監訳者／訳者まえがき

オブジェクト指向という名前は，すっかりソフトウェア関係者とその業界に定着したようです．しかし，そのわりには，クラスやリレーションという概念をソフトウェア開発の上流工程(要求記述，問題分析，仕様作成)で縦横に駆使している技術者は多くありません．とくにリアルタイム・システムや組み込みシステムのソフトウェア開発においては，オブジェクト指向どころかソフトウェア工学に基づいた記述，分析，設計という全般的な技術の利用が遅れています．プログラミング言語の制限や開発環境の多様性など，多くの問題があるからでしょう．

本書は，そうした時代の中で，果敢にリアルタイム組み込みシステムへのオブジェクト指向分析(著者は「クラス思考分析」と命名したいらしいが)の手法解説を行っています．

組み込みやリアルタイム・システムのソフトウェアが扱う世界は，注文書とか出荷予定日，仕入れ価格や販売利益などを扱うビジネス系のそれとは大きく異なっています．ビジネス系の世界が扱うのは，人の約束事であり，社会のルールです．つまり，過去もしくは現代の人間が人工的に決めた情報操作とか情報伝達の規則です．それに対して組み込みリアルタイムの世界で一義に扱うのは，自然の現象，自然法則なのです．

自然の現象や法則は嘘をつくこともありませんが，自分からその定義を語ってくれることも決してありません．そのため，ソフトウェア開発において，たった1行の要求文書がとてつもなく難しい開発プロジェクトとなる場合もあります．たとえば，「本音声認識装置は，２文節の音声を常に認識できること」と書いてあったならば，これはとてつもなく難易度の高いシステム開発になる可能性をもっています．

こうした問題は，コンピュータの性能が上がるにつれて，ソフトウェア技術者に深刻な問題を投げかけています．コンピュータを使ったシステムを利用したいユーザは、ソフトウェア技術者がコンピュータのすばらしい処理能力と柔軟性を徹底的にしぼり出してくれることを当然のことと期待しています．と同時にコンピュータが人間にとって危険な動作など決してしないようにプログラムは組めるものだと思っています．そして，ソフトウェア技術者は，こうした無理難題にやすやすと取り組むことが期待されています．なんとも無茶な話です．

本書は，こうした状況にあって柔軟で能率的，信頼性が高く，なおかつ自然現象や自然法則を扱えるソフトウェアを開発するためのモデリング技術を解説しています．まさに時代が必要としている１冊です．本書の前身である*How to Build Shlaer-Mellor Object Models*という書籍が，UMLがオブジェクト指向の表記法として一般的になる以前から存在していました(これは，『シュレイアー・メラー法によるオブジェクト・モデリング』というタイトルで邦訳書となっている)．本書は，その内容を継承したうえで著者の最近の知見をも取り入れた，最新のクラス・モデリングと分析の指南書です．日本には，本書を訳出した皆さんをはじめとしてExecutable UMLの分析技術を身につけて実務に適用している先進的なソフトウェア・アナリスト達がいます．こうした人々は，方法論の開発者や，本書の著者であるLeon Starr氏のような新しい分析技術を開発した人々から直接の指導を受けて育った世代です．彼らの知見がなければ，この訳出はありえなかったでしょう．これを糧に，日本でもっと多くの人々がクラス・モデリングの技術を身につけて，良質なソフトウェアを開発することを願います．

2004年8月　　二上 貴夫

ポストUMLには，MDA（モデル駆動型アーキテクチャ）という，モデルからプログラム・コードを自動生成する技術が注目されています．このMDAの中核になっているのがExecutable UMLです．Executable UMLは，Shlaer-Mellor手法の提唱者であるSteve Mellor氏が中心になり，アクション・セマンティクスおよびMDA技術としてOMGで標準化を行いました．MDAを知るには，まずはExecutable UMLを習得することが近道です。

本書は，Executable UMLの基本的な要点をわかりやすく解説しています．前提知識は必要ありません．本書を読んでいきながらUMLの知識が身に付いていきます．さらにオブジェクト指向技術についても，本書により学習することができます．クラス，属性といった基礎的な要素を詳細に解説しています．

ビジネス系のシステム開発ではUMLの普及が進んでいますが，組み込み系の開発にはこれから使用されていきます．組み込み系の技術者にとっては，UMLは目新しい技術といえます．また，ビジネス系のシステムと違い，組み込み系では厳密性が求められます．UMLモデルは，ビジネス系より実装に近い内容になります．

厳密なモデルを作成するためにも，本書のような基礎をきっちり学習できる書籍は重要です．また本書は，姉妹書である『Executable UML－MDAモデル駆動型アーキテクチャの基礎』（翔泳社）の入門書にあたります．本書の後に，さらに知識を深めたい読者はこちらを読まれるとよいでしょう．すでに読まれた読者は，本書で基本的な技術を確認すると，より確実になります．

最後になりましたが，本書の訳者であるExecutable UML研究会の有志，訳の手直しを行ってくれた柴田みちる氏，編集作業を担当していただいたCQ出版株式会社の相原洋氏，もう一人の監訳者である二上貴夫氏に感謝いたします．

2004年8月　　長瀬 嘉秀

UMLに関する書籍は山のようにあります．しかし，本書は山のようにある本とはかなり異なっています．まず，例題が何かのデータを管理するシステムではありません（図書情報管理システムなどの例題はもう見飽きました……）．次に，あえてユースケースやシーケンス図を話題にせずに，オブジェクト指向の本質であるクラス図のみに焦点を当てています．そして，クラス図にとことんこだわり，さまざまな情報をクラス図で表現する方法を紹介してくれています．われわれ訳者チームは組み込みシステムのソフトウェア・エンジニアなのですが，組み込みシステムの場合は何をクラスにしたらいいのかがわからない，ということがよくあります．この点で，本書の例題は非常に役立ちました．実際，本書の原書を部署内の勉強会で使用していたのがきっかけで，本書の翻訳に関係することになったのです．

われわれ訳者チームが初めてこの本の前の版にあたる*How to Build Shlaer-Mellor Object Models*に出会ったのは，かれこれ7～8年前のことです．そのときはまだUMLは存在せず，Shlaer-Mellor法という表記で書かれていましたが，そのクラス図の例の多さ，精密さ，正確さに非常に驚きました．クラス図の参考書として滅多にない貴重な本だと思っていたのですが，UML表記でなかったためにあまり売れていなかった（?）ようで，書店で目にすることもほとんどなくなってしまい残念に感じていました．その本にさらに内容が追加され，UML表記になったのが本書です．

本書を読んでいくと、関連の線や多重度によって，クラス図で表現することがここまでも変わってしまうのかと，驚かれると思います．そして，一見何をクラスにしたらいいのかわからなテーマにも，多くのクラスや関連があることに気がつき，クラス図のモデリングの魅力にどんどんはまっていってしまうかもしれません．クラス図のモデリングは一種パズルのようなものです．四角と線（クラスと関連）だけで，システム内の情報を表現していくのです．そこで，パズルを解く楽しみのようにクラス図をどんどん書きたくなっていきます．しかし大事なのはパズルを解くようなモデリングではなくて，何をモデル化するのかの“分析”であるとも本書では説明されています．たしかに，多少パズルが解けなくてもかまわないのです．どこまで厳密なクラス図を作成するべきかのバランス感覚が必要なのでしょう．読者の方々にも，ぜひ“はまりすぎない程度に”クラス図のモデリングを楽しんでください．

最後に紙面を借りて，CQ出版の編集部の方々，及び，素人の訳者チームの翻訳をまとめてくださった監訳者の方々にお礼を申し上げます．

2004年8月　　訳者一同

はしがき

本書は，要件の収集から実行可能なモデル構築の間に必要なステップについて述べている．それは，いままでだれも説明しなかった「抽象化」である．

要件の収集についてはよく知られている．ユースケースや要求仕様書，ユーザとの非公式な対話などによって，要件を集めることができる．また，モデリングの概念も知られている．モデリングでは，まず概要をスケッチし，そして問題の意味を実行可能な形式で正しくとらえていく．しかし，どのようにして，完全に詳細化された実行可能なモデルを要件から作成するのだろうか．

それに対する解答は(だれも聞きたくないかもしれないが)，「思考」である．思考とは，問題をなんとか解決する方法を見つけ出し，解答を正確に書きとめていくことである．

しかし，多くのクラスや属性，関連を書きなぐり，さまざまなインスタンスによる違いを表すために制約(OCL；Object Constraint Language)を奇妙な意味で使用して，「問題は解決した」，と宣言するのは簡単である．このような方法では，ソース・コードを修正するプロセスを促進しても，何が実際の問題かを根本から指摘することはできない．

そこで，われわれが選択した抽象化を明白に考えられるようにするための正確なモデリング言語が必要になってくる．そのような言語がExecutable UMLである．Executable UMLはUMLのプロファイルで，モデルを適切なソフトウェア・プラットフォームへコンパイルすることができる．Executable UMLは，クラス図，ステートチャート図，アクション“記述”から構成されている．Executable UMLでは，正確な表現に重点がおかれている．例えば，「航空機」クラスに「座席数」という属性がある場合，個々の航空機は「座席数」に対する異なる値をもつことができる．航空機がボーイング747-400だからといって，座席数が270席に決まっている，と主張することはできない．個々の事実は別々にとらえられなくてはいけない．

つまり，Executable UMLには規則があり，その規則には意味がある．Leon Starr氏のような熟練した技術者によって，それらの規則はさらに正確かつ表現豊かになる．そしてそれらによって，抽象的概念を検討しそれが適切かどうかを決めることが可能になる．

また，嬉しいことに，正確かつ表現豊かに書かれたモデルはソース・コードにコンパイルすることができる．

本書は，Leonの技術を読者に伝える．幸い，Leonの書籍は常に容易とは限らないが，楽しく読めるだろう．Leonは多くの経験で得たことを本書で明らかにし，それらをさまざまな例で示している．実際のプロジェクトで得た実例である．読者はこれらの例をパターンとし，自らの問題に適用できるだろう．

本書を読み，理解し，楽しんでいただきたい．そして，抽象化の技術を取得することが成果につながることを理解していただければ幸いである．

Stephen J. Mellor
(出身地)
Worcester
Worcestershire
United Kingdom

著者まえがき

本書は，私が実際のプロジェクトから得たモデルやモデルの断片，分析のテクニックを収集したものである．実行可能なオブジェクト指向のモデルを15年間作成してきた中での経験もいくつかまとめている．本書は，Executable UMLやシュレイアー・メラー法などの実行可能なモデリング手法の概要を理解している人を対象としている．これから実際のプロジェクトにExecutable UMLを適用しようとしている人や，Executable UMLのモデルを1，2年ほど作成している人を読者として想定している．そして，アプリケーションの要求仕様を十分に検討し，その要求仕様を完全に詳細なモデルに定式化し，そのモデルを動作するソフトウェアに変換することを真剣に行いたい人におすすめしたい．

本書は，Executable UMLのモデリング言語の文法を完全に説明するものではない．文法を知りたい人は，本書の「参考文献」に載せてあるSteve Mellor氏の書籍を読んでいただきたい．本書では，オブジェクト・モデリング言語の重要な特徴の使用方法を示している．さらに，いくつかの難しい問題をどう扱うかを示している．

さまざまなプロジェクトでの経験から多くのことを学んだ．その中でいちばん重要なのは，「クラス・モデルをおろそかにしてはいけない」ということである．アプリケーションの要件を完全に詳細にクラス・モデルで表現しなければいけない．これは時間を要する，大変な仕事である．しかし，クラス・モデルで手を抜くと，必ず次のような結果になった．(1)状態モデルがとても複雑になった．(2)一つか二つの重要な要件がモデル化のプロセスの最後まで隠れていて，最後に時間を浪費する再作業が必要となった．(3)予想できた新しい要件がモデル化のプロセスの最後に追加され，それによりモデルが破綻した．一方，優れたクラス・モデルがあると，単純で安定した状態モデルを作成できる．本書では，成功への鍵となる優れたクラス・モデルの作成方法に焦点を置く．

われわれと同様に機能中心のプログラミングをしていた人には，クラス・モデルの作成を学ぶのはかなり骨の折れる作業だろう．オブジェクト指向分析を新たに学ぶ技術者は皆，私がもった疑問と同じようなことを質問することに気が付いた．どのモデル構造が正しいのか(これを汎化関係にできるのか)．クラス・モデルでどの程度まで詳細に表現する必要があるのか．要求仕様が変わっても崩れないモデルをどのように作成するのか．優れたクラス・モデルと悪いクラス・モデルの違いは何か．モデル記述には何を書けばよいのか．どのように特定の数の(0でも1でも多でもない)の関連を表現すればよいのか．ものの階層をモデル化する最善の方法は何か．本書はこれらの質問への回答についても述べている．

成功へのもう一つの大きな鍵は，時間を効果的に使うことである．よくある共通の間違いは，モデリングに時間をかけすぎて分析に時間をかけないことである．分析とモデリングの違いを理解している技術者は少ない．いままでに何人もの分析の初心者が，要求仕様だと認識しているものに対して完全なモデルを作成しようと何時間もかけている．後に要件が変わったり，要求仕様がまちがっていたり，要件を確定することができないような変わりやすいシステムを基本としていたりする．結果として，モデル全体が崩れていく．第2部では，このような惨事をどうすれば防げるかを述べる．

優れたプログラマになるには，コードをたくさん書くだけではなく，他の人が書いたコードを読む必要がある．分析とモデリングに関しても同じである．実際に多くのモデルを作成していくうえで，本書のモデルが多いに役立つと確信している．

Leon Starr
San Francisco, California