

クラス構造の進化モデルと統計的進化パラメータの分析

大木 幹雄† 秋山 正二郎†

† 日本工業大学 工学部情報工学科

E-mail: ohki@nit.ac.jp

あらまし

筆者らは概念モデリングに関する判断基準の形式化を試み、実際に初心者にも理解しやすく、かつ根拠が明確な3つの判断基準を導くことができた。そこで用いた発想を発展させ、クラス構造とデザインパターンの生成メカニズムについて考察し、メカニズムをモデル化した結果を報告する。同時に、Java クラスライブラリをもとに、モデル化の正当性を検証するために行ったのクラス構造分析の途中経過についても報告する。モデルがもつクラス構造の生成メカニズムは、クラス階層の進化と共通するものであり、分析を行ったクラスライブラリの範囲内では、サブクラス化による属性やメソッドの進化形態は、自己相関的な振る舞いをするロジスティック写像で近似できることが判明した。

キーワード 概念モデル 判断基準 進化モデル 場の理論 クラス構造 デザインパターン

A Consideration of Class Structure Evolutional Mode and a analysis of statistical evolution parameters

Mikio OHKI Shojiro AKIYAMA

Nippon Institute of Technology

Abstract

We have tried to formalize the criteria for conceptual modeling and have derived three criteria. From such a experience, we next considered the generating mechanism for the design patterns. This Paper reports the result of these considerations. Also, this Paper reports the results of Class Structure Analysis for verifying the propriety of our modeling approach that were done by analysis the Java Class library. Because of the Class structure generating mechanism and the evolution of class hierarchy have same properties, it became clear that the evolution form of attributes and the methods are approximately represented by Logistic projection acting self-correlated manner.

Keyword conceptual model, classification criteria, field theory, class structure, design pattern

1. はじめに

最近、システム分析設計をオブジェクト指向で行うとき、デザインパターンを積極的に利用するようになってきている。しかし実際にデザインパターンを活用するときとなると、目的や動機等からどのようなデザインパターンが適用可能か、あるいは適用すべきかを判断しなければならない問題が伴う。適用対象のシステムが単純であるときには、このような問題が生じることは少ないが、複雑化し、いくつかのパターンを組み合わせる必要があるときには、対象システムの仕様と特性を注意深く分析して、パターンの組み合わせやカスタマイズ部分を決定してゆかなければならない。これらの問題は、デザインパターンのみならず、何らかの定石を活用する際に必ず生じる問題であるが、定石を活用すべきかの判断や適用手順等の決定は、経験から修得すべきものとされている。

しかしながらデザインパターンに関して、かつてのプログラム論理パターンの再利用と同様に、適用の判断は「経験に依存する」としたら、経験的学問としてのソフトウェア工学の真価が問われかねないことになる。ソフトウェア教育に携わる者は、このような現状でも、経験に換わる客観的で理解しやすい判断基準を初心者提供してゆかなければならない。

筆者らは、業務データの分析から概念モデリングに関する判断基準の形式化を試み、実際に、理解しやすく、かつ根拠が明確な3つの判断基準を導くことができた[1]。そこで次の段階として、概念モデリングの分類基準の導出に用いた手法、すなわち「属性を基本的な構成要素として、概念のもつ基本特性から属性を概念に分類する基準を導出し、機械的に概念モデルを形成させる」手法がデザインパターンにも適用可能かどうかの考察を行った。その結果、デザインパターンを『クラスの構成要素である属性、メソッドおよびクラス間に一定の“力”の場の存在を仮定し、場の相互作用によって構成要素の安定的な集団（すなわちクラス）が配置・形成されたもの』と考え、クラスおよびデザインパターンの生成過程をある程度説明できることがわかった。

デザインパターンの生成過程が簡単なルールで再現できれば、基本的なデータの収集や機能仕様の洗出しが終わった後に、ルールを機械的に適用してデザイ

ンパターンと等価なクラス構造を導けることから、経験に依存すること部分が少なくなる。

本稿では、このような発想のもとに、クラス構造とデザインパターンの生成メカニズムについて考察し、モデル化した結果を報告する。同時に、Java クラスライブラリをもとに、モデル化の正当性を検証するために行ったクラス構造分析の途中経過についても報告する。これらのモデルは必然的にクラス階層の進化を説明する要因を含んでおり、Delphi クラスライブラリによる妥当性の検証も進行中であることから、本稿に「統計的進化パラメータの分析」の標題を付している。

2. 概念モデル抽出の判断基準

デザインパターン生成過程のモデル化について述べる前に、本研究の基礎となった概念モデル抽出の分類基準について概要を示しておきたい。

2.1 概念と属性の関係

概念モデリングの第1段階は、どのような概念（以後、クラスの原型を意味する）が存在するかを抽出することにある。経験の少ない分析者が、何の手掛かりもなく概念を思い浮かべるのは一般に困難であり、何らかの手掛かりが必要になる。

概念はそれがもつ属性と表裏一体をなしており、どの属性に着目するかによって、抽出する概念も変化することが多い（例えば、同じ“雨”でも粒度に着目すれば“霧”であり、季節に着目すれば“さみだれ”になる）ことから、手掛かりを属性に求めると概念抽出が行いやすくなる。

2.2 基本データの分類基準

データ中心分析法では、基本データ（他のデータから導出できないデータ）の集合を分類し、概念の属性として格納する手順がとられる。この方法を発展させ、次のような分類基準をもとに、基本データをボトムアップに分類してゆくと概念モデルの導出が容易になる（分類をもとにした類似のボトムアップアプローチとして、クラスがもつサービスの視点から CRC カードを用いて体系化する責任駆動アプローチ[2]があるが、責任といったあいまいな分類基準しかない）。

【分類基準 C_1 】初期値決定時点の同時性

初期値が決定される時点 t_0 が同一の基本データ d_i は、同じ概念に属する属性とする。

$$C_1 \equiv \{ d_i \mid (\exists t_0) (\exists s) d_i \in \Delta \wedge t = f^{-1}(d_i, s) \wedge t = t_0 \} \quad \text{----- (式1)}$$

【分類基準 C_2 】実現値の構造型

多値や選択的に実現値が決まる基本データは、単値基本データと混在する形で概念の属性となりえない。別の概念の属性として分離しなければならない。なお、 $\# \xi(d)$ は、基本データ d の実現値集合要素数を意味する。

$$C_2 \equiv \{ d \mid (\exists n) d \in C_1 \wedge \# \xi(d) = n \} \quad \text{----- (式2)}$$

【分類基準 C_3 】初期値の決定状況単一性

初期値の決定時点が複数の状況に依存する基本データは、状況によって一意的に決まる初期値の決定時点をもった概念の上位概念に属する属性である。このとき、上位概念に置くとき属性名は、一致しなければならない。なお、 $\# s(d)$ は、基本データ d の状況数を意味する。

$$C_3 \equiv \{ d \mid (\exists m) d \in C_2 \wedge \# s(d) = m \} \quad \text{----- (式3)}$$

このような分類基準を用い、図1で示す基本データ集合の分類を機械的に進めると、図2のような概念モデルが抽出される(詳しくは[1])。

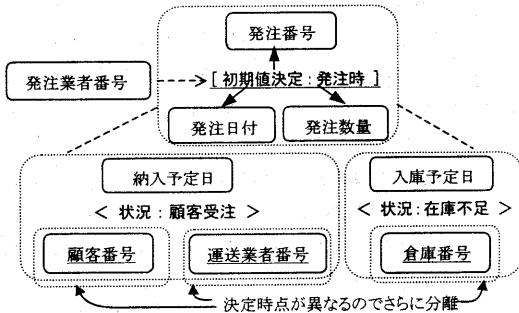


図1 分類基準3.1を用いた基本データ集合 ϕ_4 , ϕ_5 の分類

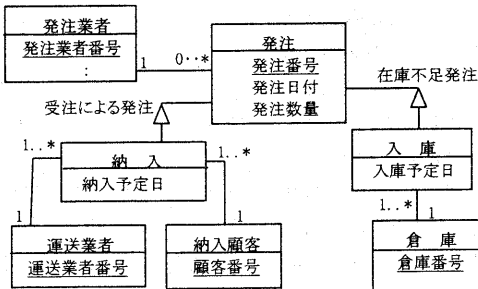


図2 基本データ集合 ϕ_4 , ϕ_5 の分類に対応する概念モデル

3. デザインパターン生成過程のモデル化

デザインパターン生成過程のモデル化と判断基準を検討するために、以下に属性、メソッド、クラスを場の概念を用いてモデル化する。

3.1 オブジェクト構造のモデル化

属性が変化すると新しいメソッドが生じ、メソッドが変化すると新しい属性が生じる。このような側面は、属性とメソッドを独立の構成要素とする何らかの場 $\phi(x, \tau)$ が存在して、それぞれが相互に影響を及ぼしあう系がもつダイナミクスとしてモデル化できる。ここでいう場はいわば不確定な要求や分析設計の対象となる“もやもや”状態を表現したものと考えてもよい。 x はオブジェクトが配置された抽象的な位相空間軸、 τ は事象の発生前後の関係を表す位相時間軸を意味し、独立した軸であり、 ϕ , x , τ は共に離散的な値をとるものとする。

このとき、属性 $i(\phi_a)$ 、メソッド $j(\phi_\mu)$ 、クラス (ϕ_a, ϕ_μ) を次のように捉える。

① 属性 $i(\phi_a)$ ($i=1 \sim k$):

$\partial \phi(x, \tau) / \partial \tau = \phi_a = \text{Constant}$ の曲線上の場合で同じ $\tau = \tau_0$ をもつものを量子化したもの。すなわち、任意の $\phi_a (= \text{Constant})$ に対して $\partial \phi / \partial \tau = \phi_a$ で $\tau = \tau_0$ をもつ離散点集合を数え上げたもの(図3参照)。

② メソッド $j(\phi_\mu)$ ($j=1 \sim m$):

$\partial \phi(x, \tau) / \partial x = \phi_\mu = \text{Constant}$ で場 ϕ を量子化したもの。すなわち、任意の $\phi_\mu (= \text{Constant})$ に対して $\partial \phi / \partial \tau = \phi_\mu$ で $x = x_0$ の離散点集合を数え上げたもの(図3参照)。

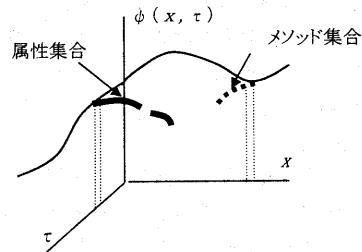


図3 場をもとにしたオブジェクト構成要素の抽出

③ クラス $\Psi(\phi_a, \phi_\mu) = \{\text{属性 } i(\phi_a) (i=1 \sim k); \text{メソッド } j(\phi_\mu) (j=1 \sim m)\}$ 。すな

わち、①②で離散点集合を各成分とする離散点。

④ インスタンス $|\Psi|$

クラスは、多くの状態をもつ位相空間内の場の1状態（通常状態ベクトルとして表現する）である。その状態ベクトルが張る空間（図4参照）には、属性やメソッドの実現値を組とする点が多数存在する（メソッドの実現値とは、メソッドの実装を意味するものとする）。この実現値化した組をインスタンスとして捉え、形式的に次のように表す。

・実現値化 $\equiv$ インスタンス生成 $=$

$$\text{クラスの絶対値}|\Psi|$$

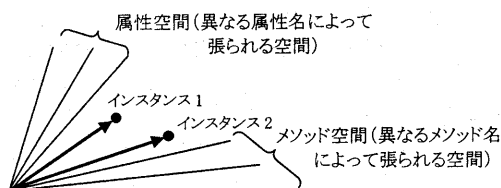


図4 属性-メソッド空間に生成されるインスタンス

⑤ $\phi(x, \tau)$ の x 軸成分、 τ 軸成分の変化：

場は相互に他軸の変化を誘発するものとする。これは場が相互に影響しあひながら位相空間を伝播する（進化する）ことを意味する。

3.2 構成要素のモデル化

任意の定数 ϕ_a, ϕ_b によって量子化された属性 i やメソッド j を“構成子”と呼ぶことにし、それぞれ共通する次のような特性をもつものとする。

① v ：状況レベル

構成子をもつエネルギーレベルに相当する。以下に述べる多重度、存在寿命とは独立した特性である。同じ存在寿命、同じ多重度をもつ構成子でも、異なる状況レベルをもちうる。

② π ：多重度

1かそれ以外の値をとり、同じ状況レベルでは、属性としての構成子は1以外の多重度をとるものは存在できない。メソッドとしての構成子の場合、同じ状況レベルに1以上の多重度をとるものが存在できる。

③ λ ：存在寿命

属性あるいはメソッドが、場 $\phi(x, \tau)$ の中で同じ状態を保つ時間を示すものである。属性をもつ関

連や多対多関連で用いるが、本稿ではこの特性については触れない。

3.3 概念モデル抽出の分類基準の再解釈

前節まで示した場と構成子の特性を用いて「2. 概念モデル抽出の分類基準」を再解釈してみよう。

(1)属性分離基準1：構成子としての属性 $i(\phi_a)$ ($i=1\sim k$)は時間変化が定数で同じ時間 τ_0 を満たすものを集めたことから自動的に導かれる。

(2)属性分離基準2：多重度 π の定義から、同じ状況レベルで1以外の多重度をもつ属性 i は存在できないことから、別の概念として分離するとすれば、おのずと導かれる。分離しても、属性 i を量子化させた操作とは無関係なので、分離後の概念は、分離のもとになった概念に存在従属している。

(3)属性分離基準3：同じ時間変化、同じ時間 τ_0 の属性で、かつ1以外の多重度でも、異なる状況レベルであれば属性 i の存在が許される。これらを別の概念として分離するとすれば、この基準はおのずと導かれる。状況レベルが異なるために分離後の概念と分離のもとになった概念は同じ属性 i を共有する、すなわち継承する。

以上から概念モデル抽出の分類基準を、場と構成子の特性から整理できたことになる。このほかにも、この考え方を用いて、クラスの階層構造がもつ次の特徴を再解釈できることになる。

(a)メソッドの継承構造形成操作 α ：パラメータ部分を含め、全く同一のメソッド名をもつメソッドが同じクラスに n 個存在とき、メソッドの多重度を n と定義する。すると、構成子としてメソッドの多重度が1以上のメソッドは、同じクラスに存在できないため、異なる状況レベルをもつ別クラスに分離し、構成することになる。

(b)メソッドの多相定義操作 β ：状況レベルが異なれば、1以外の多重度のメソッドも許される。すなわち、状況レベルの異なるクラスではメソッドを再定義できることになる。

ただし、メソッドの場合、 α, β の操作は非可換である。すなわち、 $\alpha\beta \neq \beta\alpha \neq 0$ であり、操作を交換すると構造は一般に等しくならない。また、継承に関する属性とメソッドの相違（すなわち、属性は、クラス階層上で1度しか定義できないが、メソッドは幾度

も再定義できる) についても, 多重度 π の定義から図 5 で示すように考えれば説明できる.

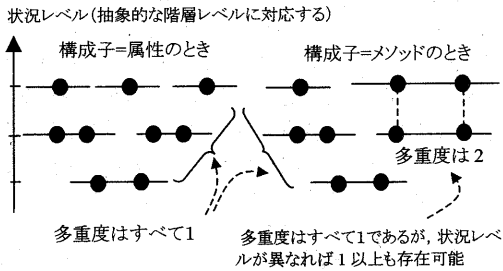


図5 属性とメソッドの継承に関する制約の相違の説明

同様にして責任駆動アプローチやユースケースアプローチによるクラス構造の抽出基準を再解釈することが考えられるが, 本稿では割愛する.

場と構成子の特性のほか, 以下の力場の概念を導入すると, デザインパターン生成過程をモデル化できる (以後, 力場モデルと呼ぶ. 図 6 参照).

- ① 構成子を1つに結合する力の場としての ϕ_α , ϕ_μ
- ② ϕ_α , ϕ_μ を1つのクラスとしてまとめる交換力 ε
- ③ 結合する力の場に逆らって継承構造を作る進化の源泉である力 ξ
- ④ クラスを結びつける継承構造以外の力 ρ

両者を併合したものをクラス $\Psi_1 (= \phi_\alpha^1 + \phi_\mu^1)$ とする

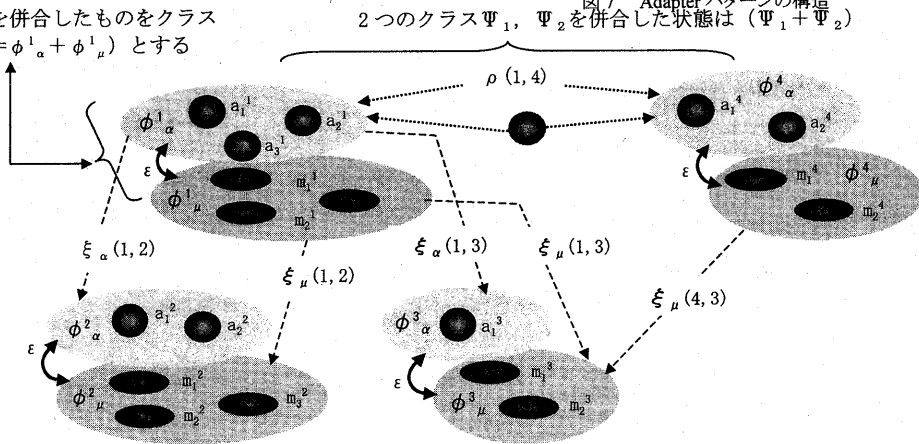


図6 量子化された ϕ_α , ϕ_μ と力を導入したクラス構造の力場モデル

3.4 力場によるデザインパターン生成のモデル化

図 6 で示した力場に関する簡単な原理から, デザインパターンの生成過程を導出するにはどのように考えるべきであろうか. 本稿では, まず構造を決定する次のデザインパターンに限定して, その導出方法を考える.

- ① Adapter : オブジェクトへのインタフェース
- ② Bridge : オブジェクトの実装
- ③ Composite : オブジェクトの構造と構成
- ④ Decorator : サブクラス化を伴わないオブジェクトの責任

(1) Adapter

属性やメソッドと同様に, クラスにも状況レベルに類似した状態レベルを考える. すると, Adapter は, 図 7 に示すように, 2つの安定した状態にあるクラスを結合するインタフェースクラスを導入することを意味する.

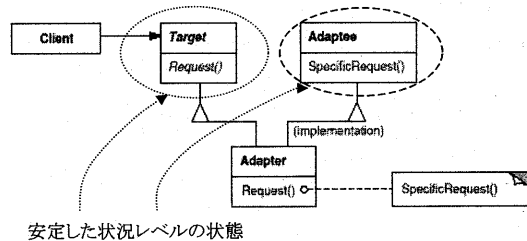


図7 Adapter パターンの構造
2つのクラス Ψ_1 , Ψ_2 を併合した状態は $(\Psi_1 + \Psi_2)$

これは図6で示した力場モデルの考え方では、2つの安定したクラスから構成される系の相互作用（系の積で表現する）の表現形式の問題になる。系の実現値は式4のように表せば、インタフェースを担うクラスは、クラス同士を結びつける交換力を実現化したクラスを意味することとなる。ただし、クラス自身の二乗はクラスを表し、実現化数2は意味をもたない。

$$\begin{aligned} & \{ \Psi(\text{Target}) + \Psi(\text{Adaptee}) \}^2 \\ &= | \Psi(\text{Target}) |^2 + | \Psi(\text{Adaptee}) |^2 + \\ & \quad 2 | \Psi(\text{Target}) \Psi(\text{Adaptee}) | \quad \text{----- (式4)} \end{aligned}$$

↑
インタフェースクラス

(2) Bridge

オブジェクトがもつメソッドの抽象的な定義と実装の分離を目的とした Bridge は、場と構成子の特性から概念モデルの分類基準が抽出できたように、次のような分類基準を設定すること導出することができる。

【分類基準 m1】 メソッドの実現化が異なる時刻で行われるものは、別クラスとする。

【分類基準 m2】 実現値の多重度が1以上のメソッド(=同じメソッド名&パラメータで複数回実装されるメソッド)は、別クラスとして分離する。

【分類基準 m3】 異なる状況レベルにあるメソッドは別クラスとして分離する。

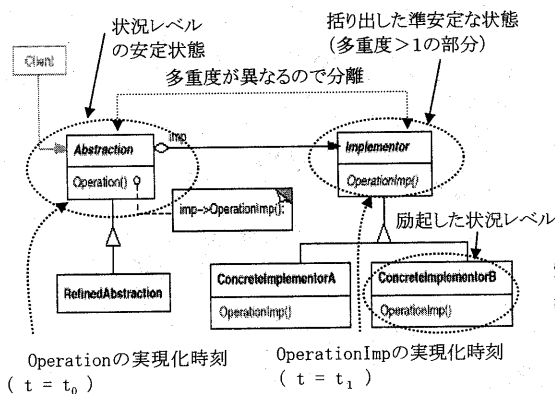


図8 Bridge パターンの構造

図8で示した Bridge パターンの構造は、まさにこのような分類基準を適用して、実現化時刻、多重度、状況レベルの異なるメソッドを分類、あるいは分離してできあがった構造と見ることできる。

(3) Composite

図9で示すような複合的なオブジェクトと構成要素の階層的な構成を再帰的なクラス構造によって表現する Composite は、“状況レベルが安定したクラスと準安定的なクラスに分離する”との考え方で導出することができる。準安定的なクラスは、状況レベルの安定状態から複雑な状況レベルに発展してゆくクラスを意味し、メソッドに関して1以上の多重度をもつことができる。

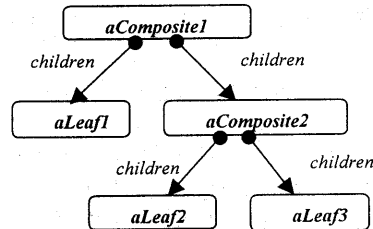


図9 Composite パターンをもつオブジェクト構造

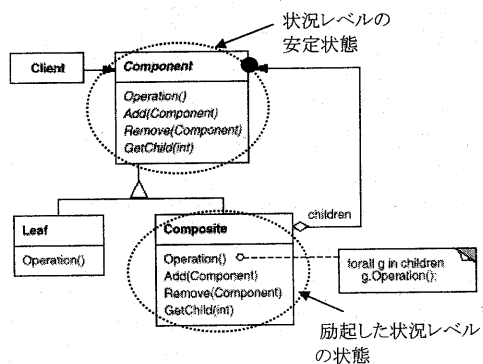


図10 Composite パターンの構造

しかし残念ながら、Composite がもつ構成関連の再帰構造は、クラスの抽出に関する3つの分類基準では導出できない（この点については、概念モデル抽出に関する判断基準でも同様）。敢えて導出するには、クラスの“役割”としての側面から見た基準を追加しなければならない。役割については、クラスの導出後に行われる関連の結合度の決定で別途、詳細に検討を行うべき作業に属するため、ここではこれ以上検討することは避ける。

(4) Decorator

図11で示す複合的な機能をもつコンポーネントを

線形的に分割し、オブジェクトの責任を階層分類化する Decorator パターンは、“メソッドの多重度が1以上のものを異なる状況レベルをもつクラスに分類すると共に、準安定の状況レベルにあるクラスを発展させる”考え方で導出することができる。構成関連の再帰構造は、Composite と同様にここでは深く検討することは避ける。

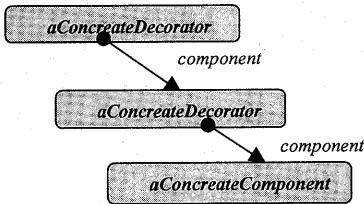


図11 Decorator パターンをもつオブジェクト構造

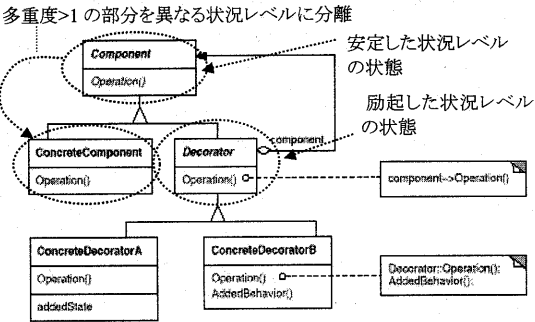


図12 Decorator パターンの構造

なお前述の3つのデザインパターンに関しては、メソッドの分類基準の適用順序がデザインパターンの導出に深く関係しており、分類基準の適用順序は図13で示すとおりになっている。m1, m2, m3 はメソッドの分類基準で、それぞれメソッド実現値化の同時時刻性、多重度の制約、状況レベル分離の基準を意味している。

(a) Bridge パターン

(m1, m2) → m3

(b) Composite パターン

(m2)*

(c) Decorator パターン

m2 → (m3)*

(注) * 記号は繰返し適用を意味する。

4. 力場モデルの検証

前節では力場モデルの一部と簡単なメソッドの分

類基準をもとに、いくつかのデザインパターンの骨格が導かれることを示した。次にデザインパターン生成過程のモデル化において前提とした力場モデルが実際に妥当であるかを検証する。さらに、継承構造を作る進化の源泉である力ξがどのような数式で近似されるかについて、Java クラスライブラリをもとに行った分析について述べる。

4.1 属性とメソッドの独立性検証

(1) 理論的な出現頻度

図6で示した力場モデルでは、属性とメソッドは互いに独立として考えている。すなわち、クラス構造が継承性に基づいて進化して行くとき、属性とクラスは相互に独立して進化するものとしている。この仮説を確認するために、ξ, λをクラスのもつ属性とメソッド、Δξ, Δλをそのサブクラスで新規に追加される属性とメソッド、“継承性を利用したサブクラス生成”をクラスがもつ合成状態φ=ξ+λからの変位φΔφとして、それぞれサブクラス生成によって属性の追加されたクラス、メソッドが追加されたクラスの出現頻度を調査した。理論的な出現頻度ρは、式5のように表される。

$$\begin{aligned} \rho(\phi \Delta \phi) &= \rho((\xi + \lambda)(\Delta \xi + \Delta \lambda)) \\ &= \rho(\xi \Delta \xi + \xi \Delta \lambda + \lambda \Delta \xi + \lambda \Delta \lambda) \quad \text{--- (式5)} \end{aligned}$$

属性ξのみを下位クラスで追加するケース(ξΔξ)の出現頻度ρ(ξΔξ)は、現実には少ないことから無視できるものとする。式5から属性を追加するクラスλΔξの出現頻度ρ(λΔξ)と、メソッドを追加するクラス(ξΔλ+λΔλ)の出現頻度ρ(ξΔλ+λΔλ)は、ほぼ1:2となる。

(2) Java クラスライブラリでの出現頻度

クラスにおける属性とメソッドの独立性の仮説を確認するため、Java クラスライブラリにおいて継承関係にある上位一下位クラス間で、属性のみが追加されたクラス、メソッドが追加されたクラスの出現頻度を計測した。871箇所の継承関係を計測した結果、各クラス出現頻度ρの割合を以下の通りになった。

$$\begin{aligned} \rho(\lambda \Delta \xi) &= 0.30999 \\ \rho(\xi \Delta \lambda + \lambda \Delta \lambda) &= 0.69001 \end{aligned}$$

ρ(λΔξ)とρ(ξΔλ+λΔλ)は、ほぼ1:2となっている。これから属性とメソッドが独立して進

化すると考えてよいことがわかる。

4.2 継承構造を構成する力の表現形式

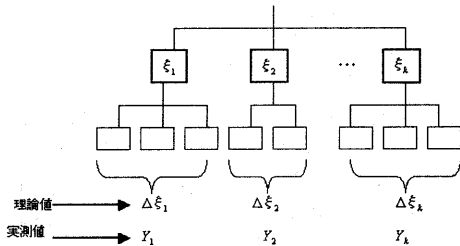
次に、属性とメソッドをクラスとして結合する力に逆らって継承構造を作る進化の源泉である力の具体的な表現形式について求めてみた。属性の継承とメソッドの継承は独立していることから、異なる力が作用すると考えるのが妥当である。属性は静的な存在であるが、メソッドは継承関係にあるクラス間で数や構成がかなりの頻度で変化する。そこでまず継承構造を形成するメソッド間で働く力の形式を求めた。

力の表現形式は、上位クラスがもつメソッド数に比例して決まると考えるのが妥当であることから、上位クラスに対して、式6で示しめすような関係が成り立つものと仮定し検証を行った。

$$\Delta \xi_i = \sum \xi_{ij} = \alpha (1 - \beta \xi_i) \xi_i \quad \text{----- (式6)}$$

ここで式6の記号は以下を意味する。

- $\Delta \xi_i$: 階層nのクラスiのサブクラスのメソッド数計
- ξ_i : 階層nのクラスiがもつメソッド数
- Y_i : 計測した階層のクラスiのサブクラスのメソッド数計



式5が成り立つためのパラメータ α , β を最小二乗法によって求めた結果、以下が判明した。ただし、調査したクラスは、特にメソッドの継承が意味をもつ ComponentUI クラスライブラリの継承木と Component クラスライブラリの継承木に限定している。また、調査対象としたクラスのメソッドの数は、分析の都合上、25 以下のものに限定している。

(1) ComponentUI クラスライブラリの分析結果

対象クラス数 $n=21$

$\alpha=1.28708264$ $\beta=0.019433742$

自由度調整済み相関係数=0.727 (1% 有意)

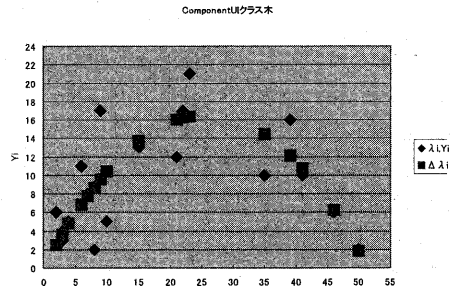


図14 ComponentUI クラスライブラリへの式6の当てはめ

(2) Component クラスライブラリの分析結果

$\alpha=1.811489653$ $\beta=0.027567953$

自由度調整済み相関係数=0.763 (1% 有意)

$n=13$

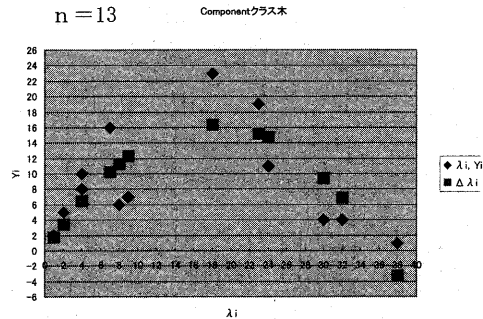


図15 Component クラスライブラリへの式6の当てはめ

5. 検討

継承関係にあるクラス間の継承力を式6のように想定したのは、上位クラスのメソッド数が下位クラスのメソッド数に影響を与えるとのカオス的な進化モデルを検証する意味も併せもっていた。式6が統計的に有意であると判明したことから、継承を利用したサブクラス化は次のような意味をもつものと考えられることができる。

『サブクラス化に伴うメソッドの総数は、上位メソッドの総数に自己相関的に(力場としての)影響を受け、その一定の誤差範囲に収まる』

6. おわりに

力場と場の考え方に基づくクラス構造の進化、デザインパターン生成過程のモデル化は、はじまったばかりの段階であり、今後、鋭意研究を進めて行きたい。

参考文献

- [1] 大木幹雄, 秋山構平, “概念モデリングにおける判断基準の提案とその有効性評価,” 電子情報通信学会論文誌 VOL. J84-D- I No. 6(2001) pp. 723-735
- [2] R. Wirfs-Brock, B. Wilkerson, “Object-Oriented Design: A Responsibility-Driven Approach,” Proc of OOPSLA' 89, ACM, pp. 71-75, 1989.