第6章　モデル・パターンの提案

6.1　モデル・パターンとは

これまでの章では、フレームワーク (Boxed Economy Foundation Model) における要素と、ソフトウェア (Boxed Economy Simulation Platform) におけるコンポーネントという部品を組み合わせることで、モデル全体を作成する仕組みについて提案してきた。このような仕組みは、大規模で複雑なモデルを作成する場合に、強力な支援の道具となるだろう。

しかし、モデル作成者に対する支援は、はたしてこれだけで十分だろうか。私は、時間の流れに伴う変化・生成を扱うモデルの作成支援としては、不十分だと考える。なぜなら、Boxed Economy Foundation Modelでは、モデル要素とその関係性について整理しているが、動的な振舞いの構成については構文的な定義しか行っていないからである。これに加えて、種々の動的な振舞いをどのように実現できるのかという整理が必要であると思われる。

そこで本章では、モデル要素をどのように組み合わせて、動的な振舞いを構成すればよいのかということを、「パターン」の考え方を取り入れてまとめることにしたい。パターンの考え方は、もともと建築デザインのために考案され、その後ソフトウェア・デザインに取り入れられたものである。本章ではさらに、モデル・デザインへ応用することを提案し、それを「モデル・パターン」と呼ぶことにしたい。本章でのモデル・パターンは、次章とその次の章におけるモデル事例を作成する際に用いることにする (各パターンの詳細は、付録 C にモデル・パターン カタログとして示してある)。

6.2　パターンによる記述

6.2.1　パターンとは何か

ここで取り上げる「パターン」とは、分析や設計の際に繰り返し現れる問題を明らかにし、その問題の解法をまとめたもののことである。パターンを利用することによる利点は、大きく分けて二つある。一つは、熟練者が自らの経験から得た経験則を明文化しているため、その問題の初心者であっても、効率的かつ洗練された方法でその問題を解決することができるという点である。もう一つは、その設計原理についての共通の語彙を提供するので、これまで直接指し示すことができなかった関係性などに

ついて、簡単に言及することができるようになるという点である。それゆえ、パターンは「複雑な設計を行うための建築素材である」(Gamma et al., 1995) といわれる。

6.2.2 パターンの基本構造

パターンは、「状況」、「問題点」、「解決策」という 3 つの観点で構成されるルールである (Alexander, 1979, 邦訳 p.202)。 状況とは、どのような時にそれを用いるのかということであり、パターンの適用条件である。問題点とは、何の問題を解決したいのかということであり、パターンを適用する目的である。解決策は、設計の要素や、それらの関連、責任、協調関係などである。これらに対し、問題とその解法を簡潔に記述した「名前」をつけることで、パターンが作成される(53)。

これらのパターンは、単独で使うというよりは、使う人が状況に合わせて組み合わせて使うことが前提となっている。それゆえ、それらを体系づけたパターン言語やパターンシステムが志向される。パターンの体系を得ることができれば、一つのパターンを適用した結果、新しい文脈が生まれ、そこに次のパターンを適用する ··· というように連鎖的な適用を行うことができるようになる。

なお、パターンを集めて体系化して記述したものを、パターンの「カタログ」と呼ぶ。カタログ内の各パターンは、統一したフォーマット (テンプレート) に従って書かれている。

6.2.3 パターンの役割

パターンを明示化し共有する第一の意義は、巧みな設計の再利用とその生成力にある。パターンは一種のルールであり、その目的を実現するためにどうすべきかを述べている。このような経験に基づいたパターンを知っていれば、そのような巧みな設計を再発見する必要がなくなるため、効率的によい設計を実現できる。それゆえ、その設計問題に対する初心者であってもパターンを身につけていれば、生成力のあるパターンの力を借りて、これまでの定石やよい設計を行うことも可能となる。「私たちの頭にあるパターンは動的であり、力をもち、生成力を備えている。それは私たちになすべきことを教えてくれる。それをいかに生成すべきか、または生成できるかを教えてくれる。さらに、一定の環境ではそれを作り出すべきだと教えてくれるのである。」(Alexander, 1979, 邦訳 p.151)。パターンは、繰り返し現れる問題と解法における関係性を抽象レベルで表しているため、それでどのような具体的な問題を扱うのかということは限定しておらず、多様な具体物を生成するための生成機構となり得るのである(54)。

パターンを明示化し共有する第二の意義は、設計に関する共通の語彙 (ボキャブラリ) を増やし、コミュニケーションを支援するということである。「パターンに名前を

付けることで、設計における用語の語彙を増やすことになる。それによって高い抽象レベルで設計することが可能となる。パターンに関する語彙が増えれば、同僚と議論したり、文書に記録したり、自分自身で考えを整理するのにも役立つ。設計に関して検討したり、設計上のトレードオフを人に伝えることも容易になる。」(Gamma et al., 1995, 邦訳 p.15)[55]

6.2.4 これまで提案されてきたパターン

建築におけるパターン

クリストファー・アレグザンダーは、建物や街の形態に繰り返し現れるものを観察し、それが要素の関係性、すなわちパターンであることを突き止めた。『時を超えた建設の道』で、建物や街を組み立てる本質を探っている際に、「『要素』は単純な積み木に見えても、実は変化しつづけ、現れるたびに異なる」(Alexander, 1979, 邦訳 p.73) ため、「この要素と称するものは空間を構成する究極の『原子』とは言えない」(同上) としている。そして、「要素と要素との関係も、要素そのもの以上にくり返し発生する」(同上, 邦訳 p.75) ということに注目する。ここで、各要素間の関係のパターンという考え方が登場するのである[56]。

アレグザンダーは、建物や街を構成するためのパターンの数はそれほど多くないと指摘し、建物は 2、3 ダースのパターンで定義でき、都市も 2〜300 のパタンで定義できるという (Alexander, 1979)。このように、比較的少ないパターンで世界が構成されるのは、「私たちは自分の頭にある類似のパタンからこの世界の現実のパタンをイメージし、心に描き、作り出し、建設し、そこに住む」ためであるという[57]。

Alexander (1977) では、8 年間の建設作業や計画作業をまとめた 253 のパターンが紹介されている。アレグザンダーは、このようなパターン言語を普及させることにより、デザイン・プロセスへのユーザー参加を実現しようとした。「これを活用すれば、隣人と共に自分の町や近隣を改良したり、家族と共に自宅を設計することができる。また関係者と力を合わせて、オフィス、作業場、学校のような公共建物も設計できる。さらにこのランゲージは、実際の工事手順の手引きとしても使える」(Alexander, 1977, 邦訳 p.ix) のである。

ソフトウェア開発におけるパターン

1980 年代後半に、建築の分野で提案されたパターンの考え方を、ソフトウェアの分野に適用するというヴィジョンが提案された (Beck and Cunningham, 1987; Rochat and Cunningham, 1988)。また、同じ頃、E. Gamma は GUI フレームワークの設計上のパターンを抽出・記述した (Gamma, 1991)。

ソフトウェア開発におけるパターンには、大別すると、「アナリシスパターン」、

「アーキテクチャパターン」、「デザインパターン」、「プログラミングパターン」などの種類がある。アナリシスパターンは、分析の際に繰り返し現れるパターンであり、ソフトウェアの設計や実装ではなく、ビジネスドメインの概念構造を反映したものである (Fowler, 1996)。アナリシスパターンに分類されるもののなかで代表的なものには、Fowler (1996) のアナリシスパターン、Coad et al. (1995) のパターン、Hay (1996) のデータモデルパターンがある (中谷および青山, 1999)。典型的な分析モデルの例を示しており、パターン言語の形式を採用していない。

アーキテクチャパターンは、典型的なソフトウェア全体の構造 (ソフトウェア・アーキテクチャ) に関するものである。構成要素とその役割、それらの関係について記述するのである。アーキテクチャパターンのカタログとして有名な Buschmann et al. (1996) では、「混沌から構造へ」「分散システム」「インタラクティブシステム」「適応型システム」などが提案されている[58]。このほか、Shaw et al. (1996) によってまとめられたアーキテクチャパターンもある。

デザインパターンは、設計段階において、優れた設計に繰り返し現れるオブジェクトとその構成を記述したものである (Gamma et al., 1995)。「種々状況における設計上の一般的な問題の解決に適用できるよう、オブジェクトやクラス間の通信を記述したもの」(Gamma et al., 1995, 邦訳 p.15) というように、アーキテクチャパターンに比べると、局所的な構造や振舞いをパターン化したものだといえる[59]。

プログラミングパターンは、イディオムとも呼ばれ、プログラミング言語に特化したプログラミングスタイルのことである。Coplien (1992) では、良いプログラムを書くのに重要なのは、文法だけでなく、「イディオム」と「スタイル」であると指摘している。例えば、Beck (1997) などのように、プログラミング言語に依存しているのが普通である。記述は、直接的にソースコードなどを提示することで、パターン言語よりも簡潔に書かれることが多い。

プロジェクトマネジメントのパターン

プロジェクトマネジメントのパターンは、開発組織やプロセスに関するパターンである。組織をパターンの観点で分析するのは新しいことではないが、Coplien (1995) はそれに生成的パターンとして持たせることを最初に提案した。Coplien (1995) は、「パターンは、とりわけ組織の構築と発展に適している。パターンは、関係のパターンによって文化を定義する、より現代的な文化人類学の基礎を形成する。」(Coplien, 1995, 邦訳 p.347) と述べている。その後、Ambler (1998); Ambler (1999) としてまとめられている。

この他にも、これまでの優れた手本としてのパターンではなく、悪い見本のパターンというのもまとめられている。このような反面教師のパターンは、Koenig (1998) で「アンチパターン」と名づけられ、その後、Brown et al. (1998); Brown et al.

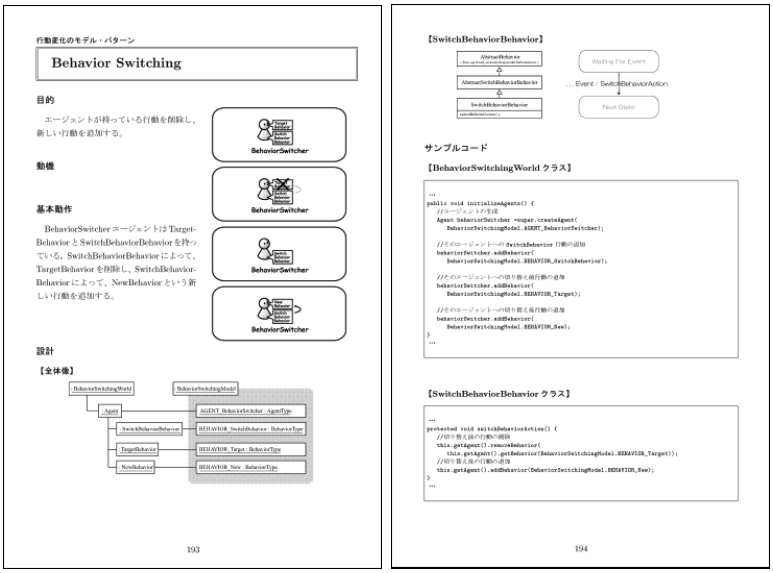

図 6.1: カタログ形式で記述されたモデル・パターンの例

(1999); Brown et al. (2000) で広く知られるようになった。Brown et al. (1998) は、ソフトウェアプロジェクトの失敗の原因となることを解説しており、Brown et al. (1999) は、「ソフトウェア構成管理（SCM）」のアンチパターンについて解説している(60)。

6.3 提案モデル・パターン

ここでは、モデル要素をどのように組み合わせて、動的な振舞いを構成すればよいのかということを、「モデル・パターン」(model patterns) としてまとめることにしたい。提案するモデル・パターンは、Boxed Economy Foundation Model に基づいてモデルを作成する際に頻繁に遭遇する問題に対して、どのようにモデル化すればよいのかを提示するものである。建築分野やソフトウェア開発におけるパターンと同様、モデル・パターンも、巧みな設計の再利用を可能とし、モデルの変化についての共通の語彙を増やしてコミュニケーションを支援する。

モデル・パターンは大きく分けて、「エレメンタリーなモデル・パターン」、「行動変化のモデル・パターン」、「コミュニケーションのモデル・パターン」、「アクティベーションのモデル・パターン」の 4 つに分けられる。各パターンの基本動作やサンプルコードなどの詳細は、本論文の付録 C のモデル・パターン カタログにまとめてある (図 6.1)。これらのパターンを分類すると、表 6.1 および図 6.2 のようになる。

6.3.1 エレメンタリーなモデル・パターン

第一のエレメンタリーなモデル・パターンは、モデル・パターンのうち、「エレメンタリー・パターン」(Wallingford, 1998) として捉えることができるものである。エレメンタリー・パターンとは、初心者に良いプログラミングを教えるための手段として用いるパターンである。エレメンタリーなモデル・パターンは、初心者が Boxed Economy Foundation Model に基づくモデルをどのように作成すればよいのかを示すものである。そのほとんどが、Behavior のアクション内で、1、2 行のプログラムで実行できる程度のものである。

6.3.2 コミュニケーションのモデル・パターン

第二のコミュニケーションのモデル・パターンは、他のエージェントとの Goods や Information のやりとりに関してのモデル化をまとめたものである。複数のエージェントに質問を投げて回収するなど、相互作用を含むモデルで頻繁に登場するパターンを集めてある。

6.3.3 行動変化のモデル・パターン

第三の行動変化のモデル・パターンは、行動の生成・削除・組み換え等のモデル化をまとめたものである。これらの行動変化は、すでに本論文の最初の方で述べたように、複雑系の記述において不可欠である。

6.3.4 アクティベーションのモデル・パターン

第四のアクティベーションのモデル・パターンは、エージェントの活性化に関するモデル化をまとめたものである。ある時間ステップにおいて、一部のエージェントにだけ行動を起こさせたい場合などのパターンがある。

6.4 発展のための覚書

本章では、Boxed Economy Foundation Model に基づくモデル化のためのモデル・パターンについて述べてきた。今後、これらのパターンは経験的に検証されて修正されるべきである。なぜなら、「パタンは単なる仮説にしかすぎない」(Alexander, 1979, 邦訳 p.218) からである。また、新しいパターンを追加していくことも重要である。これらのモデル・パターンを組み込んだモデル作成支援ツールを開発することも考えられるが、これは今後の課題である。

表 6.1: 本論文で提案するモデル・パターンの一覧

モデル・パターンの分類	モデル・パターン名
エレメンタリーなモデル・パターン	Agent Creation Relation Creation Related Agent Creation Agent Destruction Goods Creation Information Creation
コミュニケーションのモデル・パターン	Information Sending Blank Information Sending Internal Information Sending Immediate Reply Collect Immediate Replies Appointed Destination Reply Super BehaviorType Calling
行動変化のモデル・パターン	Behavior Creation Behavior Destruction Behavior Switching Temporary Behavior Attachment Requested Behavior Attachment Forced Behavior Attachment
アクティベーションのモデル・パターン	TimeEvent Distributer Agent TimeEvent Filtering TimeEvent Distributer Behavior Time-Consuming Behavior

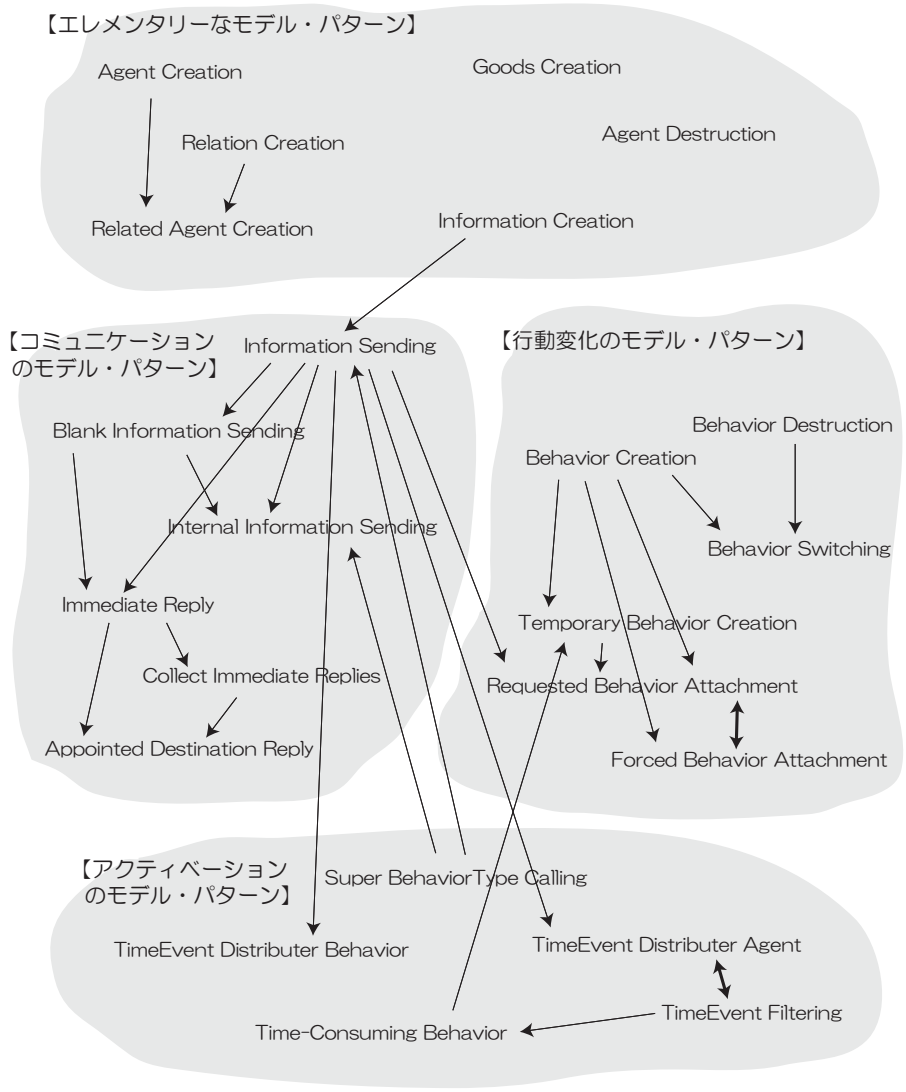

図 6.2: パターン間の関連