オブジェクト指向の思想と方法論

第1回　クラスーオブジェクトーポリモーフィズム
——認識の3つ組構造

富士ゼロックス情報システム㈱
企画部技術企画課
羽生田 栄一

オブジェクト指向を理解するとはどういったことであろうか。Smalltalk、C++といったオブジェクト指向言語の文法を覚え、プログラミングスタイルを理解すること？　それは重要である。それと共に、そのパラダイム、方法論といったものを把握することも忘れてはいけないであろう。オブジェクト指向を考える時、構造化プログラミングが提唱された時のことを思い出すとよいのかもしれない。言語処理系として構造化を実現する機能が提供されたが、プログラムをシステムとして構造化するためには、設計方法自体の構造化も求められた。オブジェクト指向はこれからのプログラミングパラダイムといわれているが、現在はまだ、具体的なプログラミングのノウハウも、根本的な考え方や方法論も、両面共に理解されるに至っていない、というのが現状であろう。そこでこの連載では、後者の考え方や方法論を掘り下げてもらおうと思う。ただし、これまで繰り返し述べられてきた単なる描象的な話としてではなく、現場指向的部分を含め、それに有効な考え方、方法論といった形で進めてもらうことにする。読者の方々からも、いろいろな意見を寄せてもらえれば幸いである。[編]

人が樹立する事物間の絆は、事物に先立って存在し、事物を決定する働きをなす。人はこの視点によって2次的に事物を創造する。いかなる事物も、いかなる瞬間においても即自的には与えられていない。

F. ソシュール：『一般言語学講義』

1　はじめに

この世の中は奇妙なもの、不思議な現象で満ちている。

たとえば、桜は毎年春になるとみごとな花をつける。例年、その開花時期は、気象庁によって1日の単位で見事に予測される。なぜこんなことが可能なのだろうか？　日本に桜の名所は数知れないが、そこに咲く桜のほとんどは、「染井吉野」という品種である。染井吉野は実をつけないから、授精により増えることはできない。接ぎ木により繁殖したものなのである。つまり、日本中に（いやワシントンのポトマック河畔の桜も含めて世界中に）ある染井吉野はただ1本の木なのである。したがって、気候変化に応じて規則正しく反応するので、桜前線と名付けて開花予測ができるわけである［中尾 90］。

さて、今回から数回にわたり、オブジェクト指向の話をすることになった。しかしすでに、オブジェクト指向の表面的な説明は、さまざまなメディアを通じてイヤというほど見聞きされていることと思う。それどころか、もしみなさんのデスクトップにワークステーションでも置いてあろうものなら、ディスプレイ上に当然のごとくウインドウ・システムが動いており、ユーザーが意識するしないにかかわらず、オブジェクト指向でデザインされたインターフェイスに触れていたりなんかするわけである。もう、オブジェクト指向とかいう題名で、ありきたりの解説をするのは、書くほうも読むほうもうんざりだ、というのが正しい状況認識だと思う。

そこで、本連載では、オブジェクト指向に対する2つの対局的なアプローチを取り上げ、それらが実は同じことを目指しているのだ、という結論にもっていくというアクロバットを演じてみることにしたい。2つの対局的なアプローチとは、1つは哲学の分野で2千年来論じられてきた認識論からのアプローチであり、もう1つはソフトウェア開発の現場で実際に必要とされているソフトウェア開発方法論からのアプローチである。

桜も風流にめでてこそ花というもの、無粋な詮索は無用とおっしゃられる方も多いことと思う。これから書き連ねることは、オブジェクト指向をソフトウェア開発の現場でまさに実践している方からみれば無用の詮索になっているかもしれない。筆者としてはこの連載が、2つの対局的なアプローチを通して、花も団子も追いかけて一兎も得ず、とならないことを祈るばかりである。

1.1 オブジェクトだって？

オブジェクト指向という言葉を聞いたのはかれこれ10年ほど前である。その時に、

「世界をモノ中心にみる考え方だ」

「モノがメッセージを受けて反応するんだ」

といった説明を聞いたわたしは、今どきまだそんなことをいっているのか、なんと古くさい考え方なのだろう、と思った。ほんとうにそれが第一印象だったのである。当時から哲学にかぶれていたわたしは、現代哲学の中心的なテーゼとして「ものからことへ」あるいは「実体から関係へ」という発想の転換を聞きかじっていた。いわく、物というのは属性の束である、物というのは物象化の結果生まれる錯視であって、世界を表現しているもろもろの関係のネットワークがたまたま絡まり結節した点が一瞬、物のように見えるに過ぎない……

そうした目から見ると、オブジェクト指向はアニミズムの発想、それが言い過ぎなら、プラトン、ア

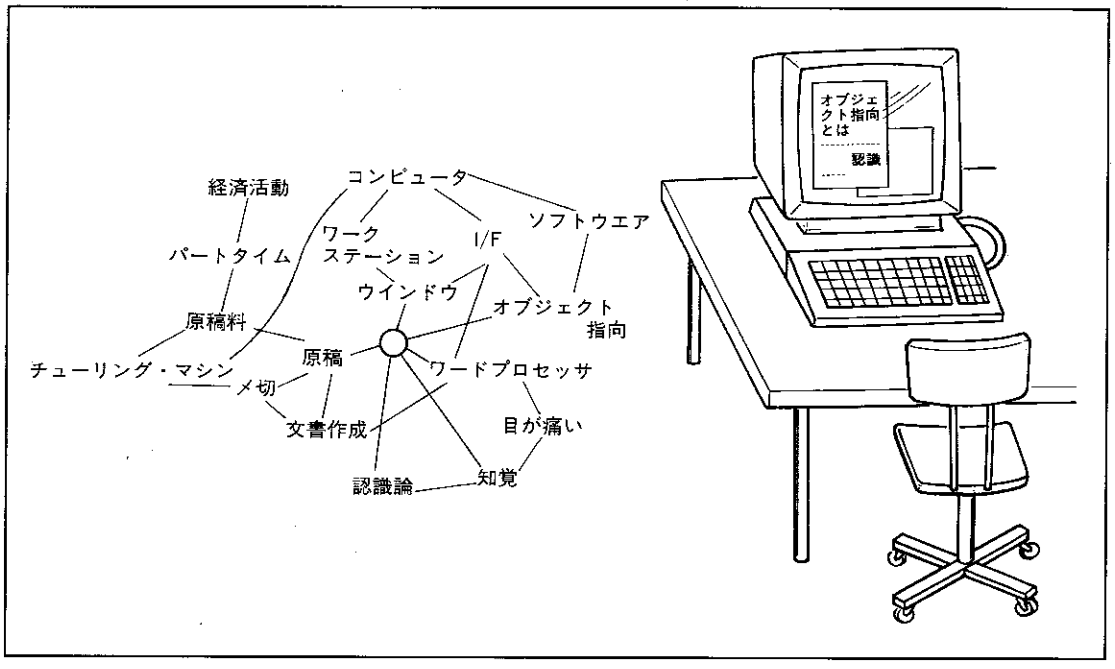

図1　現象と理解のための概念枠

リストテレスの頃からなんら進歩していない、いかにも古くさい思考法であるように思えた。いまさらオブジェクトだって？　というのが本音だった。そうこうするうちに、自分が最も先進的なオブジェクト指向環境であるSmalltalkに仕事で携わるようになっていた。それでも、人がいうようにオブジェクトというのはモノなんだ、という偏見に囚われつづけ、その裏に潜むオブジェクト指向の構造主義的視点に思い至らないでいた。最近になってようやく、オブジェクト指向をプログラミング言語のパラダイムとしてでなく、ソフトウェアの生産と利用のプロセス全体から見るようになって、初めて、オブジェクト指向の実体主義的側面より関係論的、構造主義的側面の重要性に気付くようになってきた。

本連載では、この新しい視点からオブジェクト指向を取り上げていこうと思う。今回はややオブジェクト指向の基礎論的なテーマに焦点を当てざるをえないが、次回以降は具体的なオブジェクト指向分析／設計／プログラミングに焦点を当てたい。そして最終回では、構造主義の限界そしてそれを乗り越えるための「ソフトウェアの生態学」に対応した新しい方法論の必要性をも論じられればと思う。

1.2　現象と理解

この世の中は、いろいろな現象で満ちあふれている。不思議な現象もあれば日常的な現象もある。自然現象もあれば人間関係における現象もある。たとえば、最も身近かな例として知覚を取り上げよう。目の前に部屋の中の風景が広がっている。椅子があり、机があり、机の上にはワークステーションが載っかっており、その画面には何枚かのウインドウが映し出されているとしよう。しかし、待ってもらいたい。実際に目の前に展開しているのは、わけのわからない、四角だのマルだのの形をした色のモザイク模様であって、それ以上のデータではないはずではなかったか？　なのに、われわれは、4本足のオブジェクトを見てはすぐさま「椅子がある」と認識し、ディスプレイ上の白い四角い枠を見て中にインクのシミのようなものが浮き出た白いモザイクとは思わず、ウインドウ・システムだと理解する。ある場合には、ウインドウ・システムであることさえ意識に昇らず直接そこに映し出された文章の内容に注意が集中し、「雑誌のワープロ原稿」と理解されるかもしれない（図1）。

いずれにせよ、われわれは、現象を生のまま体験することはできない。必ず、それら現象に対してある意味付けを行なって物事を認識し理解する。この事実を「認識の理論負荷性」と呼ぶことにしよう。通常、この言葉は、科学哲学の分野において、純粋な観察事実というものは存在しない、どんな観察もそれに関与する理論の枠組みを通して行なわれざるをえない、というマイナスの意味合いを込めて使われる。しかし、ここでは、われわれの認識は必ず、既存の概念枠を再利用することによって行なわれる、という積極的な意味合いで用いることにする。

1.3　機能主義と構造主義

ソフトウェア工学において、システム分析の手法の1つに機能分析がある。別名、構造化分析（Structured Analysis、略してSA法）と呼ばれるものである。ここではこの手法の根底にある考え方を機能主義と呼ぶことにしよう。機能主義では、世界を現象に即して記述しようとする。すなわち、問題となる世界をいくつかの枠で区切って、その枠への情報の入力と出力を記述してやるのである。その枠のことをブラックボックスという。そのブラックボックスへの入力と出力をペアにしたものを「機能」と呼ぶ。ブラックボックスの内部にさらにいくつかのブラックボックスを設定して記述を詳細化していくことが可能である。しかし、注意してほしいのは、ここで行なわれているのはあくまで現象の記述であって、こうした行為をいくら積み上げても理解には到達しないということである。実際、機能主義において目指されるのは、現象を記述していると称される「仕様」であって、理解ではない（図2）。

一方、もし仮にソフトウェア工学の世界において構造主義と呼ばれる流派があったならば、彼らは次のように仕事を進めるであろう。現象を前にして彼らのまず考えることは、

現象の裏にはなにか構造が潜んでいるはずだ

ということである。そして、その構造とは前節で示唆したように、認識する側であるわれわれの頭の中にある概念枠の中から見つけ出される。すでにもっている概念枠をうまく利用して、現象と整合するように概念枠に沿ってモデルを形成し、それを通して現象を理解しようとするだろう。ここで大事なことは、対象の記述ではなく理解が目指されているということである。われわれは、数学の証明を追いかけ

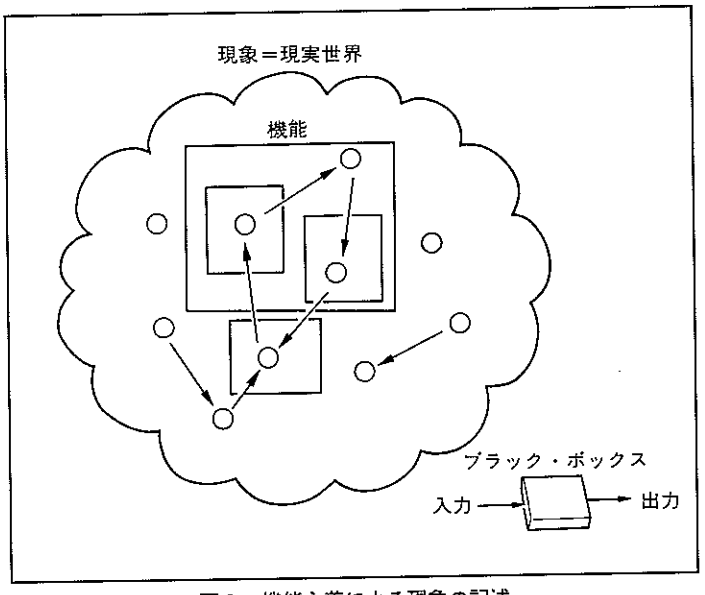

図2　機能主義による現象の記述

て、その1行1行は理解でき、しかも前の1行から次の1行への推論も正しいと理解できるにもかかわらず、なるほどというこころからの得心がいかず悲しい思いをすることがある。この場合、理解に到達できないのは、モデルの構成がうまくできないためなのである。ここでの1行1行と推論ステップの1段分が、機能主義でいうところの「機能」であり、「仕様」である。

ここまで読まれた方は薄々お気付きのことと思われるが、まさに、オブジェクト指向とは上で述べた構造主義の方法論になっているのである。現象の中から、オブジェクトを見い出すと同時にそれらを既存の認識の概念枠に結び付け、現象に対応するモデルを作り出すのである。そのモデルのことを構造だと思ってもらえばよい。したがって、オブジェクトは単独で存在するのではなく、モデルの構成要素であって、モデルを通して他のオブジェクトと関係付けられ、しかも、概念枠を通して既存の世界モデルの中にも位置付けられるのである。こうして、オブジェクトは現在対象となっている現象のモデルの中で位置価を与えられるのみならず、その新しいモデルも既存の世界モデル（認識のための概念枠）の一部を修正・拡張するという形でその中に埋め込まれる。この既存概念枠への新規モデルの埋め込みのことをわれわれは理解と呼んでいるのである（図3）。

以上述べたことは、具体的なソフトウェア開発の方法論として次回詳細を述べる。したがって、これは次回の話題だが、機能主義とは異なりオブジェクト指向では、ユーザーとシステム分析者、およびシステム分析者と設計者／開発者とのコミュニケーションを「モデル」を通した相互「理解」という形で促進することになる。

2　オブジェクト指向の基本概念

2.1　Smalltalk早わかり

ここで、オブジェクト指向の基本的な概念を説明しておこうと思う。今後の話の展開を簡潔にするために、説明のための共通言語としてSmalltalkを採用することにする。といっても概念の説明に必要な最小限度にとどめておくつもりである。この連載を理解するには必要でないが、Smalltalkのより詳細に関心のある向きは、［ピンソン 90］を参照してほしい。また、本誌の小嶋隆一氏による記事も参考になると思う。

［Ingalls 81］に述べられているように、Smalltalkプロジェクトの目的は、すべての人の創造性をコンピュータによって支援することであり、焦点を2つの基礎分野に絞った。

(1)　記述の言語（プログラミング言語）
　　＝I/F（人間の心のモデル、コンピュータ・アーキテクチャ）

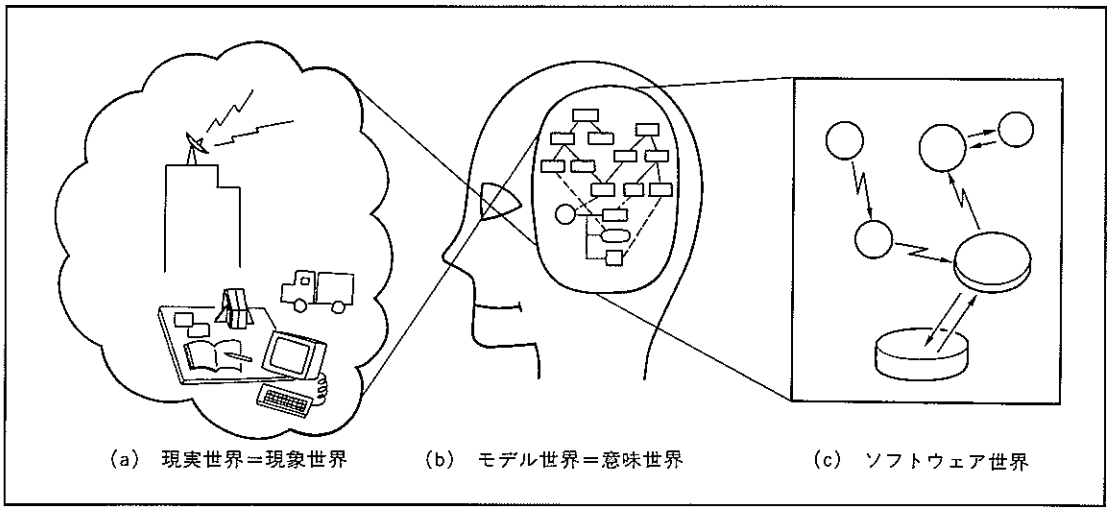

図3 現象のモデル化としての理解

(2) 対話の言語（ユーザー・インターフェイス）
＝I/F（人間のコミュニケーション・システム、コンピュータのコミュニケーション・システム）

プロジェクトは2年から4年の周期をとり、それは科学の方法論に沿ったものといえよう。

(1) [観察]
現在のシステムの枠組みを使ってアプリケーション・プログラムを構築する。

(2) [理論の構築]
経験に基づいて、言語を再設計する。

(3) [検証可能な予測]
新しい設計に基づいて新しいシステムを構築する。Smalltalk-80はこうした5回目のサイクルにおいて作り出されたものである。

ここで興味深いのは、Smalltalkの成立過程もまさに前章で述べた、モデルの形成による現象の理解というパラダイムにのっとっている点である。ただし、ここでは、このサイクルがスパイラル状に何回も繰り返されてそのたびにシステムは進化を遂げている。もう1つ興味深いのは、システムの枠組という語と言語という語が同じ意味に使われている点である。これは、Smalltalkが言語であると同時に、世界をモデル化する際に活用できる既存の認識枠をも提供しているということを示唆している。今まで、この認識枠のことを環境という言葉でかたづけ、その認識論的重要性を見えなくしてきた。

今まで、なんの説明もなく、オブジェクトだとかオブジェクト指向という言葉を使ってきた。ここで、これらの言葉に一応の定義を与えてみよう。人により若干の差異があるのだが、ここでは次のように宣言してしまう。

オブジェクト指向＝オブジェクト構造＋クラス構造＋ポリモーフィズム

これらについては、以下で順々に述べていく。ちなみに、

オブジェクト構造は、インスタンス変数とメソッド
クラス構造は、クラスとインヘリタンス(継承)
ポリモーフィズムは、メッセージとシソーラス

のメカニズムによって実現されている。

2.2 オブジェクトとは

まず、オブジェクトの定義だが、ここでは、

オブジェクトとは、データの集合と手続きの集合をパックしたカプセル

のことだと割り切ってしまおう。データ構造とそれを操作するための手続きとをひとまとめにしてしまうのである。これは、プログラミング言語の世界では「抽象データ型」として愛用されてきた考え方である。Smalltalkでは、こうしてカプセル化されたデータをインスタンス変数、手続きをメソッドと呼んでいる。メモリ中に、なんの制約もなく漂っていたデータの群れとそれらを操作するための手続きの群れが、オブジェクトというカプセルによって境界を設定され、データはオブジェクト内に隠蔽される(情報隠蔽)。したがって、

オブジェクトとは、抽象データであるということができる。また、ソフトウェアの管理という面から見ると、手続きをそれが関与するデータの種類によって分類することになる（図4）。

2.3 メッセージ交信

オブジェクトがデータの集合と手続きの集合をパックしたカプセルだとしても、それらのオブジェクトはどのようにして計算なり仕事なりを進めていくのだろうか。世の中にオブジェクトが1個しかなければ、そのオブジェクトの内部だけで仕事は済んでしまうかもしれないが、実際には多くのオブジェクトが協力して仕事を進めていかざるをえない。しかも中のデータはカプセルのおかげで直接アクセスすることができない。したがって、仕事は相手にお願いしてやってもらうことになる。そのための通信手段がメッセージである。

仕事を頼みたいオブジェクトに対して、やってもらいたい仕事の名前を指定するのである。数オブジェクトに「たし算」をしてほしいのか、「掛け算」をしてほしいのか、その手続名をメッセージとして渡すだけでよい。なぜなら、その手続名が指す手続き本体は、そのメッセージを受けたオブジェクトがもっているからである。結果はオブジェクトとしてメッセージの送り手に返ってくる。メッセージによってオブジェクトのもつ手続きを起動してやるといってもよい。こうして、オブジェクトのネットワークを送信メッセージと返信オブジェクトが飛び交い、計算が進行していく（図3(c)を参照）。

2.4 クラスとインスタンス

オブジェクト指向では、同じ性質をもつ、似たような振る舞いをするオブジェクトが複数あった時、それらをまとめて抽象化した存在を作り、それをクラスと呼ぶ。そして共通の性質をもつ個々のオブジェクトをそのクラスのインスタンスと呼ぶ。たとえば、AさんもK氏もM君もみんな人間の実例である。この時、オブジェクトA、オブジェクトK、オブジェクトMはすべてクラス人間のインスタンスである、というわけである（図5）。ここまでは、今までのプログラミング言語でも、型（タイプ）を用いて行なっていたことである。3や4や−1はみな整数型のオブジェクトであった。ただし、型と手続きを一体化し、インスタンスのテンプレートであるクラスとして利用できるようになったのは、抽象データ型言語が現われてからである。

2.5 インヘリタンス（継承）

オブジェクト間の共通の性質を記述したクラスの間にもある種の関係が成り立つ。たとえば、生物の分類において、雀クラスは鳥クラスの下位概念だし、動物クラスは鳥クラスの上位概念になっている。この時、雀クラスは鳥クラスのサブクラス、鳥クラスは雀クラスのスーパークラスであるという。動物も雀のスーパークラスになっているといってよい。この時、サブクラスはスーパークラスの構造と機能を引き継ぐことになるが、これをインヘリタンス（継承）という。雀は鳥から、「くちばし」や「翼」という構造を継承すると同時に、「空を飛ぶ」という機能

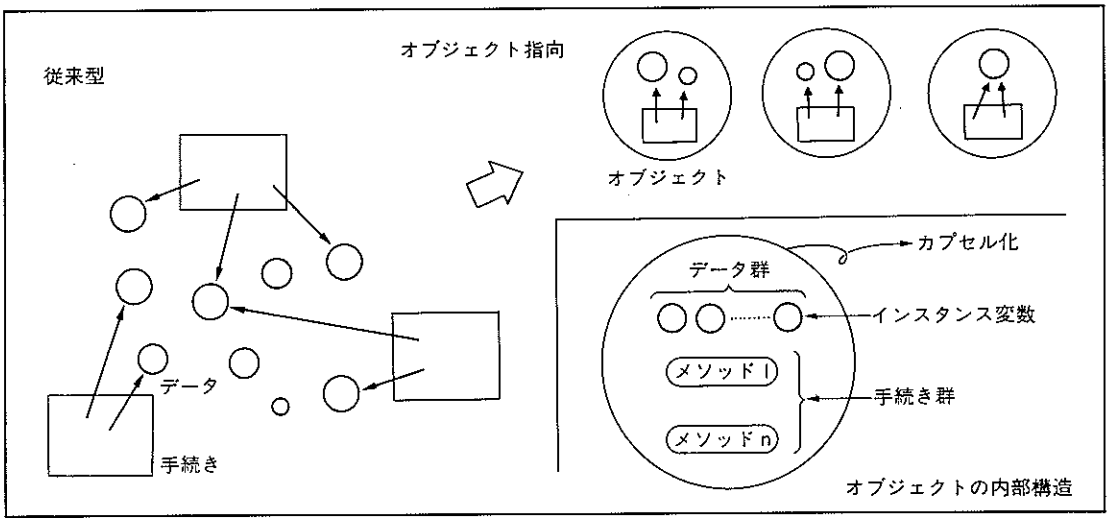

図4　オブジェクトによるデータと手続きのカプセル化

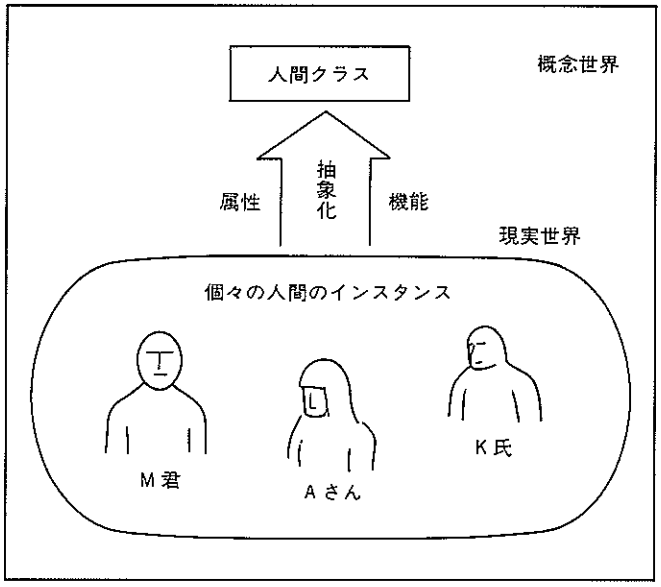

図5　クラスとインスタンス

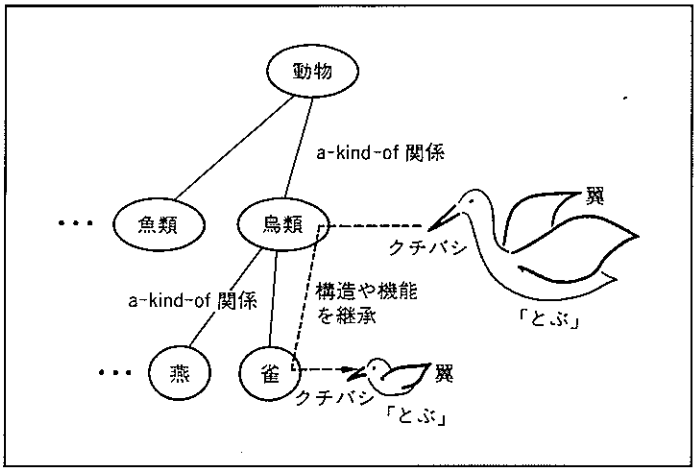

図6　インヘリタンス(継承)の概念

を継承する。オブジェクト指向においては、構造の継承はデータ構造すなわちインスタンス変数の継承に対応し、機能の継承は手続きすなわちメソッドの継承に対応する。継承関係は、知識表現におけるa-kind-of関係に相当すると考えてよい(図6)。したがって、クラスは全体として階層構造を形成することになる。もし、複数のスーパークラスからの継承を許すマルチプル・インヘリタンス（多重継承）の場合には、クラスはセミ・ラティス構造をとり、一種の概念ネットワークを形成する。ちなみに、Smalltalkの現在のバージョンは単一継承である。

オブジェクト指向においては、インヘリタンスを用いて既存クラスの構造と機能をできるだけ流用し、コードの再利用を行ない、新たな構造や機能あるいは既存クラスとは異なる構造や機能のみを追加的に記述する差分定義をすることになる。こうして、インヘリタンスの機能は、オブジェクトの概念分類に貢献すると同時にコードの管理をも同一の枠組みで行なっているのである。ちなみにSmalltalkの概念階層は図7のようになっている。

2.6 ポリモーフィズム（多相性）

2.3節でオブジェクトはメッセージのやりとりに

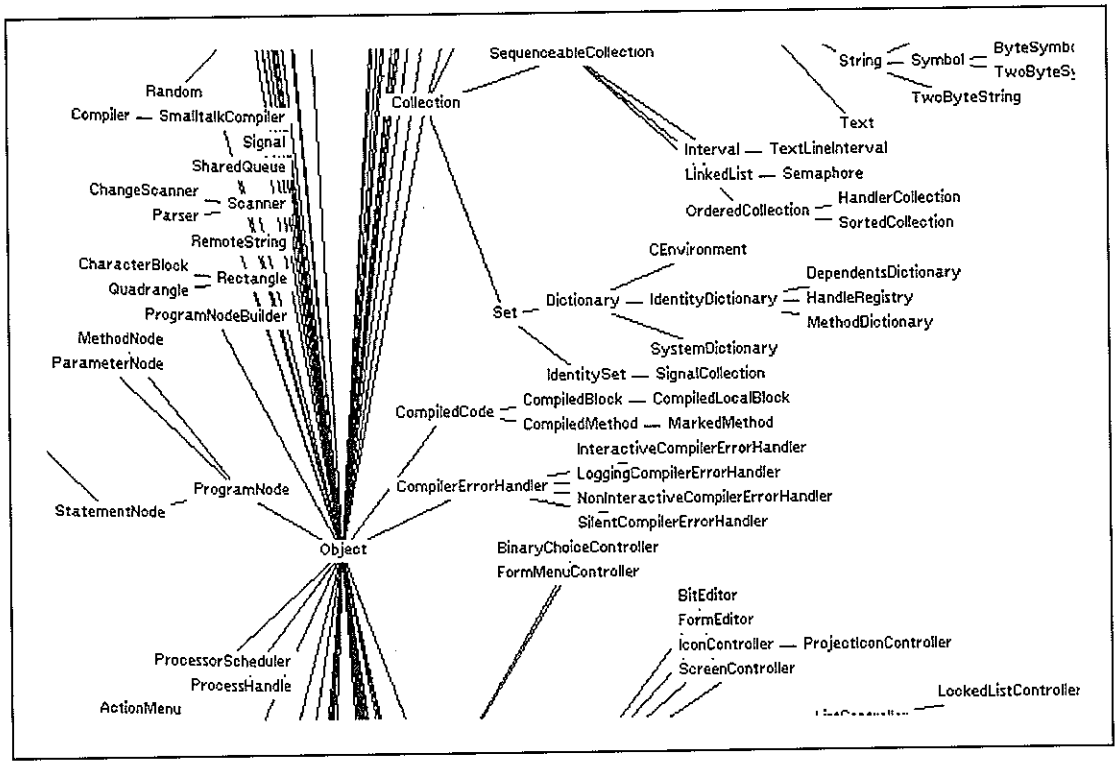

図7　Smalltalkの概念階層の一部

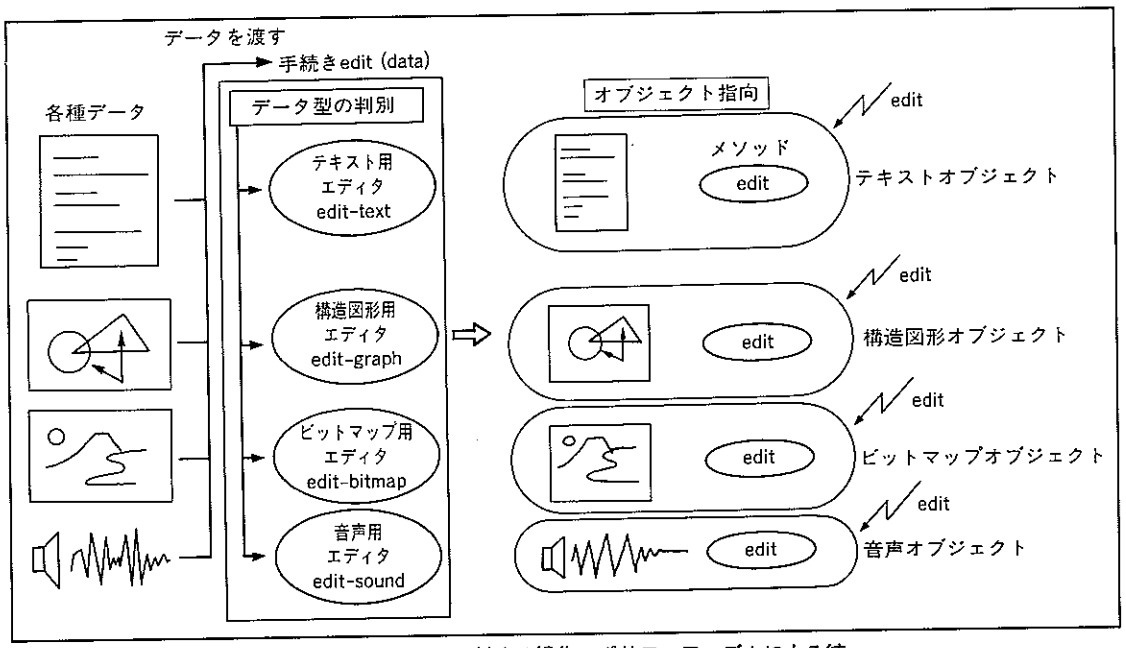

図8　多様なデータに対する操作のポリモーフィズムによる統一

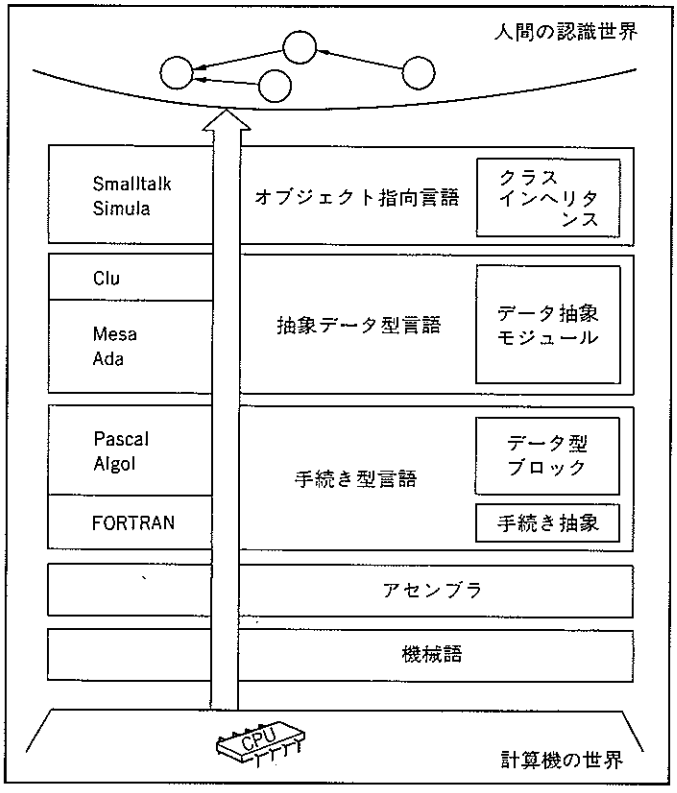

図9　プログラミング言語の進化

よって計算を進めていくことを説明した。メッセージはそれを受信するオブジェクトのもつメソッドを指定するための単なる合い言葉、送信者と受信者の間の取決めであった。すなわち、この取決め＝プロトコルを知っている相手であればどんなオブジェクトにでもこのメッセージを送ることができる。そして、メッセージを受け取った以上は、そのあとの処理はそのオブジェクトの責任となる（手続き名と手続き本体との実行時束縛)。この柔軟さは非常に大きなメリットを生む。

よく使われる例としてマルチ・メディア環境におけるさまざまな形態のデータの取扱いがある。たとえば、テキスト、(構造)図形、ビットマップ、音声の4種類のデータがあったとしよう。これらを編集するプログラムeditがあったとすると、従来は、このプログラムeditの中でデータ型ごとに分岐をして、それぞれの型に対応するエディタを呼び出していたことだろう。しかし、オブジェクト指向では、テキスト・オブジェクト、(構造) 図形オブジェクト、ビットマップ・オブジェクト、音声オブジェクトがeditというメッセージを受けた時点で、それぞれが自分の編集用メソッドを起動すればよいのである（図8)。

すなわち、オブジェクト指向におけるポリモーフィズムとは、メッセージの字面の同一性に基づいて、操作や機能に共通するある種の意味をオブジェクトに伝達することである。

2.7　まとめ

ここで、プログラミング言語の歴史を、その言語を構成する概念の発展という視点から振り返ってみよう（図9)。

図9のように、問題領域のモデル化にできるだけ自然に反映できる形にプログラミング言語は進化してきたといえる。逆に計算機の側からいうと、フォン・ノイマン型アーキテクチャをどれだけうまく隠蔽して人間の頭に見せるかという発展の歴史ともいえる。オブジェクト指向言語についても、細かくいうとそこでサポートしている概念とメカニズムに応じて分類が可能である。[Wegner 88]を参考にした図をあげる(図10)。クラスレスの言語にもインヘリ

タンスの概念を入れられるのかと疑問をもたれた方もいるかもしれないが、典型的なインスタンスをプロトタイプとしてその性質を転用するデリゲーション(delegation)というメカニズムを利用して実現する。また、このWegnerの分類ではポリモーフィズムを分類の観点に利用していないが、それを取り入れるとC++はSmalltalkに比べてポリモーフィズムの機能が弱いということになる。

3 認識の枠組みとオブジェクト指向

認識論との関係でオブジェクト指向をとらえる場合、認識の現場における問題とその知識を実践に応用していくプロセスの問題の2点が取り上げられる必要がある。

(1) 理解の枠組みとしてのクラス構造とポリモーフィック・ボキャブラリ
⇨オブジェクト指向分析の初期フェイズ

(2) 乗って航海しながら修繕するしかないノイラートの舟
⇨ソフトウェア開発・保守プロセス(スパイラル・モデル)

ここでは、(1)について主に話をする。(2)については、オブジェクト指向分析／設計／保守(＝進化)の話の際に改めて取り上げることにしたい。

3.1 認識の「として—構造」とクラス

オブジェクトは、現実世界に存在するのではなく、人間が現実世界の現象を理解しようとして、作り出すものである。イスやネコやタマネギは、そのようなモノとして世界から切り出すことが人間にとって有用だから、われわれは、おのおの独立したモノとして認識しているのである。ここで、注意しなければならないのは、人間の認識の構造である。わたしたちは、必ず、

モノaをXとして見る

という構造を通してしかモノを認識できない。この「モノaを」と「Xとして」とは同時に起こることに注意してほしい。「4本の棒の上に板の乗っかっている茶色の何か」を純粋にその形と色のまま認識する者はいない。歩き疲れた者にとってはそれは「座るための椅子」として立ち現われてくるし、高い所に置かれた荷物を取ろうとしている者には「乗っかるための台」として立ち現われるだろう。強盗と出くわしたものにとっては、この4本足のモノがとっさに「相手を叩くための武器」としてそこに置かれているように思うに違いない。このようにモノは、人間に

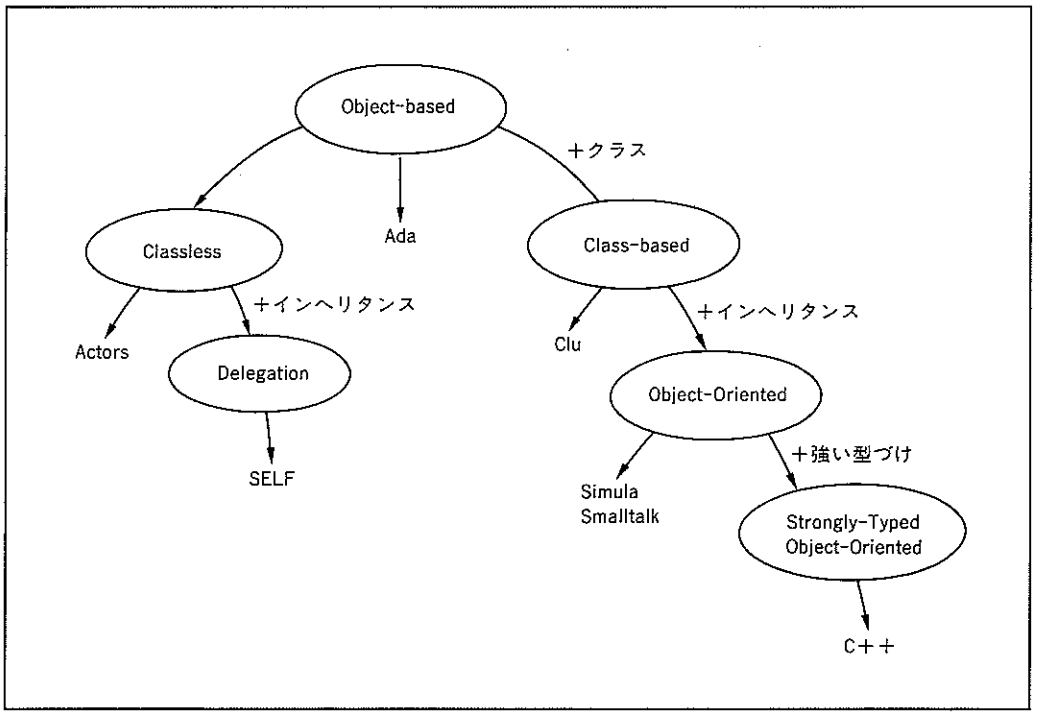

図10 オブジェクト指向言語の分類

認識される際には必ずあるカテゴリーの実例（インスタンス）として把握される。オブジェクト指向の用語では、このモノのカテゴリーのことをクラスといい、このクラスの実例として把握されたモノのことをオブジェクトという。すなわち、認識の「として－構造」をこれらの用語を用いて言い直せば、

オブジェクトaをクラスXのインスタンスとして理解する

ということになる。オブジェクトaはそのモノを指すための単なるポインタであり、その記述はクラスにおいてなされることになる。

ところで、われわれは、言語の習得を通じて世界には何が存在するのかについての基本的かつまわりの人間と共通な理解の枠組み（信念・知識のネットワーク）を保持するようになる。新しい現象・事象を認識する際には、必ずこの枠組みを通して理解をすることになる。したがって、どんな単純な対象の観察・分析も、それに先立つなんらかの理論を前提にして初めて成り立つわけである。

(1) 対象aの認識とは、「aをXとして見る」ということであり、認識プロセスには必ず「として－構造」が介在している。

(2) として－構造「aをXとして見る」におけるカテゴリーXは、さらに広い概念枠と連関してすでに理解されている。

この事実を認識のパラダイム依存性と呼んでもよい。したがって、この概念枠（カテゴリーのネットワーク構造）が変化すれば、それに伴って、その理論が把握する世界のありかたも変化してくる、すなわち、世界の分節化の仕方（オブジェクトの切出し方やそれらの関係付けの仕方）が変わってくる、ということにご注意いただきたい。

オブジェクト指向においては、以上述べてきたような概念枠は、オブジェクトの切出し方を示唆する「クラスの分類構造」とオブジェクト間の関係を規定する「インスタンスの部分－全体構造」を指すことになる。

3.2 クラス分類

分類というのは、認識の際に人間がとる最も基本的な戦略である。分類の基本は、ある物を他の物から区別・識別するということである。この時、それらのオブジェクトの間の属性や振る舞いに関する差異を基にしてクラス分類を作成していく。分類学では、クラスのことをタクソン、その分類の際の基準をクライテリアと呼んでいる。オブジェクト指向におけるクライテリアは、そのオブジェクトのもつ構造と機能、言い換えれば、どんなインスタンス変数をもっているか、どんなメソッドをもっているかという基準である。また、階層的分類体系では、同一階層に並んでいるクラスをパラレル・タクソン、それより相対的に上位のクラスをマクロ・タクソン、相対的に下位のクラスをミクロ・タクソンと呼んで区別している。

クラスがどのように分類され階層化されているかは、その人の属する社会なり文化なりが世界をどのように分類して見てきたかに依存している。その社会の歴史的な認識の遺産が認識の概念枠であり、オブジェクト指向においては、それはクラス分類構造であり、クラス・ライブラリということになる。よく整備されたクラス・ライブラリなしでオブジェクト指向を実践しようとするのは、脳なしで眼だけで物を見ようというのと同じである。像は写ってもそれを理解することはできない。

したがって、3.1節で述べたように、新しい対象を前にした時、その対象をどう分類するかは、すでにもっているクラス階層を基にして検討されることになる。とりあえず既存のクラスのインスタンスとして同定し、詳細がわかってくるに従って、サブクラス（ミクロ・タクソン）を作ってより細かい分類を行ない、分類体系を豊かにしていく。また、似たようなクラス（パラレル・タクソン）がいくつも並んでしまった場合には、それらの共通点を抽出してスーパークラス（マクロ・タクソン）を新たに設定することにより、分類体系の整合性を保つように努める必要がある。

オブジェクト指向においては、クラス分類のクライテリアは大きく分けて2つある。オブジェクトの構造（データ構造）による分類（構造継承）と機能・振る舞い（メソッド）による分類（機能継承）である。これについてはあとで述べる。

3.3 部分と全体、オブジェクト構造

オブジェクトは、あるクラスに属していると同時に、それ自身の内部構造をもっている。これはいわゆるデータ構造であるが、通常インスタンス変数の集合として表わされる。しかし、このインスタンス変数の解釈・意味付けに関しては、少なからず混乱

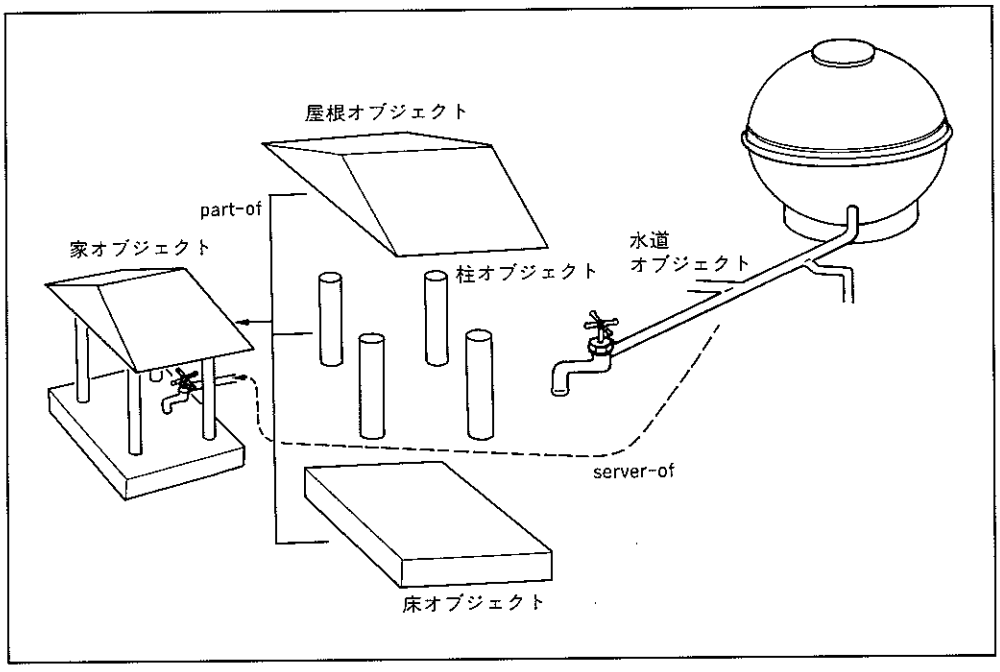

図11　オブジェクトの組立て構造

が存在している。簡単に言ってしまうと、部分－全体関係とサーバー－クライアント関係をごっちゃにしている場合が多いのである。

家が、屋根と柱と床から構成されている時、その家オブジェクトを全体とすれば、屋根オブジェクトと4本の柱オブジェクトと床オブジェクトという部分からなるといえる。これが典型的な部分－全体構造である。これを知識表現の用語でpart-of関係といったりする。一方、その家に水道が引かれていたとする。その水道はその家だけで利用しているのではなく、当然近隣の他の家でも引いて使っているとする。この場合、水道はこの家の部分ではない。この家では水道の供給を受けているのであり、水道システムを使わせてもらっているのである。すなわち、家はクライアントであり、水道システムはサーバーということになる。この場合にも、オブジェクト指向においては、インスタンス変数に水道システムを束縛することになる（図11）。

さらに、オブジェクトの内部状態を表現するのにもインスタンス変数が使われる。たとえば、あるオブジェクトが何回アクセスされたかを自分で記録しておくような場合、そのアクセス頻度はそのオブジェクトのインスタンス変数として保持しておく。しかし、このような内部状態は、そのオブジェクトの部分であるといえないことはないので問題は少ない。

このようにオブジェクト間の関係を規定するオブジェクト構造にはおおまかに2種類の関係が含まれる。part-of関係とserver-of関係である。part-of関係は、オブジェクトの内部構造にかかわり、オブジェクト間の結合は強固である。逆にserver-of関係はオブジェクト間の協調関係にかかわり、緩やかな結合を意味する。オブジェクト構造を考える場合、両者の関係をともに考慮する必要がある。

また、オブジェクト構造を情報、メッセージの流れに関する制約関係と見る視点も存在する。すなわち、全体から部分にメッセージを送る、あるいはクライアントからサーバーにメッセージを送ることはできるが、その逆は行なえない。ここでは、メッセージは木構造に沿って流れることになり、その木構造はオブジェクト構造を反映している。こうした意味からも、オブジェクト構造は部分－全体関係といった実体的なものではなく、参照可能関係とでもいうべきものであることがわかる。この参照可能関係は、MVCの話のところで出てくる依存性：更新従属関係とはメッセージの伝搬に関して双対の概念である。この話題はオブジェクト指向設計のところで詳しく行なう予定である。

3.4 メッセージ、メタファー、ポリモーフィズム

オブジェクトの共通の属性や振る舞いを抽出・抽象化したものがクラスであった。これは基本的には人間の認知能力によっている。人間の認識に寄与する能力としてもう1つ言語の運用能力がある。特にメタファー（隠喩）による表現能力が人間の認識において大きな役割を果たしている。現代レトリック論の始祖の1人I. A. リチャーズは『新修辞学原論』[Richards 36]において、

・「隠喩は言語に遍在する原理」であり、われわれの世界は「投射された世界」であり、「思考は隠喩的」である。

と述べている。また、レトリックと哲学の協働の可能性を開拓したブラックは、その画期的な論文『メタファー』[Black 62]において、

> ときにはメタファーは、既存の類似性を確認するというよりも、類似性を新たに生み出すと言ったほうが適切な場合がある。

と述べ、メタファー(隠喩)による創造的な認識の可能性を示唆している。具体的にはチェスの比喩を用いて戦争を記述するという例をあげて説明している。

> チェスの用語は、わたしが戦争の記述を行なう時、その記述に一定の方向性を与える一群の含意体系を決定する。チェスの用語を選択することによって必然的に戦争のいくつかの面が強調され、ほかは無視されることになる。

これは、すなわち、チェスの用語を戦争という対象に対してもポリモーフィックに適用するということである。これにより、戦争のチェス・ゲーム的な側面に焦点を当てることができる。さらに、もし戦争というものを知らない人間に戦争のことを説明しようとすれば、こうせざるをえないだろう。この例からもわかるように、オブジェクト指向におけるポリモーフィズムという概念は、まさにわれわれが日常の思考において頻繁に利用しているメタファーによる理解方式に対応するのである。

たとえば、「走る」という語を考えてみよう。この語は、「ポチが走る」や「ケンが走る」というように犬クラスや人間クラスのインスタンスに対して適用できる。ここで、次のような状況を想定してほしい。人類が初めて「車」を発明（発見？）した時のことである。地面の上を転がっていく「車」を目の当たりにして、彼／彼女はそれをなんと表現しただろう。みんななかなかうまい表現を思いつかず、当時の詩人が「車が走る」という表現を開発した。それを聞いた一般人はなんとうまい表現だといって使い始めたに違いない。詩人がもともとメタファーとして使ったものが徐々に不自然さを感じなくなり、通常の表現となっていき、初めの驚きは薄れていくと同時に、だれもが「犬が走る」や「人が走る」のと同じ現象として「車が走る」という現象を認識できるようになる。すなわち、同一の「走る」という既存の認識の枠組みを新しい現象に再利用したということになるのである。したがって、同一のメッセージによるポリモーフィズムとは人間のメタファーの能力を計算機へ導入することにほかならない。

人間は新しい現象に出会った時、それを表現するのに今までの手持ちの語彙をうまく再利用せざるをえない。そうした人間のメタファーに基づく認識の特性をうまく取り込み、積極的にシステムの分類・整理に利用していこうというのが、オブジェクト指向におけるポリモーフィズムである。それによって、認識の幅を広げることができる。2.6節の例で言えば、メッセージedit（編集する）によって「文章を編集する」から「図形を編集する」まで意味し、さらに今では「音声を編集する」が不自然ではなくなってきている。いずれは、「アイデアを編集する」や「建築物を編集する」ことまでもカバーするまでに認識は広がっていくのだろう。今は、メタファーとして使っている「プログラムが計算機上を走る」という表現もいずれは一般社会に市民権を得て日常表現となる可能性がある。その時、ソフトウェアやプログラムといった概念は今よりずっと具象的、実体的なイメージでとらえられるようになっていることだろう。すなわち、オブジェクト指向が日常化しているかもしれない。

3.5 オブジェクトのアイデンティティ

オブジェクトの特性の1つにアイデンティティをもつ、というのがある。オブジェクトは単なる値の集合ではなく、たとえオブジェクトの内容がわからなかったとしても、そのオブジェクトを識別できるべきだ、という考えに立っているのがオブジェクト指向である。

関係データベースにおいては、あるエンティティaとエンティティbがあった時に、もしそれらの間ですべての属性値が一致すれば、たとえ偶然の一致で

あってもその2つのエンティティは同一だとみなさざるをえない。属性の値以外に識別の手段がないからである。これは名前と年齢と性別のフィールドからなるデータベースにおいて、同年齢で同性の同姓同名の2人の人間を区別できないということである。

ところで、われわれはモノを識別するのにそれらの属性にのみ頼っているわけではない。端的に指示代名詞を用いて「あの人」とか「それ」とかいう形でモノを直示することができる。モノに対するこの直示の機能がオブジェクト指向におけるオブジェクトIDあるいはオブジェクト・ポインタといわれるものである。これによって、オブジェクトの識別をその属性値とは独立に行なうことができる。これによって、オブジェクトが存在することはわかっているのだが、その値はまだわからない（測定できない）といった状況のモデル化にも有効に対処できる。

これら2つの考え方の違いは、固有名詞に対する2つの解釈に対応する。関係データベース流に、個物とはその属性の束であると考えるのは、ラッセル流の記述理論に対応し、固有名はその個物のもつすべての属性を記述したものの省略形だと解釈する。一方、クリプキ流の固有名の解釈では、直示という名付けの儀式を重視し固有名の独自性を主張する。現代哲学においてはこの2つの理論は最大の争点の1つとなっているが、ことソフトウェアの世界ではオブジェクトIDの有用性に軍配が上がりつつあるようだ。しかし、オブジェクトIDすなわちポインタの参照の一貫性管理がユーザーに任されていることは、十分承知しておく必要がある。

冒頭の桜の例を考えてみると、染井吉野クラスの個体として何を指すべきか迷ってしまうことになる。DNAの並びというその生物の属性を基に考えると同じ個体ということになるし、各地の桜を「これ」といって名指してそれぞれを個体と識別すべきかは、現象とその認識者との間の関係で決まる。認識者の関心領域と有用性で使い分けるしかないだろう。このようにオブジェクトのアイデンティティというものも認識する側に依存した概念なのである。

4 オブジェクト指向のレトリック

4.1 隠喩、換喩、提喩——レトリックの技法

3章でメタファーについては説明した。それは、言葉の用法を進化せることにより認識を深める人間の言語能力の1つであり、ポリモーフィズムと通じるものであった。比喩あるいはレトリックの技法には、メタファー（隠喩）以外に、メトニミー（換喩）とシネクドキー（提喩）がある。これらについて簡単に説明しよう。

換喩とは、現実世界の隣接関係に基づく置換えによる比喩である。たとえば、

赤頭巾ちゃん

というのは、換喩表現になっている。1人の女の子を表現するのにその子のかぶっている赤頭巾で代用し

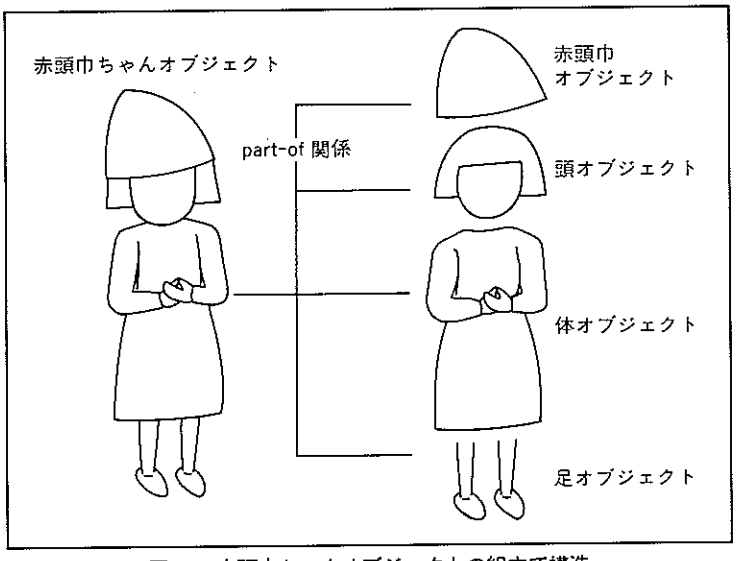

図12　赤頭巾ちゃんオブジェクトの組立て構造

ているからである(図12)。ここで注意しなければいけないのは、赤頭巾ちゃんも赤頭巾も現実世界の具体的なものであり、赤頭巾はそれをかぶっている1人の女の子の部分を構成している、という事実である。オブジェクト指向の言葉で言えば、1人の女の子のインスタンスの部分として1つの赤頭巾インスタンスが存在している、という全体ー部分の関係になっている。Smalltalkでなら、赤頭巾ちゃんオブジェクトのインスタンス変数に赤頭巾オブジェクトをバインドしておく状態といってもよい。

ヤカンが沸いている。

自転車がパンクする。

ヤカンと湯はどちらも「湯の入ったヤカン」の部分を構成し互いに隣接している。これは、同一の全体内の「部分ー部分」の関係である。一方、パンクするのは自転車のタイヤであり、ここでは、「自転車」という全体が「タイヤ」という部分を代用して表現している。これは、「赤頭巾ちゃん」とは「部分ー全体」の関係が逆になっている。これら3種類のパターンのいずれも換喩と呼ばれる。

つぎに提喩について説明する。提喩は換喩とは対照的に意味世界における包含関係に基づく置換えによる比喩である。たとえば、

花見に行く。

という表現で富良野にラベンダーの花を見に行ったりする者はいない。「花見」においては花は必ず桜を指す。「花」という上位カテゴリーで「桜」という下位カテゴリーを代用表現しているのである(図13)。この場合は、具体的な個々の菊や桜やラベンダーの花の間の関係ではなく、「花」と「桜」という意味カテゴリーの包含関係に関する言及がなされているわけである。オブジェクト指向の言葉を借りれば、ク

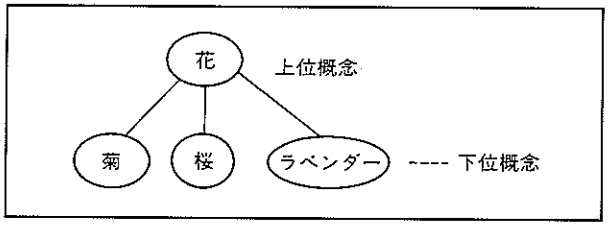

図13　意味世界における概念間の関係

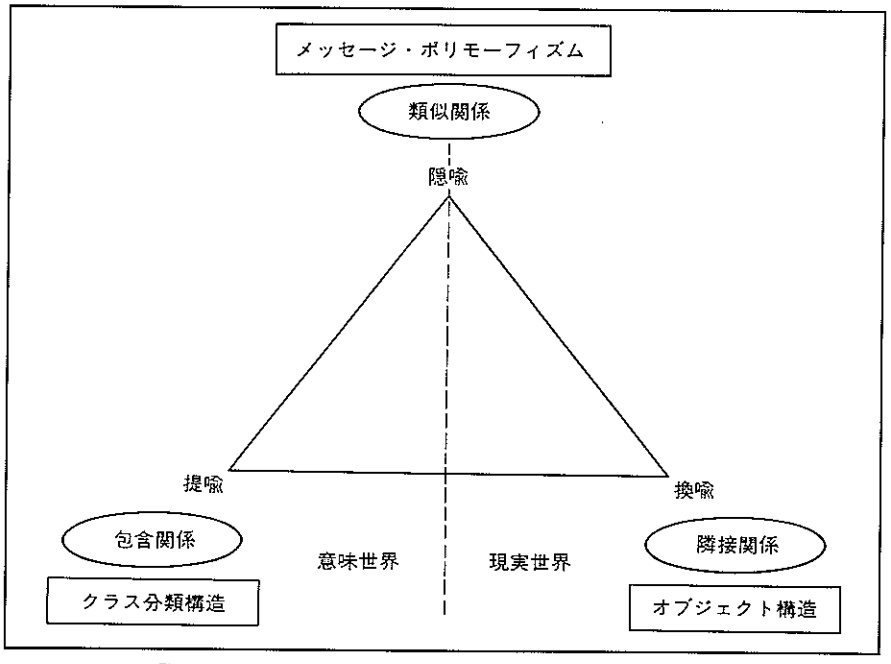

図14　レトリック技法とオブジェクト指向による認識の構造

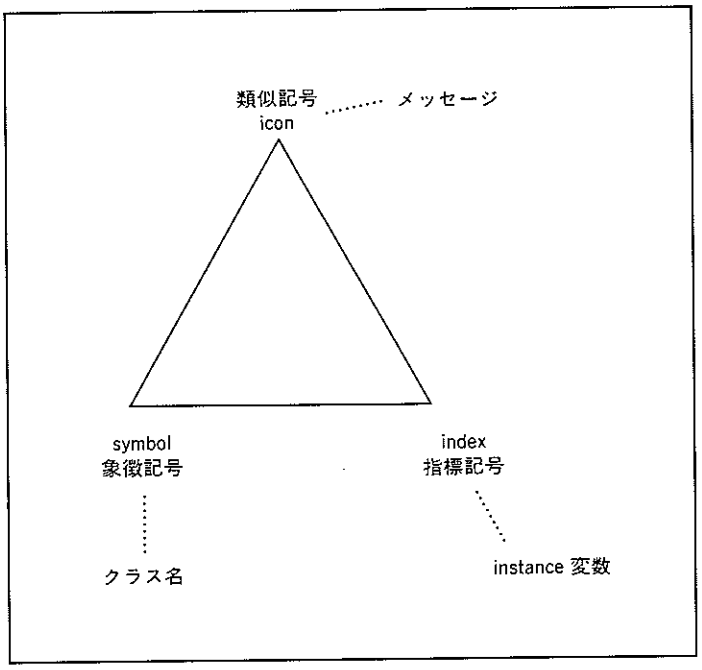

図15　パースの記号3分法とオブジェクト指向

ラス間の上位－下位関係、スーパークラス－サブクラスの関係といえる。

人はパンのみにて生くるにあらず。

この例では、パンというクラスで食物一般のクラスを指しており、パンは食物の下位クラスとなっている。

以上から、隠喩、換喩、提喩という各レトリック技法の間の関係、特に認識におけるこれら3種類の技法の役割といったものをまとめると、図14のようになる［瀬戸 86］。

4.2　オブジェクト指向の三位一体論

オブジェクト指向のキーワードは、オブジェクト組立て構造、クラス分類構造、メッセージ・ポリモーフィズムの3つにまとめられる。オブジェクト組立て構造は現実世界により近い存在であり、実際のインスタンスとそれらの間の関係に対する言及であるため、実際の現象を詳細に仕様化する際には、最も注目しなければならない構造といえよう。これはレトリックでいえば、現実世界のモノの隣接関係に基づく換喩に相当する。クラス分類構造のほうは、概念を抽象したクラス間の関係を論じるものであり、オブジェクトを認識する際の枠組みとなるものである。いわば意味世界における構造であり、これは当然、レトリックでいえば提喩に相当する。そして、メッセージ・ポリモーフィズムは、さきほど詳細に論じたように、オブジェクトのもつ機能を人間のメタファー運用能力に基づいて分類した構造であり、オブジェクトの機能を認識する際の枠組みとなるものである。オブジェクトの機能の類似性を同一のメッセージで抽象するという意味で、ポリモーフィズムは現実世界と意味世界を仲介する役割をもつというのもメタファーの機能とパラレルである。

まさしく、オブジェクト指向は、認識の3つの次元に広がる認知の枠組みを取り込んだ考え方であることがわかる。人間がレトリックを繰って言語を自在に運用していく様が自然であるならば、オブジェクト指向が人間にとって自然であるのになんの不思議もない。というわけで、今回の結論は、

すべからくオブジェクト指向のソフトウェア技術者はレトリックの素養をもつべし。

ということになろうか（きちんと「べし」で止めたので呉智英先生も許してくださるに相違ない）。

4.3　おまけ――デスクトップ・メタファーを超えて

さきほどの認識の三位一体構造図に対してパースの記号の3分類が対応する（図15）。図を見ると、シンボルがクラスに、インデックスがインスタンス変数に、そしてアイコンがポリモーフィック・メッセージに自然に対応することが理解できるだろう。こ

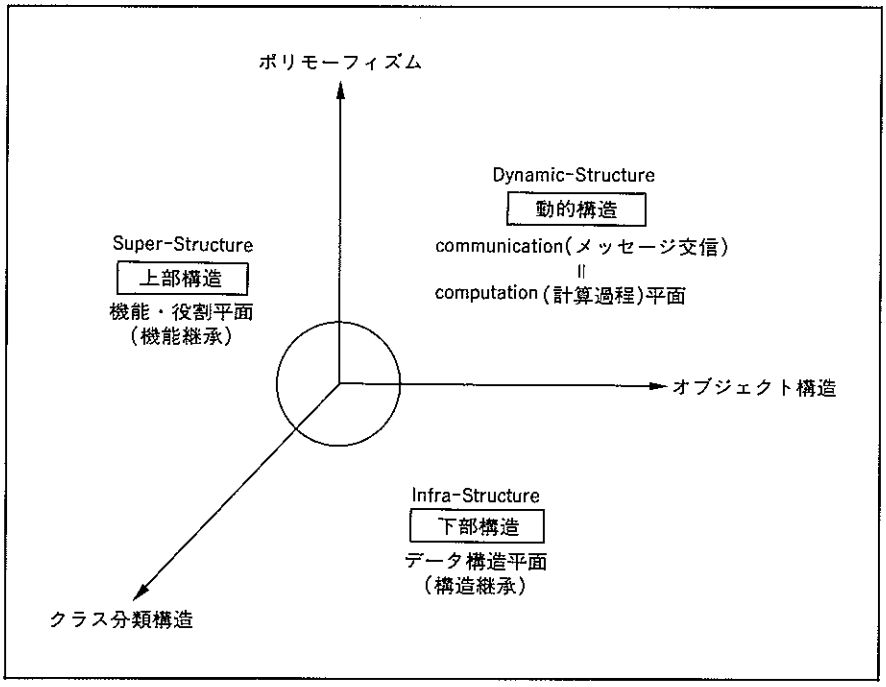

図16　オブジェクト指向を構成する多次元平面

のことから、StarやMacをはじめとして今ではWindows 3.0でお目にかかれるような通常のデスクトップ・メタファーにおける「アイコン」という語の使い方は誤っていることがわかる。あそこでアイコンと呼ばれいるものはパースに従えば、シンボルやインデックスと呼ぶのがふさわしい。そして、正しい用語法に従ったメタファーを導入するには、2次元の単なるデスクトップでは無理なのである。意味世界平面と現実世界平面そしてそれらを貫くポリモーフィズムの軸、これらを実現するにはどうしても3次元以上の表現・操作環境が要求されている。それはデスクトップ・メタファーを超えてオブジェクト指向を自然に体現した環境であるはずだ。そのイメージ図を図16としてとりあえずあげておく。この内容については次回以降説明したい。

参考文献

[瀬戸 86]「レトリックの宇宙」、瀬戸賢一、海鳴社、1986

[中尾 90]「分類の発想」、中尾佐助、朝日新聞社、1990

[Wegner 88] "Object-Oriented Software Engineering, Tutorial Notes", Peter Wegner, ICCL 88, 1988

[布施 88]「脳の中の美術館」、布施英利、筑摩書房、1988

[佐々木 85]「創造のレトリック」、佐々木健一編、勁草書房、1985

[米盛 81]「パースの記号論」、米盛裕二、勁草書房、1981

[池田 86]「構造主義生物学とは何か」、池田清彦、海鳴社、1986

[ピンソン 90]「Smalltalk：オブジェクト指向プログラミング」、L. J. ピンソン、R. S. ウィナー、富士ゼロックス情報システム訳、トッパン、1990

[ソシュール 72]「一般言語学講義」、小林英夫訳、岩波書店

[クリプキ 85]「名指しと必然性」、八木沢敬、野家啓一訳、産業図書、1985

[飯田 88]「言語哲学大全I」、飯田隆、勁草書房、1988

[Quine 53] "From a Iogical Point of View", Harper & Row, 1953

[Ingalls 81] "Design Principles Behind Smalltalk", Daniel H. H. Ingalls, Byte, 1981 Aug.