ビジネス・アプリケーション開発における状態遷移表の役割
―ビジネス分野向けオブジェクト指向開発手法―

東京国際大学

白谷　勇人 ／ 堀内　一 ／佐藤　英人

ビジネス・アプリケーション開発においてオブジェクト指向開発(OOD)の有用性は高く評価されている。しかし、実プロジェクトでの適用でのハードルは高い。本稿では、この問題を解決する手段として状態遷移表の役割について解説を行う。これは、オブジェクト指向分析と設計におけるモデリングファシリティとして位置付けられる。まず、オブジェクト指向開発プロセスを規定し、オブジェクト分析／設計の各プロセスで使用するファシリティとアーキテクチャを明示し、この中で状態遷移の位置付けについて述べる。今後、適用結果を蓄積しフレームワーク作成ガイドラインに拡充予定である。

1 はじめに

ソフトウェア開発において、オブジェクト指向開発（以下 OOD と略す）手法は高生産性と高信頼性を提供する開発手法と期待されている。しかし、実プロジェクトへの適用においてはいまだハードルは高い。実プロジェクトの問題として、

- 分析モデルと設計モデルの乖離
- 分析モデルの粒度が人に依存してしまう
- 開発ドキュメントとして分析モデルが残らず設計モデルのみしか存在しない

この要因として、

- 分析・設計における作業項目、記述内容の曖昧性
- 分析・設計モデリングファシリティの曖昧性

が考えられる。

オブジェクト指向開発（OOD）手法としては、多くの提案がなされている[1) 2) 3) 4) 5) 15) 18) 23)]。これらの提案では、分析と設計でほぼ同じモデリングファシリティを使用する。このため、分析と設計でのキャップは小さくなる替わりに、分析と設計での作業の差も小さくなり、分析モデルと設計モデルの混同の原因となっている。また、UML を代表としたモデリングファシリティの規定は、対象ドメインを規定しない汎用的なモデリングを指向している。モデリング対象をビジネス・アプリケーションと絞れば、ドメイン特有のモデリングファシリティを規定することが可能である。よってビジネス・アプリケーション開発に特化したファシリティを導入することでモデリングの効率性と属人性の軽減をはかることが可能と考える。アナリシスパターン[23)]では，分析モデルの必要性とそれを作成するためのパターンの提供を行っているが、使用しているモデリングファシリティ（例えば継承関係、状態の定義）が汎用的なため必要以上の考慮を行っていると考える。

筆者らは、UML(Unified Modeling Language)をベースとして、オブジェクト分析やオブジェクト設計で使用すべきモデリングファシリティを提案し、分析から設計へのファシリティの変換ルールの規定を行っている。

本稿では、このファシリティよりビジネス・アプリケーション開発における状態遷移表の役割の部分を切りだしその解説を行う。前提とする開発プロセスとアーキテクチャに依存している。このため、開発プロセスと規定するアーキテクチャを説明したのちに、ガイドラインの解説を行う。

2 開発プロセスの概要

開発プロセスは、大きく以下の 6 つのアクティビティから構成される。「要求仕様の定義」「システム設計」「オブジェクト分析」「オブジェクト設計」「実装」「テスト」である。これらアクティビティの概要について述べる。

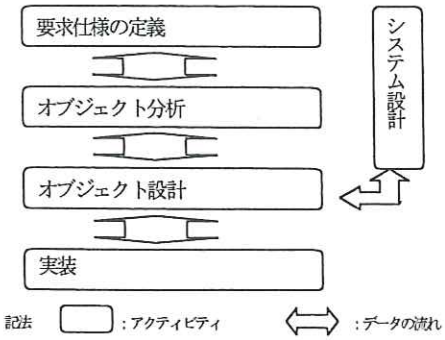


図1 開発プロセス

（1） 要求仕様の定義

ユーザの要求するシステムを構築するためには、ユーザが何を要求しているかを正確に把握することが必要となる。「要求仕様の定義」では、ユーザ要求の明確化を行う。本アクティビティでは、人間系・システム系を含めた新業務の定義から、システム化する作業の定義までを行う。本アクティビティの方法として[OOSE92][26]での要求モデルや Information Engineering[12]での計画（業務領域分析）が提案されている。筆者らは Information Engineering をベースとし、これにオブジェクトモデリングファシリティの一部を適用した方法[21]を提案している。

（2） オブジェクト分析

対象とする業務領域のモデル化を行う。要求仕様の定義で洗い出された要求に答えるためのクラスとその振舞いをモデル化する。

（3） システム設計

オブジェクト分析で明らかになったモデルを具体化するうえでのハードウェアやソフトウェアの制約事項を確認し、プログラミングを行うための基本的な方針を決定する。

（4） オブジェクト設計

実行環境に応じたモデル化及び、システム化での要素（ユーザインタフェース、通信、データベース、等）とのインタフェース設計を行う。オブジェクト設計では、クラスの属性の型、メソッドのシグニチャー、ユーティリティのシグニチャー等、アルゴリズムの内部実装以外について決定する。

（5） 実装

クラスの各メソッド、ユーティリティのインプリメント（アルゴリズムの内部設計）と単体テストを行う。

（6） テスト

組合せテスト、統合テストを行う。

2−1 開発プロセスと再利用

要求仕様の定義をオブジェクト分析と独立なプロセスとして定式化することにより次のようなオブジェクト分析でのプロダクトの再利用を行うことができる。

（1） 要求仕様の定義とオブジェクト分析の分離の利点

例えば，要求仕様の獲得を[OOSE92][26]での要求モデルや Information Engineering[12]での計画（業務領域分析）で行ったとしても、業務仕様が同じであれば同じオブジェクト分析結果（プロダクト）を再利用できる。

（2） オブジェクト分析とオブジェクト設計の分離の利点

例えば、あるシステムを異なった実行環境に移行する場合、業務仕様の変更が生じていないのであれば、以前と同じオブジェクト分析結果（プロダクト）からオブジェクト設計をはじめることができる。

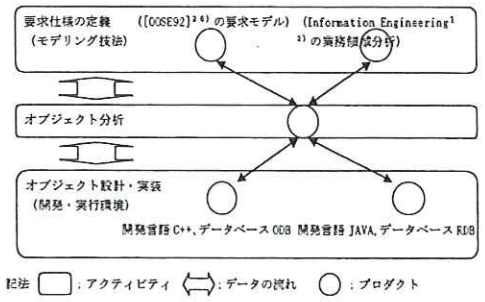


図2 開発プロセスとプロダクトの関係

3 アーキテクチャ

アーキテクチャとして情報システムのアーキテクチャを規定し、その中のアプリケーション層のアーキテクチャを規定する。まず、アーキテクチャが開発プロセスの中でどのように位置付けられるのかを

発プロセスの中でどのように位置付けられるのかを示す。つぎに情報システムのアーキテクチャを説明し最後に、アプリケーション層のアーキテクチャを規定する。

3−1 開発プロセスとアーキテクチャの関係

アーキテクチャが開発プロセスの中でどのように位置付けられるのかを示す。想定している業務分析は、筆者らが提案している業務分析[21]である。

業務分析において業務仕様を記述する要素であるビジネス・オブジェクト（例えば企業の活動のユーザである顧客や他社、その企業活動を行う組織や部門、又はその企業活動を支援するレガシーシステム）は、オブジェクト分析において詳細化される。この詳細化されたモデルは、情報システムにおいてアプリケーション層に対応する。つまりオブジェクト分析でモデリングされたプロダクトがオブジェクト設計において開発・実行環境に沿って詳細化された結果（プロダクト）は、情報システムのアプリケーション層となる。

3−2 情報システムのアーキテクチャ

情報システムのアーキテクチャは3つのレイヤーから構成される。レイヤーは、アプリケーションを大きく機能に分割した単位で、3 階層クライアント/サーバモデルのプレゼンテーション層、プロセス層、データ層に相当する。

（1）ユーザインタフェース層

システムの入出力や帳票出力を行うための機構をモデル化した枠組み。

（2）アプリケーション（ビジネスモデル）層

システム化の対象である業務のデータ構造をモデル化した枠組み。

（3）通信・データ層

オブジェクト間の通信機構やデータ管理機構をモデル化した枠組み。

3−3 アプリケーション層のアーキテクチャ

アプリケーション層のアーキテクチャは次の 4 つの層から構成されている。

（1）ビジネス・プロセス・オブジェクト

業務（ビジネス・プロセス）を抽象化したクラス。ビジネス・プロセスに対応したライフサイクル（状態）を持つ関連オブジェクト。業務ドメインに依存する。

（2）ビジネス・エンティティ・オブジェクト

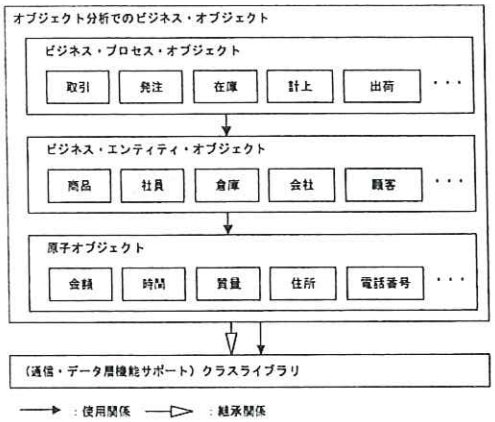
業務（ビジネス・プロセス）に参加し、状態変化が対象とするドメインのイベント（ビジネス・イベント）によらないオブジェクト。業務ドメインへの依存度は大きい。

（3）原子オブジェクト

業務を記述する最小粒度（これ以上詳細化しても業務上意味を持たない）の概念を抽象化したクラス。状態変化が対象とするドメインのイベント（ビジネス・イベント）によらないオブジェクト。業務ドメインへの依存度は小さい。

（4）クラスライブラリ

言語で設計されたソフトウェア部品。通信・データ層やインタフェース層の基本的な機能をサポートする。



出典：ビジネスオブジェクト推進協議会:CBOP/BO アーキテクチャ９８

図3 アプリケーション層のアーキテクチャ

4 ビジネス・アプリケーション開発における状態遷移表の役割

状態遷移表はシステムの振舞いのモデル化における重要なツールである。まず振舞いとは何かについて説明し、その中における状態遷移表について解説を行い、最後に簡単な事例を紹介する。

4−1 システムの振舞いのモデル化とは

システムのインプットからアウトプットまでの動作をモデル化すること、システムの制御構造の確定である。

従来の構造化分析／設計では、機能階層、機能（関数）仕様、関数（モジュール）関連、タイミングチャート、状態遷移図（表）、等でのモデル化が行われてきた領域である。これまでビジネス分野におけるシステム開発では、振舞いのモデル化において、状態遷移図（表）はメジャーなモデルとは言えなかった。原因の一つとして、開発者が状態遷移をモデル化しようとする時、開発者が管理できる範囲を超える状態遷移が生じたためと考えられる。

振舞いのモデルは、UML の相互作用図と状態（遷移）図がこれにあたる。システムの振舞いであるシナリオは数多く存在し、すべてのシナリオを記述することは通常困難である。これに対して状態遷移図は、１つのクラスの振舞いすべてを記述できる。よって、すべてのクラスについて状態遷移図を記述することで、システムの振舞いを記述することが可能となる。このため、システムの一貫性を現実的に記述できるモデルは状態遷移図と言える。

4−2 状態遷移図（表）とは

オブジェクトの状態遷移は、オブジェクトの振る舞い（生成から消滅までのライフサイクル）を明確にする。相互作用図がオブジェクト振舞いのすべてを記述することが困難であることに対して、状態遷移図（表）は、１つのクラスの振舞いすべてを記述できる。

これにより、メソッド相互の依存関係（実行順序の依存性）とメソッドの仕様が明確となる。同時にこのことがオブジェクトの振舞いを規定することとなる。

特に、予期されていないイベントを受け付けた時のエラー処理を検討するために状態遷移表は、有効である。

・メソッドの仕様の規定 ・メソッドの実行順序の規定＝制御情報の明確化 ・オブジェクトの（振る舞い）仕様の規定

（1）状態の定義

オブジェクトの状態とは、オブジェクトの各属性値（オブジェクトとの関連情報も含む）について同値分割（Equivalence Class Partition）を考慮し、これら同値分割されたクラスの属性の組合せとして定義する。状態定義の視点はこのとき、実行できるメソッド（受付可能であるイベント）は何かと言うことである。ある属性値の集合に対して実行できるメソッド（受付可能であるイベント）は限定される。このとき、このメソッドを実行できる属性値の集合をオブジェクト指向分析・設計で状態と認識する。

（2）遷移の定義

遷移とは、イベントによってメソッドが引き起こす状態の変化である。遷移先を決定する要因を下記に示す。

・イベント
・イベントが伝達するデータ値
・イベントを受け取ったオブジェクトの属性値（関連に関する情報も含む）

■ 利用目的

（1）状態遷移表からのメソッド仕様書の生成
（2）状態遷移表からの、仕様チェックリスト、プログラムチェックリストの生成

■ 状態遷移表のメリット

状態遷移を捉えることにより以下のメリットを得ることができる。

（1）オブジェクトの一貫性の保証

オブジェクトの振舞いについて、想定されていないタイミングで起こるメソッド呼び出しも含めすべての振舞いを明示的に規定できる。また、システムはオブジェクトの相互作用で動いていることからオブジェクトの振舞いが保証されるとあるレベルの動作保証されると考えられる。

（2）アプリケーションの開発において

継承関係を考える際の考慮点の一つとなる。例えば、複数クラスの汎化によりスーパークラ

スを作成する場合、そのようなクラスが持つべき振舞いを考える場合に利用できる。このようなスーパークラスは、各サブクラスの状態遷移においても汎化関係でなければならない。このことで、スーパークラスが持つべき最低限の振舞いは規定される。

別の例では、スーパークラスの特化よりサブクラスを作成する場合、そのサブクラスがもつ振舞いをを考える場合に利用できる。このようなサブクラスは、そのスーパークラスの状態遷移においても特化関係でなければならない。このことで、サブクラスが持つべき最大限の振舞いは規定される。作成したサブクラスはこの状態遷移を逸脱することない状態遷移を持つこととなる。継承関係と状態遷移に付いては下記「相互作用モデルとクラスモデルの関係」参照。

（3）既存アプリケーションのカスタマイズにおいて

フレームワークを利用したアプリケーション開発において、重要な考慮点となる。このカスタマイズの場合、原則的にフレームワークのサブクラス化作業である。あるサブクラスをどこまでカスタマイズできるのかと言うことに対して状態遷移は重要な示唆を与える。詳細は上記「アプリケーションの開発において」参照。

（4）クラスのテストにおいて

テスト工数の削減が可能となる。サブクラスクラスのテストにおいては、スーパークラスのテストケース（単体及び組合せとも）を変更することがなく利用することができる。これは、サブクラスのテストの第１段階をスーパークラスのレグレーションテストと見ることを可能とする。サブクラスのテストの第 2 段階として、新規に作成された状態遷移をテストする項目を実施する。

■ オブジェクトの状態遷移表では不充分な点

オブジェクト単体での状態遷移では、システムの中での下記のような状況が生じる可能性を排除できていない。よってこれらの状況を検知することを考慮しなければならない。これは、従来の構造化分析／設計と同様である。しかし、オブジェクト指向では、その検知にオブジェクト図を用いることができる。オブジェクト図（クラスは状態を持つと考える）をペトリネット図と見ることにより下記状況が起こる可能性のあるクラスそしてメッソドを絞り込んでいくことが可能である。このとき、クラスを“プレース（プロセス）”と“トランジッション”と読み替え方向を持った関連を“入（出）力アーク”とする。このように考えると下記状況の検知にペトリネットによる解析を利用することができであろう。

（1）無限ループの検知

あるシーケンスが想定していない無限ループに陥ることの検知を行う必要がある。

（2）デッドロックの検知

あるシーケンスにおいてイベント待ちによるデッドロックに陥ることの検知を行う必要がある。

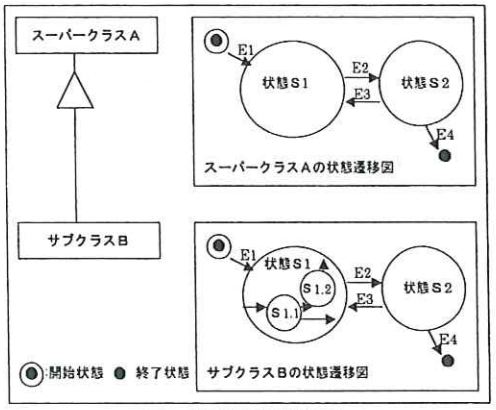


図4 継承関係と状態遷移

4−3 振舞いのモデルとオブジェクトモデルの関係

振舞いのモデルを利用し、メリットを得るためには、振舞いのモデルとオブジェクトモデルの関係（具体的には状態遷移とクラスの継承関係）を以下の様に規定する必要がある。

我々は、クラスが継承関係にあるときは、そのクラスの状態についても継承関係があるものとする。このときの状態の継承関係とは、「サブクラスの状態は、必ずスーパークラスの状態のネストで表現でき

る」と言うことである。

5 振舞いのモデルの事例

振舞いのモデルを利用したモデリングについて述べる。まず例題の説明を行う。

5−1 例題説明

■ 要求仕様例（概要）

説明のための例題として、あるファミリーレストランの販売管理の要求仕様を示す。

（1） システムの目的

・会計処理業務の迅速化

・顧客層ごとの売れ筋商品の分析

（2） システムの機能

・会計処理

・顧客層ごとの商品売上げ数の管理

（3） 業務仕様

モデルを簡単にするため以下のような制限を付ける。

・各テーブル毎（ごと）に会計は行われ、テーブルに着けるお客様は1名のみとする。

・注文はお客様一人一品までとする。

・お客様は、必ず支払いをする。

・商品は売り切れることもある。

・客層は、年齢と性別で分ける。

以下に、業務の仕様を示すシナリオを記述する。

◇ シナリオ 「注文・実施・会計処理1（正常）」

・ウェイタが来店客から注文をとる。

・ウェイタが注文を注文伝票に入力する。

・ウェイタが料理を来店客に運ぶ（サーブ）。

・ウェイタが実施チェックを注文伝票に入力する。

・食事が終了し、来店客が料金を支払う。

・会計係は注文伝票に入金済みチェックを入力する。

◇ シナリオ 「年度末決算処理1（正常）」

・年度末決算において注文は一括決済され、これまでのデータは削除する。

■ オブジェクト指向分析例（概要）

要求仕様に基づいた分析例を示す。

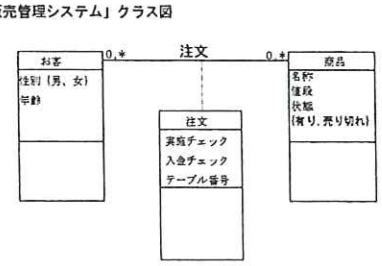


図5「販売管理システム」オブジェクト図

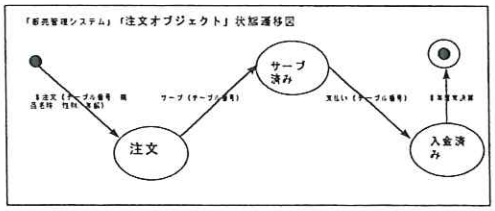


図6「注文オブジェクト」状態遷移図

5−2 オブジェクトの振舞いの抽出（分析）

各オブジェクトが並行動作を行う場合やイベントの受取順序のパターンが数多くある場合は、状態遷移表を使用して系統的に振舞いについて規定する。特に例外的な処理についての考慮を行う。（表1）

表1 "注文"オブジェクトの状態遷移表

	$注文	サーブ	支払い	年度末決算
S1:●生成前	S2へ	−	−	−
S2:注文	−	S3へ	無視する	無視する
S3:サーブ済み	−	無視する	S4へ	無視する
S4:支払い済み	−	無視する	無視する	S5:消滅へ
S5:消滅	−	−	−	−

−は、起こり得ないイベントであることを示す。
$は、クラスメソッドであることを示す。

表2 メソッド仕様書記述例

メソッド仕様		クラス名「注文」
メソッド名 「サーブ」	返り値	引数リスト
不変値(invariants)		
プレコンディション1		S2:注文状態
ポストコンディション1		S2:サーブ済み状態
アクション1		ー
プレコンディション2		ー
ポストコンディション2		ー
……		ー

5−3 状態遷移とメソッド仕様の規定

状態遷移を決定することにより、メソッドの外部仕様を決定することが可能。

5−4 状態遷移図とシーケンス図の整合性チェック

オブジェクトのイベントシーケンスを記述したシーケンス図で示されるイベント相互の順序依存性と状態遷移図（表）に示されているイベント相互の順序依存性は整合している必要がある。このことを説明するために以下に状態を記入したシーケンス図と状態遷移図を示す。

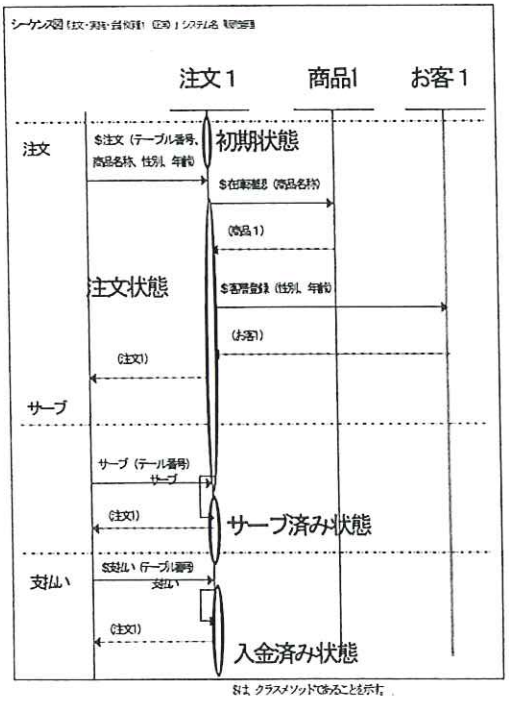


図7 シーケンス図「注文・実施・会計処理1（正常）」

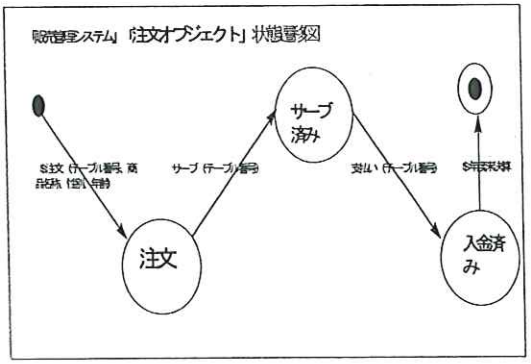


図8「注文オブジェクト」状態遷移図

6 まとめ

本稿は、OOD 手法における状態遷移の役割についての解説を行った。状態遷移表が OOD においても大きな役割を果たしていることを示した。特に、オブジェクト指向特有の継承関係において、サブクラス、スーパークラスの規定において状態遷移の役割の明確化を行った。今後、本手法の適用結果を蓄積しフレームワーク作成ガイドラインへの拡大を行うと共に新たなアナリシスパターンの抽出を計画している。

参考文献

[1] James Rumbaugh,Michael Blaha,Wiliam Premerani,Frederick Eddy,William Lorensen.OBJECT-ORIENTED MODELING AND DESIGN.EnglewOODCliffs,New Jersey :Prentice-Hall,1991. 邦訳 羽生田 栄一 監訳「オブジェクト指向方法論 OMT モデルと設計」,株式会社トッパン、1992

[2] Sally Shlaer,Stephen J.Mellor .OBJECT-ORIENTED SYSTEMS ANALYSIS Modeling the World Data.EnglewOODCliffs,New Jersey :Prentice-Hall,1988. 邦訳 本位田 真一 他訳「オブジェクト指向シス

テム分析」,啓学出版、1990

[3] Sally Shlaer,Stephen J.Mellor .OBJECT-LIFECYCLE Modeling the World in States.EnglewOODCliffs,New Jersey :Prentice-Hall,1991. 邦訳 本位田 真一 他訳「続・オブジェクト指向システム分析」,啓学出版、1992

[4] Grady Booch.OBJECT-ORIENTED DESIGN WITH APPLICATION.The Benjamin/Cumming Publishing Company,1991.

[5] Peter Coad,Edward Yourdon.OBJECT-ORIENTED ANALYSIS Second Edition. EnglewoodCliffs, New Jersey : Prentice-Hall,1991

[6] 片岡 雅憲 著.「ソフトウエア生産工学ハンドブック技術士ソフトウェア研究会編」第4章「設計技術」.株式会社フジ・テクノシステム,1991

[7] 馬場 勇 .「ソフトウエア開発の実践技法」.株式会社技術評論社1989

[8] 酒井 博敬.堀内 一 .「オブジェクト指向入門」.オーム社 1989

[9] James Martin,Carma McClure.DAIGRAMMING TECHNIQUES FOR ANALYSTS AND PROGRAMMERS.EnglewOODCliffs,New Jersey :Prentice-Hall,1985 邦訳 J.マーチン,C.マックルーア 著/国友 義久・渡部 純一 訳「ソフトウエア構造化技法」.近代科学社,1990

[10] 情報資源スキーマ調査研究委員会.「A Data Modeling Facility JDMF/MODEL-92」,1993

[11] 佐藤英人「JDMF－92解説(暫定版)」,1993

[12] Texas Instruments.A Guide to Information Engineering the IEF™,:Texas Instruments. ,1989 邦訳 日本テキサス・インスツルメンツ株式会社訳CASE実用ガイド:日経BP社,1992

[13] Object Management Group.The Common Object Request Broker Architecture and Specification Appendix A (Rev 1.1),1991

[14] (株)日立製作所 ビジネスシステム開発センタ,情報システム事業部「オブジェクト指向入門ガイド」,1993

[15] Grady Booch.OBJECT-ORIENTED ANALYSIS AND DESIGN WITH APPLICATION,second edition.The Benjamin/Cumming Publishing Company,1994.

[16] Scott Meyers.Effective C++.Additio-Wesley Publishing Company Inc,1993. 邦訳 スコット・マイヤーズ 著/岩谷 宏 訳C++の50の急所 Effective C++.ソフトバンク株式会社,1993

[17] 酒井 博敬.堀内 一 .「オブジェクト指向設計」.オーム社 1993

[18] David A. Taylor ,Business Engineering with Object Technology ,John Wiley & Sons,Inc.,1995
邦訳 デビット・A・ティラー著/錦田博樹 訳コンバージェントエンジニアリング入門 オブジェクト指向によるBPR,トッパン,1996

[19] Sheldon R.Object-Oriented Business Engineering,1993
要約; 日経コンピュータ.「最新オブジェクト指向実践ガイド」.日経BP社,1995

[20] Erich Gamma,Rihcard Helm,Raiph Johnson,John Vlissides.Design Patterns.Addison-Wesley Publishing Company,1995
邦訳 エリック・ガンマ,リチャード・ヘルム,ラルフ・ジョンソン,ジョンブリシディス著/本位田真一、吉田和樹 監訳オブジェクト指向による再利用のためのデザインパターン,ソフトバンク(株),1995

[21] 白谷勇人,葉木洋一,大成宣行,斎藤浩. 「ビジネス分野向けオブジェクト指向開発手法」,情報処理学会第52回全国大会 平8年3月

[22] 五十嵐直樹,森山宣郎,高橋久,森田真理,白谷勇人. 「要求仕様定義支援システムにおけるオブジェクト指向技術の適用」.情報処理学会第52回全国大会 平8年3月

[23] Martin Fowler. "Analysis Patterns Reusable Object Models" Addison-Wwsley、,1997
邦訳 Martin Fowler 著.邦訳児玉、友野、堀内. 「アナリシスパターン」. アジソン・ウェスレイ・パブリッシャス・ジャパン,1998)

[24] マーク・ローレンツ+ジェフ・キッド著.監訳者 宇治邦明「オブジェクト指向ソフトウェアメトリックス」.プレンティスホール出版,1995

[25] MISCO オブジェクト指向研究会著,「オブジェクトモデリング表記法ガイド」, プレンティスホール出版,1998

[26] Jacobson,I.,M. Christerson,P. Jonsson, and Overgaard. Object-Oriented Software Engineering : A Use Case Driven Approach. Wokingham,England:Addison-wesley,1992. (ヤコブソン著.邦訳 西岡、渡辺、監訳者 梶原「オブジェクト指向ソフトウェア工学 OOSE USE-CASE によるアプローチ」. アジソン ウェスレイ・トッパン,1995.)

[27] Martin Fowler,Kendall Scott 著.監訳者 羽生田栄一. 「UML モデリングのエッセンス－標準オブジェクトもデ言語の適用」. アジソン・ウェスレイ・パブリッシャス・ジャパン,1998

[28] Wolfgang Pree.Design Patterns for Object-Oriented Software Development.The ACM Press,1995. (邦訳 佐藤啓太、金澤典子.「デザインパターンプログラミング」.アジソン ウェスレイ・トッパン,1995.)

[29] 佐藤啓太. 「クラスライブラリ自由自在」. トッパン,1995.

[30] OMG, "Common Facilities RFP-4, Common Business Objects and Business Object Facility", 1996, Oct.

[31] CBOP ビジネスオブジェクトガイドライン(ワーキングドラフト:R1)、ビジネスオブジェクト推進協議会、1998.2.

[32] Bjarne Stroustrup.The C++Programming Language,2nd edition.AT&T Bell Telephone Laboratories,Inc.1991(B. ストラウストラップ著.邦訳 斎藤信男、三好、追川、宇佐美「プログラミング言語 C++ 第 2 版」.株式会社トッパン.1993)

(しらたにはやと／ほりうちはじめ／さとう ひでと)