ソフトウェア工学 124－6
（1999．10．22）

ソフトウェアパターンのモデル化とパターン進化のパターン

青山 幹雄
新潟工科大学 情報電子工学科
〒945-11 柏崎市藤橋 1719
mikio@iee.niit.ac.jp

デザインパターンを嚆矢として，ソフトウェア開発の多くの局面でソフトウェアパターンの研究開発が行われている．しかし，従来の研究はパターンのカタログ化が中心であり，パターンを利用する観点から適切なパターンを探索し，組み合せる方法に関する研究は少ない．本稿は，デザインパターンを中心にパターンの体系化とその表現を提案する．さらに設計進化の観点からパターン間の関係の意味を議論する．まず，ソフトウェアパターンのファミリ化の概念を提示し，ファミリ内とファミリ間の構造のモデル化を考察した．また，設計を絞り込むためにパターン言語におけるフォースを体系化に用いる方法を議論した．さらに，設計進化の観点から，パターン間の進化を構造進化を引き起こす原子進化オペレーションで表す方法を議論した．このような方法に基づき，例を用いて，パターンを組み合せる方法を提示した．

Models of Software Patterns and the Patterns of Patterns Evolution

Mikio Aoyama
Department of Information and Electronics Engineering
Niigata Institute of Technology
1719 Fujihashi, Kashiwazaki, 945-11
mikio@iee.niit.ac.jp

This article proposes a methodology for developing a pattern system, by which we can analyze, classify, and systematize software patterns. Although numbers of pattern catalogues have been published, little known about how to choose and use appropriate patterns from such a big sea of patterns. First, this article proposes a concept of pattern family and a method to characterize the family in problem space and solution space. To squeeze the problem-solution space, we propose the use of "forces". Then, we propose a model of pattern evolution in a family as well as across the family. Atomic evolution-operations have been identified to represent the structure of evolution. Based on the techniques, we propose a pattern composition method with examples.

1. はじめに

デザインパターン[Gamm94]を嚆矢として，ソフトウェア開発の多くの局面で「ソフトウェアパターン」の研究，開発が行われている[Naka99][Schm96]．しかし，これまでの，ソフトウェアパターンに関する研究の多くはパターンのカタログ化を中心としている．ここ数年，多くのパターンが発掘，提案され，パターンの数は増大した．さらに，類似のパターンも少なくない．しかし，パターンを利用する観点から適切なパターンを探索し，組み合せる方法などに関する研究は少ない[Aoya99a]．ソフトウェア開発の現場でパターンを活用するためには，パターンを利用した開発方法論の研究開発が必要である．このようなパターンを活用する開発方法はパターン指向開発(Pattern-Oriented Development) [Yama98, Yosh99]と呼ばれる．

パターンは孤立してはありえないと言われるように，パターンを活用するためにはパターン群を分析，体系化し，パターン間やパターンとそれ以外の部分とのインタフェースや関係を明らかにする方法が必要である．パターンを体系化する一つの試みとして，Buschmannらはパターンを整理体系化したパターンシステム(Pattern System)を提案している[Busc96]．しかし，アーキテクチャパターンやデザインパターンの単位で捉えているため，設計に適用できる指針とまではなっていない．本稿の一つの目的は，より実践的なパターンシステムの構築とその構築方法論にある．

また，自然界における進化のパターン[Elde99]のように，デザインパターンやアーキテクチャパターンの関係をデザインの進化のパターンとして捉えることができると考えている[Aoya99b]．進化のパターン化は多様なデザインパターンやアーキテクチャパターンの共通性と個別性を識別する手ががりとなり，パターンの発見，抽出，選択，利用の助けとなるだろう．

本稿では，デザインパターンを中心にパターンの体系化とその表現を提案し，さらに進化の観点からの意味付けを議論する．

なお，本稿では，アーキテクチャパターンやデザインパターンをまとめてソフトウェアパターンあるいは単にパターンと呼ぶ．これらの進化のパターンを進化パターンと呼ぶ．

2. ソフトウェアパターンモデルとその表現

パターン群を分析，体系化し，パターン間の関係を分析するためにはパターンをモデル化し，表現する必要がある．多くのソフトウェアパターンは Alexander のパターン言語[Alex77][Alex79]に基づいてカタログ化されているが，その表現は作成者により異なり，統一がとれているとはいえない．GoF(Gang of Four)[Gamm94]，PLOP [Mesz97]，あるいは著者固有の表現など幾つかの表現がある．パターンを広く実践するためには，このような表記法のいわば標準化と共通の意味モデルが必要である．

パターン言語の概念を表すために，主要要素の関係を図-1に示す．パターン言語の核となる構成要素は，前提，問題，フォース，解決方法であると考えたメタモデルを図-2に示す[Maex97]．パターン言語の表現に多様性が生まれる一つの原因は問題記述の難しさにある．具体的な手本となるための具体性と異なる前提のもとで共通に適用できるための抽象性とのバランスをとることが難しいからである．このため，本稿では，パターンの問題を次の2つに分けて表現することを提案したい．

1)問題：機能的な視点から中心となる問題を記述する．

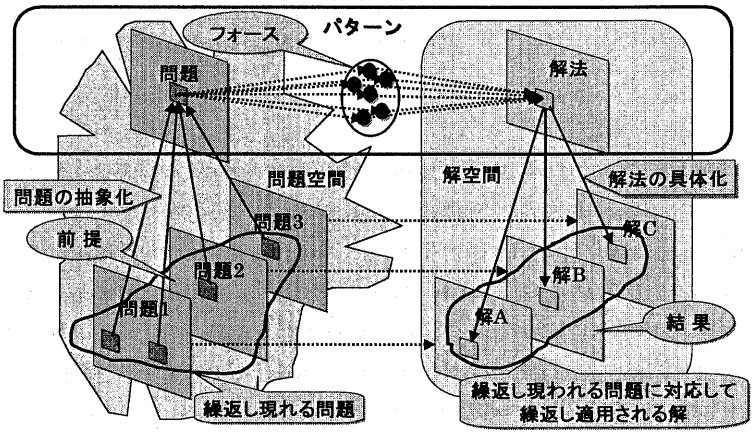

図-1 ソフトウェアパターンのモデル

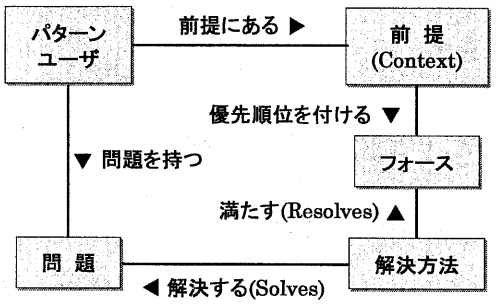

図-2 パターン言語のメタモデル

2)フォース(Forces)：問題を分析した結果，問題解決の考慮点(Concerns)，あるいは解が持つべき特性(Aspects)を記述する．これは，機能的特性では補助的な機能であったり，性能や分かりやすさなどの非機能的特性である．

3. パターンファミリ

オブジェクト生成のファクトリパターンには，ビルダやプロトタイプなど問題やフォースとアーキテクチャが似通った幾つかのパターンがある．このような問題やフォースが類似のパターン群はグループを形成し，その設計原理は共通であると考えられる．これを Parnas のプログラムファミリと同様に，パターンファミリと名付ける．

図-3に示すように，パターン全体の構造は問題-解空間をファミリへ分割するファミリ間の構造とファミリ内での各パターン間の構造の2つの階層で議論できる．

パターンシステムを構築することは，まず，パターン群をファミリへグループ化する方法とファミリ内のパターン間の関係などのファミリの構造を分析することにある．

3.1 パターンのファミリ化

デザインパターンなどのパターン化の鍵は図-2にメタモデルを示すパターン言語の枠組みにある．パターンは前提－問題－フォース－解決方法が対となって意味がある．したがって，パターンのファミリ化は，従来のプログラムや設計方法の分類に当たる解決方法，すなわち解の構造の分類だけでは不十分である．例えば，パターンの適用実験の結果，解の構造の適合性に加え目的の適合性を加えた2次元の尺度による分類が適切であったとの報告がある[Fshu96].

図-2に示すように，フォースは解決方法と直接関連を持ち，解決方法が満たすべき特性を規定しているので，パターンの選択に問題，フォース，解決方法の3つ組

PFS=(Problem, Force, Solution)

で考える．

3.2 パターンファミリの構造とフォース

同一ファミリ内のパターンは設計原理を共有していると考えられる．この設計原理に対する目的をファミリの**主要フォース(Principal Forces)**と呼ぶことにする．例えば，ファクトリパターンファミリでは，「実装を隠蔽して任意のオブジェクトを動的に生成する」ことである．この主要フォースを満たすパターンをファミリの**基本パターン(Base Pattern)**と呼ぶことにする．ファミリ内の基本パターン以外のパターンを**派生パターン(Derived Patterns)**と呼ぶことにする．

例として，ファクトリパターンファミリのフォースを分類した結果を表-1に示す．ファクトリの主要フォースを満たす解決方法は抽象ファクトリであると考えられる．従って，抽象ファクトリを基本パターンとする．この他のパターンは，抽象ファクトリパターンからの派生パターンであると考えられる．

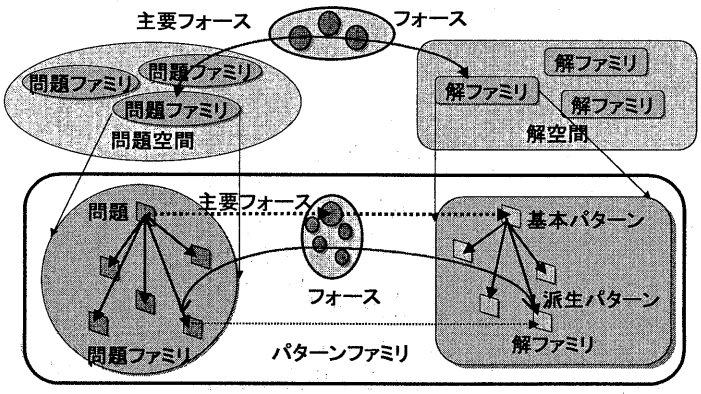

図-3 問題・解空間におけるパターンファミリの構造

表-1 ファクトリパターンのフォース分析

パターン名	フォース
抽象ファクトリ	生成対象を可変とする.
ビルダ	生成対象の複雑化：複合オブジェクトの生成
プロトタイプ	生成対象の単純化：同一オブジェクトのコピー生成
ファクトリメソッド	生成対象の簡略化：生成対象の種類を固定
オブジェクトプール	生成速度の高速化：オブジェクトのリアルタイムな生成
シングルトン	生成個数の限定：生成個数を1個に限定

図-3に示すように，パターンのファミリ化は問題空間における問題ファミリと解空間における解ファミリの対となる．フォースは問題ファミリと解ファミリとの関連を示す．主要フォースは，問題と基本解との間の関連である．その他のフォースは，問題空間における派生問題をと解空間における派生パターンとの関連を付ける．したがって，問題ファミリの問題群のフォースを分析すると対応する派生パターンを選択することができる．すなわち，*問題空間内*で解空間における解の選択のナビゲーションが可能となる．これによって，問題－解決方法を対として表現するパターン言語の最大の特性を活かすことができる．

3.3 解空間におけるパターンファミリ構造モデル

同一のパターンであっても設計のトレードオフを考慮したパターンの詳細な設計が必要な場合がある．このプロセスを支援するために，解空間内で設計の観点からパターン間の詳細な関係を明らかにする必要がある．これは，パターン進化の観点で後述する．

3.4 パターンファミリ間の構造モデル

異なるファミリのパターンを探索したり組み合せることも必要となる．このため，図-3の中で，パターンファミリを構成要素とする問題－解空間全体における要素間の関係の構造が探索の鍵となる．

PFSの対として，ファミリを特徴づけるのは次の3つ組 PHF_0である．

PFS_0=(問題，主要フォース，基本パターン)

ここで，主要フォースは基本パターンの主要フォースである．したがって，基本パターンがパターンファミリの代表パターンであると言える．PHF_0は問題－解空間全体の中でパターンファミリを探索する鍵となる．

4. メタパターンとその表現方法

パターンファミリや組み合せなどの分析には，一つのパターンをまとめて表現できる，パターンより抽象度の高いモデルが必要である．Preeはこのようなモデルをメタパターンと呼び，一つのパターンを1個のクラスと対応付けた表記法を提案している[Pree94]．しかし，パターンを分析し，組み合せるためには，クラスの表現方法では不十分であり，次のような表現力が望まれる．

1)パターン全体が一要素として表現可能
2)パターンの主要フォースが表現可能
3)パターン内のマイクロアーキテクチャを視覚的に表現可能
4)パターンと外部とのインタフェースを表現可能

このような表現上の要求に基づき，Catalysis [DSou98]のタイプモデル(Type Model)の表記法を拡張した図-4に示す視覚的表記法を用いてパターンを表す．これをパターンタイプと呼ぶ．

図に示すように，UMLのクラス図の表現と同じように，パターン名，主要フォース，パターン構造，パターンプロトコルの4つの部分から成る．パターン構造はパターンの抽象構造を表しているので，構造上の主要なクラスのみを表す．例えば，図-4に示す例では，実際には，Factoryクラスの下に生成するプロダクトに対応する複数の具象ファクトリクラスがある．

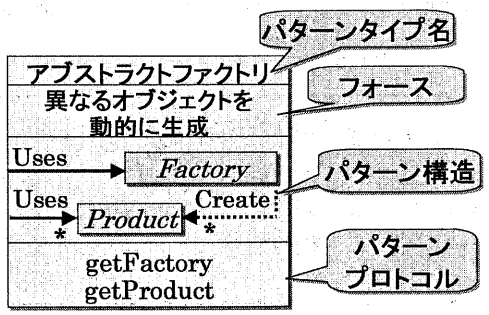

図-4 パターンタイプの表現方法

5. パターン進化パターン

5.1パターン進化の構造

パターン間の関係はパターンに対するフォースや解決方法の進化にともなう設計進化と考えることができる．これを**パターン進化(Pattern Evolution)**と呼ぶ．パターンファミリの概念によって，パターンの進化はファミリ内の**マイクロ進化**とファミリを越えた**マクロ進化**の2つに分類できる．

1)マイクロ進化パターン：ファミリ内での進化である．ファミリを代表するアーキテクチャを持つ基本パターンから派生パターンへと進化する．

2)マクロ進化パターン：ファミリを超えた進化である．例えば，複数のパターンを組み合せて新しいパターンを構成する場合がある．解の構造やフォースなどの質的変化をもたらすと考えられる．

5.2 進化を引き起こす原子進化パターン

パターンの構造は一つのアーキテクチャと捉えられるので，パターンの構造をコンポーネントとコネクタから成るアーキテクチャと考える．したがって，パターンの進化は，**コンポーネント進化**と**コネクタ(構造)進化**の2つの**原子進化パターン**で捉えることができる．ただし，コンポーネントは構造をもち，再帰的に定義できるので，コンポーネントの進化は内部で構造の進化を伴う場合を含む．

本稿では，原子進化パターンを表すために，パターンを構成するコンポーネントとコネクタの構造変化を引き起こす原子進化オペレーションによって生成的に表現する．

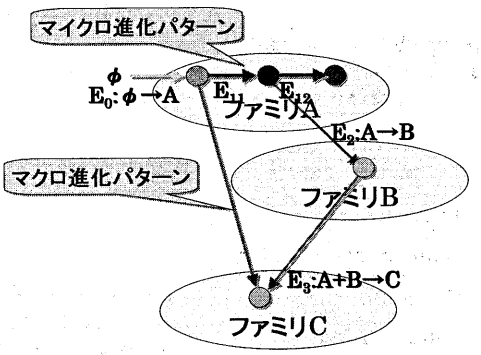

図-5 パターン進化パターン

この原子進化オペレーションとして，次の2つのオペレーションを導入する．

1)置換オペレーション

rep(x, y):x をyで置き換える

2)接続オペレーション：

con(x, y, z):x をyの前(z="pre")あるいは後(z="post")に接続する.

ここで，接続オペレーションは順序に意味があることに注意する必要がある．

5.3 マイクロ進化パターン

図-6はファクトリパターン[Gamm94][Gran98]ファミリ内のパターンの進化を示すパターン進化図PED (Pattern Evolution Diagram)である．各パターンを図-4に示したパターンタイプで表す．パターンを表すボックス間の矢印付き線がパターン間の進化の方向を示し，その上に示す丸四角形の中に進化を構成する原子進化オペレーション群を順序を含めて示す．アブストラクトファクトリパターンを基本パターンとして，派生パターンが原子進化オペレーションにより表現できる．

ファクトリの設計においても様々な制約などのフォースを考慮する必要がある．PEDは設計制約に応じて適したパターンを発見するナビゲータとして利用できる．

6. マクロ進化パターンとメタパターン

複数のパターンやアーキテクチャを組み合せて新しいパターンやアーキテクチャを構成するためにはアーキテクチャレベルの原子進化パターンを定義する必要がある．これは，**メタパターン**あるいは**ジェネリックパターン**[Rieh97]と言える．アーキテクチャの原子進化パターンも，パターンファミリ内の原子進化オペレーションと本質的に同一と考えられるので，次の2つの進化パターンに分けて考える．

1)**コンポーネント進化パターン［置換(replacement)メタパターン］**

複数のパターンあるいはコンポーネントを組み合せたり変更したパターンを一つの複合パターンで置き換える．デザインパターンの一つであるコンポジットパターンがメタパターンとしても適用できる[Rieh97]．例えば，オブザーバ，ストラテジとコンポジットパターンを組み合せ

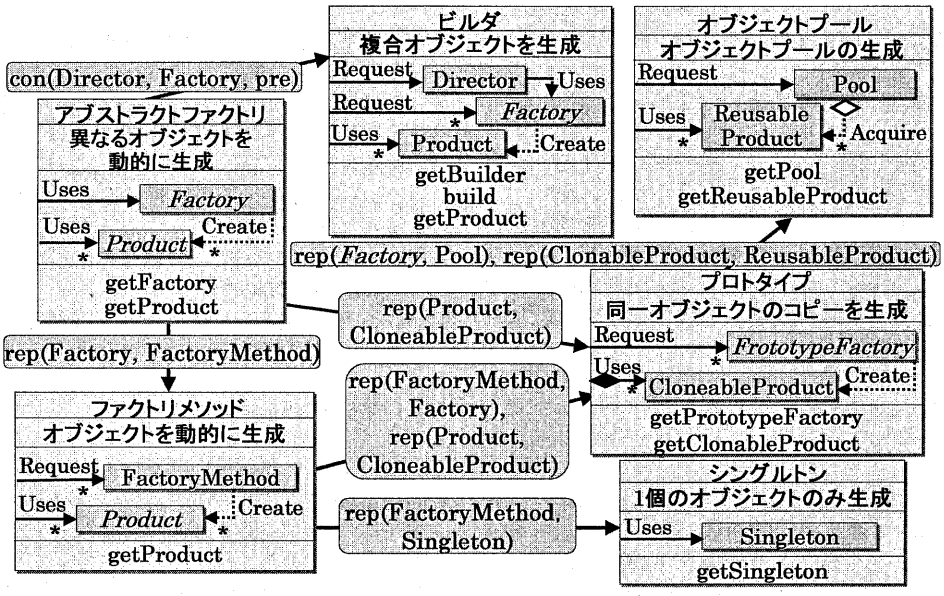

図-6 ファクトリパターンファミリのマイクロ進化パターン

てMVCフレームワークが構成される．

2)コネクタ進化パターン［接続(connection)メタパターン］

複数のパターンを接続するメタパターンには，オブジェクトの接続を支援するデザインパターンが適用できる．接続はクラスなどを共有する静的接続（構造パターン）とイベントによる動的接続(振舞いパターン)の2つがある．

静的接続メタパターンとしては，カスケードパターン[Fost99]などが適用できる．

カスケードパターンでは，同一階層内の接続と異なる階層間の接続がある．接続の相手は単一クラスだけでなく，複合オブジェクトであってもよい．カスケードパターンを用いて複数のパターンを接続した例を図-7に示す．図の左側は，State, Reactor, Chain of Responsibility を接続して POS 端末のフレームワークを作成した例[1]である．この場合，異なる階層間でパターンを接続している．図の右側の2つのオブザーバパターンの接続例[Noda97]は同一階層内で接続したものである．

一方，動的接続メタパターンとして，メディエータパターンやブローカパターンなどが適用できると考えている．

上記の2つの組み合せメタパターンは，表-2に示すように分類できる．

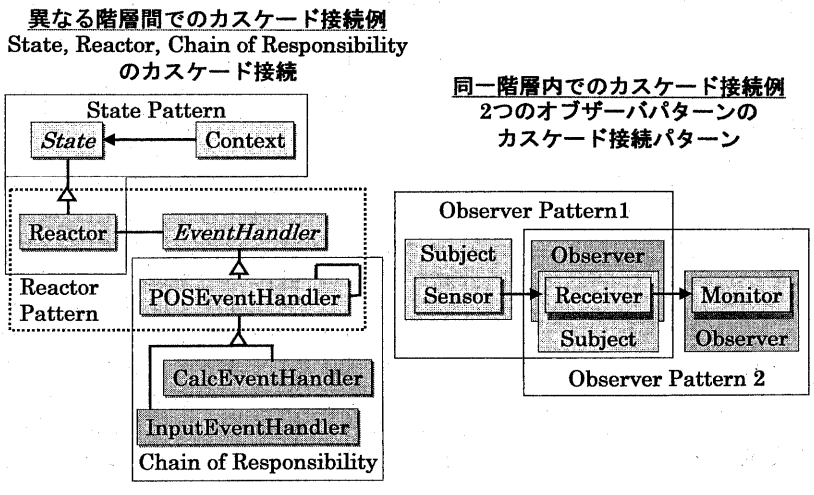

図-7 カスケードによる接続メタパターンの例

[1]本事例は情報処理振興事業協会より先進的情報処理技術促進に係わる「次世代コンポーネントウェア技術に関する研究開発」の発注を受けて実施したプロジェクトでの成果です．

表-2 メタパターンの性質

メタパターン	置き換え(コンポーネント)	接続(コネクタ)	
接続方法	委譲(コンテナ・コンポーネント)	静的(オブジェクト共有,または継承)	動的(状態とイベント)
ライフサイクル	共有(Composite集約)あるいは独立(集約)	部分的共有	独立

7. 関連研究

パターンに関する研究は，パターンの発掘とカタログ化が中心であり，パターンの組み合せや設計進化の観点からの研究は少ない．

文献[Fost99], [Rieh97]ではパターンを組み合せる観点から，それぞれ，カスケードパターン，コンポジットパターンのみを個別に議論している．本稿では，進化を含むより広い視点で，パターンの組み合せとの進化をパターン化した．また，原子進化パターンの置換，接続オペレーションは Subject-Oriented Programming の Composition Rules[Ossh95]の Replace, Joint に対応している．

8．今後の課題

本稿では，考察したパターン指向開発の概念や開発方法の厳密な検討とその支援環境の開発を検討している．

また，パターンはコンポーネントの表現モデルとして適していると考えている[Aoya98a][Aoya98b]．パターンを用いたコンポーネント指向開発の支援についても検討したい．

9．まとめ

ソフトウェアパターンの選択，適用のためにはソフトウェアパターンを整理し，体系化する方法が必要である．このための基礎概念として，ソフトウェアパターンのファミリ化とファミリ内およびファミリ間の構造のモデル化などを考察した．また，設計を絞り込むためには機能的特性だけでは不十分である．問題の非機能的な要求などの情報を活かして，設計を絞る込む方法が求められている．この観点から，パターン言語におけるフォースをパターンの選択，適用に用いる方法を議論した．

今後，本稿での議論に基づき，パターンを活用した設計方法について検討を進める．

参考文献

[Alex77] C. Alexander, et al., *A Pattern Language: Towns-Buildings-Construction*, Oxford University Press, 1977[パタン・ランゲージ:町・建物・施工, 鹿島出版会, 1984].

[Alex79] C. Alexander, The Timeless Way of Building, Oxford University Press, 1979 [平田 翰那(訳), 時を超えた建設の道, 鹿島出版会, 1993.

[Aoya98a] 青山 幹雄, デザインパターンコンポーネント, 情報処理学会ウィンターワークショップ・イン・恵那論文集, Jan. 1998, pp. 39-40.

[Aoya98b] 青山幹雄, 中所武司, 向山 博 (編著), コンポーネントウェア, 共立出版, Jul. 1998.

[Aoya99a] 青山 幹雄, コンポーネント指向ソフトウェアのアーキテクチャと開発方法:フレームワーク, パターン, コンポーネントを組み合せる開発技術, オブジェクト指向 '99 シンポジウム論文集, Jul. 1999, pp. 217-224.

[Aoya99b] 青山 幹雄, パターン進化パターン, 情報処理学会サマーワークショップ・イン・小樽論文集, Sep. 1999, pp. 47-48.

[Busc96] F. Buschmann, et al., *Pattern-Oriented Software Architecture*, John Wiley & Sons, 1996[金澤 典子ほか(訳), ソフトウェアアーキテクチャ, トッパン, 1999].

[Dsou98] D. F. D'Souza and A. C. Wills, *Objects, Components and Frameworks with UML: The Catalysis Approach*, Addison Wesley, 1998.

[Elde99] N. Eldredge, *The Pattern of Evolution*, W. H. Freeman and Company, 1999.

[Fost99] T. Foster and L. Zhao, Cascade, *J. of OOP*, Feb. 1999, pp. 18-24.

[Fshu96] F. Fshull, et al., *An Inductive Method for Discovering Design Patterns from Object-Oriented Software Systems*, CS-TR-3597, Univ. of Maryland, Jan. 1996.

[Gamm94] E. Gamma, et al., *Design Patterns*, Addison-Wesley, 1994[本位田 真一, 吉田 和樹(監訳), デザインパターン, ソフトバンク, 1995].

[Gran98] D. Grand, *Patterns in Java, Vol. 1*, John Wiley & Sons, 1998.

[Mesz97] G. Meszaros and J. Doble, A Pattern Language for Pattern Writing, R. Martin, et al. (eds.), *Pattern Languages of Program Design 3*, Addison Wesley, 1997, pp. 529-574.

[Naka99] 中谷 多哉子, 青山 幹雄, 佐藤 啓太(編), ソフトウェアパターン, bit臨時増刊, 共立出版, Oct. 1999.

[Noda97] 野田 夏子, 岸 知二, パターンを用いたアーキテクチャ設計, オブジェクト指向最前線－情報処理学会OO'97シンポジウム, 朝倉書店, 1997, pp. 23-30.

[Ossh95] H. Ossher, et al., Subject-Oriented Composition Rules, *Proc. OOPSLA '95*, Oct. 1995.

[Pree94] W. Pree, *Design Patterns for Object-Oriented Software Development*, Addison Wesley, 1994 [佐藤 啓太, 金澤 典子(訳), デザインパターンプログラミング(補訂版), トッパン, 1998].

[Rieh97] D. Riehle, Composite Design Patterns, *Proc. OOPSLA 97*, Oct. 1997, pp. 218-228.

[Schm96] D. C. Schmidt, et al (eds.), Software Patterns, *Comm. of the ACM*, Vol. 39, No. 10, Oct. 1996, pp. 36-82.

[Yama98] 山本里枝子, パターンを用いたチーム開発のためのプロジェクト管理, 上原 三八, 鯵坂 恒夫(編), オブジェクト指向最前線'98－情報処理学会OO'98シンポジウム, 朝倉書店, 1998, pp. 203-210.

[Yosh99]吉田裕之, 森崎雅稔, パターン指向アプリケーション開発, *FUJITSU*, Vol. 50, No. 3, May 1999, pp. 170-174.