ソフトウエア工学 93−17
（1993．7．9）

オブジェクト指向CASEはなぜオブジェクト指向か

中谷　多哉子

富士ゼロックス情報システム株式会社

1980年初頭にSmalltalk-80が実用化されて以来、オブジェクト指向ソフトウェア技術者は独自の仕様化技術によってシステム開発を行ってきた。しかし、90年代に入ってから急速に大規模システムへの実用化が具体化してきた背景には、オブジェクト指向分析／設計方法論が研究されてきたことと、方法論を支援するCASEツールが開発されてきたことが考えられる。

本稿では、いくつかのオブジェクト指向CASEツールの調査を行い、その動向について述べる。また、現在のCASEツールが抱えている問題点を挙げ、インレグレーション技術について今後の研究の方向を検討する。

SURVEY for the reason Why the Object-Oriented CASE-Tool used Object-Oriented Technology

Takako Nakatani

Fuji Xerox Information Systems Co.,Ltd.

Smalltalk-80 was commercialized in early 80's. Since then, Object-Oriented Software Engineers have used their own methodologies to develop their systems. In the 90's, we will use Object-Oriented technology for the large scale systems because we have realized how to define objects using Object-Oriented Analysis/Design methodology, and how to use CASE and when.

This paper describes the trend of Object-Oriented CASE by describing problems in today's CASE tools and examining the direction of Integration Technology.

1.はじめに

本稿では、オブジェクト指向によるシステム開発を支援するCASE（Computer Assisted Software Engineering）ツールについて調査した結果をもと、ソフトウェア開発におけるCASEの重要性と今後の位置づけについて検討する。

CASEツールの背景にはオブジェクト指向プログラミング言語の研究のみならず、他の多くの技術進歩の成果がある。しかし、最も大きい影響力を持っていたのはオブジェクト指向ソフトウェア開発方法論の研究である。多くの方法論が発表された時期と、オブジェクト指向CASEツールが発表された時期との時間的な隔たりは非常に短く、Coad&Yourdon法のOOATool のように、方法論研究とCASEツール開発が同時に進行していた例も見られる。

オブジェクト指向CASEツールは、従来の構造化手法を支援するものも多い。また、コンピュータのダウンサイジングによる一人一台のワークステーション時代とネットワーク時代を向かえ、CASEツール統合のためのフレームワークの標準化も進みつつあることを考慮すると、オブジェクト指向CASEは９０年代のソフトウェア生産性向上の鍵を握っていると言うことができる。

2.オブジェクト指向CASEの背景

背景を言語，開発環境，方法論の観点から検討する。

2.1.オブジェクト指向プログラミング言語と開発環境

1980年初頭に最初の実用オブジェクト指向プログラミング言語Smalltalk-80が発表された[Goldberg 84][Goldberg 87]。Smalltalk-80には、プログラミング言語のためのコンパイラだけではなく、再利用可能なクラスが提供され、コーディング，部品管理，部品検索のためのSystemBrowser、デバッグのためのデバッガー、開発管理のための変更履歴管理ツール等が提供されていた。Smalltalk-80は、内部のソフトウェア資源をオブジェクトとしてモデル化することにより、統合された開発環境を実現している。

1992年末のVisualWorksはGUI（Graphical User Interface）構築ツールや関係データベースへのインタフェースを追加し、Smalltalkの開発環境を更に充実させるものになった[PPS 93]。しかし、これらの開発環境は開発行程の下流に位置するものであり、分析／設計のためのツールは提供されていない。

一方、オブジェクト指向の継承機能による高生産性、情報隠蔽機能による高信頼性を既存言語に導入する目的で、既存言語を拡張したC++[Stroustrup 86] ,Objective-C[Cox 92J]をはじめ、ObjectPascalなどの言語が開発された。これらの言語にも下流CASEが開発されている。このように、オブジェクト指向プログラミング言語は、ワークステーションという個人使用を対象としたプラットフォーム上で使われるということと、既に下流CASEが実用化されている言語が多く存在するため、開発環境の提供が特に求められたようである。筆者の所属する部署の調査によると、オブジェクト指向の場合、下流CASEの支援機能の程度の差が、生産性に３～４倍の影響を及ぼすことが確認されている。

2.2.オブジェクト指向によるソフトウェア開発のための方法論

オブジェクト指向によるソフトウェア開発のための方法論の主なものを以下に示す。

・Booch法[Booch 86][Booch 91]
・Coad&Yourdon法[Coad 91-1][Coad 91-2]
・Shlaer&Mellor法[Shlaer 88][Shlaer 92]
・RunbaughらによるOMT（Object Modeling Technique）法[Rumbaugh 91]
・JacobsonによるOOSE（Object-Oriented Software Engineering）[Jacobson 92]
・WassermanのOOSD（Object-Oriented Structured Design）[Wasserman 90]
・James MartinのOOIE（Object-Oriented Information Engineering）[Martin 92]
・GoldbergのOBA(Object Behavior Analysis)[Goldberg 92]

その他の方法論も多く発表されている[Monarcho 92]。これらの方法論はシステムの実装言語への依存度は小さく、オブジェクト指向によるモデル構築における視点の定義と知覚の仕方を論じている。

上流CASEは方法論を支援することを目標としているため、開発方法論はCASEの最も重要な技術である。プログラミング言語環境とともに、上記の方法論とそれらの方法論を支援するCASEツールがほぼ同時に発表されているのは、上流工程における開発環境の需要に関して、次に示すようなことが原因と考えられる。

まず第1に、オブジェクト指向CASEツールが世に出る前に、要求定義工程支援環境の必要性に対する認識が十分高まっていた。第2に、オブジェクト指向分析／設計方法論が、構造化手法で使用されていた実体関連図[Chen 76]，データフロー図[DeMaeco 79]，状態遷移図[Harel 87]などを拡張して適用しているように、CASEツールも既存のものをオブジェクト指向に適用させることができた。第3に、CASEツールの開発とともに方法論の検証が行われ、完成されて行った。OOSEは、ツールを含めたすべての開発環境および方法論に対する総称である。このツール自身がOOSEの方法論を適用して開発されている。ObjectCastLightにおいても、CASEツールの開発において、方法論とツールの有効性の検証が進められている。第4にCASEツール自身がオブジェクト指向で作られており、クラス部品の再利用などによってツールの完成が早かったことが考えられる。Coad&Yourdon法によるオブジェクト指向分析の著書に使用されている分析図は、Smalltalk/Vによって開発されたOOATool が使用されている。ObjectCastLightにおいても、その開発言語にはSmalltalkが使用されており、ノードとアークを表現するクラスを再利用して開発された。CASEツールの第5の背景にGUI環境の発達と、X-Windowsをはじめとした標準化の普及が指摘できる。

CASEツールはGUIが大きな部分を占めるため、その標準化は必須である。このGUIは、人と人が会話を行うように、利用者がコンピュータと会話を行えるような環境として開発されたものである[Kay 92J]。利用者はマウスの操作によって、コンピュータに現在位置を知らせメニューを選択することで、コンピュータは利用者の意思をメッセージとして受け取り、処理の結果をディスプレイを通して利用者に表示する。このような利用者からコンピュータへのインターラクションによって作業を進める方法が最近のGUIの主流である。図1には、ObjectCastLightのコンピュータから利用者へ送られるメッセージの例を示した。利用者はマウスの形状の変化によって、現在可能な処理対象を認知することが可能である。

以上のように、CASEツールが実用化されるためには、その背景にGUIへのオブジェクト指向技術の適用と、オブジェクト指向のための分析／設計方法論の研究が不可欠であった。

その他の技術では、ハイパーテキストやハイパーメディアなどによって仕様情報の関連づけを行い、メモ書きを仕様書と共に管理することもできるようになった。また、オブジェクト指向によってマルチベンダーのツールをモデル化し、相互運用を実現するCORBA(Common Object Request Broker Architecture)[OMG 91]も開発されている。

3.オブジェクト指向ソフトウェア開発におけるCASEツールの役割

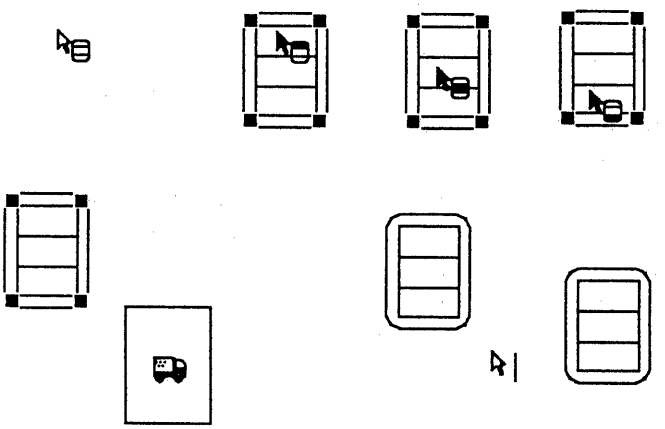


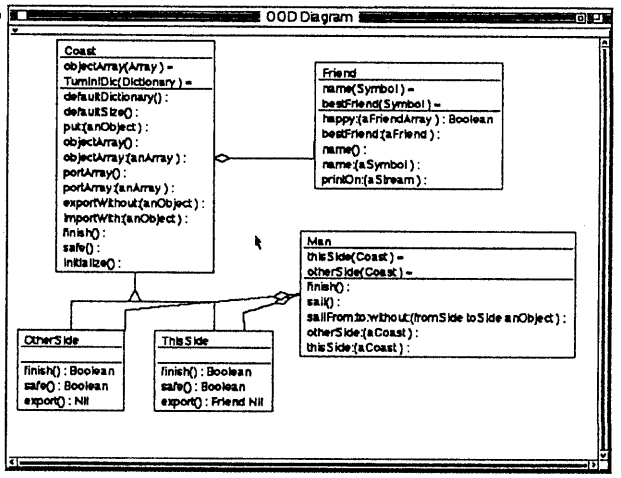
図1　ObjectCastLightのコンピュータから利用者へのメッセージ
選択されたノード内のマウスの位置によって利用者に許可される操作が決定される。
ノードの形状によって許可される操作内容を知覚することができる。

```
Object subclass: #Man
  instanceVariableNames: 'thisSide otherSide '
  classVariableNames: ''
  poolDictionaries: ''
  category: 'AI-FoxAndMan'
    "クラスManの定義"

Man methodsFor: 'sailing'
  finish   "航行の終了"
    Transcript show: 'I got it!!!!';cr

  sail   "航行の開始"
    ^self
      sailFrom: thisSide
      to: otherSide
      without: nil

  sailFrom: fromSide to: toSide without:
anObject    "航行"
    | some |
    some := fromSide exportWithout:
anObject.   "持ち込んだ荷物以外の出荷荷物を
選択する"
    some = anObject ifTrue: [^self finish].
    toSide importWith: some.
    (toSide finish and:[fromSide finish]) ifTrue:[^self finish].
    ^self
      sailFrom: toSide
      to: fromSide
      without: some
```



クラスFriendが積み荷の「キツネ」「ヤギ」「キャベツ」のクラスである。それぞれ自分の親友（食物）が近くにいると幸せな気分になる。クラスManは農夫のクラス。ただ船を航行させるだけであるが、持ち込んだ荷物を持ち帰らない条件がついている。クラスCoastは岸のクラス。出発地の岸と目的地の岸と、出荷条件と終了条件が異なるため、差分は継承を用いて２つのクラスを定義した。

（農夫のジレンマ問題）
キツネとヤギとキャベツを買った農夫が船で川を渡ろうとしている。ところが、船が小さいために、農夫以外には買ったものを１つしか載せられない。キツネがヤギを一緒に残しておくとキツネがヤギを食べるし、ヤギとキャベツを一緒に残しておくとヤギがキャベツを食べてしまう。全てをうまく向こう岸に渡す手順を与えよ。

図2　オブジェクト指向分析から実装への透過性の検証

なぜオブジェクト指向ソフトウェア開発で、CASEツールが必要なのであろうか。構造化手法によるソフトウェア開発は、分析と設計のあいだのセマンティック・ギャップが大きく、CASEツールによる全開発工程の統合支援が困難であった。しかし、「CASEツールはソフトウェア開発の全工程における支援を目指すべきである」と仮定するならば、オブジェクト指向ではそれが可能になるのであろうか。この章ではオブジェクト指向ソフトウェア開発支援について、その可能性を検討する。

オブジェクト指向は、ソフトウェア分析から設計、実装へ至る工程間でモデルの透過性が良いと言われている。図2に「農夫のジレンマ」問題のOMT法によるオブジェクト・モデルと、Smalltalkで実装されたクラスMan（農夫のクラス）のソースコードの一部を示した。オブジェクト・モデルにはメソッドの内容が定義されていないが、クラスの名前、インスタンス変数の名前、メッセージの名前が定義されている。このように、分析／設計で構築されるオブジェクト・モデルは、実装されるクラスの構造に関する情報を持っており、ソースコードとの対応がとりやすい。更に複雑な実用システムの場合は、設計時に効率の問題や部品の再利用を考慮したオブジェクトモデルの再構成を行い、メッセージのインタフェースとメソッド（手続き）を定義しなければならない。メソッドの定義には、動的モデル（または状態モデル）と機能モデルが重要な情報を提供してくれる。

CASEツールの統合には、データ統合、制御統合、表示統合の3種類の統合が論じられている[蝉坂93]。ソフトウェア開発を考える場合、垂直方向のデータ統合と水平方向のデータ統合の2つのデータ統合を検討しなければならない。図3に示した垂直方向の統合は、分析→設計→実装→テスト→保

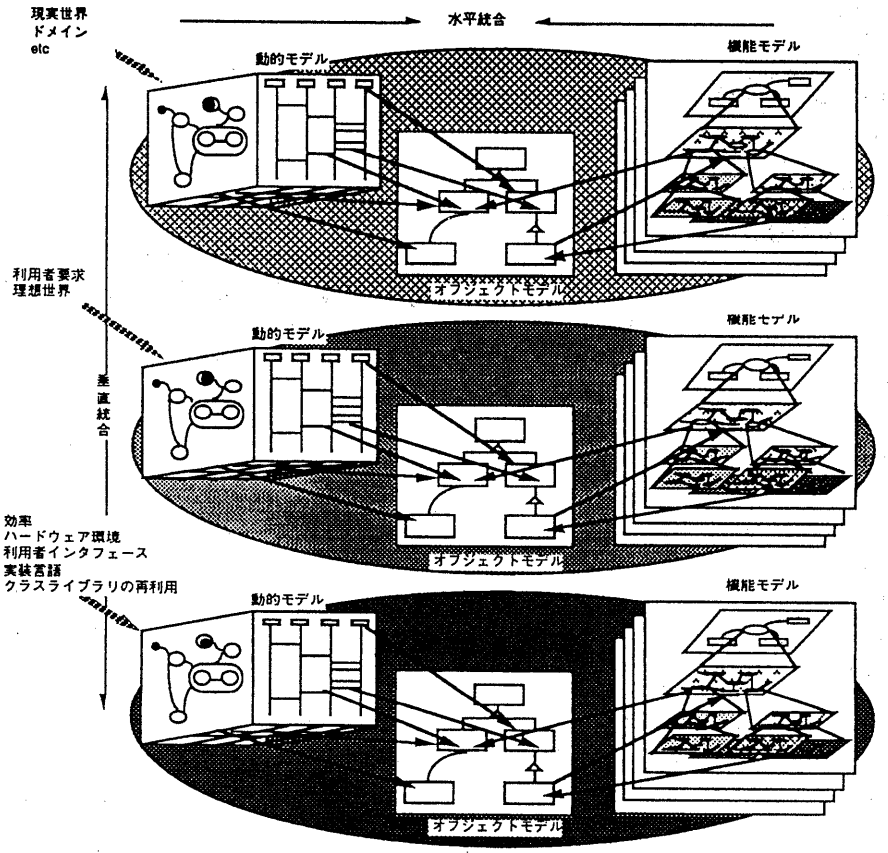


図3　オブジェクト指向によるソフトウェア開発における水平統合と垂直統合

守を支援するツール群という異なった抽象度のモデルのデータ統合であり、この統合によって、各工程間の仕様情報の関連付け、情報の相互流通によるリバース・エンジニアリングを実現することが可能になる。

水平方向のデータ統合は、静的なオブジェクトモデルを構築するツールと、オブジェクトの振る舞いを分析／設計するツール群という、異なった視点を持つモデルのデータ統合になる。現在の仕様化技術では、3つの視点を一つの図に書き表すことができないため、実体関連図，状態遷移図，データフロー図の3種類の表記を拡張して使用している。これらのモデルの視点と対象にしているシステムの範囲が異なるため、水平方向データ統合では、ツール間のデータの互換性を実現するのではなく、相互補完関係を明示することを意味し、ツール間におけるデータの一貫性を保証することを目的にしている。

開発工程の上流から下流で使われるモデルがオブジェクトという共通のモデルを使用しているため、オブジェクトモデルを構築するツール間の垂直方向のデータ統合は比較的容易であり、既にいくつかのCASEツールはこの機能を完成させている[FXIS 92]。しかし水平方向のデータ統合は、研究が進められているところであり[Rumbaugh 93]、解決されていない問題が多い。この問題を解決するためには、従来の仕様化技術をどう応用していくかを更に検討する必要がある。

オブジェクト指向によるソフトウェア開発を支援するCASEツールには以上の統合が必要である。現状はどの位まで実現されているのかを、次章で検討することにする。

オブジェクト指向CASE→	ObjectCastLight *1	ObjectMaker *2	ParadigmPlus *3	COHESION *4
支援方法論				
Coad&Yourdon	○	○	○	○
Shlaer&Meller	X	○	○	X
OMT(ObjectModel)	X	○	○	○
OMT(DynamicModel)	X	○	○	○
OMT(FunctionalModel)	X	○	○	○
Booch	X	○	○	○
データフロー図(DFD)	X	○	○	○
状態遷移図(STD)	X	○	○	○
実体関連図(ERD)	X	○	○	○
ユーザ定義の表記	X	○	公開されていないがメタCASEが存在する	X
開発フェーズ				
分析	○	○	○	○
設計	X	○	○	○
GUI設計	X	X	X	○
メソッド設計	X	○	○	○
実装	X	○	○	○
テスト	X	X	X	○
リバースエンジニアリング	X	○	○	○
プロジェクト管理	X	X	○	○
リポジトリ				
独自	X	○	○	○(CDDリポジトリ)
RDB	X			○(ATIS経由)
ODB	X			○(ATIS経由)
PCTE/ATIS/SoftBench	X	○		○
File	○	○		○

註）Coad&Yourdon,Shlaer&Mellor,OMT法の各方法論は、表記上の意味的な共通部分が多い。
オブジェクト指向方法論の多くが従来のDFD, STD,ERDを拡張したという経緯もある。
COHESIONは、DECDesignの他にマルチベンダのCASEツールが統合されている。

表1　オブジェクト指向CASEツール比較

各製品名は、各開発元の商標または登録商標です。
各ツールの詳細な仕様に関する問い合わせ先
*1) 富士ゼロックス情報システム（株）
*2)（株）ビー・エス・アイ
*3)ニチメンデータシステム（株）
*4)日本ディジタルイクイップメント（株）

4.オブジェクト指向CASEツールの現状

いくつかのオブジェクト指向CASEツールを、方法論、支援開発工程、リポジトリの観点から検討を行った。表1に結果を示す。この表を参照すると、オブジェクト指向CASEツールには大きな2つの要素があることを発見することができる。

1）複数の方法論を支援する方向からユーザ定義の表記を支援するメタCASEへ向かう流れ

2）垂直データ統合化への流れ

ObjectMakerは複数のパラダイムを支援することで1）の流れを持っている。メタCASEを提供することで、利用者自身で独自の表記に基づいたCASEツールを開発することが可能である。

CASEツールに求められる機能は、まず「お絵描き」機能である。次に「お絵描き＋意味の構造」を保持したツールである。絵と意味とを対応させる情報の構造をCASEツールが持っていると、モデルの意味と表記法とを分離して管理することが可能になる。メタCASEはユーザが使用するCASEツールの絵と絵の意味を定義し、新しいツールを構成する機能を提供している。

順に各CASEツールの概要を説明する。

1）ObjectMaker[PSI 92]

各種オブジェクト指向方法論および構造化分析／設計方法論を支援する。開発者向けにはメタCASEとしてTDL(Tool Development Kit)を提供している。支援工程は、分析／設計，コードジェネレータ，リバース・エンジニアリング，文書化支援ツールへのデータ転送を支援している。また、PCTE, HPのSoftBench, ATISなど、標準リポジトリになりつつある機構に接続可能である。米国MarkV社の製品で、日本の販売会社は(株)PSIである。

2）ParadigmPlus[ニチメン 92]

このツールの特徴はOODBMS(Object-Oriented Database Management System)であるVERSANTのオブジェクト・スキーマを定義するために使用することできる点である。複数の方法論を支援し、さらに、分析／設計，コードジェネレータ，保守，ドキュメント作成，プロジェクト管理を支援するCASEツールである。米ProtoSoft社の製品で、日本の販売会社はニチメンデータシステム(株)である。

3）COHESION[DEC 91][DEC 92]

各ベンダーのCASEツールをATIS(A Tools Integration Standards) を介してDECのCDD/リポジトリを共有し、垂直統合する環境を構築している。DECが開発したDECdesignは、米国DECが開発したCASEツールであり、COHESIONの一部となっている。他のコードジェネレータやプログラミング環境，言語コンパイラ，デバッグツール，エディタ，テスト，プロジェクト管理，ドキュメント作成等が全てNAS(NetworkApplication Support)上に統合されている。

4）ObjectCastLight[FXIS 93]

Coad&Yourdon法によるオブジェクト指向分析に、オブジェクト間のメッセージ送受信の状態を表現するオブジェクト交信図を付加したドメイン分析用のツール。分析モデルのドキュメンテーション支援として、ポストスクリプトファイルへの出力と、テキストファイルへの出力が可能。開発は富士ゼロックス情報システム(株)。

ObjectMakerは標準リポジトリへの接続を可能にしていることから、将来の垂直統合を意識してい

るものと思われる。ParadigmPlusのOODBMSのためのスキーマ定義支援機能は、アプリケーションの分析／設計とデータベース設計が同一に行えるため、プログラムとデータベースで同じモデルを使用することができる。また、COHESIONは今後もマルチベンダCASEツールの統合を進めて行くものと思われる。統合されたツールはリポジトリ内のデータ構造が明らかになること、異なるツール間の意味的な対応関係が明確になることによって、垂直統合が容易に実現できる環境が整っていると言える。

ObjectCastLightは、単機能、単ユーザに限定したツールであるが、標準リポジトリへの対応を実現することによって、他のベンダーの統合ツール環境への組み込みが可能になる。現在テキストファイルへの情報の出力が実現されているため、技術的な問題は解決している。

5.まとめ

これらのツールを概観すると、設計とソースコードの間の統合はほぼ完成されていると考えられる。これはオブジェクト指向が持つモデルの透過性の良さに起因する。

一方、分析から設計への垂直統合がまだ弱い。オブジェクト指向によるソフトウェア開発では、分析→設計の区別が明確でないため、ひとつの方法論で分析／設計を行うことが多い。しかし、現実世界のモデル化とコンピュータ世界へ移植する設計は明確に分割できるものであり、ツールは方法論間のデータの交換を実現すべきであると考える。

DECのCOHESIONは、本稿で調査した他のCASEツールとは規模が大きく異なっている。しかし、統合CASEツールはもはや1社だけで達成できる問題ではなく、統合フレームワークに準拠して開発されたツールを、利用者が自由に接続し、統合していくものである。その点で、COHESIONは将来の垂直統合CASE環境の姿を見せてくれている。

リポジトリは重要な技術であるが、CASEツール自体に提供されているより、他社のツールとのデータ互換の可能性の方がより重要である。現在、ANSI(米国標準化委員会)において、統合フレームワークの標準化の審議が進められている。PCTE, ATIS, HPのSoftBenchなどのリポジトリを介したツール間情報交換のためのフレームワークや、ネットワークを介した情報交換をオブジェクト指向でモデル化したCORBA等、統合のためのフレームワークは急速に整いつつある[OMG 91][鰺坂 93][篠木 93][ECMA 93]。

CASEツールの検討項目については、[Yourdon 92]が参考になる。今後オブジェクト指向によるソフトウェアの規模が巨大化するに従って、下流CASE環境の不備はプログラム生産性に大きな影響を及ぼすことになるが、ソフトウェアの生産性を向上させるためには、上流でどこまで詳細なレベルの検討を行えるかがキーになる。

（参考文献）

[Booch 86]Booch,G."Object-Oriented Development",IEEE transaction on Software Engineering,Feb. 1986.

[Booch 91]Booch,G.:Object-Oriented Design with Applications., Benjamin/Cummings, 1991.

[Chen 76]Chen,P.A.: "The Entity-Relationship Model Toward a Unified Approach of Data", ACM Transactions on Database Systems, March 1976.

[Coad 91-1]Coad,P. and Yourdon,E.:Object-Oriented Analysis,2nd edition., PrenticeHall,1991.（羽生田栄一監訳,オブジェクト指向分析,トッパン）

[Coad 91-2]Coad,P. and Yourdon,E.:Object-Oriented Design., PrenticeHall,1991.

[Cox 92J]Cox,B.J.& Novobilski,A.J.松本正雄訳:オブジェクト指向のプログラミング改訂第2版.トッパン,1992.

[DEC 91]日本ディジタルイクイップメント,:DECのCASE環境:COHESION ハンドブックRev.1,1991.

[DEC 92]日本ディジタルイクイップメント,:DECのCASE環境:COHESION説明パンフレット,1992.

[DeMarco 79]DeMarco,T.:Structured Analysis and System Specification., Prentice-Hall,1979.（高梨智弘, 黒田純一郎監訳：構造化分析とシステム仕様.日経マグロウヒル社）

[ECMA 93]ECMA TC33 20/May/1993 Paris,MEETING REPORT.

[FXIS 92]富士ゼロックス情報システム,:マルチクライアントプロジェクト,オブジェクト指向システム技術の実用に関するプロジェクト最終成果報告書,1992.

[FXIS 93]富士ゼロックス情報システム,:ObjectCastLightユーザーズガイド,1993.

[Goldberg 84]Goldberg,A.:Smalltalk-80, The interactive programming environment.,Addison-Wesley, 1984.（相磯秀夫監訳,Smalltalk-80対話形プログラミング環境,オーム社）

[Goldberg 87]Goldberg,A. and Robson,D.:Smalltalk-80,The language.,Addison-Wesley, 1987.（相磯秀夫監訳,Smalltalk-80言語詳細,オーム社）

[Goldberg 92]Goldberg,A. and Rubin,K.:"Object Behavior Analysis",CACM, Vol.35,No.9,Sep.,1992.

[Harel 87]Harel,D.:Statecharts: a visual formalism for complex systems.,Science of Computer Programming 8,1987.

[Jacobson 92]Jacobson,I.:Object-Oriented Software Engineering,Addison-Wesley,1992.

[Kay 92J]Kay,A.C. 鶴岡雄二訳,浜野保樹監修:アラン・ケイ.アスキー,1992.

[Martin 92]Martin,J. and Odell,J.:Object-Oriented Analysis and Design,Prentice-Hall,1988.

[Monarchi 92]Monarchi,D.E. and Puhr,G.I.:"A Research Typology for Object-OrientedAnalysis and Design",CACM,vol.35,No.9,ACM 1992,9.

[OMG 91]ObjectManagementGroup:The Common Object Request Broker: Architecture and Specification, Revision 1.1, OMG TC Document 91.12.1.（共通オブジェクト・リクエスト・ブローカ～構造と仕様～,日本オフィスオートメーション協会）

[PPS 93]ParcPlaceSystems, Inc.:VisualWorks Release 1製品紹介パンフレット,富士ゼロック情報システム,1993.

[PRI 92]ピー・エス・アイ,"ObjectMaker:Advanced Software Engineering Technology説明パンフレット",1992.5.

[Rumbaugh 91]Rumbaugh,J.,Blaha,M.,Premerlani,W.,Eddy,F. and Lorensen,W.:Object-Oriented Modeling and Design, Prentice-Hall, 1991.（羽生田栄一監訳,オブジェクト指向方法論OMT,トッパン）

[Rumbaugh 93]Rumbaugh,J."Controlling code How to implement dynamic models",JOOP,May 1993.

[Shlaer 88]Shlaer,S. and Mellor,S.J.:Object-OrientedSystems Analysis Modeling the world in Data.,Prentice-Hall, 1988.（本位田真一,山口亨訳,オブジェクト指向システム分析,啓学出版）

[Shlaer 92]Shlaer,S. and Mellor,S.J.:Object Lifecycled Modeling the world in States,Prentice-Hall, 1992.（本位田真一,伊藤潔監訳,続オブジェクト指向システム分析,啓学出版）

[Stroustrup 86]StroustrupB.:The C++ Programming Language.,Addison-Wessley,1986.（斎藤信男訳,プログラミング言語C++,トッパン）

[Wasserman 90]Wasserman,A.I.,Pircher,P.A. and Muller,R.J.:"Object-Oriented Structure Design Notation for Software Design Representation.Computer, Vol.24,No.3,IEEE,1990.

[Yourdon 92]Yourdon,:E.Decline & Fall of the American Programmer.,Prentice-Hall,1992.（松原友夫訳:ソフトウェア管理の落とし穴.トッパン）

[鰺坂 93]鰺坂恒夫,:"解放型CASEプラットフォーム",コンピュータソフトウェアVol.10. No.2. 日本ソフトウェア科学会,Mar.1993.

[篠木 93]篠木裕二,西尾高典,吉川彰弘,:"CDIF-CASEデータ交換形式",コンピュータソフトウェアVol. 10. No.2. 日本ソフトウェア科学会,Mar.1993.

[ニチメン 92]ニチメンデータシステム,"ParadigmPlusオブジェクト指向CASEツール説明パンフレット",1992.