3 ドメイン指向パターン

—効率的なシステム開発のための分析・設計ノウハウ集 —

吉田 裕之　富士通（株）
中山 裕子　（株）富士通研究所

パターンとドメイン指向

ドメイン分析・モデリングとオブジェクト指向は，ソフトウェアの再利用に関して互いに補完し合う技術である[1]．たとえば，ドメインモデルの記述にUML（Unified Modeling Language）を利用することは多いし，フレームワーク設計には対象となるアプリケーションドメインをよく分析することが欠かせない．本稿では，オブジェクト指向技術の中で特に「パターン」を取り上げて，ドメイン指向開発での位置付けを解説する．

パターンとは「ある特定のコンテキストで，何度も起こる課題に対する，典型的な解法を明文化したもの」である．たとえば，「ビジネスアプリケーションのオブジェクト指向分析では［コンテキスト］，オブジェクトの抽出［課題］のために，要求定義文の中の名詞をリストアップしてみよ［解法］」といったソフトウェア開発上のさまざまなノウハウをある一定の形式で明文化していこう，という取組みである．ただしこの定義は一面的であり，“パターンとフレームワーク”[2]でJohnsonがより多面的に定義しているので，ぜひ読んでいただきたい．パターンは，90年代のオブジェクト指向分野で主要なトピックの1つであり，数十の関連あるパターンを掲載したパターンカタログが何冊も出版されている．一般にパターンカタログでは，記述形式のテンプレートを用意して，すべてのパターンの記述項目を統一する．

パターンカタログの中でも，先駆的かつ最も影響力を持ったのは，Gamma，Helm，Johnson，Vlissidesのいわゆる GoF（Gang of Four）による“Design Patterns”[3]であろう．GoFは，「いかに再利用性が高く，かつ変更に強いソフトウェアを設計するか」をテーマに，23種のオブジェクト設計テクニックをパターンとして提供した．GoFのパターンは，オブジェクト設計を語る基本的なボキャブラリとしてすでに広く利用されている．

GoFのパターンは，対象とするアプリケーションドメインを選ばない，汎用性がきわめて高いものである．単純に考えると，パターンは汎用性が高ければ高いほど，いろいろなケースに適用できてよい気がする．パターンの執筆者にとっても，自分のアイデアをできるだけ多くの人に使ってもらえる方がうれしいだろう．ある意味でこれに異を唱えるのが，ドメイン指向の立場である．

アプリケーション開発者は，完成までの間に多数の設計上・実装上の課題を解決しなければならない．市販されているパターンカタログを読破しておけば，それらの課題のすべてがたちどころに解決してしまう，というわけではない．市販のパターンで解決できる課題はむしろ少数である．しかし，残りのすべての課題がまったく新奇のものであるはずがない．以前にも似たようなアプリケーションの開発者が同じように悩んで解決したものであろう．適用できる対象ドメインを限定してよければ，パターン化の候補となる設計テクニックはもっとたくさんあるはずである．

汎用性の高さはより抽象的であることを意味する．裏返して考えると，それだけ理解が難しくなり，適用するための具体化作業が増える．図-1は，アプリケーション開発者がパターンを適用する手順を示している．アプリケーション開発者は，解くべき具体的な課題を持っている．これに適用できるパターンを選択するためには，その課題をより一般的に捉え直して，パターンが記述されているのと同レベルの抽象度で照合しなければならない．照合に成功し，適用すべきパターンが選択できたら，次にはそれを自分の課題のレベルに具体化しなければならない．

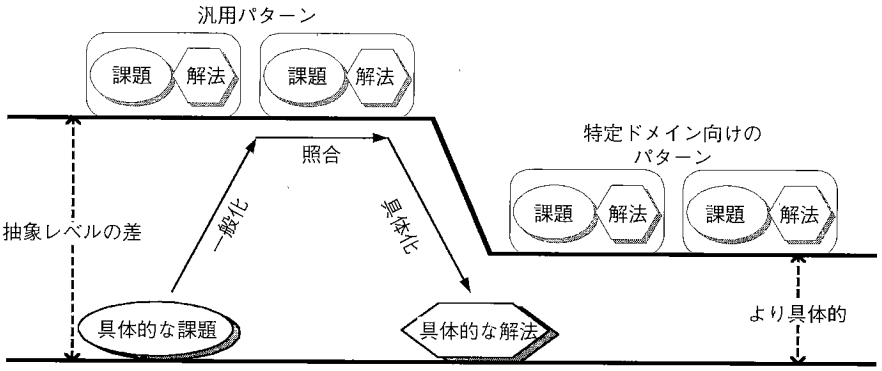

図-1　パターンの適用手順と抽象レベルの差

具体的な課題とパターンとの抽象レベルの差が大きければ大きいほど，このパターン適用作業は難しくなる．対象ドメインを適切に限定できれば，パターンを過度に抽象化する必要がなく，適用作業の困難さを緩和できる．十分に対象を限定できれば，パターンの適用をツールで自動化さえできるかもしれない．

特定の対象ドメインに向けてのパターン化技術は，より大きな効果を生み出すことが期待できる．本稿では，まず特定ドメインに向けてのパターン化の取組みを紹介し，次にドメイン指向のパターン化技術についてさらに議論する．

ドメインを絞った取組み

パターンに関する既存の取組みの中で，特定のドメインに対象を限定しているものは，大きく２種類に分類できる．１つは，既知の汎用パターンを特定のドメインでいかに利用するか，もう１つは，特定のドメインに適用範囲を限定したテクニックのパターン化である．

◆汎用パターンの特定ドメインへの適用

GoF[3] やCoad[4] などの著名な汎用設計パターンを，ある特定の対象ドメインに適用した事例は多い．抽象度が高い汎用パターンを，特定ドメインの事例で具体的に説明してもらうことは，そのドメインの開発者がパターンを理解するうえで非常に助けになる．図-1でいえば，左側の抽象レベルが高い段にあったパターンを，より具体的な適用方法を説明することで右側の低い段に降ろしたことになる．

代表的な２事例を紹介する．

①証券取引アプリケーションとデザインパターン[5]

Zhangらは，証券取引アプリケーションを題材にGoFの４つのパターンの適用方法を解説した．

まず，債券の種類ごとに異なる評価額の計算方法をプラグインにするために，StateとSingletonパターンを利用した．

次に，オプション取引のモデルと，それらを評価する各種のメソッドとを分離するために，Bridgeパターンを利用した．

さらに，利回り曲線オブジェクトの構築メソッドをそのオブジェクト自身から分離するために，Abstract Factoryパターンを利用した．

②デザインパターンに基づく製造フレームワークのアーキテクチャ[6]

Schmidは，CIMシステムのフレームワークの設計に，GoFのパターンを適用した．まず，オブジェクト指向分析を実行し，現実のCIMシステムの構造を自然に表現したオブジェクトモデルを作成した．次に，これに６つのパターンを順に適用して，次第に再利用性が高い設計になることを示した．

たとえば，CNC（Computerized Numerical Control）マシンと搬送用ロボットとのインタフェースを適合，統一するためにAdaptorパターンを利用した．また，制御処理アルゴリズムをプラグインにするためにStrategyパターンを利用した．

◆特定ドメイン向けのパターン

ここでは，ビジネスアプリケーションまたはその一部に対象ドメインを絞って，開発ノウハウをパターン化した取組み事例を紹介する．図-1でいえば，右側の抽象レベルが低い段を探すと初めて見つかるノウハウのパターン化である．

GoFのパターンはオブジェクト設計において再利用性を高めるテクニックを扱ったものであり，初期のパターン化の取組みは，こうした設計ノウハウなどの体系化を目指したものが多かった．その後，設計作業以外のすべての開発作業におけるノウハウ・テクニックの体系化にも，パターンの考え方が適用されるようになった．今日では，要求分析作業や実装作業用のノウハウ，あるいは開発プロジェクト運用のノウハウなどもパターン化されている．そこでこの節では，ビジネスアプリケーション向けのパターンの取組みを，以

下の3つに分類して紹介する.

(1) 分析・モデリングのためのパターン
対象ドメインを分析し，モデリングするためのパターンである.
(2) システム構築のためのパターン
対象ドメインに属するアプリケーションシステムを構築する際の，設計・実装のパターンである.
(3) 開発プロセスのためのパターン
対象ドメインを分析したり，アプリケーション開発したりするときの，開発プロセスのパターンである.

(1) 分析・モデリングのパターン

分析・モデリングに利用するパターンは，対象世界にフォーカスするので，必然的に対象ドメインにある程度特化したものになる．ここでは，代表的な4事例を紹介する.

①データモデルパターン[7)]

Hayは，ビジネスアプリケーションの分析・モデリングのパターンカタログを出版した.

Hayの本は，ビジネスアプリケーション向けのERモデル集である．人物・住所・企業・契約などのビジネスアプリケーションに共通にかかわるERモデルの例を62個，会計やプロセス製造などの特定のドメインに関するERモデルを60個，提供する.

②アナリシスパターン[8)]

Fowlerもまた，ビジネスアプリケーションの分析・モデリングのパターンカタログを出版した.

Fowlerの本は，Martin/Odellのオブジェクト指向表記法を使った，オブジェクトモデリングの教科書である．まず，人と組織，観測と測定などの共通なテーマを解説し，次に在庫管理，計画，トレーディング，デリバティブなどの特定ドメインにフォーカスしたモデリングテクニックを解説している.

この2つの本の共通点は，パターンカタログとしての形式化にこだわっていないことである．テンプレートを使った記述項目の統一はあえて試みず，通常の教科書のように，単純な課題からより複雑なものへモデルをどう洗練していくかを解説している.

③関連オブジェクトのビジネスパターン[9)]

Boydは，ビジネスアプリケーションに現れるオブジェクト間の関連について，どんなときにそれ自身をオブジェクトとしてモデリングすべきかを，3つのパターンで記述した.

Association Objectパターンは，関連をオブジェクト化すべきケースとして，現在だけでなく過去や未来での関連を扱う必要性に着目した．関連オブジェクトには開始日時と終了日時という属性を持たせる.

Customer Contactパターンは，Association Objectパターンの一具体例である．顧客がどの組織にいつどんな方法でコンタクトしたか，などを表現するオブジェクトを，CustomerとOrganizationの関連オブジェクトCustomer Contactとして定義する.

3-Level Orderパターンは，Association Objectパターンを3回続けて適用する例となっている．すなわち，OrderをCustomerとBusinessとの関連オブジェクト，Order RequestをOrderとProductとの関連オブジェクト，さらにOrder FulfillmentをOrder RequestとProductの関連オブジェクトとすることにより，柔軟なデータ構造を実現する.

④ビジネスリソース管理のためのパターン言語[10)]

Bragaらは，ビジネスアプリケーションを分析するための15個のパターンを体系化した．対象としているのは，リソース管理と呼ばれるアプリケーションであり，なんらかのリソースのレンタル，取引，メンテナンスを管理するものである.

各パターンには，アプリケーションの特徴ごとに次に試すべきパターンが何か，が指示されているので，15個のパターン全体がおよそ木構造を構成する．第1のパターンである“Identify The Resources”から始めて，この木構造を順にたどっていけば，アプリケーションの分析モデルが完成する.

(2) システム構築のパターン

対象ドメインに属するアプリケーションシステムを構築するときに利用する，設計・実装上のノウハウをパターン化した取組みを2例紹介する.

①D-AMEを使った金融商品モデリング[11)]

山本らは，目前に迫った金融自由化を念頭に，銀行の預金商品を対象として，ドメイン分析・モデリングを実施した．本特集で紹介したD-AMEプロセスを採用し，金融商品分析のためのさまざまなドキュメント形式，標準的な設計モデル，および設計パターンを開発した.

独自のパターンは次の2つを報告している．Deposit Business Transactionパターンは，各種の預金口座に対する入出金処理のためのクラス構造とインタラクションの一般形を定義する．図-2にクラス図を，図-3にシーケンス図をそれぞれ示す．Accounting Recordパターンは，経理科目処理（現金，小切手，他店振込などの種別ごとに入出金を管理する）の仕組みを定義する.

これらのパターンの意図は，特定の銀行や特定の金融商品に限定されない，このドメインに共通なクラス構造やインタラクションを定めることにより，設計作業を効率化することである．さらに山本らは，パターンの効用として，アーキテクチャの一貫性を

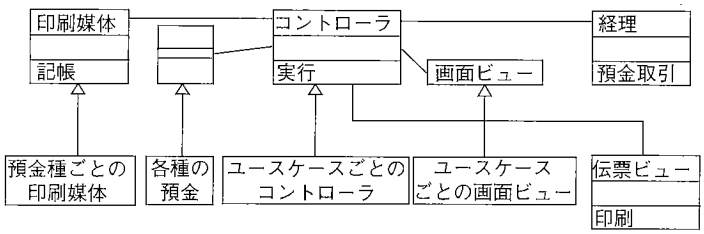

図-2　Deposit Business Transaction パターン（クラス図）

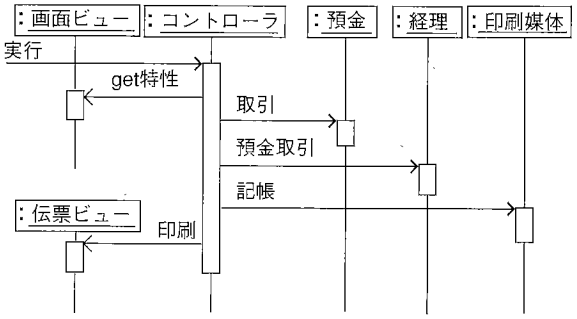

図-3　Deposit Business Transaction パターン（シーケンス図）

保てることを挙げている．たとえばDeposit Business Transaction パターンをすべての種類の預金口座の入出金処理に適用すれば，アプリケーション全体の中で，それらのクラス構造とインタラクションを統一でき，理解性・保守性が高い．

②保険システムのパターン[12]

Kellerもまた，金融の自由化と競争激化を念頭に，保険システムの構築を対象としたパターン集を開発した．7個のパターンを示し，「保険システムの最上位構造」「保険商品モデル」「ポリシー」「保険システムの分散」に分類した．

たとえば，Insurance Value Chain パターンは，保険システムのあるべきサブシステム構成を示している．山本らのパターンと同様に，特定の証券会社に限定されない，ドメインに共通なアプリケーション構造を提示するものである．

Product Tree パターンでは，保険商品を（自転車のモデルでそうするように）木構造でモデル化することを勧めている．これにより，柔軟性，専門家の理解性，再利用性が高まる．

GoF の Composite, Interpreter などよく利用する8個の汎用パターンも列挙している．

(3) 開発プロセスのパターン

ドメイン固有の開発の進め方のノウハウをパターン化した取組みを2例紹介する．

①オブジェクト指向開発のための要求分析プロセスパターン言語[13]

Whitenackは，ビジネスアプリケーションの要求分析手順を，20個からなるパターン集にまとめた．

たとえば，Managing and Meeting Customer Expectations パターンは，製品に対する顧客の期待を管理するために，一覧を作成し，それらを要求と要望とに分類せよ，としている．また，さらに詳細な分類と優先度付けのための，ユースケースマトリクスや要求テンプレートといったドキュメントを提示する．

パターン集は，Building the Right Things（正しいものを作る）パターンを根として，おおむね木構造を構成する．たとえば，Building the Right Things パターンは Customer Rapport（顧客との信頼関係）パターンや，Problem Domain Analysis（問題ドメインの分析）パターンなど，より具体的なパターンを参照する形になっている．

一連の開発手順をパターン集の形式で定義しようという，野心的な試みである．

②顧客とのお付き合いのパターン[12]

Risingは，顧客との良好な関係を保つための10個の秘訣をパターンとして記述した．パターン名を眺めるだけでも興味深いので，以下にその一覧を示す．

- It's a Relationship, Not a Sale
- Know the Customer
- Build Trust
- Customer Meetings: Go Early, Stay Late
- Show Personal Integrity
- Listen, Listen, Listen
- Listen Up
- Be Aware of Boundaries
- Mind Your Manners
- Be Responsive

Risingはパターンカタログのしきたりにしたがって記述テンプレートを設定し，Aliases, Problem, Context, Forces, Solution, Known Uses など各パターンの記述項目を統一している．

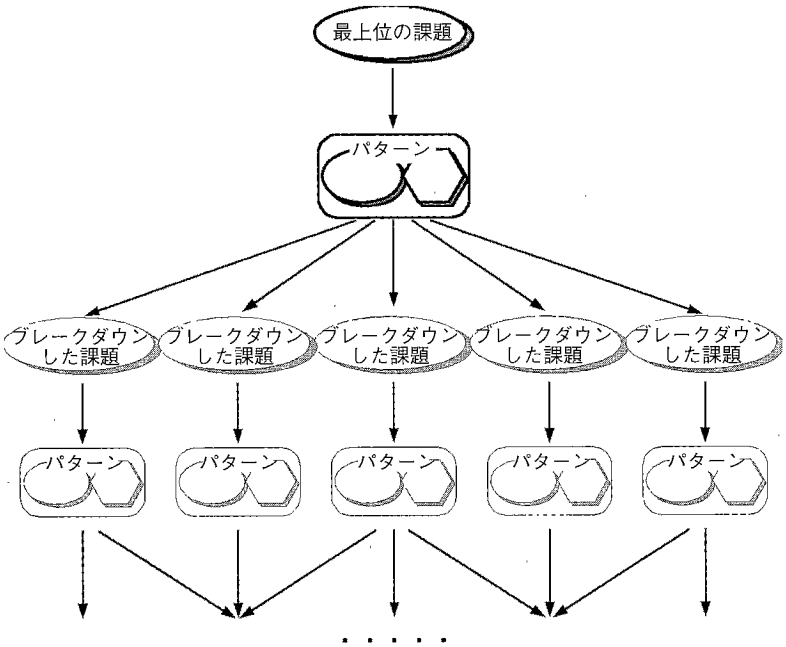

図-4　課題によるパターンの体型化

ドメイン指向アプローチへの期待

本来，ドメイン指向アプローチは，特定のドメインを対象に，体系的に課題を整理し，分析し，解決方法を蓄積する手法である．パターンに関する研究開発活動では，特定の対象ドメインに絞った取組みが増えているものの，ドメイン指向アプローチを明確に意識しているものはまだ少ない．まず先に設計・実装上のアイデアがあって，それをある程度汎用に利用できるようにパターン化した，結果的にある特定の範囲のアプリケーションにならば適用できる，というケースがほとんどである．「このパターンは○○をするようなビジネスアプリケーションに適用可能である」と述べている論文をよく見かける．対象ドメインを最初から明確に意識した取組みではなかったことの現れである．D-AMEプロセスにしたがってドメイン分析・モデリングを実施した山本らの研究開発は，むしろ稀な例といえる．

ドメイン指向アプローチは，再利用したいアイデアが先にある，ある意味でアドホックなアプローチと比べて，パターンの抽出と選択・適用の両面から有利と考えられる．

◆ドメイン指向パターンの抽出

本稿の最初で，パターンは「何度も起こる課題に対する典型的な解法」である，と定義した．典型的なよい解法があっても，その課題が何度も繰り返し起こるものでなければ，あまり有効なパターンではない．

ドメイン指向アプローチでは，まず課題を体系的に洗い出す．そのドメインに含まれるアプリケーションを開発する際に，よく問題になる部分，よく再利用される部分，繰り返し起こる部分はどこかを明らかにし，それらに対して有効なドメイン開発諸機能を対策としてあらかじめ用意することがD-AMEプロセスの目的である．

適切な対策がパターンの提供である場合もあるし，そうでない場合もあろう．たとえば，ソフトウェア部品，あるいは新しい仕様記述言語などの形態で提供することがより適切かもしれない．筆者らの経験では，ソフトウェア部品などは特定のプラットフォームやプログラミング言語，ミドルウェアに依存してしまい，適用が限定されることが多い．これに比べると，パターンはその本質が「アイデアの再利用」であるので適用時の柔軟性が高く，ドメイン開発諸機能の形態として有効であることが多いように思う．

◆ドメイン指向パターンの選択と適用

最初に述べたように，汎用性が高いパターンは適切なものを選択するのがそれだけ困難になる．このため，たとえばGoFは，似ているパターンどうしの違いの説明を加えたり，23個を「生成」「構造」「ふるまい」の3つに分類して，利用者の便宜を計ろうとした．

前節で紹介した事例のいくつかをこれと比べてみると，この点でもドメイン指向アプローチが有利であることが分かる．Kellerはパターンを「保険システムの最上位構造」「保険商品モデル」「ポリシー」「保険システムの分散」に分類している．GoFの分類と好対照である．また，BragaらやWhitenackの

パターンは，木構造を構成するので，これにしたがって順にパターンを適用していけばよい．

パターン群を木構造に構成する事例は他にもよく見受けられる．図-4に典型的な木構造の構成方法を示す．まず，そのドメインで開発者が最初に考えなければならない課題，あるいはそのドメインによく起こる課題のうちで最も規模の大きなものを選び，それに対する典型的解法をパターン化する．次に，このパターンによって分解され，より具体的になった課題を取り上げて，同じようにパターン化する．利用者は，この木構造に従っていけば，次にどのパターンを適用すればよいか，簡単に選択でき，効率よく開発を進めることができる．

また上述したように，ドメインエンジニアリングによって開発するドメイン開発諸機能は，パターンだけとは限らない．ソフトウェア部品，標準モデル，方法論，ドメイン知識など他の形態の開発諸機能とうまく連携することにより，パターンの選択と適用がより容易になる．

最も典型的な連携相手はドメイン指向フレームワークである．一般にフレームワークは，その設計者の意図を理解しなければうまく利用できない．パターンは，設計意図を明示的に記述する優れた手法である．たとえば，図-2，3に示したDeposit Business Transactionパターンは，スーパークラスとして提供される「コントローラ」や「預金」などに対して，どんなサブクラスを定義すると，どんなインタラクションを行えるかを説明している．

すべてのフレームワーク設計者は，その設計意図「どう使ってもらうつもりで設計したのか」をパターンとしていっしょに提供するように，ぜひ心がけてもらいたい．

抽象レベルと使いやすさ

ドメイン分析・モデリングの本質は，「汎用性の追求」に対するアンチテーゼとしての「特定ドメインへのこだわり」である．パターン化技術についても，GoFに代表される汎用のパターンは「初心者には手が出しにくい高度なテクニック」との印象が強い．本稿では，対象ドメインを特定することにより，パターンをより具体的に議論でき，抽出・理解・選択・適用が容易になることを説明した．具体化するほど使いやすくはなるが，反面，適用の機会が減って有効性が薄れていく．そのバランスをうまくとることが，ドメイン分析・モデリングに共通の課題である．

ドメイン指向パターンよりも，さらに具体化した「パターン」も考えられる．筆者らは，開発プロジェクトごとのノウハウを積極的にパターン化していく開発技法を提唱している[14]．プロジェクトごとに蓄積するパターンを「プロジェクト固有パターン」と呼ぶ．インクリメンタル開発を前提とし，初期開発で蓄積したプロジェクト固有パターンを，以降の開発で有効に利用する．また，複数の開発チームの工程に時間差を設けることで，先行しているチームが蓄積したパターンを利用して，後続チームは効率よく開発を進められる．

プロジェクト固有パターンは，「アプリケーション開発者の目線の高さ」のパターンであり，簡単に適用できる．また，汎用やドメイン指向のパターンが「外部の偉い人から教えてもらう」ものであるのに対して，プロジェクト固有パターンは自分たちのためのハンドメイドのパターンである．

汎用のパターン，ドメイン指向パターン，プロジェクト固有パターンという3つの異なる抽象レベルでの取組みが相補的に働くことにより，ノウハウの共有化がさらに進んでいくことを，筆者らは期待している．

参考文献

1) 伊藤 潔, 杵島修三, 田村恭久, 廣田豊彦, 吉田裕之 (編著): ドメイン分析・モデリング, 共立出版 (1996).
2) Johnson, R. E., 中村宏明, 中山裕子, 吉田和樹: パターンとフレームワーク, 共立出版 (1999).
3) Gamma, E. et al.: Design Patterns: Elements of Reusable Object-Oriented Software (1995). 本位田真一, 吉田和樹 (監訳): オブジェクト指向における再利用のためのデザインパターン, ソフトバンク.
4) Coad, P. et al.: Object Models: Strategies, Patterns, and Applications, Second Edition, Prentice Hall (1997). 依田光江 (訳), 今野 睦, 依田智夫 (監訳): 戦略とパターンによるビジネスオブジェクトモデリング, ピアソン・エデュケーション.
5) Zhang, J. Q. and Sternbach, E. J.: Financial Application Design Patterns, A Three Article Series in Journal of Object-Oriented Programming, Vol.8, No.8, pp.6-12, No.9, pp.6-12, and Vol.9, No.1, pp.6-14, SIGS (1996). http://www.irmc.com/downloads.html
6) Schmid, H. A.: Creating the Architecture of a Manufacturing Framework by Design Patterns, Proceedings of OOPSLA'95, pp.370-384, Austin, TX (1995).
7) Hay, D. C.: Data Model Patterns: Conventions of Thought, Dorset House (1996).
8) Fowler, M.: Analysis Patterns: Reusable Object Models, Addison-Wesley (1997). 堀内 一, 児玉公信, 友野晶夫 (訳): アナリシスパターン, アジソン・ウェスレイ.
http://ourworld.compuserve.com/homepages/martin_fowler/
9) Boyd, L. L.: Business Patterns of Association Objects, In Martin, R. et al. (eds.): Pattern Language of Program Design 3, Addison-Wesley (1997).
10) PLOP'99に採択された論文は以下のURLからダウンロードできます.
http://st-www.cs.uiuc.edu/~plop/plop99/proceedings/
11) Yamamoto, R. et al.: Financial Product Modeling based on D-AME, In Itoh, K. et al. (eds.): Domain Oriented Systems Development: Principles and Approaches, IPSJ, Gordon and Breach Science Publishers (1998).
12) PLOP'98に採択された論文は以下のURLからダウンロードできます.
http://st-www.cs.uiuc.edu/~plop/plop98/final_submissions/
13) Whitenack, B.: RAPPeL: A Requirements Analysis Pattern Language for Object-Oriented Development, In Coplien, J.O. et al.(eds.): Pattern Language of Program Design, Addison Wesley (1995).
14) 吉田裕之, 山本里枝子, 上原忠弘, 田中達雄: UMLによるオブジェクト指向開発実践ガイド, 技術評論社 (1999).

(平成11年10月29日受付)