# 明日出会うかもしれないもの

Apr. 27, 2000 友野 晶夫

人は、自らが獲得した事実だけでなく、その周辺について推測されることも知識として使っている。こうした知識は確実なものではないかもしれないが、人が環境に適合する上で有効である。たとえば、他者と対話する際にも、相手の価値観を「推測」することで、対話を円滑に行うことができる。必ずしもそうした「推測」の結果は正しくないかもしれないが、それが「調整」のきっかけとなり、やはり対話を円滑に進めることに寄与する。人とシステムの対話においても、このことは同様である。こうした知識領域を**「明日出会うかもしれないもの」**と名づけた。

本パターンでは、**「明日出会うかもしれないもの」**の形態について述べる。また、こうした知識形態を UML[1]による型モデル上でどのように表現するかについても述べる。

## 第1章　推測

まず、**「明日出会うかもしれないもの」**は、単純な推測により生み出される。本章では、これが、事実とどのように関与しあい、その形態を変化させるかについて述べる。本章では、次のパターンについて述べる。

１．　高価な事実
２．　事実との乖離
３．　推測の修正

### １．高価な事実

**文脈：**

ある倉庫には、特定の品物 A が入出庫されている。倉庫内の品物 A の数量を調べたいのだが、倉庫内には膨大な量の品物 A があり、これらを数え挙げるのには多大な時間がかかる。入出庫は、通常、頻繁に行われるため、日常的にこの方法を適用して倉庫内の品物 A の数量を把握することはできない。

**問題：**

ある関心事を把握したいが、そのためにかかるコストが高い。

**制約：**

- 関心事は観測可能なものであり、この値を大きく誤ると問題を起こしうる。

**解決：**

より把握しやすい事実から関心事を推測する推論ルールを用意して、関心事を推測できるようにする。

**文脈における解決例：**

倉庫への入出庫はすべて記録が残っている。そこで、知りたいと思う時刻までに起きた入庫の合計から出庫の合計を引いたものとしてその数量を計算することができる。

**型モデルへの表記：**

通常、こうした推測される事柄には、導出物であることが型モデル上に記されることになっている。ただし、導出物であるという指定は、単に冗長性を回避するという意味しか表していない。本パターンに当てはまる事柄には、それ以上の意味がある。特に、これに連なる「事実との乖離」や「推測の修正」から、いくつかのユースケースが必要となることが暗示されるため、推測であることを明示した方が有用である。そこで、このパターンに当てはまるモデル構成要素には、ステレオタイプまたはロール制約として「推測」と付記する。導出の場合と同様に、推論ルールを制約として記しておくこと。

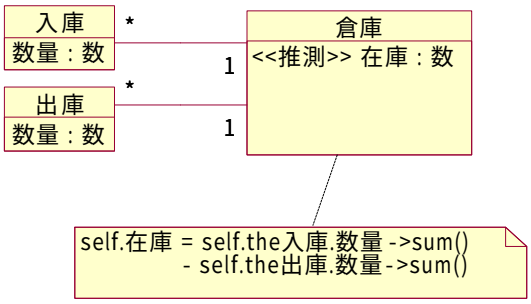


図 1 ステレオタイプ「推測」の利用例

## ２．事実との乖離

**文脈：**

ある倉庫内の品物 A の数量を直接数え上げる代わりに、知りたいと思う時刻までに起きた入庫の合計から出庫の合計を引いたものとしてその数量を計算することができる。ただし、この方法で求めた数量が実際の数量と合致するとは限らない。倉庫内での破損や入出

庫の記録ミスのように、把握しきれない事実があるためである。

**問題：**

関心事を推測するための推論ルールはあるが、把握されている事実や推論ルールが不完全なため、推測値が実際の値と乖離する。

**制約：**

- 実際の値と推測値との乖離は、常に問題となるわけではなく、ある一定の条件を満たしたときだけである。また、その条件は明文化できる。

**解決：**

推測値と事実との乖離が問題とならないように、余裕代（しろ）を設ける。こうすることで、ある程度の期間であれば、推測値と事実の乖離は余裕代の範囲内に収まると考えられる。その上で、推測値を正しいものとして扱う。

**文脈における解決例：**

この倉庫では、出庫時に在庫量が不足すると問題となる。また、出庫量はある程度予想ができる。そこで、予想出庫量と見込み損失量からなる、安全在庫量を設定し、常に安全在庫量を下回らないように品物 A の仕入れ注文を出すようにしている。

## ３．推測の修正

**文脈：**

倉庫内での破損や入出庫の記録ミスが累積していくため、そのままでは次第に推測値は実際の数量から乖離していく。倉庫では、月に１度程度の頻度で棚卸を行うため、その都度、実際の数量を把握することができる。

**問題：**

事実が把握されたとき、推測との乖離が明らかになる。これを放置すれば、この乖離は広がる一方である。

**制約：**

- ある程度の低い頻度であれば、関心事の実態を把握するのにかかるコストも許容される。

**解決：**

関心事の実態が把握される都度、それを事実として把握し、その後の推測に反映させる。

**文脈における解決例：**

棚卸により実際の数量が把握できたら、その時点での推測値との差を、その時点で入出庫したものとする。多くの場合、このときの入出庫先は、「損失」や「天使の取り分」[1]と呼ばれる。

## 第2章　事実との融合

**「明日出会うかもしれないもの」は、通常、事実と明確に区別されない。これにより、概念操作が容易となるが、一方で、そのことを表現する型モデル上に事実と推測の入り混じる構成要素が現れるため、そのままでは仕様モデルとして不完全なものとなる。そうした型モデルの構成要素に事実と推測の混在状況を明示できればこの問題は緩和される。これについて次のパターンで述べる。**

**４．事実の受容**
**５．存在の顕現**

### ４．事実の受容

**文脈：**

ある部品管理システムにおいて、製造物の属性（色、サイズ等）によって、その構成部品の仕様を推測している。しかし、生産管理者の視点から、このシステムが推測した構成部品に対して修正を加えたいことがある。生産管理者は、製造物の構成部品について責任を負っているため、生産管理者によって指定されることがらは、製品の構成について「事実」にあたることである。

**問題：**

事実が把握されたとき、推測との乖離が明らかになる。これを放置すれば、この乖離は広がる一方である。

---

[1] モルトウィスキーの熟成中に蒸発した分。英語では、angel's share。

**制約：**

- ある程度の低い頻度であれば、関心事の実態を把握するのにかかるコストも許容される。
- 把握された事実を、**「推測の修正」**で示した文脈とその解決のように、推測の元となる事実に展開できない。

**解決：**

推測に例外を与える。事実が与えられているものについては、事実を答え、それ以外のものについては、推測値を答えるものとする。

**文脈における解決：**

生産管理者が指定した構成部品については、事実として記録しておく。また、指定により変更されたものについては、無効であることも記録しておく。その後、該当する製造物の構成部品を問い合わせられた際には、指定されたものについては、記録してあることを答え、それ以外については、推測した結果を答える。

**型モデルへの表記：**

このパターンに当てはまる型モデル上の構成要素に、ステレオタイプまたはロール制約で「事実を容認し得る推測」と記す。「推測」と同様に、推測時に当てはまる推論ルールを述べておくこと。

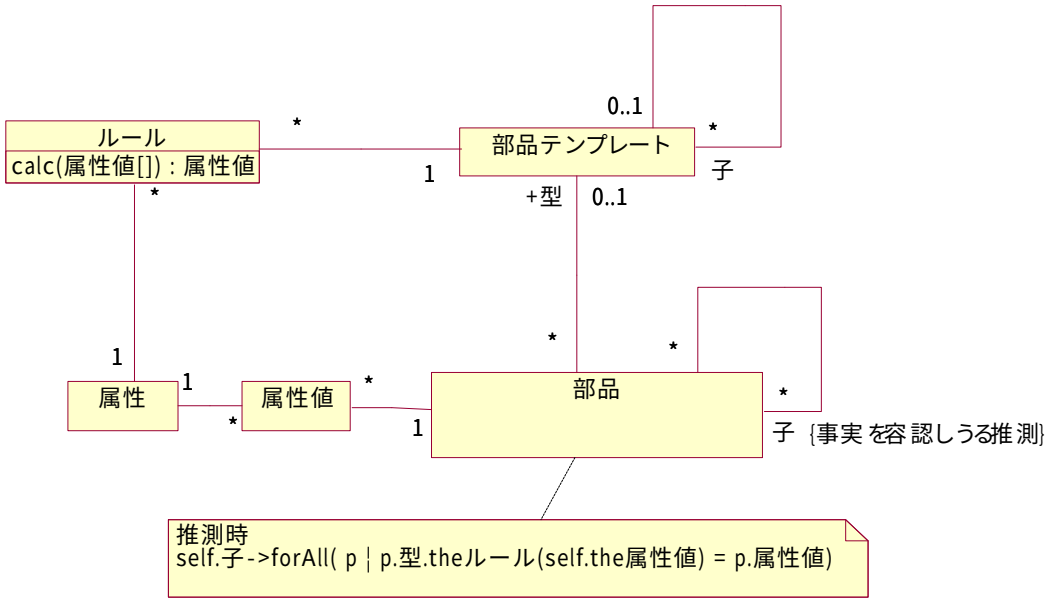


図 2 ステレオタイプ「事実を容認し得る推測」の利用例

## ５．存在の顕現

**文脈：**

**「事実の受容」**の文脈における製造物そのものが**「高価な事実」**であり、推測されるものとして表現されている（一つひとつの製造物だけを考えれば、それが**「高価な事実」**であるとは考えにくいかもしれない。しかし、通常、部品管理の文脈では大量の製品を扱うことになるため、これも**「高価な事実」**となるのである）。推測された製造物を、他の製造物の部品として生産管理者が指定する場合がある。この場合、生産管理者が推測により生み出された製造物を「事実」として認定したため、この製造物は、これ以後、事実として扱われるものとみなせる。

**問題：**

推測により導き出されたことが事実として認定されることがある。

**制約：**

- 推測は、それ以降の状況により変動し得る。
- 推測値は、推測が行われる都度、導出されるため、それを永続的に指示することができない。
- 認定された事実を、**「推測の修正」**で示した文脈とその解決のように、推測の元となる事実に展開できない。

**解決：**

推測により得られた知識を事実として捉えなおす。**「事実の受容」**と同様に、推測に例外を与える。事実が与えられているものについては、事実を答え、それ以外のものについては、推測値を答えるものとする。

**型モデルへの表記：**

このパターンに当てはまる型モデル上の構成要素に、ステレオタイプまたはロール制約で「事実と認定され得る推測」と記す。「推測」と同様に、推測時に当てはまる推論ルールを述べておくこと。

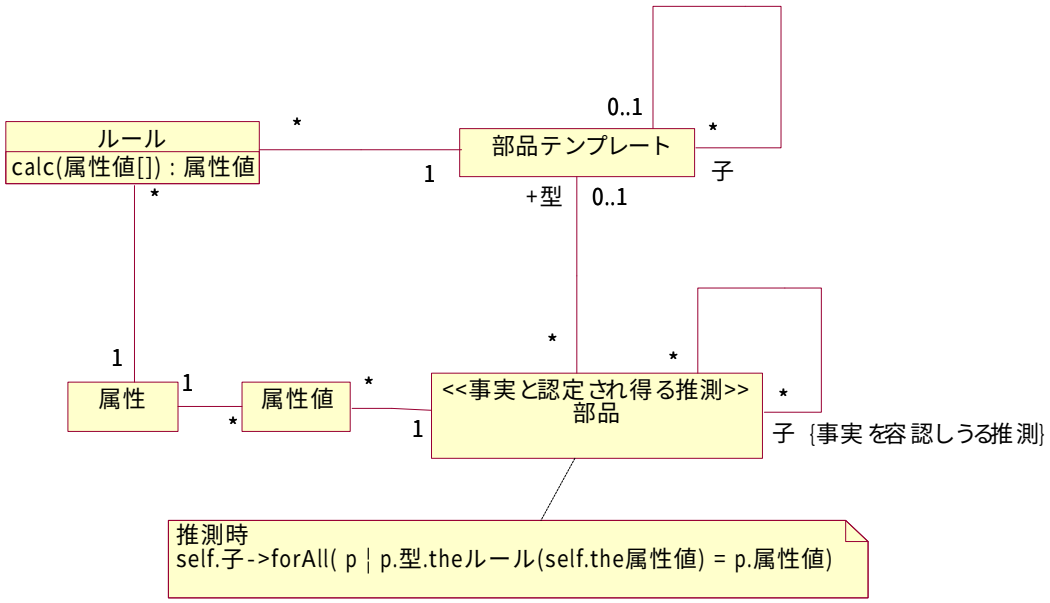


図 3 ステレオタイプ「事実と認定され得る推測」の利用例

## ６．存在の顕現の自動化

**文脈：**

ある製造物の推測された構成部品の属性を生産管理者が修正することがある。その場合、修正された結果の構成部品は、生産管理者により指定されたものと同義であるため、「事実」として認定されたものとみなせる。

**問題：**

利用者は、「事実」としての認定を必ずしも意識的には行っていない。

**制約：**

- 利用者にとって、推測と事実との区別は関心外である。
- より少ない指示により型モデルを仕様モデルとして完備できることが望ましい。

**解決：**

「存在の顕現」に当てはまるオブジェクトに、「事実を受容」しうる構成要素が付随しており、その構成要素が「事実を受容」した場合、自動的にそのオブジェクトは事実となるものとする。

## 第3章　型モデルの変化

第 1 章および第 2 章では、型モデルの変化について触れなかった。実際には、型モデルやそこに記される推測の基となる推論ルールも、**「高価な事実」**の 1 インスタンスであると考えられる（実際に、図 2 のように、推論ルールは型モデル上の型のインスタンスとして与えられる場合もある！）。本章では、そのレベルにおいてなされている努力について紹介しておく。

まず、本パターン言語でも示されるように、型モデルの構成要素についてのメタモデルだけでは、規定されない事柄がそこには残っている。そもそも、人間の思考にとって、何をして「存在」するとするかが曖昧なためである。それでも、一般的な型モデルにおける「存在」の規範は、プログラム言語機構によって与えられる「存在」の規範と同様のものとして互いに了解されているというのが大方の現状であろう。実は、本パターン言語を編み出すに至ったプロジェクトにおいても、はじめは、その視点で十分であろうと考えていたものが、結果として本パターン言語で述べたようなパターンにより、型モデルの構成要素についての定義を精緻化することが必要であると徐々に理解されてきた。

現在、モデル構成要素の精緻化に対して積極的にアプローチする方法が、AOP[2]や**アタッチャオブジェクト**[3]として発表されている。こうしたアプローチは、構築された型モデルの立脚する「世界観の変容」を促すものと捉えることができる。

また、世界は、単一の世界観で閉じていない。時間軸または地域軸（正確には、コミュニティや個といった、微細な軸の構成要素に分かれる）に沿って現れるさまざまな世界観が対話を十全に行えることが要求される。そのために「n-世界観の調停」に取り組まなければならない。これについて言及しているパターンとして、**フェデレーションパターン**[4]、および Data Knows Its Root[5]がある。

## 第4章　まとめ

本パターン言語の根底において「事実」をどう捉えているかについて述べておく。「事実」は、原初的には「個」の関心の基に現れる。それにより問題が起こらない限りにおいて、「個」が把握した「事実」をそのままにしておいてもよい。

これは、各「個」にとっての「事実」が多様になり得ることを示している。複数の「個」が社会を形成する上で、このことは問題となる。そこで、互いの「事実」について合意が形成される。こうして「同意(Accommodate)」された「事実」がそのコミュニティにおける「事実」となる。これが**「推測の修正」**および**「事実の受容」**で述べたことである。通常、システムと利用者の間では、**「事実の受容」**がシステムから利用者への一方向に「事実が受容」されると見なされやすい。ユーザによるシステムの神聖視は、この副作用としておこる。**「推測の修正」**および**「事実の受容」**をシステムに導入し、利用者からシステ

ムへの方向にも「事実が受容」され得るようにすることで、システム－人の関係を、人－人の関係に近づけることができる。

また、「個」にとっての「事実」は、「個」にとってもっとも経済的な方法で把握されうる形態の「事実」である。これを利用するための方法を述べたのが**「事実との乖離」**である。「同意」される「事実」は、「個」にとっての「事実」と比較して、その把握にかかる経済性が損なわれることに注目すると、ある「事実」について「同意」する「個」の数が増えるほどその把握が高価になることがわかる。これが「事実」が「高価」になる原因である。言い換えると、「客観性」に払うコストが必要になるのである。ただし、「客観」と見なされる「事実」も時とともに変化する。つまり、絶対的なものではない。常にそのためのコストを払いつづけることは、「対価」に見合わない。そこで、本パターン言語では、「対価に見合うだけの客観性」を要領よく取り入れるようにすることを大きなテーマとして捉えている。

私は、本パターン言語を、Peirce の次の言葉で締めくくりたい。

「かくしてわれわれの知識はその範囲が限られているだけでなく、さらに一層重要なことは、われわれが人間として知り得る限りの最善の知識でもわれわれはそれを不明確な不確実な仕方でしか知ることができないということを徹底的に知ることである。」[6]

## 参考文献


[1] UML specs, http://www.omg.org

[2] Kickzales, G., et al., “Aspect-Oriented Programming”, In *proceedings of the European Conference on Object-Oriented Programming* (ECOOP), Finland. Springer-Verlag LNCS 1241. 1997

[3] 井上 健, 「アタッチャオブジェクト」, http://www.kame-net.com/jplop/

[4] 藤野 晃延, 「フェデレーションパターン」, http://www.kame-net.com/jplop

[5] Kerth, N., “Caterpillars’ Fate”, In Coplien, J. O., Schmidt, D. C., Pattern Language of Program Design, Addison-Wesley, Reading, MA, 1994.

[6] 米盛 裕二, 「パースの記号学」, 勁草書房, 1981