オブジェクト指向のための文献紹介

妻 木 俊 彦

1. はじめに

■初歩から学習したい方のために

・土居範久編「オブジェクト指向のおはなし」，日本規格協会，1995

・岩田裕道他「ゼロからわかるオブジェクト指向の世界」，日刊工業新聞社，1996

〈寸評〉［土居 95］は，ソフトウェア工学の視点からのオブジェクト指向に関する全般的な解説書．［岩田 96］は，初心者向きの読みやすい解説書．

■基礎をあらためて確認したい方のために

・B. Meyer/二木厚吉監訳「オブジェクト指向入門」，アスキー出版，1990
（原書：“ Object-Oriented Software Construction ” Prentice-Hall, 1988；現在第 2 版が出ている）

・所真理雄他「オブジェクト指向コンピューティング」，岩波書店，1993

〈寸評〉［Meyer 88］は，プログラミング言語の立場からのソフトウェア品質論．［所 93］は，言語やデータベースなどにおけるオブジェクト指向計算モデルの解説書．

■モデリング技法を学びたい方のために

・M. Jackson/大野侑郎・山崎利治監訳「システム開発 JSD 法」，共立出版，1989
（原書：“ System Development ” Prentice-Hall, 1983）

・J. Runbaugh/羽生田栄一監訳「オブジェクト指向方法論 OMT」，トッパン，1992
（原書：“ Object-oriented modeling and design ” Prentice-Hall, 1991）

・I. Jacobson/西岡利博・渡辺克宏・梶原清彦監訳「オブジェクト指向ソフトウェア工学 OOSE」，トッパン，1995
（原書：“ Object-Oriented Software Engineering ” Addison-Wesley, 1992）

・T. Booch/山城明宏他訳「Booch 法：オブジェクト指向分析と設計　第 2 版」，アジソン・ウェスレイ・パブリッシング・カンパニー・ジャパン，1995
（原書：“ Object-Oriented Analysis and Design with Application, 2nd edition ” Addison-Wesley, 1994）

・妻木俊彦・岩田裕道著「ビジネス・アプリケーションのためのオブジェクト指向モデリング法」，1999

〈寸評〉［Jackson 83］は，オブジェクト指向ではないが，モデリングについて知るには最適．［Runbaugh 91］，［Jacobson 92］，［Booch 94］は，3 大方法論と呼ばれているもの．

■オブジェクト分析・設計のパターンを学習したい方のために

・E. Gamma/本位伝真一他監訳「デザインパターン」，ソフトバンク，1995
（原書：“ Design Patterns ” Addison-Wesley, 1995）

〈寸評〉［Gamma 95］は，デザイン・パターンの原典本，ここで紹介されているパ

ターン名は業界標準になりつつある．

■統一モデリング言語（UML）を学びたい方のために

・H. E. Eriksson/杉本宣男他監訳「UML ガイドブック」トッパン，1998
（原書：“ UML Toolkit ” Wiley, 1998）
〈寸評〉UML の解説に関する邦文の書籍では，今のところ［Eriksson 98］が最も充実している．

■オブジェクト指向データベースを学びたい方のために――

・M. E. S. Loomis/野口喜洋訳「オブジェクトデータベースのエッセンス」トッパン，1996
（原書：“ Object Database ： Essential ” Addison-Wesley, 1995）
〈寸評〉［Loomis 95］は，実際のアプリケーション・システム開発の視点からのオブジェクト指向データベースの解説．

■ビジネス・モデリングに関心を持つ方のために

・I. Jacobson/本位田真一監訳「ビジネスオブジェクト」トッパン，1996
（原書：“ The Object Advantage ” Addison-Wesley, 1995）
〈寸評〉［Jacobson 95］は，企業も一つのシステムであるという視点から，ビジネス・モデリングにユースケース分析を適用しようというもの．

2. 文 献 紹 介

Jacson, M. A.: *System Development*, Prentice Hall, 1983.（邦訳：「システム開発 JSD 法」，共立出版）

オブジェクト指向とは直接的な関係が無いかも知れないが，モデリングということについて有益な示唆を与えてくれる古典的な書籍として，最初に本書をとりあげることにする．本書は，著者である M. ジャクソンの JSD 法と呼ばれる有名なシステム開発方法論の解説書であり，日本語版の監訳者には，当「技報」のなかで，形式仕様に関する報告をされている先輩が名前を連ねておられる．若輩者が出るような幕ではないのだが，ソフトウェア工学を志す者にとって一度は手にすべき書籍と思い，ここで紹介することにした．

JSD 法は，それまでの伝統的なシステム開発方法であった‘ 機能仕様を作成する ’という作業工程に対し，‘ 実世界のモデル化 ’という作業をその中心テーマに位置づけた方法論である．すなわち，システム開発においては「まず最初にこの現実をモデル化し，そうしてからはじめてシステムが遂行しなければならない機能の詳細を考え」てゆかなければならない．何故なら，「実世界の明示的なモデルに基づいていないシステムは保守がしにくい」からであり，反対に，「実世界のモデルは開発のための安定した基盤」であり，「モデルはその上に組み立てられる機能よりも変更されることが少ない」からである．こうした認識は，多くのオブジェクト指向システム開発方法論における基本的な認識となんら変わるところがない．

JSD 法のモデリングは，実世界を構成している実体とその行動によって問題領域をモデル化する．ここで，実体とは「実世界に存在し，かつ実世界での生起の時間順に行動を実行したり行動に影響したりする」ものであり，行動とは「特定の時点に生

起するとみなされるもの」でありかつ，「システムの外の世界で生起」するものである．すなわち実世界を，実体とその行動という二つの視点で捉えることによって，静的な世界と動的な世界に分けてモデル化しようとする戦略である．こうしたモデリングの戦略は，オブジェクト指向方法論の中でもよく使われている戦略であり，静的モデルや動的モデルと呼ばれている．ただし，JSD 法では静的な実世界に対し動的な実世界が優位に位置づけられており，「時間軸がこのように重要である実世界を動的な実世界という．これに対して，時間軸をもたない，あるいはあまり重要でない実世界を静的なという」としている．静的なモデルは単なるデータベースと位置づけられ，空間軸という概念を積極的に取り込めなかったところに手続き型プログラミング言語が全盛であった時代の背景があるのかも知れない．

JSD 法は，システム開発における仕様化と実現を明確に分離することを要求している．仕様化とは「関連する実世界の問題の抽象的な記述を作成」することであり，実現とは「その抽象記述をコンピュータ内部で具体化する」ことである．すなわち，仕様化とは「何を」を定義するプロセスであり，実現とは「どのようにして」を定義するプロセスのことである．この考え方は，その後の多くのシステム開発方法論に大きな影響を与えている．前者を外部仕様，後者を内部仕様と呼ぶ場合もある．いずれにしろ，モデリングにおける基本的な規律であるあることに変わりはない．

このように， JSD 法はプロセスという側面に焦点を当てたモデリング技法であり，モデリングとは何か，プロセスをモデル化するとはどういうことかを理解するうえで格好の書籍である．

Meyer, B.: *Object-Oriented Software Construction*, Prentice Hall, 1988（邦訳；「オブジェクト指向入門」，アスキー出版局）

オブジェクト指向という技術をシステム開発に利用しようとする最初の試みは，オブジェクト指向プログラミング言語によって始まった．Simula などの抽象データ型言語やメッセージ交換による並列計算モデルであるアクタ・モデルを基盤として，1980 年に Smalltalk-80 が発表された．そこでは，様々の概念がデータとそれへの操作の組であるオブジェクトによって表現され，すべての計算がオブジェクト間のメッセージのやり取りによって進められている．そしてプログラムとは，このオブジェクトを規定するクラスを記述したものである. オブジェクト指向言語のその後の流れは，Smalltalk によって示されたこのオブジェクト指向計算モデルを C 言語や PASCAL，Lisp といった既存のプログラミング言語に適用するという方向で発展してきたといえるだろう．C＋＋や Eiffel，LOOPS，Java などのオブジェクト指向言語がこのようにして生まれた.

本書の著者である B. メイヤーは，オブジェクト指向言語 Eiffel の開発者であり，それゆえ本書は Eiffel の解説書であって，オブジェクト・モデリングに関する書籍ではない．それを，ここであえて取り上げたのは，本書の内容が単に Eiffel の説明にとどまらず，オブジェクト指向システムの特徴を，ソフトウェアの品質という視点から議論しているからである. ソフトウェアの品質は, 機能や性能とともにソフトウェア・システムを規定する重要な要因である．システム開発の現場では，ともすれば機能と

性能だけが大きく取り上げられることが多い．しかし，システムのライフサイクル全体を視野に入れたとき，ソフトウェアにかかるコストの多くが品質に依存しているということが次第に分かってきた．しからば，ソフトウェアの品質とは何か．高品質のソフトウェアとは，どのようなソフトウェアのことを指すのか．本書は，こうした問題をオブジェクト指向言語という観点から明らかにしようとしている．

メイヤーはソフトウェアの品質を決定する要因を大きく二つに分類している．外的要因と内的要因である．「一つは外的品質要因と呼ぶべきもので，スピードや使い易さのように，ユーザがその有無を認識できるようなもの」であり，「二つめは，内的品質要因と呼ぶべきもので，モジュール性，読みやすさなどの，コンピュータの専門家にしかわからない品質要因群」である．特に，外的品質要因はユーザがその利益を直接的に享受できる重要な品質要因である．本書の中でメイヤーは，この外的品質要因として次の五つの項目をあげている．正確さ(Correctness)，頑丈さ(Robustness)，拡張性（Extendibility)，再利用性（Reusability）互換性（Compatibility）である．これら五つの外的品質要因のなかでも，ソフトウェアの保守という観点から，拡張性と再利用性は特に重要な品質要因であるとしている．そして，このソフトウェアの拡張性と再利用性に，オブジェクト指向プログラミング言語のモジュール性が大きく寄与するということを詳細に論じている．

「システムは進化するにつれてその機能が劇的に変化することがある．少なくとも十分に抽象的なレベルではデータのクラスはより不変である．そして，そのことこそがシステム構造の基盤として機能ではなく，データを使う理由である」．ジャクソンがモデルと言ったところを，メイヤーはデータと言い換えている．そこに方法論開発者と言語開発者の視点の違いがあるのだろうか．いずれにしろ，機能中心の考え方から脱却することが重要であるという点では一致している．

オブジェクト指向技術の優位性を説いたメイヤーも，何をオブジェクトとするかということに関しては，「オブジェクトはただそこに転がっていて拾い上げてもらうのを待っている」としか言っていない．プログラミング言語の開発者であるメイヤーには，具体的なアプリケーション・システムは興味の対象外であったのだろうか．

しかし，あまたのシステム開発方法論が未だにシステムの品質基準を具体的に示すことができずにいるなかで，ソフトウェアの品質要件を体系化した本書の価値はいささかも下がるものではない．

Coad, P. and Yourdon, E.: *Object-Oriented Analysis*, 2nd edition, Prentice Hall, 1991.（邦訳；「オブジェクト指向分析（OOA）[第2版]」，トッパン）

1990年代に入ると様々なオブジェクト指向システム開発方法論が次々と発表されるようになった．その中で，最初の体系立った方法論を発表したのは，コードとヨードンである．その後に発表されたオブジェクト指向設計（OOD）と併せてOOAD法とも呼ばれている．

本書は，前述のメイヤーに対する批判から始まっている．「オブジェクトはただそこに転がっていて拾い上げてもらうのを待っている．・・・嘘っぱちである．問題領域やシステムの責任範囲という観点からみて適切なクラス＆オブジェクトが，ただそ

こに転がっていて拾い上げてもらうのを待っていることはまれである」．プログラミング言語がどれほど優れた機能を内蔵していても，それを適切に使用することができなければオブジェクト指向技術も単なる絵に描いた餅に過ぎないだろう．われわれの周りにも，オブジェクト指向言語によって記述された手続き指向型のプログラムが沢山あるに違いない．そこにシステム開発方法論の存在理由がある．

また，メイヤーがオブジェクト指向技術の価値をソフトウェアの品質に置いたのに対し，コードらはオブジェクト指向技術を，複雑さを管理するための技術と位置づけている．オブジェクト指向技術は「問題領域とその領域におけるシステムの責任範囲の持つ複雑さを管理するための主要な原理」である．どちらの主張が正しいかということではない．おそらく，両方とも正しいに違いない．

OOAD 法の特徴は，多重レイヤーモデルにもとづく分析と多重コンポーネントモデルにもとづく設計という構造的な枠組みを提示したところにある．

分析モデルは，サブジェクト層/クラス＆オブジェクト層/構造層/属性層/サービス層という五つのレイヤーからなっており，これを多重レイヤーモデルという．分析作業もこの五つのレイヤーに対応して，クラス＆オブジェクトの識別，構造の識別，サブジェクトの定義，属性の定義そしてサービスの定義という順に進む．JSD 法が時間軸を重視したのに対し，OOAD 法はもっぱら問題領域の空間的な構造に焦点を当てている．

一方，設計モデルは，問題領域部/対話インタフェース部/タスク管理部/データ管理部という四つの主要コンポーネントから構成される．これを多重コンポーネントモデルと呼んでいる．OOA で作成した分析モデルは問題領域部にマッピングされる．この多重コンポーネントモデルは，一種のアプリケーション・アーキテクチャである．本方法論の特徴は，このアーキテクチャに基づいた設計を行うところにある．このようにアプリケーション・プログラムの構造を規定するアーキテクチャとしては，Smalltalk で有名な MVC モデルや，後述する OOSE 法のモデルなどがある．それぞれのシステム開発プロジェクトは，これらのアーキテクチャを参考に個々のアプリケーション・アーキテクチャを設計しなければならない．

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, W. *Object-oriented modeling and design*. Prentice Hall. 1991.（邦訳；「オブジェクト指向方法論 OMT 法」，トッパン）

いわゆる 3 大方法論と呼ばれている方法論がある．すなわち，J. ランボーの OMT 法，I. ヤコブソンの OOSE 法，G. ブーチのブーチ法である．これらの方法論だけが，何故 3 大方法論と呼ばれるのかは知らない．少なくとも，ブーチが統一モデリング言語 UML の作成のためにランボーとヤコブソンを仲間に引き入れるまでは 3 大方法論などという言葉はなかったはずである．

本書は，J．ランボーのオブジェクト指向システム開発方法論である OMT 法の解説書である．OMT 法はランボーらによって作成された，最も著名なオブジェクト指向システム開発方法論の一つである．OMT 法の出現によって，オブジェクト指向システム開発方法論というものが世の中に認知されたといっても過言ではないだろう．

多々あるオブジェクト・モデリング技法のなかでOMT法が確固たる地位を築けたのは，オブジェクの構造を表すクラス図によって実際の問題領域の構造が表現可能であることを示せたからに違いない．それまでの方法論が汎化関係や集約関係を重視していたのに対し，OMT法はオブジェクト間の関連を重視し，そのことによって問題領域の構造をオブジェクト同士の関係によって表現することに成功した．つまり，オブジェクトモデルが実際に役に立ちそうだということを具体的に示したのである．コード/ヨードン法やブーチ法にも独自のクラス図があったが，OMT法のクラス図はこれら他の方法論のクラス図に較べその表記法が簡潔で明快だったことも多くの人に受け入れられた理由かも知れない．いずれにしろ，オブジェクト・モデルの中心的な図表であるクラス図の主導権を握ったことがOMT法をオブジェクト指向システム開発方法論の中に確固たる地位を築かせた大きな理由であろう．UMLのクラス図もOMT法のクラス図をベースとしている．クラス図の本質について知りたいなら，UMLではなくOMT法を勉強した方がよいだろう．個人的には，UMLのクラス図よりOMT法のクラス図の方が問題領域をより構造的に表現できるような気が今でもしている．

OMT法は，オブジェクトモデルという単一のモデルにもとづいたモデル駆動型の技法である．OMT法の分析モデルには，オブジェクトモデルと動的モデルと機能モデルという三つのモデルがある．「オブジェクトモデルはシステムの静的，構造的，データ的な側面を表現している．動的モデルはシステムの時間的，動作的，制御的な側面を表現している．機能モデルはシステムの変換的，関数的，機能的な側面を表現している」という．しかしOMT法のプロセスのなかで，動的モデルと機能モデルはオブジェクトモデルを補完するためのモデルであり，分析から実装に至るプロセスはオブジェクトの識別とその詳細化の過程であるといえる．また，OMT法におけるモデリングの中心的な課題は問題領域のモデルを作成することで，ユーザインタフェースの設計やシステム制御機構の設計などといった作業はシステム設計としてモデリングからは独立して取り扱われている．これはコード/ヨードン法や次に紹介するOOSE法とは異なったアプローチである．

OMT法のその他のモデルについて補足するなら，動的モデルの表記法はD．ハレルのステート・チャートであり，機能モデルの表記法にはデータフロー図を使用している．いまや，ハレルのステート・チャート図を知る人は少ないけれど，OMT法の状態図なら知っているという人は沢山いるのではないだろうか．また，機能モデルに関して言えば，後にブーチは「もしあなたがいかなる正当な理由であれ構造化分析をフロントエンドとして使わざるを得ないとしても，設計のにおいがしたとたん必須モデルの代わりにデータフロー図を書き出すのは止めるように忠告する」と言っている．本書には，表記法やモデリング技法が詳細かつ具体的に説明されているにとどまらず，様々な事例や，練習問題が示されている．特に，豊富な練習問題は，モデリングという作業が決して知識ではなく技能やセンスによるものであることを作者が経験として知っていたからだろうと想像される．モデラーを目指す人は，まず本書の静的モデルの練習問題をやってみることである．そして，このような問題に興味が持てて，「ある状況に対する単一の正しいモデルが存在するのではなく，ただ適切なものと不適切

なものが存在するだけである」ということを抵抗なく受け入れることができるのなら，あなたはオブジェクト・モデラーに適していると言えるかもしれない．

Jacobson, I., Christerson, M., Jonsson, P. and Overgaard, G.: *Object-Oriented Sftware Engineering*. Addison-Wesley Publishing Company. 1992.（邦訳；「オブジェクト指向ソフトウェア工学 OOSE 法」，トッパン）

本書は，ユースケース分析法で有名な I. ヤコブソンのオブジェクト・モデリング技法の解説書であり，本書の頭文字をとって OOSE 法と呼ばれることがある．

OOSE 法もモデル駆動型のシステム開発方法であるが，OMT 法が分析から設計に至るまでオブジェクトモデルという 1 種類のモデルを詳細化してゆくのに対し，OOSE 法では要求モデル，分析モデル，設計モデル，実装モデルというように異なったモデル間でのモデル変換が行なわれる．

OOSE 法はユースケース駆動モデリングと呼ばれるように，ユースケース・モデルをシステム開発のあらゆる局面で利用している方法論である．すなわち，「ユースケース・モデルの中でそれぞれのユースケースで何か行われるかを記述し，分析モデルでどのオブジェクトがその振る舞いを提供しているかを記述し，インタラクション図によってそれぞれのオブジェクトがどのように振る舞うかを示す」のである．

OOSE 法のなかでユースケース・モデルは要求モデルと位置づけられている．「システムの外部に何が存在するか（アクター）とシステムによって何がなされるか（ユースケース）を定義する」ためのものである．アクターとは「そのシステムと情報交換する必要のあるもの」であり，ユースケースとはアクターがシステムと対話するときの「一連の処理」のことである．すなわち，「ユースケースは，システムが持つ機能のある一部を実行するための特定の手順」であり，ユースケース図とユースケース・シナリオによって表記される．言い換えれば，ユースケースとはアクターの視点で捉えたシステム機能である．

シナリオというのは別段目新しいアイデアではない．以前から時系列な事象を記述するのにスクリプトという方法があったし，OMT 法にもシナリオ記述という手法がある．ユースケース・モデルの特長は，シナリオをサブシナリオ単位に分割し，拡張関係とかユース関係のようなシナリオの構造化を行うことができる点にある．ユースケース・モデルについては，その後ランボーやブーチなど他の方法論作成者たちもその有効性に注目してそれぞれの方法論に取り入れている．しかし，実際にユースケース・シナリオを記述する段になると，それはそう容易い作業ではないことに気が付くだろう．先にも述べたようにモデリングは理屈ではないから，本書を読んで頭で理解しただけでは適切なシナリオを記述することは難しい．重要なのは，プロセスを記述するとはどういうことなのかを理解することである．ユースケース・シナリオ記述の核心を掴むうえで，JSD 法を学ぶことは有効である．

OOSE 法の分析モデルには，実体オブジェクト，インタフェースオブジェクト，制御オブジェクトという 3 種類のオブジェクトの概念が導入されている．問題領域から識別されたオブジェクトは，実体オブジェクトにマッピングされることになる．システムを構成するオブジェクトの論理的な機能による構造化は一種のアプリケーショ

ン・アーキテクチャであり，先に述べた MVC モデルやコード/ヨードン法の多重コンポーネントモデルとの比較を行うのも面白いが，紙面の都合もあるのでここでは省略する．

OOSE 法は，他のほとんどの方法論が使用しているクラス図を使用しない．クラス図のような静的モデルを記述する代わりに，分析モデルや設計モデルでは UML のコラボレーション図に似た図式を使用する．分析モデルでは上に述べた 3 種類のオブジェクトの協調関係が描かれ，設計モデルでは実装を考慮した様々な仕組みを追加して行く．こうしてそのシステム独自のアーキテクチャを作成して行く．次にインタラクション図を使用してオブジェクト間の相互作用を示す．インタラクション図はユースケース・シナリオを視覚化するためのもので，問題の本質を捉えるには非常に便利な図式である．このインタラクション図は，UML ではシーケンス図と名前を変え，複雑な制御も記述できるように拡張されているが，その分，問題の本質を掴み難くなっている．

OOSE 法のモデルはすべてユースケース単位に作成される．そういう意味では，OMT 法がデータ構造中心のモデリング技法であったのに対し，OOSE 法はプロセス重視型のモデリング技法と言うことができるだろう．

Booch, T. *Object-Oriented Analysis and Design with Application*, 2nd edition. The Benjamin/Cummings Publishing. 1994.（邦訳；「Booch 法：オブジェクト指向分析と設計　第 2 版」，アジソン・ウェスレイ・パブリッシング・カンパニー・ジャパン）

本書は，雲形のクラス図で有名なブーチ法の第 2 版である．ブーチ法の特徴は，よく言えばこれまでに提案されてきたモデリング技法の集大成であり，その意味で本書はオブジェクト指向モデリング技法の歴史を知るには最適なの書籍である．

ブーチ法の独自性はモデリング技法ではなく，その開発プロセスにある．すなわちブーチ法ではシステム開発プロセスを，マクロ・プロセスとミクロ・プロセスに分離する．何故なら，「完全に合理的な設計プロセスというのはあり得ないが，開発のミクロ・プロセスとマクロ・プロセスとを調和させることによってそれに近づけることができる」からである．ここでマクロ・プロセスというのはシステムの要件定義から分析，設計，進展，保守に至る逐次型の開発ステップであり，ミクロ・プロセスとはオブジェクトの識別, クラスの設計といったオブジェクト・モデリングのことである．しかし，各プロセスで作成されるモデルが外部仕様なのか内部仕様なのかの定義もなく，それぞれのモデルの作成に際して必要な問題領域に対する視点も明確ではない．どちらかと言えば，モデル駆動型というより手続き駆動型に近いプロセスという印象を受ける．この二つのプロセスも，あえて比較すれば，マクロプロセスというのは OMT 法のシステム設計と実装に，ミクロプロセスは分析とオブジェクト設計に相当しはしないだろうか．

一方，具体的なオブジェクト・モデリング技法には，ドメイン分析，ユースケース分析，責任分析などこれまでに提案された主要な技法が網羅されている．表記法もクラス図, 状態図, 相互作用図など著名なダイヤグラムを駆使している．ユースケース・

シナリオは OOSE 法とは異なった表記事例が載っているので参考にするとよいだろう．ただし，これら多くのモデルのなかで，中核となるモデルがどれかが判然としていない．しかし，こうした様々な技法と表記法を上記のシステム開発プロセスの中に統合したところにブーチ法の価値があるのかも知れない．ブーチが UML 制定の中心的な役割を担ったのもむべなるかなというところである．

Gamma, E., helm, R., Johnson, R and Vlissidea, J. *Design Patterns*, Addison-Wesley Publishing Company. 1995.（邦訳；「デザインパターン」，ソフトバンク）

本書は，E. ガンマらによるデザインパターン集であり，デザインパターンというと本書を指すほど有名になった書籍である．

デザインパターンというのは「オブジェクト指向システムにおいて重要でかつ繰り返し表れる設計を，それぞれ体系的に名前付けし，説明を加え，評価したもの」であり，「デザインパターンを使えば，成功した設計やアーキテクチャの再利用が簡単にできる」ことになる．本書は単にデザインパターンのカタログを提示しているだけではなく，デザインパターンの概念からその利用法に至るまで，MVC モデルの概念を例に取り，丁寧な解説がついている．いまや，数多くのプロジェクトでデザインパターンが利用されている．オブジェクト指向プログラミングを学ぼうとする人にとっては必須の書籍である．本書に採録されているデザインパターンは生成に関するパターンと構造に関するパターン，および振る舞いに関するパターンで，全部で 23 の代表的なパターンが紹介されている．一気にすべてのパターンを覚える必要はない．重要なパターンから覚えて行けばよい．

本書が提案しているパターンは，その構造を変えることなく仕様の変更にダイナミックに対応できるようにジェネリックなメカニズムを多用している．こうしたアルゴリズムは往々にしてシステムの効率低下を招く場合がある．即時処理を必要とするようなアプリケーションでは，スタティックな構造のパターンが必要とされるだろう．また，本書のパターンには「インタフェースに対してプログラミングするのであり，実装そのものをプログラミングするのではない」というように，これまでのオブジェクト指向言語の概念を越えるものがあるが，これがオブジェクト指向の基本的なスタイルであると勘違いされては困る．デザインパターンは，あくまで抽象的なアルゴリズムを実装するための特殊な構造であることを忘れてはならない．

パターンは，本書でも言っているように先人の優れた経験を再利用するものである．上手に利用すれば多くの利益を得ることができる．しかし創造的な作業であるシステム開発に従事するものにとって，ただ単に先人の成果を鵜呑みにするだけで，何時か自分自身の独自のパターンを設計しようとする姿勢を失ってはならないだろう．

H. -Eriksson, and Penker, M.,: UML Toolkit, Wiley, 1998.（邦訳：「UML ガイドブック」，トッパン）

オブジェクト・モデリングにとっての最近のトピックスは，オブジェクト指向モデル記述用言語である UML（Unified Modeling Language）が国際標準として制定されたことであろう．UML の登場によって，世界中のオブジェクト指向技術者が共通

の名称と表記法を使用してモデルを記述できることになった．これは画期的なことである．しかし，UML には分析から実装までのモデル要素がひとまとめになって定義されているため，実際に UML を使うには何らかのガイドラインなり方法論が必要となるだろう．また，現在提供されている UML の仕様は最終版ではなく，今後とも小さな修正が加えられてゆくはずである．

UML は，もともと「統一方法論」を作成するためにブーチとランボーが中心になって作業を進めていたが，ヤコブソンが参加するに及んで，「プロセスは色々な企業と文化の間で異なるので，万人に使用されるような標準のプロセスを構築すること」を諦め，その目標をモデリング言語の作成に変更したという経緯がある．これは一つの見識であり，異質なものを異質なものとして認めたうえで標準とは何かを考えようとする彼らの姿勢を，標準とは ‘ みんなで渡れば怖くない'式の傘と考えがちな我々の文化と較べたとき，考えさせられることが多い．

UML の内容は，大きく意味論に関する部分と表記法に関する部分からなっている．本書は表記法に関する説明書である，邦訳されている説明書としては今のところ最も親切なものである．しかし，あくまで表記法の説明書であるので，実際に UML を使用してオブジェクト・モデリングを行うには，それなりのオブジェクト・モデリングのための方法論とともに活用する必要がある．また，UML は今後とも進化して行く言語であるし，したがって本書の説明にも必ずしも妥当とは思えない部分がある．UML を使用する基本的な態度として，あまり詳細な部分にこだわらない方が良い．

UML はこれまでのオブジェクト指向言語の概念を越えた幾つかの概念を取り込んでいる．それらの多くは，デザイン・パターンから導入されたものである．したがって，UML の理解に先立ってデザイン・パターンに目を通しておくことが必要かも知れない．

UML に関する情報は，その他市販の書籍やインターネットを通して入手することができる．意味論について知りたい方は，そちらの方を入手されると良いだろう．

執筆者紹介　妻　木　俊　彦（Toshihiko Tsumaki）
1970 年東北大学理学部卒業．同年日本ユニシス(株)入社．大型計算機の OS，ネットワーク・アーキテクチャ，人工知能システム，オブジェクト指向システムなどの企画・開発に従事．現在，情報技術部技術研究開発室所属．ACM 会員．