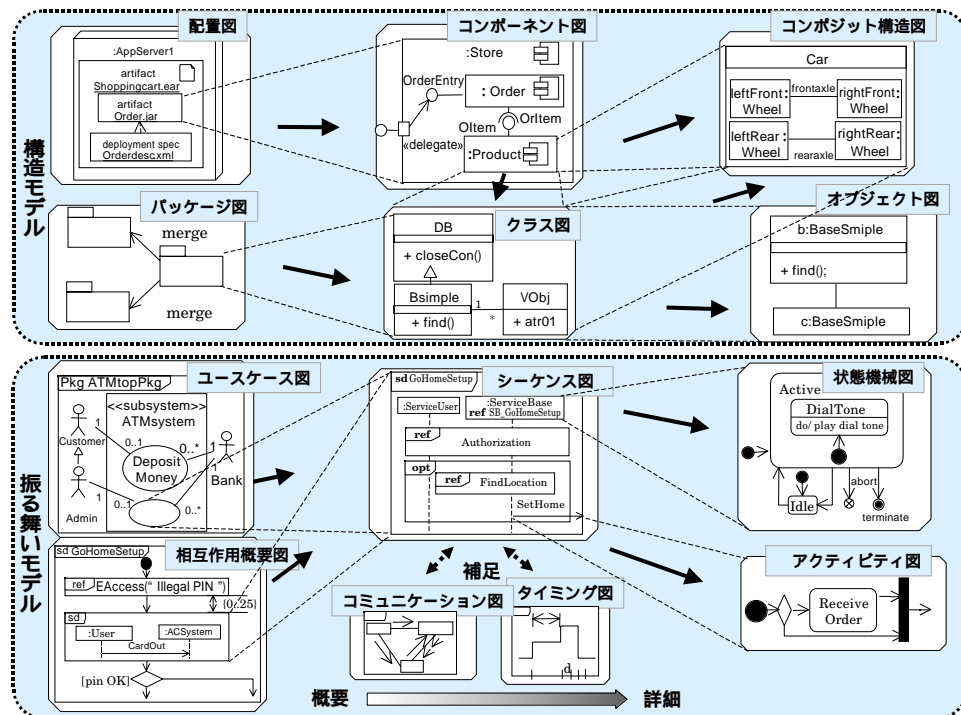


図5 ダイアグラム体系 (UML2.0)

相互作用図の一つで、ユースケースの詳細化などでオブジェクトやサブシステム間の相互作用を表現する。

・ **コミュニケーション図 (既存)**

相互作用の振る舞いを定義する。

・ **相互作用概要図 (新規)**

相互作用図間の制御フローを表現する。アクティビティ図の記法をほとんどそのまま流用している。

・ **タイミング図 (新規)**

組み込み制御系システムでのタイミングや経過時間を表現する。

・ **アクティビティ図 (既存)**

アクション (ロジック) の実行手順やビジネスフロー、データフローを表現する。

・ **状態機械図 (既存)**

オブジェクトの状態遷移とサブシステムの状態遷移を表現する。プロ

トコル状態機械図も記述可能となっている。

(4) いつから習得すべきか?

UML2.0の一番の変更点はメタモデルである。では実際に利用する開発者のメリットは何か? 2で述べたように、UML2.0の仕様は安定したとはいえ、いまだ変更される可能性がある。このため、「それならまだ知る必要がない」という声も聞く。

しかし、UML2.0は、UML1.Xと比較して、より厳密で広い表現能力を持ち合わせている。さらに、近い将来、UML2.0 (あるいはその後継) で記述されたモデルを中心にした開発 (関係者のコミュニケーションが図られる世界、つまり冒頭で述べたMDA) が広まる公算は高い。よっ

て、開発者は早めに習得しておいて損はないといえよう。

## 6 まとめ

第1回の今回は、UML2.0の背景とUML2.0の概要について紹介した。次回から個々のダイアグラムにフォーカスして、その変更点と追加点を解説していく。

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] Meyer, Bertrand, Object-Oriented Software Construction (Prentice Hall International Series in Computer Science), Prentice Hall, 2000  
 [3] OMG, UML2.0 Superstructure Final Adopted specification, <http://www.omg.org/uml/>, 2003