第3章　オブジェクト指向計算モデルの導入

3.1　どのように写し取るのか: 計算モデルの導入

モデル化方法について考えるためには、対象を「どのように写し取るのか」(how) ということを考える必要がある。本論文の対象となる複雑系のモデルは、従来の言語モデルや図式モデル、数学モデルで扱うことが困難であることから[29]、ここでは、「計算モデル」(computational model) という表現形式を導入することにしたい。計算モデルは、コンピュータ・サイエンスとソフトウェア工学の分野で考えられ発展してきたものであり、処理の種類と対象を記述してあるため、「計算」(computing)[30]を行うことができる。

計算モデルは、数学モデルに比べて、社会科学のモデル化に適していると言われている。例えば、Gilbert and Troitzsch (1999) では、計算モデルは (1) 現実との対応関係の把握が容易であり、(2) 並列的なプロセスや順序の決まっていないプロセスの扱いが容易であり、(3) モジュール性[31]をもたせることが容易であり、(4) 多様な主体を組み込んだモデルの構築が容易であるとしている。また、Knuth (1985) は、数学とコンピュータ・サイエンスを比較して、次のような暫定的な結論を導いている。数学には、「複雑度」すなわち「操作の節約」 という概念がほとんどなく、また過程の状態に関する動的な概念 (代入の概念) がないのに対し、コンピュータ・サイエンスでは、同時に実行される諸過程間の相互作用を研究するときにも状態の概念が重要となり、また、本質的に均質でない諸概念に柔軟に対処できる、としている。

計算モデルの代表的なものには、手続き型計算モデルや関数型計算モデル、論理型計算モデル、オブジェクト指向計算モデルなどがある (図 3.1)。近年の計算モデルの発展は、「命令から宣言へ、手続きからオブジェクトへ、逐次集中から並列分散へ」(青木 1993) という方向にある。第一の点は、「これをやってから、次にこれをやる」というように詳細な計算手順を命令的に記述するスタイルから、「これはそれとこのような関係にある」というように、計算の意味を宣言的に記述するスタイルになってきているということである。第二の点は、主に計算手順を記述するスタイルから、計算手順とデータをひとまとまりとして扱うようなスタイルになってきているということである。そして第三の点は、計算の実行がひとつのところで集中的に行われるというスタイルから、分散して存在する複数の実行部が、協調して計算を行うというスタイルが多くなってきたということである。このような方向性によって、計算モデルの特

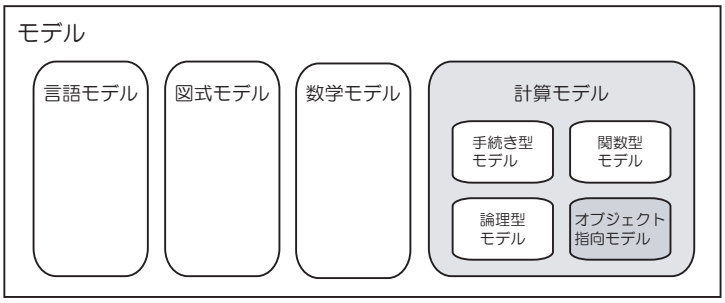

図 3.1: モデルの種類

徴がますます強化されているが、そのなかでも特に有力なパラダイムのひとつが、本稿で導入する「オブジェクト指向」である。以下では、オブジェクト指向によるモデル化について、簡単に見ていくことにしよう。

3.2 オブジェクト指向計算モデルの考え方

計算モデルのなかでも、オブジェクト指向によるモデル化は、人間の認知の仕組みに近く、より自然なモデル化が可能であるといわれている。「オブジェクト指向」では、物理的あるいは概念的なモノのひとつひとつを「オブジェクト」として捉え、その状態や関係の変化で対象となる現象をモデル化する。ここでいう「オブジェクト」とは、現実に存在する有形無形の「モノ」(thing) を表す一つのまとまりのことである。日常的な例でいうと、机や椅子、テレビ、携帯電話、新聞、人間、犬、観葉植物などは、どれもオブジェクトとして表現することができる。モデルの観点からいうと、オブジェクトとは、現実世界における個々のモノを、その「状態」と「振る舞いのルール」をひとまとめにして (カプセル化して) 表現したものであり、もっとも基本的な思考の単位となる。

オブジェクト指向では、さまざまなオブジェクトが、それぞれ役割を分担しながら相互作用することで世界が動いていると捉える。オブジェクト間の相互作用は、互いにメッセージを送り合うことで表される。オブジェクトは、自分の知っている (リンクをもつ) オブジェクトに対してメッセージを送ることができ、そのメッセージを受けたオブジェクトは自分の状態を変化させたり、必要があれば何らかのメッセージを送り返したり、他のオブジェクトにメッセージを送ったりする。これが、オブジェクト指向の基本的な世界観である (図 3.2)。状態と振る舞いのルールを保持しているオブジェクトが多数存在し、それらが相互作用しているという点が、オブジェクト指向の本質である。

オブジェクト指向が現実世界のモデル化に適しているのは、実はオブジェクト指向が考案された背景が関係している。オブジェクト指向の起源は、1960 年代にノルウェーで

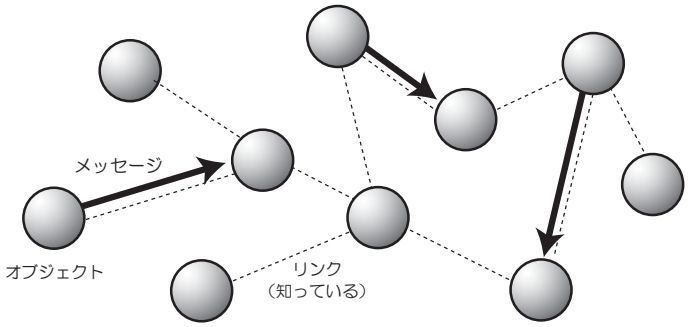

図 3.2: オブジェクト指向のイメージ

開発された Simula というコンピュータ言語にさかのぼる (Dahl and Nygaard, 1966)。Simula は ”simulation language”の略で、その名が示す通り、現実世界のさまざまな現象をシミュレートするためのコンピュータ言語であった。Simula は、何千もの構成要素からなる複雑なシステムのモデルをコンピュータ上で動かすことを目的に設計されたため、動的で複雑な現実世界をそのままコンピュータ上に取り込むための工夫がなされている。このような問題意識から生まれた考え方だからこそ、オブジェクト指向は、複雑な現実をモデル化する際の有力な方法となり得るのである[32]。

3.3 クラスによるモデル設計

3.3.1 概念とクラス

オブジェクト指向がもつ記述力を発揮するためには、「クラス」という考え方を導入する必要がある。「クラス」とは、複数のオブジェクトを共通の性質ごとに分類したもののことである。クラスを用いることによって、オブジェクトの体系的な整理が可能となる上、効率的な記述が可能となる。モデル化の対象となる現実世界はさまざまなものから構成されているため、これらを個別に把握したりモデル化したりしようとすると、時間と手間が膨大にかかってしまう。クラスの考え方を用いれば、よく似たオブジェクトを個別記述していかなくても、共通項を一括して表現できるようになる。

オブジェクトをクラスに分類するという仕組みは、実は、人間の認知プロセスにおける「概念化」と同じメカニズムである。私たち人間は世界のさまざまな物事を認知するとき、複数のものを共通する性質に着目して、概念を用いてまとめて把握したり、体系化したりしている (図 3.3)[33]。具体的なものと概念の関係、そしてオブジェクトとクラスの関係の対応を図示すると図 3.4 のようになる。認知における「具体的なもの」と、オブジェクト指向におけるオブジェクトは、それぞれ概念とクラスの「インスタンス」(実例) と呼ばれる (図 3.5)。クラスは状態をもたないが、そのインスタンスであるオブジェクトは必ず状態をもつ。個別のものをクラス (概念) で分類するとい

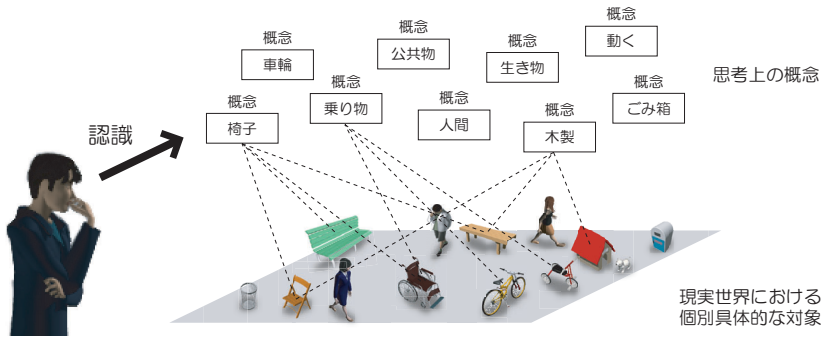

図 3.3: 現実認識における概念

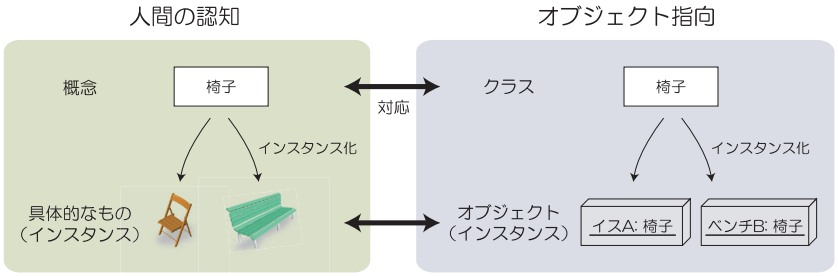

図 3.4: 人間認知における「概念」とオブジェクト指向における「クラス」

うことは、世界の複雑さに対処するためのひとつの方法なのである。

オブジェクト指向では世界の動きを複数のオブジェクトの相互作用で表現することはすでに述べた通りであるが、このときもそれぞれのオブジェクトを生成するクラス同士の関係として一括して定義することができる。オブジェクト同士の関係は、それぞれの型であるクラスどうしの関係に準ずるからである。クラスの関係は、関連 (association) と呼ばれる。関連のインスタンスはリンク (link) と呼ばれ、関連するそれぞれのクラスのインスタンスであるオブジェクト間の意味的な結び付きに用いられる (図 3.6)。あるクラスが他のクラスとまったく関連を持たなければ、そのクラスのインスタンスであるオブジェクトは、協調作業することのない孤立したオブジェクトということになる。

3.3.2 クラス間の関連

人間は現実世界を認知する際に、その複雑さに対処するために複数の概念を関係づけるメカニズムを利用しているのだが、オブジェクト指向でも同様に複数のクラスを関係づけるメカニズムを用いる。ここでは、特によく用いられる「汎化／特化」と「集

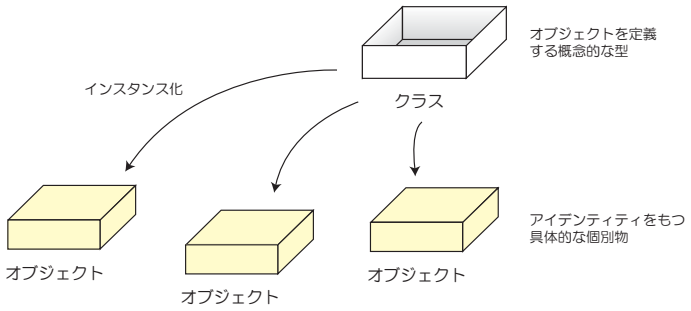

図 3.5: クラスとオブジェクト

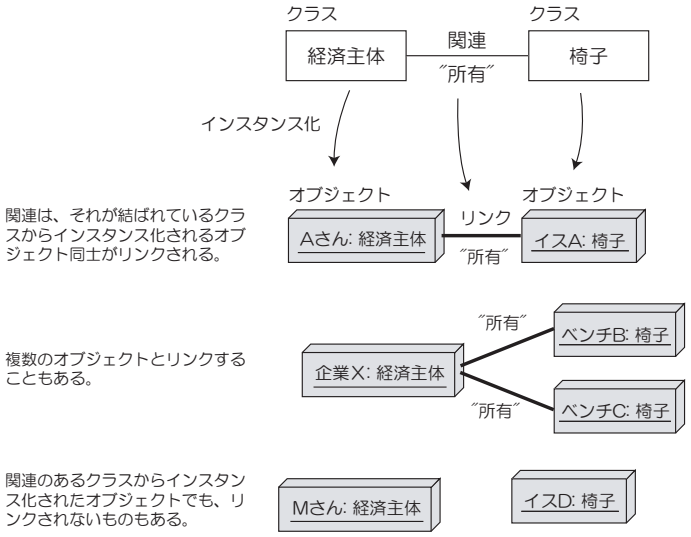

図 3.6: クラス間関係とオブジェクト間関係

約」について説明する[34]。

「汎化 / 特化」(generalization / specialization) の関係は、共通する属性や振舞いを一括して定義した上位クラスと、特化した下位クラスによる階層関係である (図 3.7)。この階層において、上位クラスは下位クラスを「汎化」したものであり、逆に下位クラスは上位クラスを「特化」したものとなる。この関係は、「a-kind-of」(〜の一種である) ということができる。汎化／特化の関係を用いる利点は、あるクラスが他のクラスよりも包括的であることを識別することで、クラスを体系化することができる点にある。また、この関係を用いて、少しだけ違うようなもの同士を体系化して整理することもできる。また、モデル記述の観点から言えば、共通の振る舞いや属性は上位のクラスで一度だけ記述すれば良く、下位クラスで記述する必要がないという利点もある。おそらく、クラス階層の下位クラスでは上位クラスにはない特化した属性や振舞いをもっていると思われるが、上位クラスの属性や振舞いはすべて下位クラスが

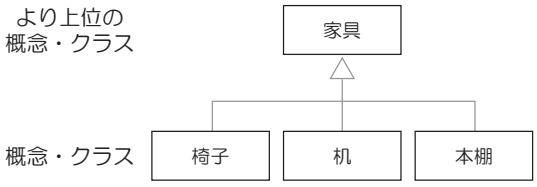

図 3.7: 概念の特化／汎化

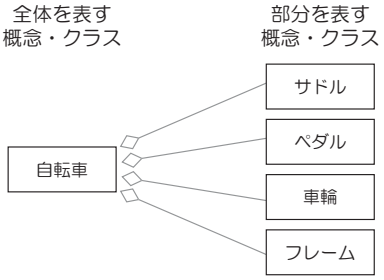

図 3.8: 概念の集約

引き継ぐ。これを「継承」というのだが、このメカニズムによって、下位クラスには、上位クラスと下位クラスとの差の部分のみを記述すればよいということになる。

複雑さに対処するためのもうひとつのメカニズムは「集約」(aggregation) である。集約は、あるオブジェクトが、別のオブジェクトから構成されていることを表す関係である。コンポジション (composition) の関係は、全体をあらわす全体クラスと、その構成要素にな部分クラスの関係であり、「composed of」(〜から構成される) ということができる (図 3.8)。コンポジションの利点は、多くの部分から組み立てられているものを 1 つの全体として扱うことができる点にある。全体に対する属性が、部分についても適用できるのである。例えば、「自転車」オブジェクトは、「サドル」、「ペダル」、「車輪」、「フレーム」などのオブジェクトから構成されている。この自転車が他の場所に移動すれば、その部品も移動するだろう。また、オブジェクトの構成要素が、時間とともに変化することを許すことでもある。先ほどの「自転車」オブジェクトにある時点で「ライト」が装着されるかもしれないが、その後は「ライト」オブジェクトも含めて移動することになるのである[(35)]。

3.4 UML(統一モデル化言語)

オブジェクト指向のモデルを記述するための記法は、近年、UML(Unified Modeling Language: 統一モデル化言語) として、標準化されている。UML は、「ソフトウェアシステムの成果物を規定、構築、可視化、文書化する言語」(Object Management Group,

2000) であり、「大規模で複雑なシステムをモデル化する上でその有効性が実証された工学上の最良の実践を収集し、代表するもの」(Object Management Group, 2000) である。UML の開発の背後にあった目標の中で、「第 1 の最も重要な目標は、UML がすべてのモデル作成者が利用することのできる汎用のモデリング言語となること」(Rumbaugh et al., 1999) であり、そのため「UML は所有権の設定されたものではないと同時に、コンピュータ業界の大多数による共通の合意の基づいたもの」(Rumbaugh et al., 1999) であるという。また、「できるだけシンプルでありながら、それでいて構築しなければならない広範な実用システムをモデリングできるようにすること」(Rumbaugh et al., 1999) が目指された(36)。

本論文で用いる UML 図の記法については、付録 A にまとめておく。