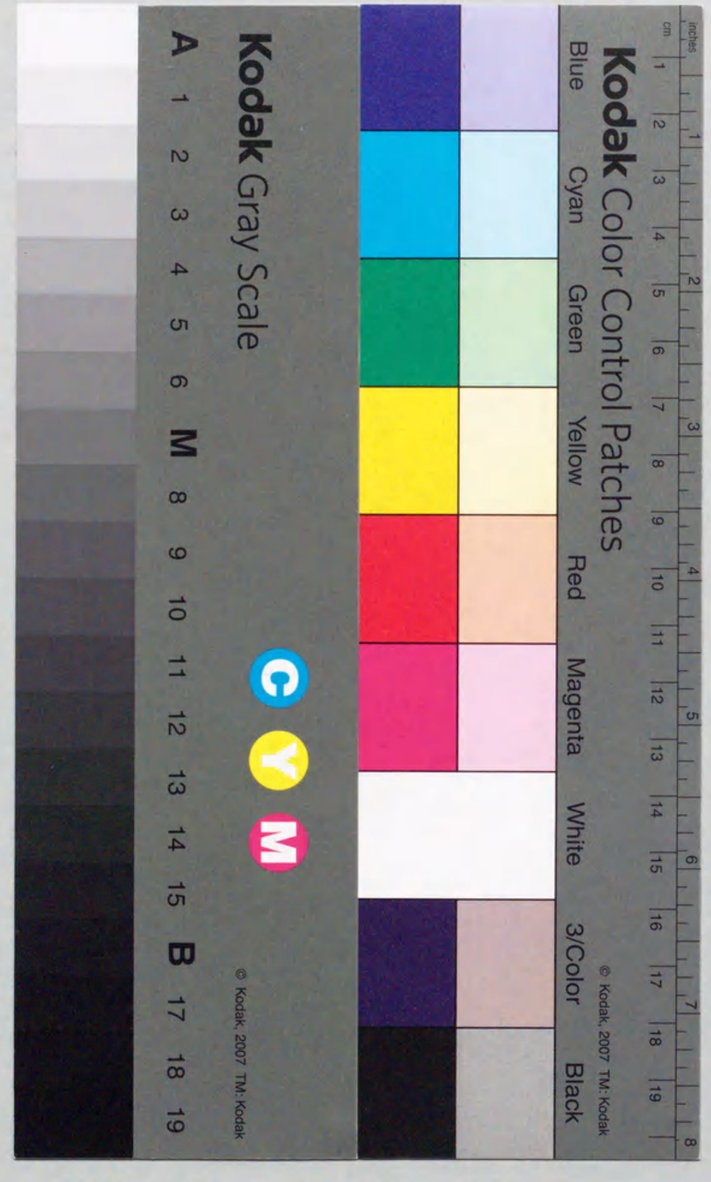


3層アーキテクチャに基づく WWW情報システム開発方式の研究

山本修一郎

(1)

3層アーキテクチャに基づく WWW情報システム開発方式の研究

山本修一郎

概要

企業情報システムの開発では 1990 年代の前半にクライアントサーバシステムが登場しメインフレーム型の企業情報システムのダウンサイジングが進展してきた．特に 1995 年以降からインターネットの普及に伴い企業における WWW を用いた情報システムの導入が進んでいる．このような背景から WWW を利用した企業情報システム開発技術の確立が求められている．

従来から情報システム開発方式として，構造化システム分析技術，オブジェクト指向分析技術，分散システム設計技術などの開発方法論や，WWW サーバなどのミドルウェア技術が研究されてきた．しかしながら，クライアントサーバアーキテクチャに従った WWW 情報システムを構築する上では次のような課題があり，WWW を利用したクライアントサーバ型の企業情報システム開発技術としては必ずしも十分ではなかった．

（課題 1）複数の担当者で同時並行的に分析設計工程の作業を実施する場合には抽出される分析情報や設計情報の重複や抜けなどが発生する可能性がある

（課題 2）クライアントとサーバ間にシステムが階層分割されるため階層間で設計の重複が発生する可能性がある

（課題 3）既存システムを再利用してクライアントサーバアーキテクチャ型の WWW 情報システムへ移行するための手順が明確になっていない

（課題 4）HTML だけではデータベース処理や手続き処理を記述できないため，HTML で表示層処理を記述し C 言語などのプログラミング言語を用いて企業情報システムに必要な機能層やデータ層の処理を実現する必要があった

（課題 5）WWW で用いられる HTTP 通信では，個々の情報ページの送信ごとに通信を切断してしまうため，異なる情報ページにまたがってデータを引継ぐことができない．したがって WWW の導入が情報検索や情報周知などの情報共有型の情報システムなどに限定されていた．

これらの問題を解決するためには，

（1）クライアントサーバアーキテクチャに基づく情報システムの分析工程を既存システムからの移行を考慮して定式化し

（2）分析工程から設計工程へ段階的な手順を確立するとともに

（3）クライアントサーバアーキテクチャにおける各階層の処理を記述できるスクリプト言語ならびに WWW ブラウザを表示層として利用できるセッション管理機能を持つミドルウェアを提供することにより

（4）クライアントサーバアーキテクチャによる WWW 情報システム構築を容易化するための開発方式を確立する必要がある．

本研究では，3 層型のクライアントサーバアーキテクチャに基づいて WWW 情報システムを実現するための系統的な開発方式を提案し，その有効性を実証的に評価する．3 層アー

キテクチャを対象にした理由は，クライアントサーバシステムを表示層，機能層，データ層に階層的に分割することにより，大規模な情報システムを柔軟に構築できるためである．

本研究で提案する 3 層 WWW 情報システム開発方式では，次のようにして 3 層 WWW 情報システムの分析・設計・構築を進める．

分析設計段階では，構造化分析に基づく 3 層設計法，オブジェクト指向分析に基づく 3 層設計法，シナリオフロー設計法を用いて階層間のインタフェースをシナリオフロー図により作成する．シナリオフロー図は 3 層アーキテクチャを構成する表示層，機能層，データ層の各層処理と処理間のデータの流れを定義する図式である．

構造化分析に基づく 3 層設計法は，システムの入出力関係からデータフロー図を自動生成する手法，モジュール構造図からデータフロー図を自動生成する手法，データフロー図からモジュール構造図を作成する手法，モジュール構造図に基づきシナリオフロー図を作成する手法からなる．データフロー図はシステム仕様をプロセスと各プロセスの入出力データおよびその流れに基づいて定義する図式である．モジュール構造図はシステム構造をモジュールとその利用関係に基づいて定義する図式である．

オブジェクト指向分析に基づく 3 層設計法は，役割分析に基づくオブジェクトモデル作成法とオブジェクトモデル図に基づくシナリオフロー図作成法とからなる．オブジェクトモデル図はシステム仕様を 7 種類に分類された類型オブジェクトと類型オブジェクト間の関係に基づいて定義する図式である．

ページフロー図に基づくシナリオフロー設計法では WWW の表示層で必要となる情報ページの遷移関係に基づいてシナリオフロー図を作成する．ページフロー図は WWW の情報ページ構造を類型化された情報ページ間の遷移により定義する図式である．

WWW システム構築技術では WebBASE スクリプトによりシナリオフロー図で仕様化された処理内容を実現する．WebBASE スクリプトは WWW 情報システムを HTML 文と処理手続き文および SQL 文で記述するためのスクリプト言語である．

本研究で提案する 3 層 WWW 情報システム開発方式の特徴は次のようである．

［特徴 1］分析工程でオブジェクトを表示層，機能層，データ層に分類しておき段階的な 3 層設計ができる（課題 1 の解決）

［特徴 2］シナリオフロー図に基づき各階層間のインタフェース設計を共通化できる（課題 2 の解決）

［特徴 3］既存システムのモジュール構造図に基づきデータフロー図を自動生成しておき，データフロー図から段階的にシナリオフロー図を設計できる（課題 1 および課題 3 の解決）

［特徴 4］WebBASE スクリプトにより HTML 文と処理手続き文および SQL 文からなる WWW 情報システムを記述できる（課題 4 の解決）

［特徴 5］WebBASE 処理系により異なる情報ページ間でデータの引継ぎができる（課題 5 の解決）

本論文では，上述した 3 層型のクライアントサーバアーキテクチャに基づく WWW 情報システム開発方式について述べ，その評価結果を明らかにする．本研究で得られた主な結果を以下に列挙する．

（1）データフロー図の自動生成ならびに復元方式

データフローの入出力関係を表現する行列を論理ベクトルの集合と見なすことにより，論理ベクトルの大小関係に基づいてデータフロー図を一意的に生成する決定性アルゴリズムを考案した．このアルゴリズムの特徴は，入出力データの関係を満足する最小のデータフロー図を一意的に自動生成できること，従来手法に較べ，効率的であること，任意の入出力データの関係に適用できることである．

またモジュール構造図から入出力関係に基づいてデータフロー図を復元するアルゴリズムを提案した．大規模ソフトウェア開発ではソフトウェアの分析設計手法の経験者数が限定される．ところが，初心者の場合，誤りや冗長なデータフロー図を作成し易いという問題がある．提案したアルゴリズムを用いることにより，人手によるデータフロー図の復元作業に比べ，約 70%から 80%の作業工数を削減できることを明らかにした．

（2）役割関係とオブジェクト分類に基づくオブジェクト分析方式

情報システムのオブジェクト指向分析を容易化するパターン分類に基づくオブジェクト指向分析手法を提案した．特定の分野に限定されない役割オブジェクトの関係から情報システム分野を対象とした一般的な 7 種類のオブジェクト間の関係を規則的に導出できることを明らかにした．POOM 手法では分析者によるオブジェクトモデルの変換手順を具体化し自動化の可能性を示唆した．また POOM 手法では 7 種類のオブジェクトを表示層，機能層，データ層に分類しているため，3 層アーキテクチャに基づくシステム設計への連続性が高いという特徴がある．

（3）3 階層クライアントサーバシステム設計方式

ページフロー図と呼ぶ WWW 情報システムのための情報ページ遷移図式を提案し，3 層アーキテクチャモデルに適した分散型情報システムを設計するための 3 層クライアントサーバシステム設計法を提案した．この設計法では，3 層インタフェースの設計を容易化するシナリオフロー図により各層モジュールの独立性の概念に基づいて 3 層クライアントサーバシステムにおける層間インタフェースの設計条件を明らかにした．

次いでデータフロー図からのモジュール構造図生成方式を提案した．モジュール構造図生成方式の主な特徴は，データフロー図に基づいて，詳細設計に必要なモジュール構造図の主要部分（約 70%）を自動生成できること，複雑なデータフロー関係を持つデータフロー図に対しても適用できること，データやモジュール名などの設計情報をデータフロー図からモジュール構造図に継承でき設計情報の再利用性を向上できることなどである．

また，モジュール構造図に基づいて分散型情報システムの 3 層アーキテクチャを設計する手法を述べ，3 層アーキテクチャモデルに適した分散型情報システムを設計するための 3 層クライアントサーバシステム設計法を提案した．これにより，構造化手法に基づいて作成された既存システムから 3 層クライアントサーバシステムへの移行方式を具体化した．この手法ではモジュール構造図からデータフロー図の復元過程とデータフロー図からモジュール構造図の生成過程までを自動化できる．またモジュール構造図からシナリオフロー図への変換手順を具体化した．

さらに，オブジェクトモデル図に基づいて分散型情報システムの 3 層アーキテクチャを設計する手法を述べ，3 層アーキテクチャモデルに適した分散型情報システムを設計するための 3 層クライアントサーバシステム設計法を提案した．これにより 3 層アーキテクチャを用いた情報システム開発における分析工程と設計工程における生産物間の追跡性問題を解消するためのひとつの手法を具体化した．この設計手法では分析モデルからシナリオフロー図への変換手順を示し自動化の可能性を示唆した．

(4) スクリプトに基づく WWW－データベース連携方式

HTML と SQL ならびに基本的な言語処理を記述できる新しいスクリプト言語 WebBASE スクリプトを考案し，その処理系 WebBASE システムを実現し，実験評価ならびに適用事例に基づき有効性を明らかにした．WebBASE の特徴は多様なイントラネットアプリケーション開発に適用できること，CGI 方式に比較して約 6 倍の応答性能であること，開発規模を約 40%削減できることである．WebBASE は，WWW 情報システム開発のプラットフォームとしてこれまでに約 740 システム以上への導入実績がある．

(5) 開発方式の実証的な評価方式

本論文で提案する 3 層 WWW 情報システム開発方式を構成する各技術の有効性の評価では，被験者や開発生産物を層別化して比較評価する手法を用いた．システム開発方式の評価尺度として生産物の自動生成率，生産物の均質性，開発工数や生産物の削減率，オピニオンポイントなどを用いた．特に生産物の均質性は，異なる開発者が同一の課題に対して作成した生産物の一致性を測るために本論文で新たに提案した評価尺度である．被験者の層別比較では，初心者と経験者，個人作業とチーム作業を用いた．生産物の層別比較では，表示層，機能層，データ層ごとに生産物の均質性などを評価した．特に個人作業とチーム作業を区別した評価や 3 層を構成する各層ごとの評価については本論文で新たに提案した評価法である．

本論文は次の 7 章から構成されている．まず，第 1 章では，本研究の目的および研究の背景について説明するとともに，本研究の内容を概説する．第 2 章では，3 層アーキテクチャに基づく WWW 情報システム開発方式の全体的な構成とその概要を述べる。第 3 章から

第 5 章で開発方式の具体的な内容について述べる．

第 3 章では，3 層 WWW 情報システムの要求分析技術に関する研究として，システム入力と出力の依存関係に基づくデータフロー図の自動生成方式，モジュール構造図からのデータフロー図の自動復元方式，オブジェクトパターンに基づくオブジェクトモデル作成方式を提案する．

第 4 章では，WWW 情報システムのページ構成の設計を容易化するページフロー図と 3 層アーキテクチャに適したシナリオフロー図に基づく設計方式を提案する．さらに，データフロー図からのモジュール構造図の生成方式，モジュール構造図に基づくシナリオフロー図の作成法，オブジェクトモデルからのシナリオフロー図の作成法を提案し，3 層アーキテクチャに基づく情報システムの設計作業を容易化する方式を明らかにする．

第 5 章では，WWW を用いた分散型情報システムで必須の機能である WWW とデータベースを連携するために開発した WebBASE について述べる．WebBASE の適用事例について述べ WWW を用いた情報システム開発を容易化できることを示す．

第 6 章では，本論文で提案するシステム開発方式の実証的な評価法とその結果について述べる．第 7 章は結論であり，本研究で得られた技術成果をまとめ，今後の技術的課題について述べる．

目次

第 1 章 序論

1.1 研究の目的

1990 年代後半に至って，クライアント／サーバや WWW に代表されるように，情報システムのアーキテクチャが集中型から分散型へ急速に移行している．これに従い，1960 年代後半から 30 年間に渡って集中型情報システムを中心に展開されてきた情報システム構築技術の研究も，クライアント／サーバ・システム開発技術や WWW システム開発技術に代表される分散型情報システムの構築技術についての研究への転換が求められている[1]．しかしながら，大規模情報システムの分散型へのダウンサイジングにおける開発効率を向上させる上で，これまでの分散型情報システムの構築技術の研究には，以下のような課題がある．

(1) 分散型情報システムの分析に関する課題

大規模情報システムの要求分析では，多数の要員が参加することから作成される仕様の均質性が重要になる．ところが，情報システム分野で最も普及している構造化手法でも従来の集中型システムについての大まかな指針しか与えられていないため，分散型情報システムの分析では，分析者によって作成される仕様の品質に大きな差が生じやすい[2]．また，大規模情報システムの要求分析では，ソフトウェアの仕様だけではなく，ソフトウェアを取り巻く環境やシステムアーキテクチャについても考慮していく必要がある[3][4][5][6]．しかし，分散型情報システムのアーキテクチャを前提とした要求分析手法はこれまで検討されていない．従来から要求分析と設計とのギャップがさまざまなソフトウェア開発方法論の問題として指摘されてきたが，その理由の一つは，これまでの要求分析手法や設計手法では，システムのアーキテクチャを考慮していないことにあると思われる．

また，大規模情報システムをダウンサイジングする場合，これまでに蓄積された大量の既存情報システムのリエンジニアリングが必要となるが，リバース分析のための系統的な手法がないため，分析者の習熟度に依存してしまい，リバースされた仕様の均質性が保証できないという問題があった[7]．WWW 情報システムへ移行するための方法論でも既存情報システムの分析が必要となる[8]．

(2) 分散型情報システムの設計に関する課題

これまでのクライアントサーバの 2 層アーキテクチャについては，オブジェクト指向設計方法論が提案されている[9]．3 層アーキテクチャは，2 層アーキテクチャよりもシステム進化の柔軟性や性能面で優れており，3 層アーキテクチャに基づく設計法が望まれる．しかし，汎用機による既存の情報システムのデータベースをアプリケーション・サーバで収集し，パソコン上の GUI により活用するための 3 層アーキテクチャ[10]を前提とした実務レベルの開発方法論は存在していない．

また，これまでに研究されている構造化手法やオブジェクト指向設計手法は，分散配置

されるコンポーネント間のインタフェースに基づいた方法論になっていない．分散型情報システムの設計では，インタフェース定義に基づいたコンポーネントの組合せを記述できる必要がある[11]．

WWW 情報システムの開発では，単に情報ページを作るというプロセスだけを考えるのではなく，企業情報システムとしてどう位置付けるかという要求分析上の問題も含めたWWW 情報システム全体に渡る系統的な開発方法論が必要である[12][13][14]．これまでのハイパーメディア開発方法論は，主にスタンドアロンを前提としており，分散型情報システムを対象としていない[15][16]．

(3) 分散型情報システムの構築に関する課題

分散型情報システムの構築では，GUI 製品やデータベース製品に依存しないオープンな分散型情報システム構築に適したスケーラブルなミドルウェアが必要である[17]．分散型情報システムでは，さまざまなコンポーネントを連携させていくため，エンドユーザでも容易に利用できる記述性の高いスクリプト言語とその高速な処理系としてのミドルウェアが必要である．また，このようなミドルウェアでは，クライアント／サーバ型情報システムとWWW 情報システムとを統合できる必要がある[18]．

(4) 開発方式の実験評価に関する課題

これまでのソフトウェア工学研究におけるソフトウェア開発方式の提案では手法の有効性を定性的に評価するだけにとどまっている場合が多い．新しいソフトウェア開発方式の有効性を評価するためには開発手法の定量的な評価尺度と評価実験の手法を確立する必要がある．

本論文の目的は，以上述べた課題について 3 層 WWW 情報システムに対する要求分析技術，3 層アーキテクチャに基づくシステム設計技術，WWW 情報システム言語処理技術ならびに情報システム開発方式の有効性の定量的な評価技術を明らかにすることにより，3 層WWW 情報システム開発方式を確立することである．

1.2 研究の背景

以下では，関連研究と本研究の位置付けを示す．

1.2.1 構造化分析設計技術

(1) 入出力データの依存関係に基づくデータフロー図の自動生成方式

Adler は，以下のような代数的な手法を用いて入出力関係行列Dからデータフロー図を生成することを提案した[19]．すなわち，まず互いに依存する入力 A と出力 X について，組（A→X）を要素とする式 L を構成する．次に，代数的な式の変換規則を用いて，入出力データが高々1 個しか出現しないように，L を最簡形 L'に変換する．この最簡形 L'に基づいてデータフロー図を生成する．

Adler の手法は，データフロー図を機械的に作成する上で有効であるが，変換規則の能力が不足しており，以下の点で必ずしも十分とはいえなかった．

(1) 変換規則の適用順序によって，異なる最簡形が生成されるため，それぞれの最簡形から作成されるデータフロー図が異なる．

(2) 入出力関係行列から変換規則を用いて生成できない最簡形が存在するため，正しいデータフロー図を作成できる入出力関係が限定されている．

(3) プロセス間で流れる内部データに対する依存関係を考慮していない．

このため，Adler の手法を拡張して内部データを考慮できるようにした代数式の変換系が提案されている[20][21]．

彼らの手法と本手法との基本的な違いは，次の点である．すなわち，本方式では，入出力関係行列を論理ベクトルの集合と見なし，論理ベクトルの比較演算により，データフロー図を生成するので，代数式の変換系で必要となるような，複雑で計算コストの高いパターンマッチング処理や与えられた入出力関係と生成したデータフロー図に対する入出力関係との一貫性の検査処理が不要となり，効率的である．

また，本方式では，内部データを含む場合への拡張にあたって，入出力データの直接的な依存関係と内部データと入出力データの依存関係が矛盾しないこと（適合性条件）を仮定している．一方，従来の内部データへの拡張方式では，この適合性条件を仮定していなかった．たとえば，従来手法では同じ内部データ S と関係する入力データ a と出力データ z があるとき，a と z は関係しない．ところが，この入出力関係行列から生成されるデータフロー図では，内部データを介さない場合と同様に，入力データ a から内部データ S を介して出力データ z に至るデータフローのパスがあるため，本論文では，内部データを外部データと区別して扱うのではなく，a と z が入出力関係行列上でも関係することとしている．

人手で作成したデータフロー図の場合，データフロー図で規定される入出力データ間の依存関係が明示的に記述されていないため，人手による入出力関係の確認が必要である．Modell は，システムの入力から始めて，すべてのデータフローを最終出力まで段階的に確認していく入出力確認法を示している[22]．入出力確認法では，入力データをデータフロー

に従って最終出力となるデータを見つけるまで追跡を続ける．もし，下位のデータフロー図に分割されているプロセスによって追跡中のデータが処理されていれば，そのプロセスの出力データを見つけるまで下位のデータフロー図上で追跡を続ける．

本論文で提案する入出力関係行列からデータフロー図を生成する方式を用いることにより，もし入出力関係行列に誤りがなければ，妥当なデータフロー図が生成されるので，このような人手による確認作業の必要はない．また，人手で作成された既存のデータフロー図がある場合については，本方式を応用することにより，入出力確認作業を自動化できる．すなわち，人手で作成された階層的なデータフロー図から入出力関係行列を導き，各入出力関係行列の適合性条件を検証することにより，入力と対応しない出力や，出力と対応しない入力を検出できる．

(2) データフロー図からのモジュール構造図の生成支援方式

構造化設計法には，データフロー図からモジュール構造図を作成する変換分析手法がある[23][24][25][26]．変換分析では，まずデータフロー図のプロセスを入力プロセス，変換プロセス，出力プロセスの 3 種に分類する．次に，変換プロセスの中から変換中心プロセスを選択する．変換中心プロセスが決まると，変換中心プロセスを最上位モジュールとし，変換中心プロセスに隣接するプロセスをその下位モジュールとする．モジュールに変換されたプロセスに対して，まだモジュールに変換されていないプロセスが隣接しているとき，そのプロセスを，隣接するプロセスから生成されたモジュールの下位モジュールとして変換する．この手順を繰り返すことにより，すべてのプロセスをモジュールに変換していく．

CASE ツールの中には，この変換分析手法に従ってモジュール構造図を作成できるものもある[27][28]が，設計者が作成するモジュール構造図と階層が深くなるなどの問題がある．しかし，この変換分析手法で作成されるモジュール構造図は，データフロー図の最長パスの長さに対応した階層の深さを持つのでモジュールの階層が深くなり易いという問題がある．これに対して，設計者が作成するモジュール構造図の階層の深さは浅いことが多い．したがって，変換分析手法で自動生成されたモジュール構造図を利用する場合，階層を浅くするなどの手直しが必要となる．また，データフロー図のプロセス間に複雑なデータフロー関係がある場合，従来の CASE ツールでは，データフロー関係をそのままモジュール間の呼び出し関係に変換するので複雑な構造を持つモジュール構造図が生成されるという問題もあった．

本論文では，プロセス間のデータフロー関係に基づいて変換分析で用いられる規則だけでなく階層を平坦化する変換規則を用いることにより，より自然なモジュール構造図を作成できる手法を提案する．本論文の評価実験では，提案するアルゴリズムで生成されるモジュール構造図では，約 70 から 80%のモジュールが設計者が作成するモジュールと一致する[29]という結果を得ている．

(3) データフロー図の復元方式

大規模ソフトウェア開発では，データフロー図とモジュール構造図に関する追跡性が重

要になる．データフロー図とモジュール構造図との間にどのような対応関係があるかを追跡する判断作業には大きな工数が必要になる．既存の CASE ツールでは，データフロー図とモジュール構造図との対応関係を予め設計者が記録したり，検索できる追跡機能を提供している．また， Benedusi らは既存のプログラムコードからデータフロー図をリバースエンジニアリングするための手法を提案している[30]．Bryne はこの手法を適用した実験結果を報告している[31]．この手法では，プログラムの変数間の依存関係から階層的なデータフロー図を作成する．しかし，生成されたデータフロー図には，同じ名称のプロセスが含まれるという問題がある．また，モジュール構造図のモジュールごとにデータフロー図のプロセスを作成するため，データフロー図に詳細な処理に対応するプロセスが出現するという問題がある．O'Hare は RE-Analyzer と称するリバースエンジニアリングツールを提案している[32]．プログラムコードから制御処理に基づいて階層的なデータフロー図を作成する．この場合の問題点は，制御処理に対応したプロセスを生成してしまうので，データフロー図に他のプロセスと互いにデータフロー関係を持たない孤立した制御プロセスが存在してしまうことである．

もし，モジュール構造図からシステム入力とシステム出力の依存関係を抽出することができれば，本論文で提案するデータフロー図の自動生成アルゴリズムを用いることにより，モジュール構造図に対応するデータフロー図をこの依存関係から自動的に作成できる[33][34]．この手法により，上述したようなデータフロー図に詳細処理に対応するプロセスが含まれるという問題を解決できる．

1.2.2 オブジェクト指向分析技術

これまでに，数多くのオブジェクト指向分析手法が提案されている．これらのオブジェクト指向分析手法は，オブジェクトとその関係に着目してオブジェクトモデルを作成する情報構造主導型とユースケースなどのシナリオに着目してオブジェクトモデルを作成する利用構造主導型に分けられる．

情報構造主導型では，現実世界を前提として，可能性のあるオブジェクトをすべて抽出してから，オブジェクトモデルを作成する．もし，現実世界が十分に具体的に定義できるなら，安定したオブジェクトモデルを作成できる．しかし，情報システムの場合，装置や物などの具体的に存在する物理的な対象が想定しにくいので，現実世界のオブジェクトを抽出しにくい．したがって，分析者が論理的に必要だと考えられるオブジェクトを個人的な経験から抽出する必要があり，分析者間でオブジェクトモデルの均質性を保ちにくいという問題がある．とくに，複数の分析者が同時に参加するような分析作業では，分析者間の意識合わせが困難になる可能性がある．また，分析段階のオブジェクトモデルと設計段階のオブジェクトモデルとが必ずしもうまく対応しないことが多い．代表的な情報構造主導型オブジェクト指向分析手法として OMT 法と SOM 法がある．

OMT 法[35]では，本論文で提案したようにオブジェクトを明確に分類していない．したが

って，オブジェクトモデルを作成する上での指針が必ずしも具体的でないため，情報システムの分析に適用する上で，分析者への負担が大きい．また，問題レベルのオブジェクトとソリューションレベルのアーキテクチャを表現するオブジェクトを区別して扱うような指針がないため，他の分析者が作成したオブジェクトモデルでは，論理的なアーキテクチャの理解や再利用が困難であるという問題がある．

Shared Object Modeling 法[36]は，情報システム分野へ容易に適用できるオブジェクト指向方法論として提案された．SOM 法では，ウィンドウオブジェクト，ユーザオブジェクト，共有オブジェクトからなる 3 階層のアーキテクチャを用いることにより，情報システムに関する見通しの良いオブジェクト指向設計を可能としている．ウィンドウオブジェクト，ユーザオブジェクト，共有オブジェクトを，それぞれ，表示層，機能層，データ層に対応づけることができる．SOM 法は設計レベルのオブジェクトモデルを作成するための手法であり，役割オブジェクトモデルを考慮していない．

利用構造主導型では，システム利用者と利用する上でのシナリオをすべて抽出してから，オブジェクトモデルを作成する．システム外部との境界条件がシナリオによって明確に規定されるので，外部境界と関連するオブジェクトについては，安定したオブジェクトモデルを作成できる．しかし，利用構造主導型では，情報システムで最も重要となる静的なオブジェクト構造を抽出しにくいという問題がある．代表的なオブジェクト指向分析手法として OOSE 法と OBA 法がある．

OOSE 法[37]の特徴は，ユースケースと呼ばれる問題分野の記述モデルを持つことと，オブジェクトを表示オブジェクト，制御オブジェクト，機能オブジェクトに分類していることである．エンドユーザの視点で作成されるという点で，役割オブジェクトモデルとユースケースの目的は同じであり，相補的に用いることができる．OOSE の 3 種のオブジェクト分類は，設計レベルのオブジェクトであり，パターン化オブジェクトモデルが対象とするオブジェクトよりも実装の観点からの分類である点が，提案したオブジェクト分類との違いである．

OBA 法[38]では，オブジェクトの役割に着目する．異なるオブジェクトがある同じ役割を持つ場合や，ひとつのオブジェクトが異なる局面で違う役割を持つことがある．たとえば，個人オブジェクトは，学生や従業員，顧客などさまざまな役割を持つことができる．このような問題分野の構造を柔軟に記述するためには，学生や従業員，顧客という役割を特定の個人とは独立に扱うことが必要になる．OBA の役割モデリングでは，サービスを利用するオブジェクトをイニシェータ，サービスを提供するオブジェクトをパーティシパントと呼ぶ．イニシェータがパーティシパントの提供するサービスを呼び出すことがアクションである．OBA では，まず，このような，イニシェータ，アクション，パーティシパント，サービスからなる 4 項組を用いて，システムの振る舞いをスクリプトとして列挙する．次いで，このスクリプトで抽出されたイニシェータやパーティシパントを候補としてオブジェクトを抽出し，オブジェクト間の関係を定義する．OBA では，スクリプトからオブジェ

クトモデルを抽出するプロセスでオブジェクトのパターン分類を考慮していない．したがって，オブジェクトモデルの均質性やオブジェクトの粒度に個人差が出る可能性がある．また，役割オブジェクトを洗練する手法として，抽象化，特殊化，統合，分解などをあげているが，イニシェータやパーティシパントをオブジェクト候補とするので，あらかじめスクリプト上でイニシェータやパーティシパントとして列挙されないトランザクションオブジェクトやデータオブジェクト等を OBA では抽出しにくいという問題がある．

本論文ではオブジェクト分類に基づくオブジェクト分析方式 POOM(Pattern Oriented Object Modeling)を提案する．POOM ではまず役割オブジェクト間の関係を表す有向グラフを用いて役割オブジェクトモデル図を作成する．役割オブジェクト間を接続する有向矢の始点がイニシェータ，終点がパーティシパントであり有向矢のラベルとしてアクションとアクションに付随する情報を付与する．次いで役割オブジェクトモデルに基づいて 3 層アーキテクチャに対応したオブジェクトモデルを規則的に作成する．

1.2.3 3 層クライアント／サーバ設計技術

これまでのクライアントサーバの 2 層モデルについては，Andleigh らによって，オブジェクト指向設計方法論が提案されている[38]．彼らの手法を要約すると，次のようになる．

(1) オブジェクト指向分析を行う
(2) 設計の最小単位となる最下層のオブジェクトとその属性を定義する
(3) オブジェクトの階層構造とグループ構造を定義する
(4) 階層構造に従って，オブジェクトの詳細な属性を定義する
(5) オブジェクト間の関係を定義する
(6) 各オブジェクトが提供するサービスを定義する
(7) 設計したオブジェクトを統合する

Andleigh らの手法では，オブジェクトモデルの設計を目的としており，3 層アーキテクチャや層間インタフェースの設計を考慮していない．これに対して，最近，Wegsheider, E. が ObjectQuest と呼ばれるアプリケーション開発環境を提案している[39]．ObjectQuest では，UI 層，ビジネス層，パーシステント層からなるオブジェクトによる 3 層アーキテクチャを対象としているが，このアーキテクチャは，本論文の表示層，機能層，データ層と一致する．ObjectQuest の開発環境では，GUI，リポジトリ，開発ツール，コード生成系，スキーマ生成系，アプリケーション実行系を提供している．Wegsheider の設計法では，まず，オブジェクトモデルを作成し，次いで UI 層，ビジネス層を設計する．このように，Wegsheider の設計法では，3 層アーキテクチャを考慮しているが，層間インタフェースを明示的に表現するための図式や層間インタフェースの設計指針を提供していない．Ning は，インタフェース定義に基づいたコンポーネントの組合せを定義できる ASL と呼ぶインタフェース定義言語を提案している[40]．しかし，3 層アーキテクチャに従ったコンポーネントを設計する上での具体的な設計指針については提案していない．イベント駆動型プログラミングのため

の設計指針を Philip が提案している[41]．この設計指針には，プログラムをイベント手続きの集合で実現するための指針として，イベント手続きのモジュール化法，イベント手続きでのデータ共有法などが述べられている．しかし，Philip の設計指針は，GUI プログラミングレベルの指針であり，3 層アーキテクチャを考慮していないという限界がある．

このように，情報システムのデータベースをアプリケーション・サーバでアクセスし，パソコン上の GUI により活用するための 3 層アーキテクチャを前提とした実務レベルの開発方法論は存在していない．

本論文で提案する 3 層アーキテクチャに基づく 3 層アーキテクチャに基づく WWW 情報システム設計法では，WWW 情報システムのページフロー図を作成し表示層の機能イベントを抽出する．次いでシナリオフロー図により表示層と機能層のインタフェースと機能層とデータ層のインタフェースを定義する．これによりシナリオフロー図に基づく明示的な層間インタフェースが定義できるだけでなく，各層のモジュールの独立性に基づく設計指針を提案している．また要求分析で作成したデータフロー図に基づくシナリオフロー図の作成手順とオブジェクトモデル図に基づくシナリオフロー図の作成手順も提案している．これにより要求分析と設計における生産物の対応関係が明確化でき生産物間の追跡性を向上できる．

1.2.4 WWW データベース連携ミドルウェア技術

WWW－データベース連携ミドルウェア研究の課題には，クライアント側の課題とサーバ側の課題がある．クライアント側の問題は，柔軟なユーザインタフェースを効率的に構築することである．たとえば，ヘルプ機能により HTML query form を自動生成するアプリケーションジェネレータの研究がある[42]．ただし，この方法では生成できるデータベース操作がアプリケーションジェネレータが提供するヘルプ機能の範囲に限定されるという問題がある．これに対して，サーバ側の研究には，種々の異なるデータベース・サーバを柔軟に接続する研究や性能の高いアプリケーションを効率的に構築する研究がある．異なるデータベース・サーバを連携する研究には，Infomaster[43]，OO-SQL wrapper[44]，DataWeb[45]などがある．Infomaster では，知識ベースを用いて，ODBC や WWW ページなどの差異を吸収している．OO-SQL では，サーバ側に異なるデータベースの違いを吸収するラッパを置き，モバイルクライアント側からは共通インタフェースを用いてこれらのデータベースを利用できる．DataWeb では，マルチメディア情報の階層的なカタログをデータベースで管理しておき，CGI で記述された QueryServer により WWW ブラウザから段階的に検索できる．

WWW－データベース連携アプリケーションの作成では，HTML 文，SQL 文などのデータベース操作の記述，制御構文を記述する必要がある[46]．本論文で提案する WebBASE Script は，HTML と SQL のようなデータベース操作言語の差違を吸収する言語の一つである．このようなアプリケーションロジックの記述方式には関数ライブラリ方式とテンプレート方式がある．

関数ライブラリ方式は HTML 文やデータベース操作文を既存のプログラミング言語の関数として用意することにより，従来のプログラミングの延長で Web データベース連携アプリケーションを作成する方式である．関数ライブラリ方式の製品事例として，Oracle 社の WebServer[47]や Netscape Communications 社の LiveWire Professional[48]などがある．WebServer では，Oracle のストアドプロシージャの記述言語である PL/SQL の関数ライブラリとして，HTML のタグを生成する HTF 関数と HTML の結果を出力する HTP 関数を提供する．LiveWire Professional では，Java script の外部関数として Open DataBase Connectivity(ODBC) とともに Informix, Sybase, Oracle などのデータベース操作インタフェースを提供する．関数ライブラリ方式の利点は，プログラミング言語をベースにしているので構文上の制約が少なく柔軟性が高いことである．しかし，アプリケーション開発の面では，HTML そのものではなく HTML を生成する関数を学習する必要があること，ベースとするプログラミング言語の学習が必要となることなどから，テンプレート方式に比較して生産性が高いとは言えない．とくに，WebServer の場合，データベース管理システムも Oracle だけに限定される．

テンプレート方式は HTML 文，SQL 文，制御構文を，一つのスクリプト言語でまとめて記述する方式である．テンプレート方式の製品事例として，Microsoft 社の IDC(Internet Database Connector)[49]，O'Reilly 社の Cold Fusion [50]Sybase 社の Web.sql [51]などがある．テンプレート方式の利点は，一つのスクリプト言語だけで WWW－データベース連携アプリケーションを作成できるので，従来のプログラミング言語を用いるよりははるかに生産性が高いことである[52]．しかし，テンプレートで利用できる構文や変数に制約がある場合，必ずしも柔軟にアプリケーションロジックを記述できないという問題がある．このような制約の例として，IDC では，一つのテンプレートでは複数の SQL 文を記述できないこと，Cold Fusion では，複数の SQL 文を記述できるが IF ELSE 構文を使えないなどがある．また，IDC の場合，UNIX サーバの上で動作できないので，アプリケーションの可搬性に限界がある．この他に，HTML ファイルとは別に SQL 文を別ファイルに記述する方式も提案されている[53]．しかし，この方式では，大規模システムを開発する場合，ファイルの管理が煩雑になるだけでなく，セションを管理できない，特定の RDBMS に限定した方式である，異なるページ間で変数を共有できないという問題もあり，試作レベルにとどまっている．

本論文では，HTML を拡張して SQL や簡易言語で記述されたプログラムをテンプレート方式で実行できるスクリプト言語機能とセッション処理機能を持つ WebBASE を提案する．また WebBASE の具体的な適用事例ならびにその評価結果について述べる． WebBASE は NTT ソフトウェア株式会社に技術開示され，同社が 1996 年に出荷を開始してからこれまでに約 740 システム以上に適用されており WWW 情報システム構築製品として市場からも高い評価を得ている．

1.2.5 情報システム開発方式の実証的な評価技術

ソフトウェア開発方式を実証的に評価する方法にはオピニオン評価法と実験評価法がある．オピニオン評価法はアンケート項目を用意しておき新手法を適用した開発者の意見をできるだけ客観的に抽出することにより開発手法の有効性を評価する手法である．

実験評価法には新手法による生産性や品質などを従来手法における標準値と比較する手法と，新手法を使用する開発チームと従来手法を使用する開発チームに分けそれぞれの結果を比較する制御実験法の 2 つがある．標準値を比較する手法では従来の標準値と新手法との比較でしかないため従来手法を用いた場合と実験条件が必ずしも一致しないという欠点がある．厳密な比較を行うためには同一の課題に対してソフトウェア開発の実験条件を制御して一致させる制御実験が必要である．制御実験法を用いてソフトウェア開発手法を定量的に評価した研究はごくわずかしかない．たとえば構造化分析設計手法に基づくソフトウェア開発支援ツールの定量的に評価した研究として岡らの研究がある[54]．この研究はできる限り同一条件の下で構造化手法と従来手法とを比較する実験を実施し，仕様，品質，工数，生産性などを比較分析している．

これまでにオブジェクト指向分析設計手法の有効性を比較した研究には，Jacobson の研究[55]と Wirfs-Brock の研究[56][57]がある．Jacobson の研究は，定性的な比較論に留まっており，具体的，定量的な研究ではない．これに対して Wirfs-Brock は，ビール醸造プロセスの制御問題に対して作成された情報構造主導型と利用構造主導型のオブジェクトモデルについて，各種の設計メトリクスを用いて定量的に比較している．Wirfs-Brock は，Shlaer/ Mellor 手法の経験者 1 名と RDD 手法の経験者 1 名が同じ問題に対して作成したオブジェクトモデルを比較しているだけであり，問題数，被験者数の点でも十分な比較評価であるとは言えない．また，設計段階のオブジェクトモデルを対象としており，要求分析段階のオブジェクトモデルについての評価ではない．

本論文では，以上述べた理由から，POOM と OMT を複数人を被験者として 3 層アーキテクチャによる情報システム分野について複数の問題に対して制御実験法を用いて定量的に比較することにより，分析効率，オブジェクトモデルの均質性，階層型アーキテクチャとの親和性，オブジェクトモデルの妥当性，習熟性などに対する評価尺度を用いて POOM が OMT よりも優れていることを定量的に明らかにする．

ソフトウェア開発方式の評価では生産性や品質に関するデータを評価するだけでなく，生産物の内容を比較して評価する必要がある．ソフトウェア生産物の自動生成方式と開発者による生産物の比較評価では，生成率と適合率とを用いた評価法を提案する．ここで生成率とは開発者が作成した生産物の構成要素をどれだけ自動生成できたかを示す比率である．適合率は自動生成した生産物に含まれる構成要素のうち開発者が作成した構成要素がどれだけ存在するかを示す比率である．生成率が高ければ開発者が作成する生産物を自動生成できる確率が高くなる．適合率が高ければ自動生成した結果と開発者の作成した生産物とが一致する確率が高くなる．生成率が高くて適合率が低い場合，自動生成した結果に

は開発者が意図しない構成要素が含まれるのでそれらを除去する必要がある．適合率が高くて生成率が低い場合，自動生成結果には高い確率で開発者の意図する構成要素が含まれているが，本来必要な構成要素が欠落している可能性があるのでそれらを補う必要がある．生産物の生成率と適合率の高いソフトウェア開発手法は属人的要素を除去できている可能性が高いからソフトウェア開発過程の定式化の完全性も高いと考えられる．

最も効果の高い評価法は制御実験法であるが実施するためのコストが高いため，評価対象に応じて評価法を工夫する必要がある．

1.3 本研究の内容

本研究では，WWW と 3 層アーキテクチャを用いた分散型情報システム開発方式を構成する次の 4 つの技術とその実証的な評価技術を提案する．

（1） 入出力データの依存関係に基づく構造化分析技術
（2） 役割関係に基づくオブジェクト分析技術
（3） 3 層アーキテクチャ設計技術
（4） WWW 情報システム構築技術
（5） 実証的な評価技術

1.3.1 要求分析技術

（1） 代数的な変換系により，システム入出力の依存関係からデータフロー図を作成する手法が提案されている．この手法では，システム入出力の依存関係と代数式との相互変換ならびに，生成されたデータフロー図と依存関係との一貫性の検証が必要だった．本論文では，プロセス数が最小なデータフロー図を入出力関係から一意的に生成できる決定論的なアルゴリズムを提案する．

（2） モジュール構造図からデータフロー図を復元する場合，モジュール構造図に含まれる詳細処理がデータフロー図に含まれてしまうという問題がある．本論文ではモジュール構造図から抽出されたシステム入出力の依存関係に基づいてデータフロー図を作成することにより論理的で最小なデータフロー図を作成する方式を提案する．

（3） 従来のオブジェクト指向分析手法では作成されたオブジェクトモデルの均質性やオブジェクトの粒度に差が生じやすいという問題があった．本論文ではオブジェクトの型を分類しておき役割関係からオブジェクトモデルを系統的に作成する方式を提案する．オブジェクトの型が 3 層アーキテクチャに対応しているので 3 層アーキテクチャを用いたシステム開発における分析手法として有効である．

1.3.2 設計技術

（1） 従来から WWW を用いたハイパーメディアシステムの設計図式として Bichler らにより SHDT 図が提案されている[58]．しかし WWW ページ間のハイパーリンクとアプリケーシ

ョンを起動するための機能的なイベントを区別していない点や処理結果に応じて異なるページへの遷移を記述できないなどの点で問題がある．本論文では Bichler らの WWW ページ設計図式を拡張したページフロー図により 3 層アーキテクチャに基づく WWW 情報システムの表示層インタフェースを作成する手法とページフロー図の記述規則を提案する．

(2) 3 層アーキテクチャを構成する表示層，機能層，データ層の処理と層間インタフェースを定義するためのシナリオフロー図を提案する．次いでページフロー図とシナリオフロー図に基づいて 3 層アーキテクチャに基づく情報システムを作成する手順を提案する．本手法により要求分析モデルと 3 層アーキテクチャを構成する各層のコードとの対応を明確化できるだけでなく，各層のコード間のインタフェースを製造工程に先立って定式化できるので開発上の重複や手戻りを抑止できる．

(3) 構造化設計法では，変換分析手法に従ってデータフロー図からモジュール構造図を作成する．しかし，変換分析手法に従って作成されたモジュール構造図は，データフロー図のデータの流れに従ってモジュールを配置するため，階層が深くなりやすく，設計者が作成するモジュール構造図と必ずしも一致しないという問題がある．本論文では，設計者が作成するモジュール構造図に近い形でモジュール構造図を生成するアルゴリズムを提案する．

(4) 従来の構造化設計手法では 3 層アーキテクチャへの階層分割手法を考慮していないと言う問題があった．このため構造化手法で開発された情報システムを 3 層アーキテクチャへの移行法が明らかではなかった．本論文ではモジュール構造図に基づくシナリオフロー図作成法を提案する．

(5) 従来は要求分析段階では 3 層アーキテクチャを考慮しておらず，コード化段階ではじめて 3 層にプログラムを分割していたため，層間への処理の配置を検討する上での負担がコード化工程に集中するだけでなく，要求分析とコードとの間での追跡性が低いという問題があった．本論文ではパターン化オブジェクトモデルに基づくシナリオフロー図の作成法を提案し，3 層アーキテクチャに基づくオブジェクト指向型の情報システム設計法を具体化する．本手法により要求分析工程から段階的に 3 層アーキテクチャ設計を進めることができるので作業上の負荷を各工程に平準化できるとともに工程生産物間の追跡性を向上できるという特徴がある．

1.3.3 WWW 情報システム構築技術

(1) 従来は WWW を用いて情報システムを作成する場合，HTML だけでなく C 言語や shell スクリプトを用いて業務処理ごとにデータベースと情報ページ間の連携処理を作成していたので開発効率が低下するという問題があった．本論文では HTML，SQL，手続き処理を統合して記述できるスクリプト言語を提案することによりこの問題を解決した．

(2) WWW のプロトコル(HTTP)ではページ毎に通信の接続／切断が行われるため，複数ページに跨ってデータベースのトランザクション制御を行うことができないという問題があった．本論文ではセッション管理機能を持ちデータベースと連携できる WWW プロセス常駐方式を開発することによりこの問題の解決を図った．

1.3.4 システム開発方式の実証的な評価技術

(1) モジュール構造図からのデータフロー図を復元する方式の有効性を人手による復元結果と比較することにより評価する．自動的に復元されたデータフロー図の有効性を評価する一つの方法として，内容と被験者を層別化して各層ごとに内容を比較する手法を考案し，評価手法を用いてモジュール構造図の例に対する構造化分析の初心者と経験者によるデータフロー図の復元結果を自動的な復元方式と比較した結果について述べる．

(2) 情報構造主導型方法論である OMT と，本論文で提案する役割オブジェクト分析に基づく利用構造主導型方法論 POOM について，両者の有効性を実証的に評価する．実際のシステム分析はチームによる協調作業となることが多いので，個人によるオブジェクトモデル分析だけではなく，チームによるオブジェクトモデル分析の結果を比較する．この場合，異なる分析者間で均質性の高いオブジェクトモデルを作成できるかどうかが重要となるので，多数オブジェクト率という新しい評価指標を考案し，この指標に基づいて，OMT と POOM についてオブジェクトモデルの均質性を比較した結果について明らかにする．

(3) データフロー図からのモジュール構造生成方式の有効性を人手による設計結果と比較することにより評価する．自動生成されたモジュール構造図の妥当性の評価尺度としてモジュールの生成率と適合率を定義し，3 つの事例に対して自動生成されたモジュール構造図に関する評価結果を述べる．

(4) 3 層システム設計法の有効性を実際のシステム開発への適用結果に基づいて評価するための一つの手法を提案する．具体的にはシナリオフロー図に出現するモジュール数やデータベース設計における実体数ならびに層間インタフェースの個数を評価尺度としてインタフェースの共通化率を評価する．また，3 層システム設計法を適用した 3 つの事例に対する評価結果を述べる．

(5) WWW 情報システム開発者へのアンケート調査に基づいて，本論文で提案する WebBASE による WWW 情報システム開発の有効性を従来型の情報システム開発と比較することにより，開発言語の種類，開発規模と期間，開発工数などの観点から評価する．

第 2 章以降で具体的な研究の内容とその結果を述べる．まず第 2 章で WWW と 3 層アーキテクチャを用いた分散型情報システム開発方式の課題を整理し本研究の背景と位置付けを明らかにする．

第 3 章では 3 層設計の前段で必要となる要求分析法を提案する．第 3 章の構成は，次の通りである．3.2 では，入出力の依存関係に基づくデータフロー図の自動生成方式を述べる．3.3 では，モジュール構造図からのデータフロー図の復元方式を述べる．3.4 では，役割オブジェクト分析に基づくオブジェクトモデルの作成手法 POOM を提案する．

第 4 章では，シナリオフロー図と呼ぶ層間インタフェースを定義できる図式を考案し，3 層クライアント／サーバアプリケーションを手順的に設計するための設計法を提案する．第 4 章の構成は次の通りである．4.2 で WWW 情報システムの表示層を構成する情報ページの流れを記述するためのページフロー図の作成法を述べる．4.3 ではシナリオフロー図について述べる．4.4 では，データフロー図からモジュール構造図を自動生成する方式を述べる．4.5 でモジュール構造図に基づくシナリオフロー図の作成方式を述べる．4.6 でオブジェクトモデルに基づくシナリオフロー図の作成方式を述べる．

第 5 章では，3 層 WWW 情報システムをスクリプト言語で効率的に構築するためのミドルウェアを提案する．第 6 章では，本論文で提案する 3 層 WWW 情報システム開発方式を構成する分析・設計・構築技術の評価手法とその評価結果について述べる．

第 6 章では，ソフトウェア開発方式の実証的な評価技術について述べる．6.2 では，モジュール構造図からのデータフロー図の復元方式の有効性を人手による復元結果と比較することにより評価する．6.3 では，POOM の有効性を評価するために実施した POOM と OMT との比較実験計画について述べる．まず 6.4 で POOM と OMT に対する個人による要求分析実験の結果を比較し POOM の有効性を明らかにする．次いで 6.5 で POOM と OMT に対する複数人からなるチームによる要求分析実験の結果を比較し POOM の有効性を明らかにする．6.6 では，データフロー図からのモジュール構造生成方式の有効性を人手による設計結果と比較することにより評価する．6.7 では，3 層システム設計法の有効性を実際のシステム開発への適用結果における層間インタフェースの共通化率に基づいて評価する．6.8 では，情報システム開発者へのアンケート調査に基づいて WebBASE による WWW 情報システム開発の有効性を従来型の情報システム開発と比較することにより評価する．最後に第 7 章で結論を述べる．

第 2 章　3 層アーキテクチャに基づく WWW 情報システム開発の特徴と課題

2.1 まえがき

企業情報システムの開発では 1990 年代の前半はクライアントサーバシステムが登場しメインフレーム型の企業情報システムのダウンサイジングが進展した．また 1995 年以降からインターネットの普及に伴い企業における WWW を用いた情報システムの利用が大きく発展した．

本章では，クライアント／サーバシステム開発や WWW 情報システム開発の特徴と技術的な課題について述べ，その解決策として 3 層 WWW 情報システム開発方式を提案する．

2.2　3 層アーキテクチャ

ハードウェアとネットワークの飛躍的な性能向上を背景として，分散処理によるアプリケーションの開発が進んでいる．1 台の大型コンピュータだけですべてのアプリケーションを実行する集中処理に対して，複数のコンピュータをネットワークで接続してアプリケーションを処理する方式が分散処理である．異なるコンピュータによって連携して処理されるアプリケーションを分散アプリケーションという．分散処理の利点としては，複数のコンピュータが仕事を分担するので，集中処理に比べて性能を向上できること，大型コンピュータによる集中処理の代わりにパソコンやワークステーションの組み合わせによる分散処理ではシステム構築経費を大幅に削減できることなどがある．

代表的な分散処理方式にはクライアント／サーバ方式がある．クライアント／サーバ方式では，データベースなどの資源を管理するサーバ処理と，その資源を用いて仕事を実行するクライアント処理とをネットワークで接続して情報処理を行う．サーバ上には複数のクライアントから利用できるサービスを配置する．利用者が端末上のクライアントからネットワークを通してサーバ処理を呼び出すことにより，サーバ上の様々な資源を利用できる．サービスの依頼者がクライアントで，サービスの提供者がサーバである．

このようなクライアント側とサーバ側に配置するアプリケーションの組み合わせにはいくつかの種類がある．小規模な分散システムの開発では 2 層アーキテクチャが用いられた．この 2 層アーキテクチャでは，GUI（Graphical User Interface）製品により，表示処理と業務処理をクライアント側で作成し，サーバ上のデータベースを SQL(Structured Query Language) によりアクセスする．SQL はリレーショナル・データベースにおけるデータの定義，操作のための構造化照会言語である．

2 層アーキテクチャの利点として，次が挙げられる．

(1) GUI によるユーザフレンドリな操作が可能である

(2) 単純なアーキテクチャであるため，小規模な分散アプリケーションの構築が容易である

(3) GUI 製品を用いたエンドユーザプログラミングによりアプリケーションの開発期間を

短縮できる

しかしながら，従来のクライアントサーバシステム用の開発環境は小規模システムを対象としているため，大規模なクライアント／サーバシステム開発に対して，これらの開発環境を適用する場合には，次のような問題があった．

(1) ビジネスフローの変更などを伴う業務仕様の拡張に対応したアプリケーションの変更が困難である．

(2) 業務量の増加に伴うデータの増加や処理能力の向上が困難である．

(3) 他システムとの情報流通が困難である．

この原因として，次の3点が考えられる．

(1) クライアント／サーバシステム開発のための分析・設計方法論が確立していない．

(2) クライアントからサーバ上のデータベースを操作するたびにSQLのための通信が必要になるので通信処理の負荷が高いことや，データベース・サーバに処理の負荷が集中するなどの性能上の問題がある

(3) アプリケーションプログラムの画面制御と業務機能とSQLによるリモートデータベース操作をクライアント側のアプリケーションで渾然一体として実現する構成をとっている．したがって，わずかな仕様変更がシステム全体に大きな影響を与える可能性が大きい．

(4) システムを構成するツールが，データベース管理システム，OS，ハードウェアなどに強く依存した仕様であるため，他の環境との親和性が高くない．たとえば，異なるデータベース管理システムやOSとの接続のために新たなプログラム開発が必要になることなどが多い．

2層アーキテクチャが持つこのような問題に対して，Donovanにより提唱された3層クライアント／サーバ・アーキテクチャ[1]が注目された．このアーキテクチャでは，クライアント／サーバシステムを，次のような表示層，機能層，データ層から構成する．

(1) 表示層では，マンマシンインタフェースを提供する．

(2) 機能層では，表示層とデータ層との接続ならびに業務処理機能を提供する．

(3) データ層では，データベース管理システムとの接続ならびに，既存システムとの接続を含むデータサービスを提供する．

このように企業情報システムでは2層アーキテクチャの拡張として3層アーキテクチャが提案されたが，3層アーキテクチャの類似概念として以下のようなアーキテクチャがこれまでにも提案されている．

(1) Wiederholdの階層的DB統合アーキテクチャ[2]は，異種分散データベースを統合するための階層的なアーキテクチャである．階層的DB統合アーキテクチャでは，ユーザ層，メディエータ層，データベース層からなる3階層アーキテクチャで分散データベースを連携させている．このアーキテクチャでは，ユーザ層からの検索要求に対応するデータベースをメディエータ層が選択してデータベースに対する具体的な検索要求を構成する．この具体的な検索要求に対するデータベース層からの検索結果をメディエータ層が得ると，各データベースからの結果を合成してユーザ層に提示する．3層アーキテクチャと対比すれば，ユーザ層が表示層，メディエータ層が機能層，データベース層がデータ層に対応すると考えられる．このアーキテクチャは，分散協調エージェントのためのアーキテクチャとしても用いられている．

(2) SmallTalkにおけるMVCモデルは，モデル，ビュウ，コントローラと呼ばれるオブジェクトから構成される対話型システムのオブジェクト指向型アーキテクチャである．3層アーキテクチャと対比すれば，モデルがデータ層，ビュウが表示層，コントローラが機能層に対応すると考えられる．

(3) OMT[3]における多面的オブジェクトモデルでは，データモデル，状態モデル，機能モデルからオブジェクト指向分析を行う．3層アーキテクチャと対比すれば，データモデルがデータ層，状態モデルが表示層での入力イベントに対する制御処理，機能モデルが機能層の機能に対応すると考えられる．また，Jacobsonのステレオタイプ[4]（OOSE）は，境界オブジェクト，制御オブジェクト，エンティティ・オブジェクトにオブジェクトを分類している．3層アーキテクチャと対比すれば，境界オブジェクトが表示層，制御オブジェクトが機能層，エンティティ・オブジェクトがデータ層に対応すると考えられる．

(4) WWWによる3層アーキテクチャでは，分散アプリケーションを，ブラウザ，WWWサーバ，APサーバから構成する．APサーバからデータベースや基幹業務システムをアクセスする場合には，これらを別の階層としてとらえれば4層アーキテクチャであると考えられる．WWWによる3層アーキテクチャが対象とするのは，クライアントやサーバの物理的な構成であることに注意する必要がある．3層アーキテクチャと対応付けると，WWWサーバに蓄積されたHTMLファイルをWWWブラウザにダウンロードして処理することから，WWWブラウザとWWWサーバが表示層，APサーバが機能層，データベースや基幹業務システムがデータ層に対応すると考えられる．

(5) インターネット上のECでは，買い手と売り手をブローカが仲介するオープンな取引モデルがオークションなどで利用され始めている．3層アーキテクチャと対応付けると，買い手が表示層，ブローカが機能層，売り手がデータ層に対応すると考えられる．

これらの概念と3層アーキテクチャの共通点をまとめると，サービスの利用者としての第1層と基本的なサービスの提供者としての第3層を中間に位置する第2層が中継あるいは連携することによって各層の独立性を高めシステムの柔軟性や拡張性の向上を図ることができる点である．このように多様な分野で3層アーキテクチャに類似した設計が提案されていることから，3層アーキテクチャに基づいて分散システムを設計することにより開発された分散システムの柔軟性や拡張性を向上できると考えられる．また，分散システムのアーキテクチャを構成する階層の段数が3層以上の場合を考えてみよう．この場合，分散システムを構成する利用者が属する階層と，データの提供者やサービス提供者が属する階層の2つが両端にあり，3層以上の階層は両端の2つの層を複数段で中継するための階層であると考えられる．このような多階層アーキテクチャは機能層を複数階層に拡張したもの

であり，基本的には 3 層アーキテクチャに帰着させることができると思われる．したがって，分散システムにおける 3 層アーキテクチャの一般性は高いと言える．

3 層アーキテクチャを企業情報システムに適用することにより以下のような効果が期待できる．

(1) 3 層をシステム構成上明確に分離し，各層間に共通のインタフェースを設定することにより，システムを再利用性の高い独立の部品群から構成できる．換言すれば，これらの部品を組み合わせてアプリケーションを構成することにより，改造が容易なシステムを構成できる．したがって業務量の増加や機能追加に際して，影響範囲を各層ごとの部品に限定し易いので，比較的容易にシステムを拡張できる．また業務処理とデータベース処理との独立性が高いので保守性が向上する．

(2) データ層に既存の外部システムとの通信機能を配置することにより，外部システムとの等価性の高い接続ができる．したがって，メインフレームシステムからクライアント／サーバシステムへの柔軟なシステム更改ができる．

(3) 3 層アーキテクチャの性能面での効果としては，クライアントとサーバ間の通信を RPC（Remote Procedure Call）で行うことにより，大規模な分散システムの構築が可能になったことが挙げられる．RPC では，同一のコンピュータの中で行われる手続きの呼び出しと同様に，ネットワークを介して接続された他のコンピュータ上の手続きを呼び出すことができる．3 層アーキテクチャでは，クライアントとサーバの間で SQL 処理を行わず，クライアントからサーバ上の処理手続きを RPC で呼び出し，このサーバ上の手続きの中からデータベースを操作するための SQL 処理を行う．したがって，クライアントとサーバ間では，このサーバ上の手続きに対する RPC 処理依頼とその結果通知だけが通信されるので，クライアントとサーバで SQL 通信を行う 2 層アーキテクチャに比べて通信の負荷が大幅に軽減できる．

2.3 WWW を用いた情報システム[5][6][7][8]

WWW（World Wide Web）を用いた分散アプリケーションによりこれまで企業内に閉じていたネットワークが一般消費者にまで接続範囲が拡大されるようになってきた．これにより企業の外部との連携を容易化できるので，インターネットという新しいマーケットに容易に進出できるだけでなく，WWW（WWW ページを閲覧するために端末上で動作するプログラム）を用いたビジュアルなインタフェースにより顧客への商品案内や問い合わせ応対などのサービスを高度化できる効果がある．インターネット技術を用いた企業情報システムはイントラネットと呼ばれている．イントラネットでは端末としてのパソコン上で動作する WWW ブラウザから，企業内ネットワークで端末と接続されているサーバ上の企業情報システムに HTTP（Hyper Text Transfer Protocol. WWW ブラウザと WWW サーバ間で HTML 情報を通信するためのプロトコル）でアクセスする．クライアントの表示層が WWW ブラウザで統一されると，WWW サーバ上に構築された企業情報システムを WWW ブラウザで

自由にアクセスできるようになった．表示処理を変更する場合，サーバ上の HTML を変更し必要に応じてブラウザのプログラムをサーバからクライアントに転送することにより対応できるため，クライアントへのソフト配布作業を軽減できるようになった．これに対して 3 層アーキテクチャの場合には，クライアント側に表示層のアプリケーション開発が必要になる．したがって，表示層のアプリケーションを変更する場合，すべてのクライアントに対して表示層のアプリケーションの入れ替え作業が発生するため，運用面や管理面での作業が必ずしも容易ではないという問題があった．また，部門ごとに異なるクライアント／サーバ製品を利用している場合，クライアントとサーバ間の通信プロトコルが異なるため，異なる部門システム間で情報共有できないといった問題もあった．イントラネットでは，表示層が WWW で統一されたことにより，表示層のアプリケーション開発が不要になった．また，異なる部門間での情報共有も HTML 文書を WWW で統一的に閲覧するという形で自然に実現できるようになった．以上述べた分散アプリケーションのアーキテクチャの発展段階をまとめると図 2−1 のようになる．

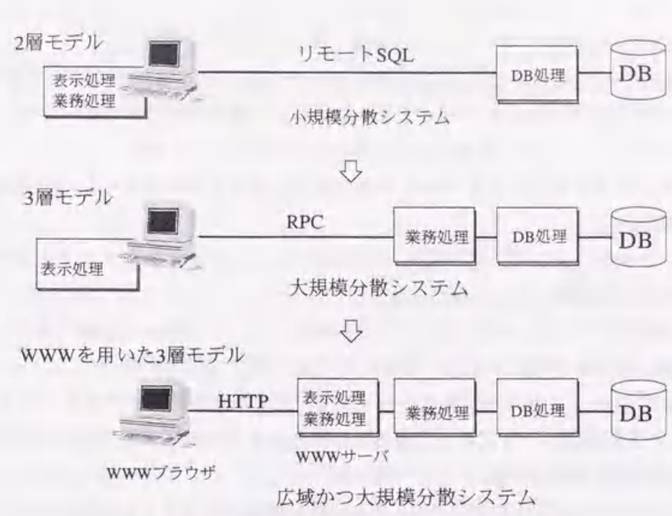


図 2−1　分散アーキテクチャの発展

2.4 3 層アーキテクチャを用いた WWW 情報システム開発の課題

以上述べたように 3 層アーキテクチャと WWW の利用により大規模な企業情報システム構築の可能性が見えてきたといえる．しかしながら，実際に 3 層アーキテクチャに従った WWW 情報システムを構築するためには次のような課題がある[9][10][11][12]．

(課題 1) 複数の担当者で同時並行的に分析設計工程の作業を実施する場合には抽出される 3 層アーキテクチャの分析情報や設計情報の重複や抜けなどが発生する可能性がある

(課題 2) クライアントとサーバ間にシステムが階層分割されるため階層間で設計の重複が発生する可能性がある

(課題 3) 既存システムを再利用して 3 層アーキテクチャ型の WWW 情報システムへ移行するための手順が明確になっていない

(課題 4) HTML だけではデータベース処理や手続き処理を記述できないため，HTML で表示層処理を記述し C 言語などのプログラミング言語を用いて企業情報システムに必要な機能層やデータ層の処理を実現する必要があった

(課題 5) WWW で用いられる HTTP 通信では，個々の情報ページの送信ごとに通信を切断してしまうため，異なる情報ページにまたがってデータを引継ぐことができないため，初期の企業情報システムへの WWW の導入は，情報検索や情報周知などの情報共有などに限定される．

これらの問題を解決するためには，

(1) 3 層アーキテクチャ型情報システムの分析工程を既存システムからの移行を考慮して定式化し

(2) 分析工程から設計工程へ段階的な手順を確立するとともに

(3) 3 層アーキテクチャにおける各階層の処理を記述できるスクリプト言語ならびに WWW ブラウザを表示層として利用できるセッション管理機能を持つミドルウェアを提供することにより

(4) 3 層アーキテクチャによる WWW 情報システム構築を容易化するための開発方式を確立する必要がある．

2.5 3 層 WWW 情報システム開発法

本研究で提案する 3 層 WWW 情報システム開発方式では，図 2−2 に示すようにして 3 層 WWW 情報システムの分析・設計・構築を進める[7][8][11][12][13][14][15][16][17][18]．

3 層 WWW 情報システム開発方式の各工程で作成する生産物は次のようになる．

(1) データフロー図[20]　システム仕様をプロセスと各プロセスの入出力データおよびその流れに基づいて定義する図式

(2) オブジェクトモデル図[15][19]　システム仕様を類型オブジェクトと類型オブジェクト間の関係に基づいて定義する図式

(3) モジュール構造図[16][21]　システム構造をモジュールとその利用関係に基づいて定義する図式

(4) シナリオフロー図[11][12][13][19]　3 層アーキテクチャを構成する表示層，機能層，データ層の各層処理と処理間のデータの流れを定義する図式

(5) ページフロー図[13][19]　WWW の情報ページ構造を類型化された情報ページ間の遷移により定義する図式

(6) WebBASE スクリプト[7][8][18]　WWW 情報システムを HTML 文と処理手続き文および SQL 文で記述するためのスクリプト言語

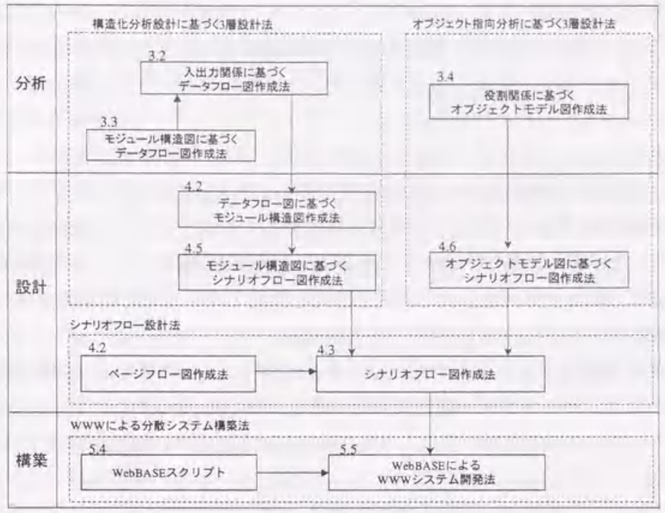


図 2−2　3 層 WWW 情報システム開発方式

本研究で提案する 3 層 WWW 情報システム開発方式の特徴は次のようである．

［特徴 1］分析工程でオブジェクトを表示層，機能層，データ層に分類しておき段階的な 3 層設計ができる（課題 1 の解決）

［特徴 2］シナリオフロー図に基づき各階層間のインタフェース設計を共通化できる（課題 2 の解決）

［特徴 3］既存システムのモジュール構造図に基づきデータフロー図を自動生成しておき，データフロー図から段階的にシナリオフロー図を設計できる（課題 1 および課題 3 の解決）

［特徴 4］WebBASE スクリプトにより HTML 文と処理手続き文および SQL 文からなる WWW 情報システムを記述できる（課題 4 の解決）

［特徴 5］WebBASE 処理系により異なる情報ページ間でデータの引継ぎができる（課題 5 の解決）

データフロー図とオブジェクトモデル図の作成法については第 3 章で明らかにする．第 3 章ではデータフロー図については入出力関係に着目することにより自動生成できることを

示す．またオブジェクトモデル図については系統的な作成手順を示す．モジュール構造図とシナリオフロー図およびページフロー図の作成法については第4章で明らかにする．第4章ではページフロー図とシナリオフロー図の作成手順を示す．またデータフロー図に基づいてモジュール構造図を自動生成できることを示す．またモジュール構造図からのシナリオフロー図の作成手順を示す．さらにオブジェクトモデル図からのシナリオフロー図の作成手順についても明らかにする．第5章ではWebBASEスクリプトの作成法について明らかにする．

2.6 まとめ

本章では，最近の企業情報システムで急速に適用が進展しつつある3層アーキテクチャとWWW情報システムの有効性と問題点を明らかにすることにより企業情報システムの発展段階を示した．また3層アーキテクチャを用いたWWW情報システムの課題を明らかにした．次いで3層WWW情報システム開発方式を提案することにより3層WWW情報システムの課題に対するひとつのアプローチを示した．

3層WWW情報システム開発方式では，構造化分析およびオブジェクト指向分析に基づく段階的な3層アーキテクチャ設計が可能であり，WWWページのセッション管理機能ならびにスクリプト言語処理系からなるWebBASEによる3層WWW情報システムの構築が可能である．

第3章　要求分析技術

3.1 まえがき

(1) 代数的な変換系により，システム入出力の依存関係からデータフロー図を作成する手法が提案されている．この手法では，システム入出力の依存関係と代数式との相互変換ならびに，生成されたデータフロー図と依存関係との一貫性の検証が必要だった．本論文では，プロセス数が最小なデータフロー図を入出力関係から一意的に生成できる決定論的なアルゴリズムを提案する．

(2) モジュール構造図からデータフロー図を復元する場合，モジュール構造図に含まれる詳細処理がデータフロー図に含まれてしまうという問題がある．本論文ではモジュール構造図から抽出されたシステム入出力の依存関係に基づいてデータフロー図を作成することにより論理的で最小なデータフロー図を作成する方式を提案する．

(3) 従来のオブジェクト指向分析手法では作成されたオブジェクトモデルの均質性やオブジェクトの粒度に差が生じやすいという問題があった．本論文ではオブジェクトの型を分類しておき役割関係からオブジェクトモデルを系統的に作成する方式を提案する．オブジェクトの型が3層アーキテクチャに対応しているので3層アーキテクチャを用いたシステム開発における分析手法として有効である．

本章の構成は，次の通りである．3.2では，入出力の依存関係に基づくデータフロー図の自動生成方式を述べる．3.3では，モジュール構造図からのデータフロー図の復元方式を述べる．3.4では，オブジェクトモデル分析方式を述べる．

3.2 入出力関係からのデータフロー図の自動生成

ソフトウェアの要求仕様を作成するための構造化分析手法[1][2][3]では，データの流れに着目してデータフロー図を階層的に作成する．ところが，上位のプロセスの入力データと出力データから下位のデータフロー図を作成する手順が必ずしも明確ではなかった．このため，Adlerが，以下のような，データフロー図の自動生成法を提案した[4]．すなわち，1) 与えられた入出力データ間の依存関係を行列で表現し，2) この関係行列に対する代数式を変換規則を用いて最簡形へ変換することにより，3) 最簡形の代数式に対してデータフロー図を生成する．

しかし，Adlerが提案した方法では，適用すべき変換規則が一意に定まらない，変換規則の適用順序に依って変換結果が異なる，必ずしも常に最簡形に変換できるとは限らない，パターンマッチングのための計算コストが高いなどの実用上の問題があった．また，上位のプロセスに関する入出力データしか考慮していないため，下位のプロセスにおける内部データと上位の入出力データとの依存関係に基づいたデータフロー図を生成することができないという適用上の制約があった．

このため，本論文では，データの入出力関係を表現する行列の各行を論理ベクトルと見

なし，これらの論理ベクトル間の大小関係に基づいてデータフロー図を一意的に生成する決定論的なアルゴリズムを提案する[5]．このアルゴリズムの特徴は，以下の3点である．

［特徴1］入出力データの関係を代数式に変換し，最簡形を求める手法に較べ，入出力データの関係行列だけを用いるので，入出力データの関係行列の他に代数式という中間表現を用いる必要がないため，見通しが良く効率的である．

［特徴2］データフロー図を入出力関係行列から一意的に生成できる．

［特徴3］外部データだけでなく内部データも含めた依存関係にも適用できる．

以下では，3.2.1 で入出力データ間の依存関係からデータフロー図を自動生成するアルゴリズムを示す．さらに，3.2.2 では，このアルゴリズムを内部データフローの依存関係を考慮した場合に拡張する．3.2.3 では，本論文で提案した方式の有効性と実際のシステム開発への適用性について考察する．3.2.4 では，関連研究との比較について述べる．最後に，3.2.5 で関連研究との比較を述べる．

3.2.1 入出力関係に基づくデータフロー図の自動生成方式

まずデータフロー図の自動生成方式で必要となる基本概念を定義する．

［定義1］入出力データ間の依存関係

上位のプロセスに対する入力データの集合 $I=\{I_i \mid 1\leqq i\leqq N\}$，出力データの集合 $O=\{O_j \mid 1\leqq j\leqq M\}$ に対する，入出力データ間の依存関係を次のような行列 D で定義する．

$D=\{d_{ij} \mid O_j$ を出力するために I_i が必要なら $d_{ij}=1$，そうでないとき，$d_{ij}=0\}$．

［定義2］論理ベクトル間の関係

論理ベクトル a，b の各成分について，$a_i\leqq b_i$ であるとき，$a\leqq b$ で表す．論理ベクトル a，b の各成分が同じ場合，$a=b$ で表す．$a\leqq b$ かつ，ある成分 i について $a_i<b_i$ のとき，$a<b$ で表す．

［定義3］論理ベクトル集合のグラフ表現

論理ベクトルの集合 R と大小関係＜に基づいて，以下のようにしてグラフ G（R，＜）を定義できる．まず，r∈R のとき，r を G のノードとする．また，a＜b のとき，a＜c かつ c＜b となる c がなければ，ノード b から a への有向辺（b，a）を定義する．

このように構成された G（R，＜）について次が成り立つ．

［性質1］グラフ G（R，＜）の構成から，グラフ上の点 r から点 s への有向辺の並び（パス）があるとき，s＜r となる．逆に，s＜r のとき，グラフ G（R，＜）上に点 r から点 s へのパスがある．

論理ベクトル集合に対するグラフ表現に基づいてシステム入出力間の依存関係行列 D からデータフロー図を自動生成することができる．以下では，まず，アルゴリズムの基礎となっているデータフロー図の記述規則とアルゴリズムを構成する上での基本的な考え方を示す．次に，具体的なアルゴリズムを示す．

データフロー図は，システム機能を視覚的に表現する柔軟性の高い記述法であるが，分析者の経験や視点の違いによって，記述上の個人差が生じ易いという問題がある．もし，このような記述規則を定めておかなければ，分析者によって記述規則が異なるデータフロー図が作成されてしまうため，複数の分析者が分担してデータフロー図を記述する場合，分担作成されたデータフロー図を統合してシステム全体のデータフロー図を作成する時に手戻りが生じ易い．データフロー図の記述規則を以下に示す．

［規則1］入力 a，b を必要とする出力の集合が一致する場合，入力 a，b を同じプロセスへの入力データフローとすること

［規則2］出力 x，y が必要とする入力の集合が一致する場合，出力 x，y を同じプロセスからの出力データフローとすること

［規則3］入力 a を必要とする出力の集合が，入力 b を必要とする出力の集合を真に含む極小な集合である場合，a を外部から入力するプロセスから，b を外部から入力するプロセスへのデータフローがあること

［規則4］出力 x が必要とする入力の集合が，出力 y を必要とする入力の集合に真に含まれる極大な集合である場合，外部へ x を出力するプロセスから，外部へ y を出力するプロセスへのデータフローがあること

［規則5］入出力データフロー名が一意であること

ここで，規則1および規則2は冗長なプロセスを記述しないための規則である．また，規則3および規則4は，プロセス間の冗長なデータフローを記述しないための規則である．規則5は，外部とプロセス間で冗長な入出力データフローを記述しないための規則である．上述した規則を満足するデータフロー図の例を図1に示す．たとえば，プロセス p の出力 x が必要とする入力は a と b である．また，プロセス q の出力 y が必要とする入力は，a，b，c である．したがって，プロセス q の出力 y が必要とする入力の集合に，プロセス p の出力 x が必要とする入力の集合が含まれている．このとき，プロセス p から q へのデータフローがあるので，図3−1のデータフロー図は規則4を満足している．

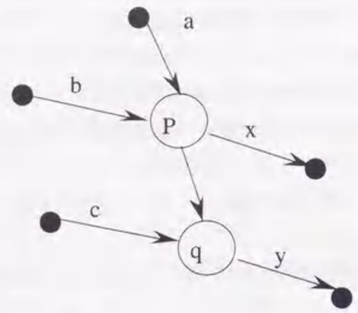


図 3−1 データフロー図の例

以下ではこの記述規則を満足するデータフロー図を自動生成するためのアルゴリズムの基本的な考え方を明らかにする.

データフロー図の記述規則 4 から，出力 x が必要とする入力集合が出力 y を必要とする入力の集合に真に含まれる極大な集合である場合,外部へ x を出力するプロセスを p とし，外部へ y を出力するプロセスを q とすれば，プロセス p からプロセス q へのデータフローがあるから，出力 x および出力 y が，プロセス p の入力を必要とすることになる.
したがって，外部へ x を出力するプロセスに対する出力集合の要素は，x 自身と，x が必要とする入力集合を真に含むような入力集合を必要とする出力 y である.
このように考えると，各出力 x に対して，x を外部へ出力するプロセスへの入力を必要とするような出力の集合を求めることができる．また，各入力については，与えられた依存関係行列の各行に基づき，それを必要とする出力の集合を直接求めることができる．さらに，プロセスの入力を必要とする出力集合に基づいて論理ベクトルを構成できる．したがって，性質 1 を用いれば，論理ベクトルの大小関係に基づいて，プロセス間のデータフローを生成することができる.

すなわち，以下に示すアルゴリズムでは，まず，依存関係行列 D の第 i 行に，入力データ I_i を必要とする出力データの集合が対応していることから,入力が対応するプロセスについては，依存関係行列の各行を各プロセスの出力集合を表現する論理ベクトルと見なす．また，出力が対応するプロセスについては，上で述べたように，依存関係行列の各列を比較して，出力が対応するプロセスの入力を必要とするような出力の集合を論理ベクトルで

表現する.

［アルゴリズム 1］データフロー図の自動生成アルゴリズム
［入力］入力データの集合 $I=\{I_i \mid 1\leqq i\leqq N\}$，出力データの集合 $O=\{O_j \mid 1\leqq j\leqq M\}$，入出力関係行列 $D=\{d_{ij}\}$．D の各行からなる論理ベクトル集合 $R=\{r_i \mid R_i$ の第 j 成分が $d_{ij}\}$ .D の各列からなる論理ベクトル集合 $C=\{c_j \mid c_j$ の第 i 成分が $d_{ij}\}$.
［出力］D で与えられる入出力関係を満足するデータフロー図 $G(E_I \cup E_O \cup P, F_I \cup F_O \cup F_P)$．ここで,$E_I$ は入力データに対する外部ノード集合,E_O は出力データに対する外部ノード集合，P はプロセスに対するノード集合である．また，F_I は外部入力ノードとプロセス間のデータフロー集合，F_O は外部出力ノードとプロセス間のデータフロー集合，F_P はプロセス間のデータフロー集合である.
［方法］
［ステップ 1］入出力関係 D の各列 c_j の大小関係を比較し,各出力データ O_j に対する行ベクトル q_j を決定する．ここで，O_j に対する行ベクトル q_j の第 k 成分は $c_j\leqq c_k$ なら 1，その他の場合 0 である（$1\leqq k\leqq M, 1\leqq j\leqq M$）．もし，このようにして決められた q_j が R の要素でなければ，R に q_j を追加する.
［ステップ 2］R のグラフ化 G（R，＜）のノード集合および有向辺の集合により，プロセスの集合 P，およびプロセス間のデータフローの集合 FP を次のように生成する.

P＝｛$r_i \mid r_i\in R$，ただし，$r_i=r_j$（$i<j$）のとき，r_i と r_j のすべての成分が一致するので最小の添え字を持つ r_i だけを要素とする．このとき，r_j を r_i に同一視するという.｝

$F_P=\{(r_j, r_i) \mid r_i<r_j,\ r_i\in P,\ r_j\in P,$ かつ $r_i<r<r_j$ となる $r\in P$ がない$\}$
［ステップ 3］外部ノードを入出力データに対して生成する．すなわち，各 I_i および O_j に対して外部ノード e_i および e_j を生成する.
［ステップ 4］入力データに対する外部ノードとプロセス間のデータフロー F_I を，以下のようにして生成する.

$F_I=\{(e_i, p_i) \mid e_i$ は入力 I_i に対して生成した外部ノード，p_i は r_i および r_i に同一視された論理ベクトルから生成されたプロセスノード$\}$

このとき，データフロー（e_i，p_i）のラベルを I_i とする.
［ステップ 5］出力データに対する外部ノードとプロセス間のデータフロー F_O を，以下のようにして生成する.

$F_O=\{(p_j, e_j) \mid e_j$ は出力 O_j に対して生成した外部ノード，p_j は O_j に対する行ベクトル q_j および q_j に同一視された行ベクトルから生成生成されたプロセスノード$\}$

このとき，データフロー（p_j，e_j）のラベルを O_j とする.

（アルゴリズム終わり）

本アルゴリズムの正当性は，以下の性質 2 から明かである.

［性質2］入出力データI，Oおよび入出力データ関係Dからアルゴリズムで生成されたデータフロー図をG（$E_I \cup E_O \cup P$，$F_I \cup F_O \cup F_P$）とする．このとき，入力データI_iの入力元ノード$e_i \in E_I$から出力データO_jの出力先ノード$e_j \in E_O$へのパスがあるとき，$d_{ij}=1$である．また，逆も成り立つ．

（性質2の証明）

入力データI_iの入力元ノード$e_i \in E_I$から出力データO_jの出力先ノード$e_j \in E_O$へのパスπがあるとする．このとき，以下の2つの場合がある．

1）プロセスノードp_iが存在してパスπをe_iからp_iへの有向辺（e_i，p_i）$\in F_I$，p_iからe_jへの有向辺（p_i，e_j）$\in F_O$に分割できる場合．

2）プロセスノードp_i，p_k（$i \neq k$）が存在してパスπをe_iからp_iへの有向辺（e_i，p_i）$\in F_I$，p_iからp_kまでのパスω，p_kからe_jまでの有向辺（p_k，e_j）$\in F_O$に分割できる場合．

場合1）のとき，（p_i，e_j）$\in F_O$から，p_iに対する論理ベクトルr_iの第j要素が1である．したがって$d_{ij} \in D$となる．

場合2）のとき，p_iからp_kへのパスωがあることからp_iおよびp_kに対する論理ベクトルr_iおよびr_kは，$r_k < r_i$を満たす．また，（p_k，e_j）$\in F_O$から，r_kの第j要素は1である．したがって$r_k < r_i$から，r_iの第j要素も1となる．故に$d_{ij} \in D$となる．

したがって，いずれの場合も，$d_{ij} \in D$となる．

逆は，アルゴリズムの構成から明かである．

（証明終わり）

アルゴリズム1の適用例を図3−2に示す．図3−2では，入力データa，b，c，d，e，出力データx，y，z，w，sに関する依存関係が次の行列Dで与えられているものとする．

	x y z w s
a	1 1 0 0 0
b	0 1 0 0 0
c	1 0 0 1 1
d	0 1 1 0 0
e	0 1 0 0 0

［ステップ1］まず，Dの各列を比較して追加すべき行ベクトルを抽出する．出力データxに対する第1列c_1は，c_2，c_3と比較不能かつ$c_4=c_5<c_1$だから，xは　q_1=（1 0 0 0 0）に対応する．q_1は，Dの各行と異なるので追加する必要がある．出力データyに対する第2列c_2は，c_1，c_4，c_5と比較不能かつ$c_3<c_2$だから，yはq_2=（0 1 0 0 0）に対応する．q_2はDの第2行および第5行と一致する．出力データzに対する第4列c_3はc_1，c_4，c_5と比較不能かつ$c_3<c_2$だからzはq_3=(0 1 1 0 0)に対応する．q_3はDの第4行と一致する．出力データw，sに対する第4，第5列c_4，c_5はc_2，c_3と比較不能かつ$c_4=c_5<c_1$だから，w，sは$q_4=q_5=$

（1 0 0 1 1）に対応する．q_4とq_5はDの第3行と一致する．このとき，Dの第1行をr_1，第2，第5行をr_2，第3行をr_3，第4行をr_4，q_1をr_5として

R= {r_1，r_2，r_3，r_4，r_5} とする．

［ステップ2］G（R，<）をグラフ化する．すなわち，Rの各要素r_1，r_2，r_3，r_4，r_5ごとにプロセスノードp_1，p_2，p_3，p_4，p_5を作成し，次のようにしてノード間のデータフローを追加する．すなわち，

$r_2 < r_1$より，p_1からp_2へのデータフローを作成する．

$r_5 < r_1$より，p_1からp_5へのデータフローを作成する．

$r_5 < r_3$より，p_3からp_5へのデータフローを作成する．

$r_2 < r_4$より，p_4からp_2へのデータフローを作成する．

［ステップ3］入力データa，b，c，d，eに対する外部ノードe_a，e_b，e_c，e_d，e_e，および出力データx，y，z，w，sに対する外部ノードe_x，e_y，e_z，e_w，e_sを作成する．

［ステップ4］入力データに対する外部ノードからプロセスノードへのデータフローを作成する．すなわち，aがr_1に対応するのでe_aからp_1へのデータフローを作成する．bとeがr_2に対応するのでe_bとe_eからp_2へのデータフローを作成する．cがr_3に対応するのでe_cからp_3へのデータフローを作成する．dがr_4に対応するのでe_dからp_4へのデータフローを作成する．

［ステップ5］プロセスノードから出力データに対する外部ノードへのデータフローを以下のようにして作成する．すなわち，xがr_5に対応するのでp_5からe_xへのデータフローを作成する．yがr_2に対応するのでp_2からe_yへのデータフローを作成する．zがr_4に対応するのでp_4からe_zへのデータフローを作成する．wとsがr_3に対応するのでp_3からe_wとe_sへのデータフローを作成する．

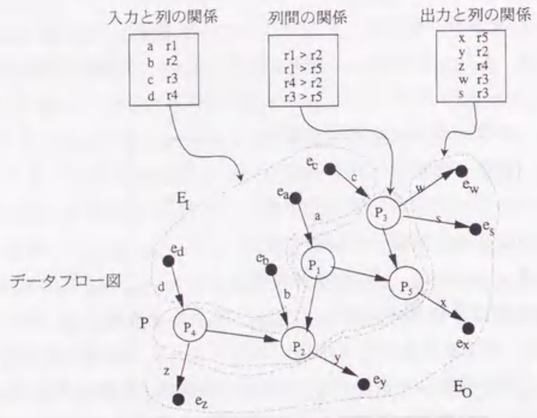


図 3−2 アルゴリズムにより生成されたデータフロー図

3.2.2 内部データを含む場合への拡張

前節で述べたアルゴリズム 1 では，内部データの依存関係を考慮していなかった．しかし，一般には，内部データとの依存関係も含めて考慮しないと，プロセスをうまく分割できないことが多い[3]．たとえば，図 3−3 に示すデータフロー図の場合，入出力データの依存関係は，次のようになる．

	x y z w
a	1 1 0 0
b	1 1 1 1
c	1 1 1 1

この場合，入出力データの依存関係だけでは，プロセス P_2，P_3，P_4 を 3 個のプロセスとして，図 3−3 のように分割できない．したがって，図 3−3 の例では，プロセス P_2，P_3 を分割するための内部データ d と，P_2 と P_4 を分割するための内部データ e を考慮する必要がある．これに対して，図 3−3 の内部データ f を考慮しなくても入出力データの依存関係だけに基づいて，これらの 3 個のプロセスとプロセス P_1 とを分割することができる．

以下では，入出力データ間の関係だけでなく，内部データとこれらの依存関係をも考慮した上で，データフロー図を生成するアルゴリズムを示す．ここで，データフロー図には，外部とプロセスとの間のデータフローだけでなく，以下の 2 つの内部データフローがある．

すなわち，プロセス間のデータフローと，プロセスとデータストアとの間のデータフローである．このうち，本論文では，内部データとしてプロセス間のデータフローの依存関係を扱う．

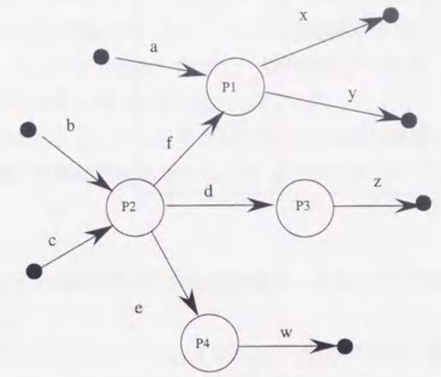


図 3−3 データフロー図

まず，入出力関係行列 D を，外部入出力データ間の関係だけでなく，プロセス間のデータフローである内部データ間の依存関係も表現できるように拡張する．

［定義 4］入力データの集合を $I=\{I_i \mid 1\leqq i\leqq N\}$，出力データの集合を $O=\{O_j \mid 1\leqq j\leqq M\}$,内部データの集合を $L=\{L_k \mid 1\leqq k\leqq K\}$ とするとき，入出力データおよび内部データ間の関係を次のような 4 個の論理行列 $\Delta=\{\delta_{ij}\}$ で表現する（図 3−4）．

（場合 1）$1\leqq i\leqq N$，$1\leqq j\leqq M$ のとき O_j を出力するために I_i が必要なら $\delta_{ij}=1$，そうでないとき $\delta_{ij}=0$.

（場合 2）$1\leqq i\leqq N, M+1\leqq j\leqq M+L$ のとき L_j を出力するために I_i が必要なら $\delta_{ij}=1$．そうでないとき，$\delta_{ij}=0$.

（場合 3）N+1≦i≦N+L，1≦j≦Mのとき，
O_j を出力するために L_i が必要なら $\delta_{ij}=1$，そうでないとき，$\delta_{ij}=0$.
（場合 4）N+1≦i≦N+L，M+1≦j≦M+L のとき,L_j を出力するために L_i が必要なら $\delta_{ij}=1$，そうでないとき $\delta_{ij}=0$．ただし，$\delta_{ii}=0$.

	出力データ	内部データ
入力データ	$\triangle_1$	$\triangle_2$
内部データ	$\triangle_3$	$\triangle_4$

図 3−4　内部データ間の関係を示す論理行列

入出力データ間の依存関係は，場合 1 に対応する部分行列Δ_1である．入力データと内部データとの依存関係は，場合 2 に対応する部分行列Δ_2である．内部データと出力データとの依存関係は，場合 3 に対応する部分行列Δ_3である．内部データ間の依存関係は，場合 4 に対応する部分行列Δ_4である．たとえば，図 3−3 で示した，データフロー図に対する依存関係行列を以下に示す．

	x y z w d e f
a	1 1 0 0 0 0 0
b	1 1 1 1 1 1 1
c	1 1 1 1 1 1 1
d	0 0 1 0 0 0 0
e	0 0 0 1 0 0 0
f	1 1 0 0 0 0 1

Δでは，入出力データ間の依存関係が部分行列Δ_1だけでなく，部分行列Δ_2，Δ_3，Δ_4の

組合せとしても表現される．このため，Δ_1で示される入出力関係がΔ_2，Δ_3，Δ_4の組合せで表現される入出力関係と互いに矛盾しないための条件を明らかにしておく必要がある．
［定義 5］依存関係の適合性条件
（1）$\delta_{ik}\in\Delta_2$，$\delta_{kh}\in\Delta_4$，$\delta_{hj}\in\Delta_3$が存在して，$\delta_{ik}=1$，$\delta_{kh}=1$，$\delta_{hj}=1$ であるとき，$\delta_{ij}=1$ となる$\delta_{ij}\in\Delta_1$がある．
（2）$\delta_{ik}\in\Delta_2$，$\delta_{kj}\in\Delta_3$が存在して，$\delta_{ik}=1$，$\delta_{kj}=1$ であるとき，$\delta_{ij}=1$ となる$\delta_{ij}\in\Delta_1$がある．

適合性条件を満足するΔでは，内部データを介して入力データと出力データが依存するならば，Δ_1で入力データと出力データが依存関係にある．
たとえば，図 3−3 で示した，データフロー図に対する依存関係行列の場合，内部データ d と入力データ b および出力データ z について依存関係行列を調てみると，確かに$\delta_{bd}=1$，$\delta_{dz}=1$ のとき，$\delta_{bz}=1$ である．同様に，他の内部データ e，f と入出力データの組み合せについても適合性条件を満足している．
適合性条件が成立しているかどうかは行列演算により簡単に確かめることができる．
以下では，適合性条件を満足するΔに対してデータフロー図を生成するアルゴリズムを示す．

［アルゴリズム 2］入出力データおよび内部データ間の依存関係に基づくデータフロー図生成アルゴリズム
［入力］入力データの集合 $I=\{I_i \mid 1\leqq i\leqq N\}$，出力データの集合 $O=\{O_j \mid 1\leqq j\leqq M\}$，内部データの集合 $L=\{L_k \mid 1\leqq k\leqq K\}$，適合性条件を満足する入出力データおよび内部データ間の依存関係$\Delta=\{\delta_{ij} \mid 1\leqq i\leqq N+L,\ 1\leqq j\leqq M+L\}$
［出力］Δで与えられる入出力および内部データの依存関係を満足するデータフロー図 G（$E_I\cup E_O\cup P$，$F_I\cup F_O\cup F_P$）．ここで，E_I，E_Oは，それぞれ入力データに対応する外部ノード集合，出力データに対応する外部ノード集合，P は,プロセスノード集合である．また，F_I，F_O，F_P はそれぞれ，外部入力ノードとプロセス間のデータフロー集合，外部出力ノードとプロセス間のデータフロー集合，プロセス間のデータフロー集合である．
［方法］
［ステップ 1］入出力関係Δから構成された論理ベクトルの集合 R に対してアルゴリズム 1 の［ステップ 1］と同様に出力データに対する行ベクトルの候補を追加した集合を R'とする．
［ステップ 2］R'のグラフ表現 G（R'，＜）のノード集合および有向辺の集合から，それぞれプロセスの集合 P，およびプロセス間のデータフローの集合 F_P を生成する．$P=\{r_i \mid r_i\in R'$．ただし $r_i=r_j$（$i\neq j$）のとき,r_iと r_jとを同一視する$\}$
$F_P=\{(r_j,\ r_i) \mid r_i,\ r_j\in P,\ r_i<r_j$かつ $r_i<r<r_j$ となる $r\in P$ が存在しない$\}$

このとき，データフロー（r_j，r_i）のラベルを行ベクトル r_j に対する内部データ名 L_j とする．

［ステップ 3］入出力データに対する外部ノードを生成する．すなわち，各 I_i および O_j に対して外部ノード e_i およびを e_j を生成する．

［ステップ 4］入力データに対する外部ノードとプロセス間のデータフローF_Iを，以下のようにして生成する．

F_I＝｛(e_i，p_i）｜e_i は入力 I_i に対して生成した外部ノード,p_i は,r_i および r_i に同一視された論理ベクトルから生成されたプロセスノード｝このとき，データフロー（e_i，p_i）のラベルを I_i とする．

［ステップ 5］出力データに対する外部ノードとプロセス間のデータフローである F_O を，以下のようにして生成する．

F_O＝｛(p_j，e_j）｜e_j は出力 O_j に対して生成した外部ノード，r_j は出力 O_j に対する行ベクトル，p_j は,r_j および r_j に同一視された論理ベクトルから生成されたプロセスノード｝．

このとき，データフロー（p_j，e_j）のラベルを O_j とする．

（アルゴリズム終わり）

アルゴリズム 2 は，アルゴリズム 1 と基本的に同じである．異なる点は，ステップ 2 で内部データ名をプロセス間のデータフロー名として付与している点である．

アルゴリズム 2 の正当性は，以下の性質から明かである．

［性質 3］入出力データ I，O，内部データ L，入出力および内部データ間の関係Δからアルゴリズム 2 で生成された データフロー図を，G（E_I∪E_O∪P，F_I∪F_O∪F_P）とする．このとき，I_i（または L_i）に対するノードを p，O_j（または L_j）に対するノードを q とすると，p から q へのパスがあるとき，δ_{ij}＝1 である．また，逆も成り立つ．

3.2.3 データフロー図自動生成方式の有効性

入出力関係行列からデータフロー図を作成する方式が有効であるためには，生成されたデータフロー図が与えられた入出力関係と矛盾しないこと（正当性）だけでなく，次の各条件を満足する必要がある．

［最小性］各プロセスが 少なくとも 1 つの外部入出力データを互いに重複することなく入出力し，プロセス数が最小なデータフロー図を生成すること

［一意性］異なるデータフロー図を生成しないこと

［完全性］最小なデータフロー図がすべて生成できること

以下では，本方式が，これらの条件を満足していることを示す．議論を簡単にするため，内部データを考慮しない場合について述べる．内部データを考慮した場合の証明については紙数が限られているので割愛する．内部データを考慮した場合について，これらの性質を証明するためには，入力集合 I および出力集合 O の定義を，それぞれの要素として内部データを含むように拡張することにより，内部データを含まない場合と同様に証明することができる．

（1）最小性

アルゴリズム 1 の構成から，同一の入出力データ名が複数箇所に出現しないこと，各プロセスが少なくとも 1 つの外部入出力データを入出力することは明かである．

次に，アルゴリズム 1 で生成されたデータフロー図のプロセス数が最小であることを以下のようにして示すことができる．

もし，アルゴリズム 1 で入出力関係 D から生成された データフロー図が最小でないとすると，あるプロセスノード p，q（p≠q）が存在して，以下のいずれかが成立する．

1）I（p）＝I（q）かつ O（p）≠O（q）

2）I（p）≠I（q）かつ O（p）＝O（q）

3）I（p）＝I（q）かつ O（p）＝O（q）

ここで，プロセス p の出力データが必要とする入力データの集合を I（p），プロセス p の入力データを必要とする出力データの集合を O（p）で表す．

場合 1 のとき，O（p）≠O（q）より，以下の何れかが成立する．

（1−a）任意の x∈O（p）に対して x≠y となる y∈O（q）が存在する

（1−b）任意の y∈O（q）に対して y≠x となる x∈O（p）が存在する

（1−a）の場合，a∈I（p）なら a∈I（q）より d_{ax}＝d_{ay}＝1．また a∈I（p）でなければ a∈I（q）でないことより d_{ax}＝d_{ay}＝0．したがって，D の列ベクトルについて c_x＝c_y が成立する．このとき，アルゴリズム 1 の構成から x∈O（p）ならば，y∈O（p）となり矛盾する．（1−b）の場合も同様に矛盾する．

場合 2 のとき，I（p）≠I（q）より，以下の何れかが成立する．

（2−a）任意の a∈I（p）に対して a≠b となる b∈I（q）が存在する

（2−b）任意の b∈I（q）に対して b≠a となる a∈I（p）が存在する

（2−a）の場合，x∈O（p）について x∈O（q）より d_{ax}＝d_{bx}＝1．また x∈O（p）でなければ x∈O（q）でないことより d_{ax}＝d_{bx}＝0．したがって， D の行ベクトルについて r_a＝r_b が成立する.このとき，アルゴリズム 1 の構成から，a∈I（p）ならば，b∈I（p）となり矛盾する．（2−b）の場合も同様に矛盾する．

場合 3 のとき，アルゴリズム 1 が生成するデータフロー図では，同一の入出力データ名が複数のプロセスに直接入出力されないこと，各プロセスが少なくとも 1 つの入出力データを入出力することに矛盾する．

したがって，アルゴリズム 1 により，入出力関係行列から生成されるデータフロー図は最小である．

（2）一意性

入出力関係行列に基づく論理ベクトルの集合 R'の生成は一意的であり，また，そのグラフ化も一意的である．したがって，アルゴリズム 1 では，入出力関係行列から異なるデータフロー図が生成されることはない．

また，逆に性質 2 で示したように，異なる入出力関係から同じデータフロー図が生成されることはない．

(3) 完全性

アルゴリズム 1 から生成されるデータフロー図が与えられた入出力関係行列に矛盾しないことは性質 2 から明かである．したがって，最小なデータフロー図 G に対して入出力関係行列 D が必ず存在して，D からアルゴリズム 1 によって G を生成できることを示す．すなわち，最小なデータフロー図 G の入力データを直接入力する各プロセス P について，P からの各出力データへの到達可能性に従って論理ベクトルを作り，それを入出力関係行列 D の行とする．このようにして構成された D から，アルゴリズム 1 に従って，G を生成できることは明かである．

3.2.4 データフロー図自動生成方式の適用性

(1) 計算量の評価

データフロー図自動生成アルゴリズムの基本となる処理は入出力関係行列の各列に対して抽出する行ベクトルの計算である．入出力関係行列 D が N 行 M 列であるとする．アルゴリズムのステップ 1 で M 個の出力データごとに各列を比較している．各列の比較回数は高々 M-1 回である．したがってステップ 1 の計算量は $O(M^2)$ である．またステップ 1 の結果として抽出される行ベクトルの個数は高々M 個である．ステップ 2 以降ではステップ 1 で抽出された行ベクトルを D の行ベクトルからデータフロー図を生成する処理である．この一連の処理に対する計算量は O(N+M)である．

(2) 処理効率の評価

データフロー図を用いたシステム分析作業に，本方式を適用する場合，応答時間の早さが必要である．以下では，本アルゴリズムを C 言語で記述した実験結果について述べる．プログラム規模は約 2K，入出力データと入出力関係行列から NTT ソフトウェア研究所で開発した CASE ツールである SoftDA[6][7][8]のデータフロー図ファイルを生成する．SPARC Station2 で測定したところ，入力データ数 15，出力データ数 8 の入出力関係行列に対するデータフロー図を生成した場合，入出力の依存関係を入力してからデータフロー図のレイアウトまでを含めた一連の処理に対する応答時間は約 0.35 秒である．このうち，データフロー図の生成アルゴリズムに関する処理時間は約 0.07 秒である．この応答時間は，以下の理由から，十分実用的である．すなわち，入出力関係行列からデータフロー図を人手で作成した場合，まず，分析者が入出力関係行列から記述規則を満たすようなデータフロー図の構成を案出するまでに少なくとも数十分が必要である．また，分析者がデータフロー図に対する十分な構成を把握してから，データフロー図エディタを使って記述するためには 1 シンボル当たり数秒が必要である．さらに，分析者が作成したデータフロー図が与えられた入出力関係行列に矛盾しないことを確認するための時間が必要になる．

また，本方式を用いて，より複雑なデータフロー図を自動生成する場合には，入出力の

依存関係からデータフロー図の構成要素を自動生成するアルゴリズムの効率よりも，データフロー図の構成要素を配置するレイアウト処理の品質および効率を改善することが重要な課題となる．

(2) 実際のソフトウェア開発への適用性

大規模なソフトウェア開発では，複数の担当者がデータフロー図を記述するので，データフロー図の記述上の差が大きくならないように，担当者間でデータフロー図の記述規則を統一しておく必要がある．本方式では，最小性の概念を用いているので，データフロー図の標準化が容易である．

また，実際のソフトウェア開発プロジェクトでは，構造化分析手法に習熟した分析者が十分いるとはいえないのが現状である．このような場合，入出力データの依存関係を正確に満足するようなデータフロー図の記述が困難であるだけでなく，同じ名称のデータフローを複数箇所で記述したり，同じ入出力を持つ異なるプロセスを作成するといった冗長なデータフロー図を作成し易いという問題がある．本方式を用いることにより，このような冗長性のない適切なデータフロー図を作成できる．

ただし，データフロー図を用いたシステムの機能分析だけでなく，エンティティ・リレーションシップ図を用いたデータベース分析も必要となるようなソフトウェア開発へ本方式を適用する場合には以下のような限界がある．すなわち，本方式ではデータの構造を考慮していないため，自動生成したデータフロー図とエンティティ・リレーションシップ図やデータ構造図との関係が厳密に規定できない．したがって，自動生成したデータフロー図を手作業で修正して，これらの図式が表現する構造を持つデータとの対応関係を付加する必要がある．この問題を解決するためには，構造を持つデータの依存関係とデータフロー図との対応関係を規定した上で，新たな自動生成方式を考案する必要がある．

3.2.5 関連研究との比較

Adler は，以下のような代数的な手法を用いて入出力関係行列 D からデータフロー図を生成することを提案した．すなわち，まず $d_{ij}=1$ となる i，j について，$(I_i \rightarrow O_j)$ を要素とする式 L を構成する．次に，代数的な式の変換規則を用いて，入出力データが高々1 個しか出現しないように，L を最簡形 L'に変換する．この最簡形 L'に基づいてデータフロー図を生成する．

Adler の手法は，データフロー図を機械的に作成する上で有効であるが，変換規則の能力が不足しており，以下の点で必ずしも十分とはいえなかった．

(1) 変換規則の適用順序によって，異なる最簡形が生成されるため，それぞれの最簡形から作成されるデータフロー図が異なる．

(2) 入出力関係行列から変換規則を用いて生成できない最簡形が存在するため，正しいデータフロー図を作成できる入出力関係が限定されている．

(3) プロセス間で流れる内部データに対する依存関係を考慮していない．

このため，Adler の手法を拡張して内部データを考慮できるようにした代数式の変換系が提案されている[9][10]．

彼らの手法と本手法との基本的な違いは，次の点である．すなわち，本方式では，入出力関係行列を論理ベクトルの集合と見なし，論理ベクトルの比較演算により，データフロー図を生成するので，代数式の変換系で必要となるような，複雑で計算コストの高いパターンマッチング処理や与えられた入出力関係と生成したデータフロー図に対する入出力関係との一貫性の検査処理が不要となり，効率的である．

また，本方式では，内部データを含む場合への拡張にあたって，入出力データの直接的な依存関係と内部データと入出力データの依存関係が矛盾しないこと（適合性条件）を仮定している．一方，従来の内部データへの拡張方式では，この適合性条件を仮定していなかった．たとえば，従来手法では同じ内部データ S と関係する入力データ a と出力データ z があるとき，a と z は関係しない．ところが，この入出力関係行列から生成されるデータフロー図では，内部データを介さない場合と同様に，入力データ a から内部データ S を介して出力データ z に至るデータフローのパスがあるため，本論文では，内部データを外部データと区別して扱うのではなく，a と z が入出力関係行列上でも関係することとしている．

3.3 モジュール構造図からのデータフロー図の復元方式

大規模ソフトウェア開発では，データフロー図とモジュール構造図に関する追跡性が重要になる．データフロー図とモジュール構造図との間にどのような対応関係があるかを追跡する判断作業には大きな工数が必要になる．既存の CASE ツールでは，データフロー図とモジュール構造図との対応関係を予め設計者が記録したり，検索できる追跡機能を提供している．また， Benedusi らは既存のプログラムコードからデータフロー図をリバースエンジニアリングするための手法を提案している[11]．Bryne はこの手法を適用した実験結果を報告している[12]．この手法では，プログラムの変数間の依存関係から階層的なデータフロー図を作成する．しかし，生成されたデータフロー図には，同じ名称のプロセスが含まれるという問題がある．また，モジュール構造図のモジュールごとにデータフロー図のプロセスを作成するため，データフロー図に詳細な処理に対応するプロセスが出現するという問題がある．O'Hare は RE-Analyzer と称するリバースエンジニアリングツールを提案している[13]．プログラムコードから制御処理に基づいて階層的なデータフロー図を作成する．この場合の問題点は，制御処理に対応したプロセスを生成してしまうので，データフロー図に他のプロセスと互いにデータフロー関係を持たない孤立した制御プロセスが存在してしまうことである．

もし，モジュール構造図からシステム入力とシステム出力の依存関係を抽出することができれば，3.2 で示したアルゴリズムを用いることにより，モジュール構造図に対応するデータフロー図をこの依存関係から自動的に作成できる[14][15]．この手法により，上述したようなデータフロー図に詳細処理に対応するプロセスが含まれるという問題を解決できる．

［アルゴリズム 3］データフロー図の自動復元アルゴリズム

［ステップ 1］モジュール構造図 S の入力パラメータと出力パラメータとの関係 D を決定する．ここで，

D＝｛(a，x) ｜S の任意のモジュール m に関して，a が m の入力で x が m からの出力でありかつ，m において a から x への変換処理が存在する｝

入力しか持たないようなモジュール m に対して，D の要素は m のパラメータから抽出されない．この理由は，入出力関係が存在しないからである．同じ理由で，出力しか持たないモジュールに対しても，D の要素は抽出されない．また，モジュールの中で引き継がれるだけで処理されないデータについても入出力関係を抽出しない．この理由は，このパラメータがモジュールの中で変換されないためである．

［ステップ 2］もし，D が（a，b）と（b，x）を含むが，（a，x）を含まないとき，（a，x）を D の要素として追加する．この手続きを，D に新たな要素が追加されなくなるまで繰り返す．

［ステップ 3］すべての入力 a∈I について，出力集合 Ra を決定する．ここで，

R_a＝｛x｜（a，x）∈D｝

［ステップ 4］各 R_k についてデータフロー図のプロセスを以下の条件に基づいて作成する．ここで，R_k は集合 R＝｛R_a ｜ a∈I｝ の要素である．

（1）R_k∈R に対して R_j∈R かつ R_k＝R_j となる j（≠k）が存在しないとき R_k に対するプロセスを作成する．

（2） R_k∈R に対して R_j∈R かつ R_k＝R_j となる j（≠k）が存在するとき集合 ｛R_j ｜ R_j＝R_k，j≠k，R_j∈R，R_k∈R ｝ に対するプロセスを作成する．

［ステップ 5］もし R_a を含むような出力集合 R_k が存在しなければ，R_a について外部環境からプロセス P への入力データフローa を作成する．ここで， P は R_a に対して作成されたプロセスであり，R_k は集合 R＝｛ R_a ｜ a∈I ｝ の要素である．

［ステップ 6］もし，R_a が R_b を含み，R_a⊇R_c かつ R_c⊇R_b となる R_c が存在しなければ，Ra に対して策されたプロセス P から，Rb に対して作成されたプロセス q へのデータフローを作成する．

［ステップ 7］もし R_a に含まれるような出力集合 R_k が存在せず，R_a が x を含んでいるならば，プロセス P から外部環境への出力データフローx を作成する．ここで，P は R_a に対して作成されたプロセスであり，R_a および R_k は集合 R＝｛ R_a ｜ a∈I ｝ の要素である．

［ステップ 8］データフロー図のプロセスの名称を以下のようにして決定する．

（1）もしプロセス P とモジュール m が同じ入力と出力を持つとき，P の名称を m の名称とする．

（2）もしプロセス P が入力として a だけを持つとき，a を入力するモジュールの名称 P の名称とする．

(3) もしプロセス P が出力として x だけを持つとき，x を出力するモジュールの名称 P の名称とする．

（アルゴリズム終わり）

［具体例］

［ステップ 1］次のような入出力関係 D を図 3−5 に示すモジュール構造図から抽出する．

D = { (a, b), (b, c), (b, d), (d, x), (d, y), (c, z) }

たとえば，モジュール P は入力パラメータ a と出力パラメータ b を持つので，(a, b) を D の要素とする．

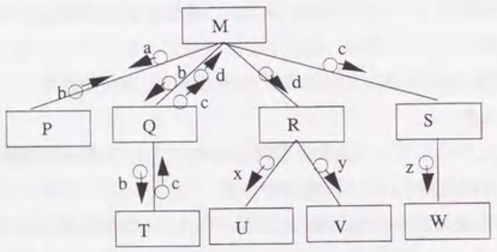


図 3−5 モジュール構造図

［ステップ 2］ 入力 a に対して，(a, b) と (b, c) が D に含まれるので，(a, c) を D に追加する．同様にして (a, d), (a, x), (a, y) および，(a, z) を D に追加する．

また，入力 b に対して，(b, d) と(d, x) が D に含まれるので，(b, x) を D に追加する．同様にして，(b, y) と(b, z) を D に追加する．入力 c と d に対しては， x，y および z が D の要素の左辺に出現しないので，D に追加する入出力関係の要素はない．結果として D は次のようになる．

D = { (a, b), (a, c), (a, d), (a, x), (a, y), (a, z) ,

(b, c), (b, d), (b, x), (b, y), (b, z), (c, z), (d, x), (d, y)}

［ステップ 3］D に基づき，R_a, R_b, R_c および R_d を決定する．

R_a = { b, c, d, x, y, z}

R_b = { c, d, x, y, z}

R_c = { z}

R_d = { x, y}

［ステップ 4］R_a，R_b，R_c および R_d に対するプロセスを作成する．

［ステップ 5］R_a に対するプロセスについて，R_a を含む出力集合 R が存在しないので，外部からの入力データフロー a を作成する．

［ステップ 6］R_a に対するプロセスから R_b に対するプロセスへのデータフローを作成する．R_b に対するプロセスから R_c に対するプロセスへのデータフローを作成する．R_b に対するプロセスから R_d に対するプロセスへのデータフローを作成する．

［ステップ 7］R_c に含まれる出力集合が存在しないので，R_c に対するプロセスから，外部境界への出力データフロー z を作成する．同様にして，R_d に対するプロセスから，外部境界への出力データフロー x および y を作成する．

［ステップ 8］プロセス R_a，R_b，R_c および R_d に対する名称を P, Q, R および S とする．たとえば，R_a は入力 a と出力 b を持つ．モジュール P も入力 a と出力 b を持つ．したがって P を R_a に対するプロセスの名称とする．

図 3−5 のモジュール構造図から生成されたデータフロー図を図 3−6 に示す．

（具体例終わり）

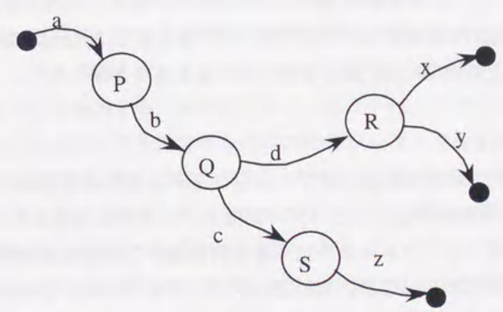


図 3−6 モジュール構造図から生成されたデータフロー図

3.4 オブジェクトモデル分析法

これまでに，数多くのオブジェクト指向分析手法が提案されている[16]．これらのオブジェクト指向分析手法は，オブジェクトとその関係に着目してオブジェクトモデルを作成する情報構造主導型とユースケースなどのシナリオに着目してオブジェクトモデルを作成する利用構造主導型に分けられる[17]．OMT 法[18]，Shlaer/Mellor 法[19]，Fusion 法[20]などは情報構造主導型である．OBA 法[21]，OOSE 法[22]，RDD 法[23]などは利用構造主導型である．

情報構造主導型では，現実世界を前提として，可能性のあるオブジェクトをすべて抽出してから，オブジェクトモデルを作成する．もし，現実世界が十分に具体的に定義できるなら，安定したオブジェクトモデルを作成できる．しかし，情報システムの場合，装置や物などの具体的に存在する物理的な対象が想定しにくいので，現実世界のオブジェクトを抽出しにくい．したがって，分析者が論理的に必要だと考えられるオブジェクトを個人的な経験から抽出する必要があり，分析者間でオブジェクトモデルの均質性を保ちにくいという問題がある．とくに，複数の分析者が同時に参加するような分析作業では，分析者間の意識合わせが困難になる可能性がある．また，分析段階のオブジェクトモデルと設計段階のオブジェクトモデルとが必ずしもうまく対応しないことが多い．

利用構造主導型では，システム利用者と利用する上でのシナリオをすべて抽出してから，オブジェクトモデルを作成する．システム外部との境界条件がシナリオによって明確に規定されるので，外部境界と関連するオブジェクトについては，安定したオブジェクトモデルを作成できる．しかし，利用構造主導型では，情報システムで最も重要となる静的なオブジェクト構造を抽出しにくいという問題がある．

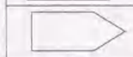
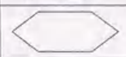
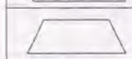
以下では，役割オブジェクト分析に基づくオブジェクトモデルの作成手法 POOM(Pattern Oriented Object Modeling)を提案する[24] [25] [26] [27] [28][29]．第 6 章では，比較実験に基づき，POOM が生産性や均質性などの点で OMT よりも優れていることを示す．

3.4.1 オブジェクトモデル

POOMでは，効率的で均質性の高いオブジェクトモデルを作成するために，オブジェクトモデルを役割オブジェクトモデルとパターン化オブジェクトモデルからなる2段階に分けて分析する．役割オブジェクトモデルは，ある役割を持つ業務担当者間の関係に着目したオブジェクトモデルである．これに対してパターン化オブジェクトモデルは，表示層，機能層，データ層からなる階層型クライアント／サーバ・アーキテクチャに着目したオブジェクトモデルである．パターン化オブジェクトモデルでは，各層のオブジェクトとそれらの関係を定義する．パターン化オブジェクトモデルでは3層アーキテクチャを構成する各層にオブジェクトが対応するので，3層アーキテクチャの設計を容易化できる．パターン化オブジェクトモデルに基づいて分散型情報システムの3層アーキテクチャを設計する手法については，第4章で説明する．

POOMのオブジェクト分類を表3－1に示す．POOMでは，表示層，機能層，データ層からなる分散型情報システムの3層アーキテクチャに基づいて，オブジェクトを7種類に分類している．

表3－1　オブジェクトの分類

分類	図形	名称
表示層		役割型
		所有者型
		アクセス境界型
機能層		トランザクション型
データ層		アクセス対象型
		発生源型
		流通対象型

表示層オブジェクトには，役割オブジェクト，所有者オブジェクト，アクセス境界型オブジェクトがある．役割型オブジェクトは，業務担当者の役割を表現するためのオブジェクトである．アクセス境界型オブジェクトは，トランザクション型オブジェクトに対するユーザインタフェースを表現する．所有者型オブジェクトは，アクセス境界型オブジェクトや役割オブジェクトの所有者を表現するためのオブジェクトである．

機能層に対応するオブジェクトには，トランザクション型がある．機能層オブジェクトは，データ層オブジェクトを操作して表示層オブジェクトに処理結果を返す機能を表現するオブジェクトである．表示層に対応するオブジェクトには，アクセス境界型，所有者型，役割型がある．アクセス境界型はユーザインタフェースに対応するオブジェクトである．所有者型は，アクセス境界型や役割型など表示層オブジェクトを所有するオブジェクトである．役割型は，役割分析モデルで抽出した役割オブジェクトである．

データ層に対応するオブジェクトには，アクセス対象型，流通型，発生源型がある．アクセス対象型は，データベース内に格納されトランザクションで操作されるオブジェクトである．流通型オブジェクトは，商品や代金など物流や出納の対象となるオブジェクトである．発生原型オブジェクトは，外部からのイベントに対応するオブジェクトである．た

とえば，商品を顧客が購入する場合，商品が流通型，代金を支払う顧客が発生源型，販売管理簿がアクセス対象型となる．

POOMにおける7種類の基本オブジェクト間の関係を図3－7に示す．

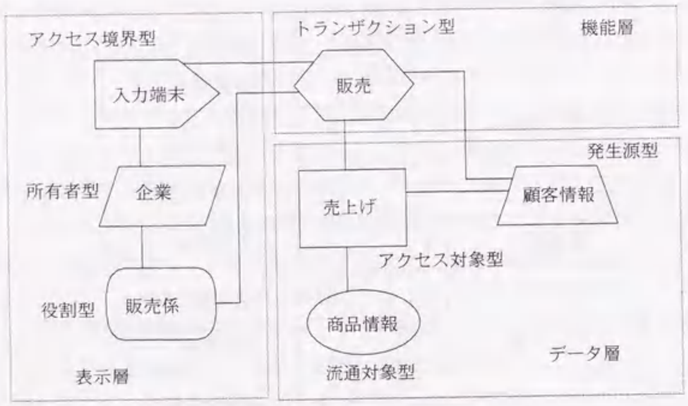


図3－7　7種類の基本オブジェクト間の関係

パターン化オブジェクトモデルでは，オブジェクトの分類ごとに 7 種類の図形を利用できるように，OMT のオブジェクトモデルの表記法を拡張している．オブジェクト間の関係に関する表記については OMT の表記法をそのままパターン化オブジェクトモデルでも使用する．パターン化オブジェクトモデルの表記法はオブジェクトの種別を明示的に区別した OMT 表記法である．パターン化オブジェクトモデルでは，抽出すべきオブジェクトとその関係が予め制約条件として明確化されているので，この制約条件をガイドラインとしてオブジェクトを抽出することができる．したがって，パターンを考慮しないオブジェクトモデルを作成する手法に比べて，パターン化オブジェクトモデルの方が効率的にオブジェクトモデルを作成できると考えられる．

［具体例］

OMT の教科書[18]で例題として採り上げられている ATM 問題に対して，本手法を適用してパターン化オブジェクトモデルを作成した例を図 3－8 に示す．図 3－8 では，表示層のオブジェクトとして，所有者型オブジェクトである銀行と銀行組合，役割型オブジェクトである会計係，アクセス境界型オブジェクトである ATM および会計端末，入力装置がある．

機能層のオブジェクトとして，会計トランザクション，遠隔トランザクション，更新トランザクションおよびその上位オブジェクトとしてのトランザクションなどのトランザクション型オブジェクトがある．データ層のオブジェクトとしては，アクセス対象型オブジェクトカード認証と口座，流通型オブジェクトキャッシュカード，発生源型オブジェクトの顧客がある．この例からも分かるように，パターン化オブジェクトモデルでは，オブジェクトの種類を図形の形状で直感的に把握できるだけでなく，表示層，機能層，データ層に属するオブジェクトを互いに自然に関係付けることができる．

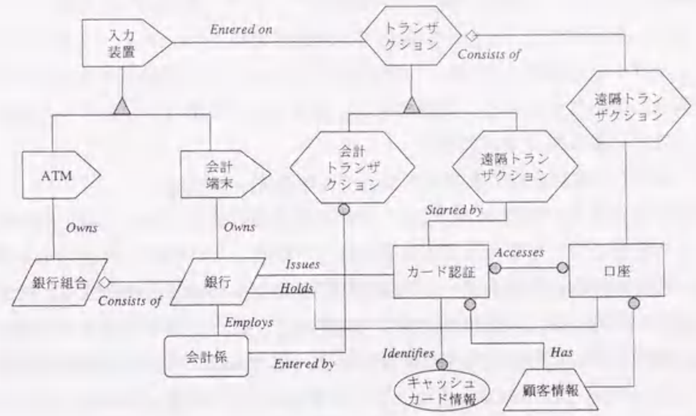


図 3－8　POOM による ATM 問題の記述例

3.4.2 分析手順

POOM では，役割オブジェクトモデルから段階的にパターン化オブジェクトモデルを作成する．以下では，POOM の分析手順を具体的に述べる．

(1) 役割オブジェクトモデルの作成

役割オブジェクトモデルでは，業務担当者間の役割に着目して，役割間の関係を明らかにする．したがって，役割オブジェクトモデル図では，役割オブジェクトをグラフ上のノードで表し，役割間の関係を2つのオブジェクト間の有向矢として記述する．役割オブジェクトAから役割オブジェクトBへ，有向関係Sがあるとき，始点の役割オブジェクトAに対して，終点の役割オブジェクトBがサービスSを提供することを表す．役割オブジェクトAは，役割オブジェクトBのサービスを利用するためのメッセージを役割オブジェクトBに対して通知する必要がある．したがって，役割間の関係は，役割オブジェクト間でのメッセー

ジの流れを分析することにより明確化できる．ここで，業務の手順ではなく役割に着目する理由は，役割という概念を用いることにより，本質的な業務だけを抽出するためである．

役割オブジェクトモデル図の作成手順は以下のようになる．まず対象となる業務に関連する担当者の役割をオブジェクトとして抽出する．もし担当者が複数の役割を持つならば，これらの役割を異なるオブジェクトとして抽出する．次に，担当者間でやりとりされる情報に着目して，役割オブジェクト間のメッセージの流れを役割オブジェクト間の関係として抽出する．役割オブジェクトモデルでは，設計者の立場ではなく，エンドユーザの立場でモデルを作成することが重要である．

役割オブジェクトモデルの記述規則を示す．

［規則 1］孤立した役割オブジェクトが存在しないこと

他の役割オブジェクトと関係を持たない役割オブジェクトを孤立した役割オブジェクトという．孤立した役割オブジェクトが発生する原因は次の 2 つである．すなわち他のどの役割オブジェクトとも関係しないか，他の役割オブジェクトと関係があるにも関わらず関係付けを失念しているかである．前者の場合には孤立した役割オブジェクトを削除する．後者の場合には必要な関係を追加する．

［規則 2］共通する役割を持つ役割オブジェクトが存在しないこと

同じ役割を持つ複数の役割オブジェクトが存在する原因として使用する用語が統一されていないため重複している場合と本来の業務自体が重複している場合の 2 つの可能性が考えられる．前者の場合は用語を統一して役割オブジェクトの重複を排除する．後者の場合はシステム化の対象としての業務を見直し，役割オブジェクトから共通する役割を分割して新たな役割オブジェクトを作成する必要がある．たとえば，共通する役割を持つ役割オブジェクトとしてＡとＢがあるとする．このとき，ＡとＢの共通する役割に対する共通オブジェクトをＣとして抽出し，ＡとＢからＣを除いた役割に対するオブジェクトをA'とB'とすることにより，役割オブジェクトＡとＢを，互いに重複する役割を持たない役割オブジェクトA', B', C を作成する．

［規則 3］役割オブジェクト間に意味のない関係が存在しないこと

役割オブジェクト間の関係が役割オブジェクト間の情報交換につながらないとき，このような関係をトランザクションに関して意味のない関係であるという．この理由は役割オブジェクト間で情報交換が発生しないのであれば，役割オブジェクト間の関係に対して情報交換のためのトランザクションを生成する必要がないことになるからである．意味のない関係の例としては，具体的な情報交換が存在しないような職場の上下関係などがある．

以下の例では販売係から会計係に販売伝票を通知し，会計係が店長に会計報告を提出する役割オブジェクトモデルを図3－9に示す．

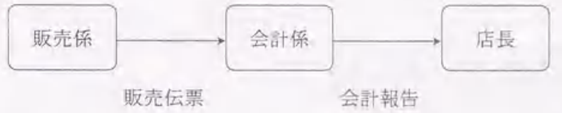


図3－9　役割オブジェクトモデルの例

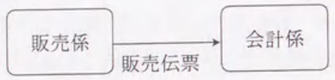
(2) メッセージ分析

役割オブジェクトモデルでは，役割間の関係がメッセージとして表現されている．メッセージには，一方の役割型オブジェクトが他方の役割型オブジェクトに対して提供するサービスの内容を規定するデータが含まれている．したがって，役割オブジェクト間の関係を分析することによって，アクセス対象型オブジェクトを抽出することができる．

また，メッセージの送り手である役割型オブジェクトでは，このアクセス対象型オブジェクトを生成するトランザクションが必要となる．メッセージの受け手である役割型オブジェクトでは，このアクセス対象型オブジェクトに基づいてサービスを提供するためのトランザクションが必要となる．このようにして，アクセス対象型オブジェクトと役割型オブジェクトに対してトランザクション型オブジェクトを抽出することができる．

以下では図 3－10（a）に示した販売係から会計係に販売伝票を通知する役割オブジェクトモデルの部分に対してパターン化オブジェクトモデルの導出過程を具体的に説明する．まず，アクセス対象として「販売伝票」を抽出する．また，販売伝票に対するトランザクションには，役割オブジェクトごとに存在する．この例では販売係が販売伝票を登録するための販売トランザクションと会計係が登録された販売伝票を参照する集計トランザクションがある．このようにしてメッセージ分析に基づいて作成されたオブジェクトモデルの例を図 3－10（b）に示す．

(a) 役割オブジェクトモデル



(b) パターン化オブジェクトモデル

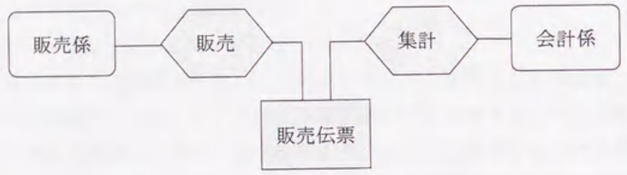


図3－10　役割オブジェクトモデルとパターン化オブジェクトモデル

(3) データ層オブジェクト分析

データ層オブジェクト分析では，メッセージ分析に基づいて抽出されたアクセス対象型オブジェクトに対して，発生源型オブジェクトや流通対象型オブジェクトを追加する．

図 3－10(b)のオブジェクトモデル図にデータ層オブジェクトを追加することを考える．販売伝票では，「商品を顧客が購入したこと」を記録するので，商品に対応する流通オブジェクト「商品情報」と，顧客に対応する発生源型オブジェクト「顧客情報」を抽出する．このようにしてデータ層のオブジェクトが拡充されたパターン化オブジェクトモデル図を図 3－11 に示す．

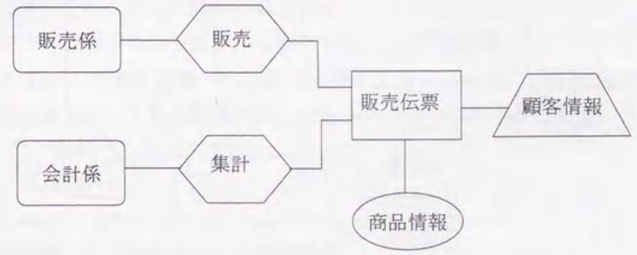


図 3－11　データ層オブジェクト分析に基づいて作成したオブジェクトモデル図

(4) 表示層オブジェクト分析

表示層オブジェクト分析では，役割型オブジェクトとトランザクション型オブジェクトに対して，トランザクション型オブジェクトに要求を出すためのアクセス境界型オブジェクトと，アクセス境界型と役割型が所属する組織としての所有者型オブジェクトを抽出する必要がある．

図 3－11 のオブジェクトモデル図に表示層オブジェクトを追加することを考える．販売トランザクションと集計トランザクションに対して，それぞれ，「販売端末」と「会計端末」を抽出する．また，所有者型オブジェクトとして「企業」を抽出する．図 3－11 のオブジェクトモデル図に対して，所有者型オブジェクトとアクセス対象型オブジェクトを追加したオブジェクトモデル図を図 3－12 に示す．

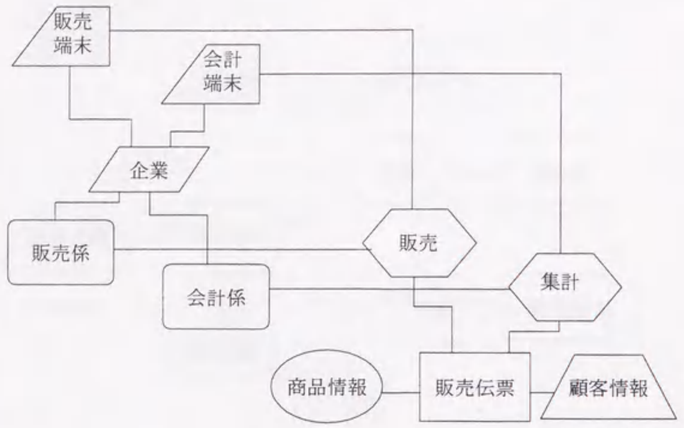


図 3－12　所有者型オブジェクトとアクセス対象型オブジェクトを追加した
オブジェクトモデル図

(5) オブジェクトモデルの洗練

オブジェクトモデルに含まれるオブジェクト間の共通性を分析して，オブジェクトの種別ごとに上位の一般化されたオブジェクトを抽出することによって，オブジェクトモデルを洗練する．たとえば基本的なオブジェクトが抽出された後で，オブジェクト間の共通性や関係の次数（1：1 関係，1：n 関係など）を分析してパターン化オブジェクトモデルを洗練する．図 3－12 のオブジェクトモデル図に対して，アクセス対象型オブジェクトとトランザクション型オブジェクトの上位オブジェクトを追加したオブジェクトモデル図を図 3－13 に示す．

上述したように，役割オブジェクトモデルからパターン化オブジェクトモデルを系統的に作成することができるので，役割オブジェクトモデルとパターン化オブジェクトモデルとの一貫性規則を次のようにして自然に定義できる．

役割オブジェクトモデルを ρ とし，ρ に対するパターン化オブジェクトモデルを π とする．このとき以下のようにして ρ と π との間での一貫性規則を定義できる．

［規則 1］ρ に役割オブジェクト A と B があり，A から B へデータ d が通知されるとき，π は以下の条件を満たす必要がある．

(1) π に，役割オブジェクト A，B ならびにデータオブジェクト d が存在する．

(2) トランザクションオブジェクト τ_a，τ_b が π に存在し A と τ_a，τ_a と d，B と τ_b，τ_b と d に関係が存在する．

(3) π の役割オブジェクト A に対して，所有者型オブジェクト γ，アクセス境界オブジェクト λ が存在し A と関係するトランザクション型オブジェクト τ に対して，γ と τ，γ と λ，γ と A との間に関係がある．

［規則 2］π に役割型オブジェクト A があるとき，ρ にも A が存在する必要がある．

［規則 3］π にトランザクションオブジェクト τ と，τ が関係するデータオブジェクト d と役割オブジェクト A があるとき，π には，d と関係する τ' ならびに，τ' と関係する B が存在する．また，ρ には，A と B が存在し，A と B の間で d に関する役割間の関係が存在する．

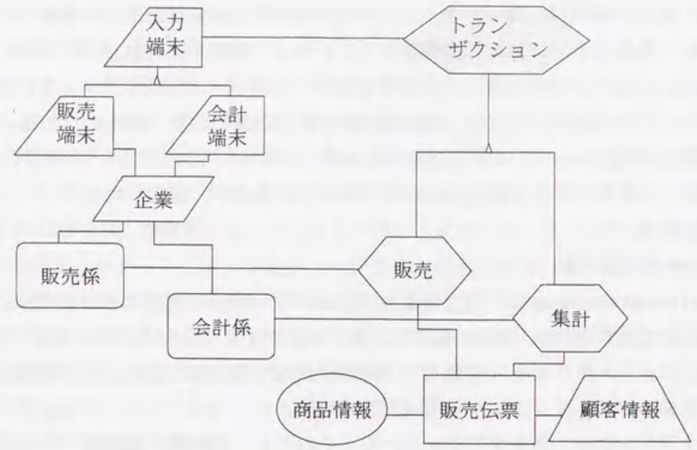


図 3－13 上位オブジェクトを追加したオブジェクトモデル図

3.4.3 従来のオブジェクト分析手法との関係

本論文が関連する研究分野として，オブジェクト指向分析手法とデザインパターンがある．以下では，POOM とパターン化オブジェクトモデルについて従来の研究と比較する．

（1）OMT 法との比較

OMT 法[18]では，本論文で提案したようにオブジェクトを明確に分類していない．したがって，オブジェクトモデルを作成する上での指針が必ずしも具体的でないため，情報システムの分析に適用する上で，分析者への負担が大きい．また，問題レベルのオブジェクトとソリューションレベルのアーキテクチャを表現するオブジェクトを区別して扱うような指針がないため，他の分析者が作成したオブジェクトモデルでは，論理的なアーキテクチャの理解や再利用が困難であるという問題がある．

（2）OOSE 法との比較

OOSE 法[22]の特徴は，ユースケースと呼ばれる問題分野の記述モデルを持つことと，オブジェクトを表示オブジェクト，制御オブジェクト，機能オブジェクトに分類していることである．エンドユーザの視点で作成されるという点で，役割割オブジェクトモデルとユースケースの目的は同じであり，相補的に用いることができる．OOSE の 3 種のオブジェクト分類は，設計レベルのオブジェクトであり，パターン化オブジェクトモデルが対象とするオブジェクトよりも実装の観点からの分類である点が，提案したオブジェクト分類との違いである．

（3）SOM 法との比較

Shared Object Modeling 法[30]は，情報システム分野へ容易に適用できるオブジェクト指向方法論として提案された．SOM 法では，ウィンドウオブジェクト，ユーザオブジェクト，共有オブジェクトからなる 3 階層のアーキテクチャを用いることにより，情報システムに関する見通しの良いオブジェクト指向設計を可能としている．ウィンドウオブジェクト，ユーザオブジェクト，共有オブジェクトを，それぞれ，表示層，機能層，データ層に対応づけることができる． SOM 法は設計レベルのオブジェクトモデルを作成するための手法であり，役割オブジェクトモデルを考慮していない．

（4）役割モデリングとの比較

さまざまなオブジェクトがある役割を持ったり，ひとつのオブジェクトが異なる局面で違う役割を持つことがある．たとえば，個人オブジェクトは，学生や従業員，顧客などさまざまな役割を持つことができる．このような問題分野の構造を柔軟に記述するためには，学生や従業員，顧客という役割を特定の個人とは独立に扱うことが必要になる．たとえば，OBA の役割モデリング[21]では，サービスを利用するオブジェクトをイニシェータ，サービスを提供するオブジェクトをパーティシパントと呼ぶ．イニシェータがパーティシパントの提供するサービスを呼び出すことがアクションである．OBA では，まず，このような，イニシェータ，アクション，パーティシパント，サービスからなる 4 項組を用いて，システムの振る舞いをスクリプトとして列挙する．次いで，このスクリプトで抽出されたイニシェータやパーティシパントを候補としてオブジェクトを抽出し，オブジェクト間の関係を定義する．

POOM では，OBA のスクリプトを役割オブジェクトモデル図で記述できる．役割オブジェクトモデル図は，役割オブジェクト間の関係を表す有向グラフである．役割オブジェクト間を接続する有向矢の始点がイニシェータで，終点がパーティシパントとなる．POOM では，有向矢のラベルとしてアクションと，アクションに付随する情報を付与する．したがって，役割オブジェクトについては，OBA が表形式で POOM が図式であるという点を除けば，両者には差がないといえる．しかし，OBA では，スクリプトからオブジェクトモデルを抽出するプロセスでオブジェクトのパターン分類を考慮していない．また，役割オブジェクトを洗練する手法として，抽象化，特殊化，統合，分解などをあげているが，イニシェータやパーティシパントをオブジェクト候補とするので，あらかじめスクリプト上でイニシェータやパーティシパントとして列挙されないトランザクションオブジェクトやデータオブジェクト等を OBA では抽出しにくいという問題がある．したがって，OBA は POOM に較べると，オブジェクトモデルの均質性やオブジェクトの粒度に個人差が出る可能性がある．また，POOM では，3.4.2 の一貫性規則で示したように役割オブジェクトモデルとパターン化オブジェクトモデルの間に明確な対応関係があり，両者の間での追跡可能性が高い．一方，POOM ではオブジェクトのパターン分類を前提としているので，OBA に比較すると POOM では，役割オブジェクトからパターン化オブジェクトモデルへの展開を規定しているので，パターン化オブジェクトモデルへの展開の適用分野が限定される可能性がある．ただし，すでに述べたように，POOM の役割オブジェクトモデルについては，OBA のスクリプトと同等なので，両者の適用分野に差があるとはいえない．これらの点を厳密に検証するために，実証実験を行う必要がある．

3.4.4 デザインパターンとの関係

（1）デザインパターンとフレームワーク[31]

ソフトウェア・アーキテクチャの再利用を目的とした個々の具体的な設計問題に対するソリューションがデザインパターンである．デザインパターンでは，ある分野の問題に対するソリューションで必要となる一般的なコンポーネントの構造やコンポーネント間の関係を定義している．したがって，デザインパターンでは，あるソフトウェア・アーキテクチャにおけるコンポーネントの構造やコンポーネント間の関係を，特定のオブジェクトやクラスに関するオブジェクト指向言語によるソースコードよりも高いレベルで表現している．このため，デザインパターンでは，論理設計レベルの開発知識を表現しており，特定の OS やプラットフォームが異なる場合でもデザインパターンを再利用することができる．これに対して，アルゴリズムや詳細設計レベルの知識の場合，特定の OS やプラットフォームに依存するので，再利用できる範囲が限定される．このように，デザインパターンの目的は，論理設計レベルでソフトウェア・アーキテクチャを再利用することである，換言す

ると，実装によって影響を受けない論理的なアーキテクチャがデザインパターンであると言えよう．このようなデザインパターンに対して，フレームワークでは，論理設計レベルでソフトウェア・アーキテクチャを再利用するのではなく，より，直接的にオブジェクトの設計やコードを再利用できる．デザインパターンとフレームワークの異なる点は，デザインパターンが言語に依存しないのに対してフレームワークが特定の言語で実現されていることである．

(2) オブジェクト分類とデザインパターン

すでに述べたように，役割オブジェクトモデルでは，ソフトウェア・アーキテクチャではなく問題分野における役割オブジェクトとそれらの相互関係を定義しており，問題の構造を明らかにするためのモデルである．これに対して，デザインパターンでは，論理設計レベルでソフトウェア・アーキテクチャを定義している．また，パターン化オブジェクトモデルでは，情報システム分野における階層型クライアント／サーバ・アーキテクチャを前提としてアプリケーションで必要となるオブジェクトを定義している．Gamma ら[32]は，情報システムのトランザクションの実現で利用できるデザインパターンとして，コマンド・パターンを示した．このように，デザインパターンは，実現法を対象としているのに対して，オブジェクトモデルでは，デザインパターンで実現される対象を分析している．

3.5 まとめ

3.2 では，データフローの入出力関係を表現する行列を論理ベクトルの集合と見なすことにより，論理ベクトルの大小関係に基づいてデータフロー図を一意的に生成する決定性アルゴリズムを提案した．このアルゴリズムの特徴は，以下の 3 点である．

［特徴 1］入出力データの関係を満足する最小のデータフロー図を一意的に自動生成できる．

［特徴 2］従来手法に較べ，効率的である．

［特徴 3］任意の入出力データの関係に適用できる．

したがって，本方式は，任意の入出力データの関係に対してデータフロー図を効率よく自動生成できるので，実用性が高い．また，入出力関係行列からデータフロー図を生成するだけでなく，次のような応用が考えられる．

(1) 複数のデータフロー図の合成手法

互いに共通部分を持つ複数のデータフロー図を 1 個の データフロー図に矛盾なく合成することができれば，データフロー図の共通化や統合が容易になり，ソフトウェアの保守作業を軽減できる．たとえば，旧版のデータフロー図 G に対して別々の拡張が施されたデータフロー図 G_1, G_2 がある場合，G_1, G_2 を統合して G の新版を作る場合などが考えられる．このような場合，本方式を用いることにより，複数のデータフロー図を以下のようにして，効率よく合成できる．すなわち，まず，各データフロー図 G_k に対する入出力関係行列 D_k を求め，D_k を合成した入出力関係行列 D を作成する．D_k 間の互いに矛盾しない入出力関係

については，各 D_k の和として D の入出力関係を作る．もし，D_k 間で入出力関係に矛盾があれば，矛盾する入出力関係についてともに削除するか，あるいは矛盾する入出力関係の一方だけを残すことにより，D の入出力関係を作成する．このようにして作成された D に本方式を適用することにより，複数のデータフロー図 G_k を合成したデータフロー図 G を自動生成することができる．

(2) データフロー図の確認手法

人手で作成したデータフロー図の場合，データフロー図で規定される入出力データ間の依存関係が明示的に記述されていないため，人手による入出力関係の確認が必要である．Modell は，システムの入力から始めて，すべてのデータフローを最終出力まで段階的に確認していく入出力確認法を示している[24]．入出力確認法では，入力データをデータフローに従って最終出力となるデータを見つけるまで追跡を続ける．もし，下位のデータフロー図に分割されているプロセスによって追跡中のデータが処理されていれば，そのプロセスの出力データを見つけるまで下位のデータフロー図上で追跡を続ける．

本論文で述べた入出力関係行列からデータフロー図を生成する方式を用いることにより，もし入出力関係行列に誤りがなければ，妥当なデータフロー図が生成されるので，このような人手による確認作業の必要はない．また，人手で作成された既存のデータフロー図がある場合については，本方式を応用することにより，入出力確認作業を自動化できる．すなわち，人手で作成された階層的なデータフロー図から入出力関係行列を導き，各入出力関係行列の適合性条件を検証することにより，入力と対応しない出力や，出力と対応しない入力を検出できる．

3.3 では，モジュール構造図から入出力関係に基づいてデータフロー図を復元するアルゴリズムを提案した．大規模ソフトウェア開発ではソフトウェアの分析設計手法の経験者数が限定される．ところが，初心者の場合，誤りや冗長なデータフロー図を作成し易いという問題がある．提案したアルゴリズムを用いることにより，モジュール構造図から最適なデータフロー図を生成できるので，初心者でも標準的なデータフロー図を作成でき，ソフトウェアの保守工数を削減できる．今後の課題としては，データフロー図やモジュール構造図と入出力関係との一貫性の管理や，修正時の影響波及分析などがある．また，エンティティ・リレーションシップ図など他の要求分析図式と統合した要求分析のリバースエンジニアリング手法の検討も今後の課題である．

3.4 では，役割関係に基づくオブジェクト指向分析手法を提案した．POOM には，役割関係モデルとパターン化オブジェクトモデルの 2 つのオブジェクトモデルがある．役割関係モデルは，特定の分野に限定されてはいない．パターン化オブジェクトモデルでは，OMT の教科書の ATM の例題に基づいて一般化した情報システム分野を対象とした 7 種類のオブジェクトパターンを使用している．この点では，提案した 7 種類のオブジェクト分類に属さないようなシステム分野については，本論文のオブジェクト分類では限界があると思われる．しかし，その場合でも適切なオブジェクト分類を導入することによりこの問題は解

決できると思われる．また，7種類のオブジェクトは，役割関係モデルから自然な展開により抽出できるオブジェクトであることから，役割関係モデルの適用分野については POOM も適用できると思われる．7種類のオブジェクトを表示層，機能層，データ層に分類しているが，この分類自身も一般的なものである．これらの点を考慮すると POOM の適用分野は情報システム分野に限定されないと思われる．しかし，制御システム分野などへのパターン化オブジェクトモデルの適用性については未検証であり，今後の評価が必要である．また，POOM 手法では分析者によるオブジェクトモデルの変換手順を示しているが自動化までにはいたっていない．したがって役割関係に基づいてオブジェクトモデルを自動生成する研究を進める必要がある．

データフロー図とオブジェクトモデルとの関係について考察する．データフロー図は外部境界ノード，プロセスノード，データストアノードとノード間のデータフローから構成される．オブジェクトモデルは表示層オブジェクト，機能層オブジェクト，データ層オブジェクトとオブジェクト間の関係から構成される．したがって，次のようにしてデータフロー図とオブジェクトモデルとを対応付けることができる．外部境界ノードを表示層オブジェクトに対応付ける．プロセスノードを機能層オブジェクトに対応付ける．データストアノードをデータ層オブジェクトに対応付ける．ノード間のデータフローをオブジェクト間の関係に対応付ける．この対応付けは 3 層アーキテクチャを前提とすることにより可能になったものである．

従来からデータフロー図を用いてオブジェクトモデルを作成する手法が考えられてきた．たとえば，データストアをオブジェクトの属性とし，そのデータストアとデータフローで接続されたプロセスをオブジェクトの操作とする方法が考えられる．しかし，この方法ではオブジェクトの型を考慮していない．また，3層アーキテクチャも考慮していない．

3 層アーキテクチャに基づいてデータフロー図とオブジェクトモデルとを対応付けることにより構造化分析とオブジェクト指向分析とを比較研究する必要があると思われる．たとえばデータフロー図とオブジェクトモデルの相互変換条件を 3 層アーキテクチャに基づいて明らかにする必要がある．

4.1 まえがき

(1) 従来の WWW を用いたハイパーメディアシステムの設計図式では WWW ページ間のハイパーリンクとアプリケーションを起動するための機能的なイベントを区別していない点や処理結果に応じて異なるページへの遷移を記述できないなどの点で問題がある．本論文では 3 層アーキテクチャに基づく WWW 情報システムの表示層インタフェースを作成する手法としてページフロー図とその記述規則を提案する．

(2) 3 層アーキテクチャを構成する表示層、機能層、データ層の処理と層間インタフェースを定義するためのシナリオフロー図を提案する。次いでページフロー図とシナリオフロー図に基づいて 3 層アーキテクチャに基づく情報システムを作成する手順を提案する。本手法により要求分析モデルと 3 層アーキテクチャを構成する各層のコードとの対応を明確化できるだけでなく、各層のコード間のインタフェースを製造工程に先立って定式化できるので開発上の重複や手戻りを抑止できる。

(3) 構造化設計法では，変換分析手法に従ってデータフロー図からモジュール構造図を作成する．しかし，変換分析手法に従って作成されたモジュール構造図は，データフロー図のデータの流れに従ってモジュールを配置するため，階層が深くなりやすく，設計者が作成するモジュール構造図と必ずしも一致しないという問題がある．本論文では，設計者が作成するモジュール構造図に近い形でモジュール構造図を生成するアルゴリズムを提案する．

(4) モジュール構造図に基づくシナリオフロー図作成法

従来の構造化設計手法では 3 層アーキテクチャへの階層分割手法を考慮していないと言う問題があった．このため構造化手法で開発された情報システムを 3 層アーキテクチャへの移行法が明らかではなかった．本論文ではモジュール構造図に基づくシナリオフロー図作成法を提案する．

(5) オブジェクト指向分析法に基づくシナリオフロー図作成法

従来は要求分析段階では 3 層アーキテクチャを考慮しておらず，コード化段階ではじめて 3 層にプログラムを分割していたため，層間への処理の配置を検討する上での負担がコード化工程に集中するだけでなく，要求分析とコードとの間での追跡性が低いという問題があった．本論文ではパターン化オブジェクトモデルに基づくシナリオフロー図の作成法を提案し，3 層アーキテクチャに基づくオブジェクト指向型の情報システム設計法を具体化する．本手法により要求分析工程から段階的に 3 層アーキテクチャ設計を進めることができるので作業上の負荷を各工程に平準化できるとともに工程生産物間の追跡性を向上できるという特徴がある．

本章の構成は次の通りである．4.2 でページフロー図作成法を述べる．4.3 でシナリオフロー図の作成法を述べる．4.4 でデータフロー図に基づくモジュール構造図作成法を述べる．

4.5 でモジュール構造図に基づくシナリオフロー図作成法を述べる．4.6 ではオブジェクト指向分析法に基づくシナリオフロー図作成法を述べる．

4.2 ページフロー図の作成法

従来から WWW を用いたハイパーメディアシステムの設計図式として Bichler らにより SHDT 図が提案されている[1]．しかし WWW ページ間のハイパーリンクとアプリケーションを起動するための機能的なイベントを区別していない点や処理結果に応じて異なるページへの遷移を記述できないなどの点で問題がある．本論文では Bichler らの WWW ページ設計図式を拡張したページフロー図[2][3]により 3 層アーキテクチャに基づく WWW 情報システムの表示層インタフェースを作成する手法とその記述規則を提案する．

ページフロー図の目的は，WWW の各ページの入出力特性に関する情報と，各ページ間の関係を明らかにすることにより，ユーザインタフェース仕様を定義することである．このためページフロー図では WWW による画面入出力情報および画面遷移を明らかにする．図 4−1 にページフロー図の例を示す．

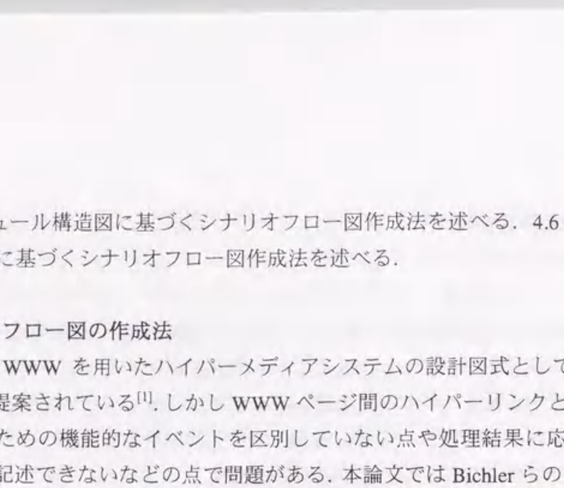


図 4−1 ページフロー図の例

画面遷移イベントには，異なる画面への状態遷移と同一画面内での状態遷移の 2 つがある．後者の例として，ある画面内で複数の情報フィールドごとにデータを入力して，その都度ボタンをクリックしながら処理を実行する業務がある．入力イベントに対してデータベースアクセスや業務処理を行い新たなページを動的に生成して結果を表示するとき，こ

の入力イベントは機能イベントである．ページフロー図では，予め用意された固定的なページと，スクリプトなどにより動的に生成されるページを区別し，動的なページへの有効矢のラベルとして機能イベント名を付与する．

ページフロー図では WWW のページ特性をリンクページ，親ページ，選択ページ，入力ページに分類している．リンクページは，他のページへのリンクを持つページである．親ページは，複数の子ページに詳細化されるページである．選択ページは一覧表から項目を選択するページである．入力ページは情報の入力項目を持つページである．ページ間の遷移関係には，静的に常に遷移関係が存在する場合と，ある条件が成立したときに限り動的に遷移関係が発生する場合がある．このようなページフロー図の表記法をまとめて表 4−1 に示す．

表 4−1 ページフロー図の表記法

記号		記号の説明
ページ記号		予め用意された固定的なページであることを示す
		スクリプトなどから動的に生成されるページであることを示す
		上位のページへ復帰することを示す
ページ特性記号	◊	他のページへのリンクを持つページであることを示す
	○	複数のページに詳細化されるページであることを示す
	▤	一覧表から項目を選択するページであることを示す
	▭	情報入力項目を持つページであることを示す
ページ関係記号	⟶	ページ間の遷移関係が常に存在することを示す
	⟶	ページ間の遷移関係がある条件の下で存在することを示す

ページフロー図の設計手順は次のようになる．

［手順 1］情報システムで必要となる機能の概要を概要ページフロー図で記述する．

図 4−1 の例では，問題処理票の管理には発行，受け付け，対処，承認，検索があることを概要ページフローで記述している．

［手順 2］概要ページフロー図の構成要素として記述された各ページについて詳細ページフロー図で具体的なページ構成とページ間の遷移を記述する．

図 4−1 の例では問題処理票の検索ページ処理の内容が詳細ページフロー図で記述されている．

4.3 シナリオフロー図の作成法

3層アーキテクチャでは，表示層処理，機能層処理，データ層処理ならびに表示層機能層間インタフェース（P-F インタフェース），機能層データ層間インタフェース（F-D インタフェース）により，システムを構成する．表示層機能層間インタフェースと機能層データ層間インタフェースを総称して，層間インタフェースと呼ぶ．この層間インタフェースは，他の層への処理要求と応答結果の対からなる．処理要求は，インタフェースを一意に識別し機能を特定する識別子と，機能に対する入力情報からなる．応答結果は，機能の処理結果を判定するための制御情報と出力情報からなる．この3層アーキテクチャに従えば，3層クライアント／サーバアプリケーションの設計問題は，与えられた要求仕様に対して，適切な P-F インタフェースと F-D インタフェースの集合を抽出することと言える．

本論文では，層間インタフェースを記述するための表記法としてシナリオフロー図を提案する[4][5][6]．シナリオフロー図はモジュール構造図から作成する方法とオブジェクトモデル図から作成する方法とがある．いずれの場合でもページフロー図に基づいて，表示層インタフェースを抽出しておく．以下では，シナリオフロー図と層間インタフェース設計手順について述べる．

シナリオフロー図の目的は，3層に分割された処理の組合せと，各層間の入出力情報フローを明らかにすることにより，ユーザインタフェースで示された処理要求と応答結果の対を実現できることを論理的に示すことである．

このような層間インタフェースと各層の処理を記述するために，シナリオフロー図では，各層ごとに，インタフェースを記述するインタフェースブロックと，このインタフェースに対応する処理を記述する処理ブロックを設けた．

インタフェースブロックでは，処理ブロックへの処理要求と結果応答の対を記述する．処理ブロックでは，インタフェース名称と，処理要求を応答結果に変換するための処理を記述する．表示層インタフェースブロックでは，表示層処理に対する処理要求と応答結果の対を記述する．以下では，表示層インタフェースを分かり易さのために，ユーザインタフェースと呼ぶことがある．機能層インタフェースブロックでは，機能層処理に対する処理要求と応答結果の対を記述する．データ層インタフェースブロックでは，データ層処理に対する処理要求と応答結果の対を記述する．機能層インタフェースブロックでは，P-F インタフェースを記述する．データ層インタフェースブロックでは，F-D インタフェースを記述する．

シナリオフロー図では，データ層処理ブロックが操作するデータ実体を明らかにするために，データブロックを設けている．

シナリオフロー図では，各層間で発生する情報のフロー関係を，インタフェースの利用側となる処理ブロックから提供側となるインタフェースブロックへの矢線で示す．

シナリオフロー図の表記法をまとめて表4−1に示す．また，シナリオフロー図の例を図4−2に示す．

表4−2　シナリオフロー図の表記法

記号	説明
⟶	データフロー．要求（request）と応答（response）がある
	処理ブロック．処理内容を記述する． 表示層，機能層，データ層ごとに　処理ブロックを作成する
	データブロック．データ実体を記述する
	外部システム．

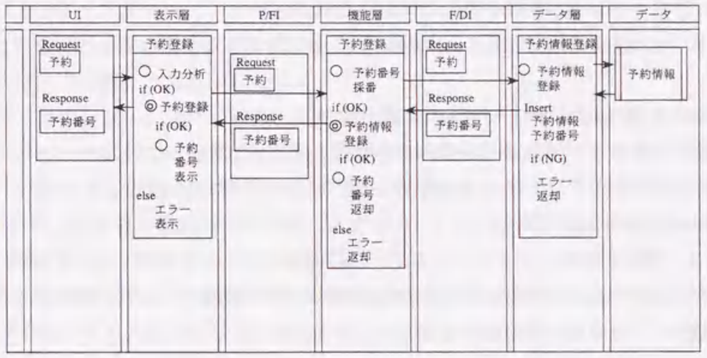


図4−2　シナリオフロー図

画面操作で発生するイベントには，データベース操作を伴う機能イベントとデータベース操作を伴わない表示イベントがある．たとえば，フィールドに入力した情報に基づいてデータベースの内容を更新する機能に対応するイベントは，機能イベントである．これに対して，画面の一部をクリックすることにより，選択した部分が反転表示されるような機能に対応するイベントは，表示イベントである．機能イベントに対してシナリオフロー図を作成する．表示イベントについては，表示層に閉じた処理であるため，シナリオフロー図を作成する必要はない．

［設計手順 1］層間インタフェース設計

［ステップ 1］表示層の設計

［ステップ 1−1］ユーザインタフェース決定

ページフロー図から機能イベントを抽出し，対応するシナリオフロー図の名称を決定する．抽出した機能イベントの入出力情報（要求情報と応答結果）を決定し，シナリオフロー図のユーザインタフェースブロックに記述する．次いで，表示層処理ブロックで，機能イベントに対応する処理要求と結果応答の対を識別するための表示インタフェース名称を付与する．

このとき，以下の設計指針に基づいて，可能な限り独立な機能イベントを抽出する．

［設計指針 1］機能イベントの独立性

異なる機能イベントが互いに同じ要求情報と応答結果を持たないこと

［設計指針 2］機能イベントの単一性

一つの機能イベントが異なる業務に対応しないこと

また，以下の設計指針に基づいて，機能イベントに対応する最小な表示インタフェースを抽出する．

［設計指針 3］表示インタフェースのの最小性

(1) 異なる表示インタフェースが一つの機能イベントに対応しないこと

(2) 異なる機能イベントが一つの表示インタフェースに対応しないこと

この理由は，次の通りである．

(1) もし，異なる表示インタフェースが一つの機能イベントに対応するとすると，それぞれの表示インタフェースが対応する異なる業務が一つの機能イベントに対応することになり，機能イベントの単一性に反する．

(2) もし，異なる機能イベントが一つの表示インタフェースに対応するとする．このとき，機能イベント E_i，E_j の入出力が異なるので，各機能イベントが異なる業務に対応する．したがって，異なる業務の切り替え処理を表示インタフェース処理が持つことになり，表示層が業務処理を含むことになる．これは，表示層と機能層の処理の独立性に反する．

［ステップ 1−2］処理種別決定

機能イベントに対するデータベース処理をデータベース操作の種別で分類することにより，機能イベントの処理種別を，挿入，更新，削除，参照の 4 種のいずれかに分類する．この理由は，機能イベントの入出力情報が一致する場合でも，データベースの操作が異なる場合には，処理が異なるため，機能イベントに対する処理ブロックを区別しておく必要があるからである．

［ステップ 1−3］表示層処理ブロックの作成

表示層では，画面入出力のみに依存する処理を可能な限り機能層処理を利用して作成する．表示層処理ブロックの基本的な構成は，ユーザインタフェースからの情報の入力，P-F インタフェースの呼び出し，結果情報の表示である．情報入力では，入力結果のチェックとチェック結果の表示処理が必要になる．P-F インタフェースを呼び出すためには，まず，P-F インタフェースを決定する必要がある．P-F インタフェースは，処理要求と結果応答の対から構成される．処理要求は，P-F インタフェース識別子，機能への入力データと機能を制御するためのフラグからなる．結果応答は，出力データと処理結果の状態を示すためのフラグからなる．ユーザインタフェースの機能イベントの入出力情報から表示処理に依存する情報を捨象することにより，可能な限り論理的な入出力データを抽出し，これを P-F インタフェースの入出力データとする．P-F インタフェース識別子は，出力データの内容が機能を規定する場合が多いことに着目して，出力データの内容と機能イベントの処理種別の組合せにふさわしい適切な名称を付与する．

このとき，以下の設計指針に基づいて，可能な限り最小な P-F インタフェースを抽出する．

［設計指針 4］ P-F インタフェースの最小性

(1) 異なる P-F インタフェースが一つの表示インタフェースに対応しないこと

(2) 異なる表示インタフェースが一つの P-F インタフェースに対応しないこと

この理由は，次の通りである．

(1) もし，異なる P-F インタフェースが一つの表示インタフェースに対応するとすると，それぞれの P-F インタフェースが対応する異なる業務が一つの表示インタフェースに対応することになる．したがって，異なる業務の切り替え処理を表示インタフェース処理が持つことになり，表示層が業務処理を含むことになる．これは，表示層と機能層の処理の独立性に反する．

(2) 異なる表示インタフェースが一つの P-F インタフェースに対応するとする．このとき，P-F インタフェースの入出力は一定であるから，それぞれの表示インタフェースが対応する機能イベント E_i，E_j の入出力が同一になる．これは，機能イベントの独立性に反する．

結果情報の表示には，出力データの表示処理と，P-F インタフェースの処理結果状態に対する表示処理がある．処理結果の状態を決定するためには，入力情報と実体情報の組合せを明らかにする必要がある．この組合せは，機能層処理ブロックで決定されるので，機能層処理ブロックの内容が設計された段階で，すべての応答結果の場合を列挙して，それぞ

れに対するユーザインタフェースでの表示処理を表示層処理ブロックで設計する．

［ステップ 2］機能層の設計

［ステップ 2−1］F-D インタフェースの抽出

機能層処理ブロックの基本的な構成は，P-F インタフェースからの情報の入力，F-D インタフェースの呼び出し，P-F インタフェースへの応答結果の出力である．情報入力では，入力結果を各 F-D インタフェースに分ける分配処理が必要になる．情報出力では，各 F-D インタフェースの応答結果を統合して P-F インタフェースへの応答結果として統合する統合処理が必要になる．F-D インタフェースを呼び出すためには，まず，F-D インタフェースを決定する必要がある．F-D インタフェースも，P-F インタフェースと同様に，処理要求と結果応答の対から構成される．処理要求は，F-D インタフェース識別子，機能への入力データと機能を制御するためのフラグからなる．結果応答は，出力データと処理結果の状態を示すためのフラグからなる．

P-F インタフェースで与えられた出力データについて，出力データを作成するために必要な実体をＥＲ図から抽出し，実体のアクセス順序および，出力データを作成するための検索要求と検索結果を各実体ごとに明らかにする．抽出した実体と検索要求，検索結果ごとに，F-D インタフェースを作成する．

F-D インタフェースの入出力データは，実体の属性についての検索条件と実体の属性だけから作成されるデータである．F-D インタフェース識別子に対して，実体名と機能イベントの処理種別の組合せにふさわしい適切な名称を付与する．

［ステップ 2−2］トランザクション範囲決定

実体のアクセス順序に基づいて，1 つのトランザクションで処理すべき実体の集合をトランザクション範囲として抽出し，各実体に対するトランザクションの一貫性を保証する処理を機能層処理ブロックで作成する．

F-D インタフェースの組合せから，機能層処理ブロックの処理結果の状態を決定する．このとき，機能イベントが必要とする情報を扱う実体に対するトランザクションは機能イベントに対応する機能層処理に含まれていなくてはならない．また，機能層処理に含まれるトランザクションが操作する実体は機能イベントが必要とする情報を扱う実体でなくてはならない．また，F-D インタフェースを実体へのアクセスとしたので，実体間の一貫性保証を機能層で処理する必要がある．このとき，以下の設計指針に基づいて，可能な限り最小な F-D インタフェースを抽出する．

［設計指針 5］ F-D インタフェースの最小性

(1) 異なる F-D インタフェースの組合せが一つの P-F インタフェースに対応しないこと

(2) 異なる P-F インタフェースが一つの F-D インタフェースの組合せに対応しないこと

この理由は，次の通りである．

(1) もし，異なる F-D インタフェースの組合せが一つの P-F インタフェースに対応するとすると，それぞれの F-D インタフェースが対応する異なる実体が一つの P-F インタフェースに対応することになる．このとき， P-F インタフェースの入出力は一定であるから，一つの入出力の組に対して，異なる実体集合の組合せが存在することになり，データモデルの最小性に反する．

(2) 異なる P-F インタフェースが一つの F-D インタフェースの組合せに対応するとする．このとき，F-D インタフェースの組合せに対する入出力は一定であるから，それぞれの P-F インタフェースが対応する機能イベント E_i，E_j の入出力が同一になる．これは，機能イベントの独立性に反する．

また，以下の設計指針にしたがって，F-D インタフェースにおける入出力データの操作回数を可能な限り最小化する．

［設計指針 6］ F-D インタフェース入出力の最小性

(1) 実体の要素に対する操作があるとき，必要なデータだけ交換するような実体への操作に変換する

(2) 異なる実体操作間で中間データの持ち回り操作があるとき，これらの実体をグループ化して中間データの持ち回り操作を省略し，結果だけを取得する操作に変換する

［ステップ 2−3］機能層処理ブロックの作成

機能層では，表示処理で必要なデータ操作の結合，データの計算処理を可能な限りデータ層処理を利用して作成する．

［ステップ 3］データ層処理ブロックの作成

データ層では，F-D インタフェースの機能を可能な限り実体操作を用いて作成する．また，外部システムとの入出力操作を作成する．

（設計手順終わり）

以下では図 4−3 に示す商品受注情報照会に対するページフロー図に基づいてシナリオフロー図を作成する手順を具体的に示す．

商品受注情報照会の機能イベントの処理種別は「参照」である．このとき入力は「顧客 Id」であり，「顧客名」と「受注商品一覧」が正常時の出力である．例外時の出力は「例外内容」である．したがって表示層のユーザインタフェースとしてこれらを記述する．P-F インタフェースとしての入出力は，「顧客 Id」であり，「顧客名」と「受注商品一覧」が正常時の出力である．例外時の出力は「例外 Id」である．ここで「例外 Id」は機能層処理から表示層処理に返却されるコードであり，表示層処理はこの「例外 Id」に基づいて「例外内

容」をユーザが理解しやすい形式で提示する．

次に F-D インタフェースを抽出する．商品受注情報を検索するためには「顧客」「受注」という 2 つのデータベースの実体にアクセスする必要がある．ここで「受注」には顧客から受注した商品が受注案件ごとに「顧客 Id」と「商品名」とを対応付ける形で登録されているとものする．このとき商品受注情報を顧客ごとに参照するためには，まず指定された「顧客 Id」が存在するかどうか「顧客」実体にアクセスして確認してから「受注」実体にアクセスして「顧客 Id」に関する受注商品の一覧を検索する必要がある．したがって，データ層にデータ実体として，まず「顧客」を配置しその下に「受注」を配置する．次にデータ層処理として顧客情報照会と受注商品情報照会を順に配置する．この 2 つのデータ層処理ごとに，F-D インタフェースを作成する．顧客情報照会に対する F-D インタフェースは入力が「顧客 Id」で正常時の出力が「顧客名」，例外時の出力が「顧客 Id 例外」である．受注商品情報照会に対する F-D インタフェースは入力が「顧客 Id」で正常時の出力が「受注商品名」，例外時の出力が「受注例外」である．

このようにして図 4−3 のページフロー図に基づいて作成されたシナリオフロー図を図 4−4 に示す．

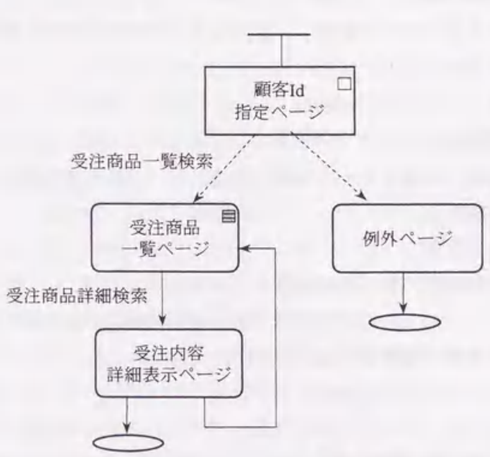


図 4−3　受注商品情報照会に対するページフロー図

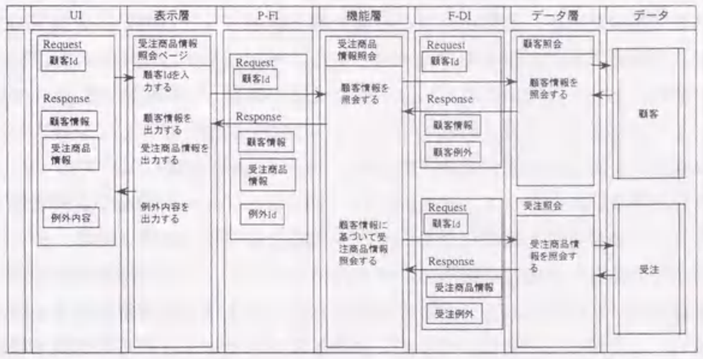


図 4−4　受注商品情報照会に対するシナリオフロー図

ここで提案したシナリオフロー図に基づく 3 層システム設計手法（ParaDISE）の適用例には OCN 運用管理業務支援システムやゴルフ場予約システムなどがある[6][7][8]．OCN 運用管理業務支援システムは NTT が開始したインターネット系サービスのための業務サイクル全般を支援するためのシステムでありネットワーク設備，サービスオーダ，障害，設備計画などの情報を管理する．OCN 設備管理システムの予想開発規模約 40.6 人月に対して ParaDISE の適用により実際の開発工数を約 26.5 人月に押さえることができた．したがって ParaDISE の適用により開発工数を約 35%削減できたことなる．この理由はシナリオフロー図により約 60%のモジュールの開発を抑止できておりモジュールの共通化を推進できたためである．システムの開発規模に比例してモジュール共通化の効果が大きくなるので ParaDISE は大規模システム開発に適していると考えられる．また，シナリオフロー図は NTT コミュニケーションウェアに実用的な 3 層クライアント／サーバ設計技法として導入されており，同社において各種の大規模な分散システム開発に適用されている[9]．

4.4　データフロー図からのモジュール構造図の生成方式

構造化設計法には，データフロー図からモジュール構造図を作成する変換分析手法がある[10][11][12][13]．CASE ツールの中には，この変換分析手法に従ってモジュール構造図を作成

できるものもある[14][15]が，設計者が作成するモジュール構造図と階層が深くなるなどの問題がある．

変換分析では，まずデータフロー図のプロセスを入力プロセス，変換プロセス，出力プロセスの 3 種に分類する．次に，変換プロセスの中から変換中心プロセスを選択する．変換中心プロセスが決まると，変換中心プロセスを最上位モジュールとし，変換中心プロセスに隣接するプロセスをその下位モジュールとする．モジュールに変換されたプロセスに対して，まだモジュールに変換されていないプロセスが隣接しているとき，そのプロセスを，隣接するプロセスから生成されたモジュールの下位モジュールとして変換する．この手順を繰り返すことにより，すべてのプロセスをモジュールに変換していく．

この変換分析手法にしたがって，データフロー図からモジュール構造を生成する CASE ツールがある．しかし，この変換分析手法で作成されるモジュール構造図は，データフロー図の最長パスの長さに対応した階層の深さを持つのでモジュールの階層が深くなり易いという問題がある．これに対して，設計者が作成するモジュール構造図の階層の深さは浅いことが多い．したがって，変換分析手法で自動生成されたモジュール構造図を利用する場合，階層を浅くするなどの手直しが必要となる．また，データフロー図のプロセス間に複雑なデータフロー関係がある場合，従来の CASE ツールでは，データフロー関係をそのままモジュール間の呼び出し関係に変換するので複雑な構造を持つモジュール構造図が生成されるという問題もあった．

本論文では，プロセス間のデータフロー関係に基づいて変換分析で用いられる規則だけでなく階層を平坦化する変換規則を用いることにより自然なモジュール構造図を作成できる．以下ではアルゴリズムの内容について述べ，このアルゴリズムの有効性に関する実験的な評価については第 6 章で述べる．

提案するモジュール構造図生成アルゴリズムは以下の手順からなる[17]．

(1) データフロー図のプロセス種別を推定するアルゴリズム

(2) プロセス種別と，プロセス間のデータフロー関係に従ってデータフロー図をモジュール構造図に変換するアルゴリズム

4.4.1 プロセス種別の推定アルゴリズム

本アルゴリズムでは，データフロー図を外部境界とデータストアに隣接するプロセスから出発して内側に向かって解析していくことにより，入力プロセス，変換プロセス，出力プロセスに分類する．

まず，外部境界ノードの集合を E，プロセスの集合を P で表す．ノード n からノード m へのデータフローがあることを，n⇒m で表す．ノード集合 A，B の和集合および差集合を A∪B および A−B で表す．要素 n が集合 S に含まれることを n∈S で表す．

［アルゴリズム 1］ プロセス種別推定アルゴリズム

［ステップ 1］ 初期化

探索済みノード集合 Visit←E∪D

［ステップ 2］ 停止条件を判定

Visit が E∪D∪P のとき，終了．そうでない限り，以下を繰り返す．

［ステップ 3］ 探索段階

Next←{ p |（p∈ E∪D∪P−Visit）かつ（n∈Visit）かつ（n⇒p または p⇒n）}

［ステップ 4］ 種別を推定

p∈Next についてプロセス種別を推定する．

(1) n⇒p となる n∈Visit が存在するが，p⇒n となる n∈Visit が存在しないとき，p のプロセス種別を入力とする

(2) p⇒n となる n∈Visit が存在するが，n⇒p となる n∈Visit が存在しないとき，p のプロセス種別を出力とする

(3) n⇒p となる n∈Visit および，p⇒n となる n∈Visit がともに存在するとき，p のプロセス種別を変換とする

［ステップ 5］ 次の Visit を計算

Visit ←Visit∪Next

ステップ 2 に戻る．

（アルゴリズム終わり）

［具体例］ 以下のようなノード集合からなるデータフロー図についてプロセス種別を推定する．

外部境界ノード集合 E＝ $\{e_1, e_2\}$

データストアノード集合 D＝ $\{f_1, f_2\}$

プロセスノード集合 P＝ $\{P_1, P_2, P_3, P_4, P_5, P_6, P_7\}$

データフロー関係 R＝ $\{e_1 \Rightarrow P_1, P_1 \Rightarrow P_2, P_2 \Rightarrow P_3, P_3 \Rightarrow P_5, P_7 \Rightarrow P_3, f_1 \Rightarrow P_7, P_3 \Rightarrow P_4, P_4 \Rightarrow f_2, P_5 \Rightarrow P_6, P_6 \Rightarrow e_2\}$

このデータフロー図を図 4−5 に示す．

［段階 0］ 初期化．Visit ← $\{e_1, e_2, f_1, f_2\}$

［段階 1］

Next← $\{P_1, P_7, P_4, P_6\}$

$e_1 \Rightarrow P_1$ かつ $P_1 \Rightarrow n \in$ Visit が存在しないから，P_1 のプロセス種別を入力とする．$f_1 \Rightarrow P_7$ かつ $P_7 \Rightarrow n \in$ Visit が存在しないから，P_7 のプロセス種別を入力とする．$P_4 \Rightarrow f_2$ かつ $n \Rightarrow p_4$ となる $n \in$ Visit が存在しないから，P_4 のプロセス種別を出力とする． $P_6 \Rightarrow e_2$ かつ $n \Rightarrow P_6$ となる $n \in$ Visit が存在しないから，P_6 のプロセス種別を出力とする．

Visit← $\{e_1, e_2, f_1, f_2\} \cup \{P_1, P_7, P_4, P_6\}$

［段階 2］

Next← $\{P_2, P_3, P_5\}$

$P_1 \Rightarrow P_2$ かつ $P_2 \Rightarrow$ n∈Visit が存在しないから，P_2 のプロセス種別を入力とする．$P_7 \Rightarrow P_3$ かつ $P_3 \Rightarrow P_4$ となるから，P_3 のプロセス種別を変換とする．$P_5 \Rightarrow P_6$ かつ $n \Rightarrow P_5$ となる n∈Visit が存在しないから，P_5 のプロセス種別を出力とする．

Visit← $\{e_1, e_2, f_1, f_2, P_1, P_7, P_4, P_6\} \cup \{P_2, P_3, P_5\}$

［段階 3］

Visit＝E∪D∪P となり，すべてのノードを探索したので終了する．

（具体例終わり）

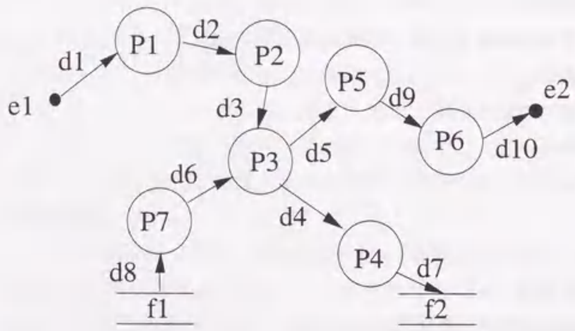


図 4−5 データフロー図のプロセス種別の判定

4.4.2 データフロー図からのモジュール構造作成アルゴリズム

［記法］

（1）モジュールの呼び出し関係

モジュール M からモジュール N が呼び出されるとき，M →N で表す．

（2）入出力データフローの個数を表す関数

プロセス p からプロセス q へのデータフローが存在する場合，通常は，モジュール q がモジュール p を呼ぶ形でモジュール構造図を作成する．すなわち，q が p の親モジュールとなる．ところが，入力プロセス p から複数のプロセス q_1，q_2 へのデータフローが存在する場合，同じように q_1 と q_2 が p を呼ぶ形でモジュール構造図を作成すると，モジュール p の親

モジュールが複数となるので，木構造ではなくなるという問題がある．同様に，p が出力プロセスの場合にも，p へのデータフローが複数あると同じ問題が生じる．このため，以下の関数を用いて，各プロセスの入力および出力データフローの個数を調べておく必要がある．

InDeg(p)=|{q| q⇒p }|

OutDeg(p)=|{q| p⇒q }|

ここで，集合 x の要素数を|x|で表す．

（3）主制御モジュール

すべてのモジュールの根となるモジュールを主制御モジュールという．入力プロセス p から複数のプロセス q_1，q_2 へのデータフローが存在する場合，主制御モジュール Main を導入して，Main がモジュール p，q_1，q_2 を呼び出すことにより，モジュール構造図を木構造で実現できる．すなわち，p からのデータ d を Main が受け取り，d を入力パラメータとして q_1，q_2 を呼び出す．出力プロセスへのデータフローが複数存在するときも同様にしてこの問題に対処できる．

（4）プロセスのレベル

主制御モジュール Main から直接呼び出されるモジュールに対するプロセスのレベルを 1 とする．上位のモジュールに対するプロセスのレベルが 1 であるとき，下位モジュールに対応するプロセスのレベルは 2 である．すなわち，プロセスのレベルを以下のように定義する．ここで，プロセス p から作成されるモジュールを Module(p)で表す．

level(p)=1 ここで，Main → Module(p)

level(p)=k+1 ここで，Module(q) → Module(p)かつ level(q)=k

（5）データフローとモジュールの呼び出しパラメータ

データフロー図のノード p から q へのデータフローd を，Data（p⇒q）で表す．また，モジュール構造図のモジュール m がモジュール n を呼び出すときの入力パラメータと出力パラメータを，それぞれ，Input(m→n)と Output(m→n)で表す．

データフロー図からのモジュール構造図への変換規則を以下に示す．

［規則 1］$p_1 \Rightarrow p_2$ のとき，p_1 と p_2 が以下の条件 a から条件 e のいずれかを満たす場合，Main → Module(P_1)および Main → Module(P_2)を作成する．また，Input（Main→ Module(p_1)）＝ Output（Main→ Module(p_2)）＝Data（ $p_1 \Rightarrow p_2$）とする．このとき，主制御モジュールの名称をデータフロー図名とする．また，プロセスから作成されるモジュールの名称をプロセス名とする．

（条件 a）p_1 が入力プロセスである．p_2 が変換プロセスもしくは出力プロセスである．

（条件 b）p_1 が出力プロセスである．p_2 が変換プロセスもしくは入力プロセスである．

（条件 c）p_1 が変換プロセスである．

（条件 d）p_1 と p_2 がともに入力プロセスである．Level(p_1)＝ Level(p_2)＝1.

（条件 e）p_1 と p_2 がともに出力プロセスである．Level(p_1)＝ Level(p_2)＝1.

［規則 2］$p_1 \Rightarrow p_2$ のとき，p_1 と p_2 がともに入力プロセスであり，Level(p_1)＞1 かつ Level(p_2)＞1 ならば，Module(p_2) → Module(p_1)を作成する．また，Input（ Module(p_2)→ Module(p_1)）＝Data（ $p_1 \Rightarrow p_2$）とする．このとき，プロセスから作成されるモジュール名をプロセス名とする．

［規則 3］$p_1 \Rightarrow p_2$ のとき，p_1 と p_2 がともに出力プロセスであり，Level(p_1)＞1 かつ Level(p_2)＞1 ならば，Module(p_1) → Module(p_2)を作成する．また，Output(Module(p_1)→ Module(p_2))＝Data($p_1 \Rightarrow p_2$)とする．このとき，プロセスから作成されるモジュール名をプロセス名とする．

［規則 4］外部境界 e とプロセス p について，e⇒p のとき， Module(p) → Module（e）を作成する．このとき，外部境界 e に対応するモジュール Module（e）の名称をデータフロー名 d＝ Data(e⇒p)に基づいて「d を入力する」とする．

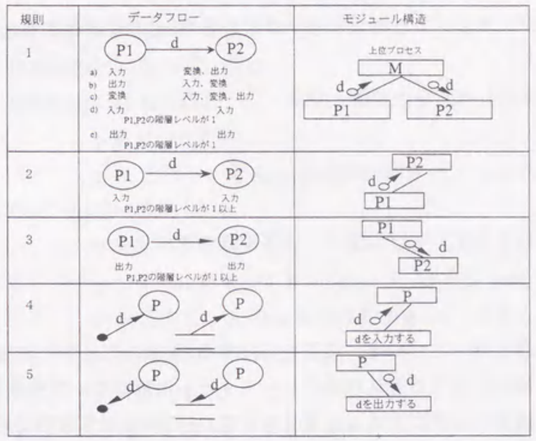
同様に，データストア s とプロセス p について，s⇒p のとき，Module(p) → Module(s)を作成する．このとき，データストア s に対応するモジュール Module(s)の名称をデータフロー名 d＝ Data(s⇒p)に基づいて「d を入力する」とする．

［規則 5］外部境界 e とプロセス p について，p⇒e のとき， Module(p) → Module(e)を作成する．このとき，外部境界 e に対応するモジュール Module(e)の名称をデータフロー名 d＝ Data(p⇒e)に基づいて「d を出力する」とする．

同様に，データストア s とプロセス p について，p⇒s のとき，Module(p) → Module(s)を作成する．このとき，データストア s に対応するモジュール Module(s)の名称をデータフロー名 d＝ Data(p⇒s)に基づいて「d を出力する」とする．

以上の変換規則をまとめて表 4−3 に示す．

表 4−3 データフロー図からモジュール構造図への変換規則



［アルゴリズム 2］ モジュール構造図作成アルゴリズム

［ステップ 1］

外部境界およびデータストアと接続するプロセスを Next の要素とする.

［ステップ 2］

すべてのプロセスのレベルを未定義とする.

［ステップ 3］

すべてのプロセスをノーマークとする.

［ステップ 4］

Next が空集合（φで表す）でない限り，以下を繰り返す.

（1）Current を Next とする.

（2）Next をφとする.

（3）Current の各要素 p について，以下を実行する.

（3−1）p に隣接するプロセス以外のノード n と p に対して，変換規則を適用する.

（3−2）p に隣接するプロセス q と p に対して，p のプロセス種別に応じて，以下のようにして変換規則を適用する.

a）p が入力プロセスの場合

a1）InDeg(p)＞0 の場合

OutDeg(p)＜1 ならば，誤り.

OutDeg(p)＝1 ならば,

q⇒p となるすべての q がマークされていれば,

p をマークする.

レベル 1 の q が存在すれば，p のレベルを 1 とする.

p と q に対して変換規則を適用する.

p⇒r となる r がマークされていれば,

p と r に対して変換規則を適用する

p⇒r となる r がマークされていなければ,

r を Next の要素とする.

そうでないとき，p を Next の要素とする.

OutDeg(p)＞1 ならば,

q⇒p となるすべての q がマークされていれば,

p をマークする.

p のレベルを 1 とする.

p と q に対して変換規則を適用する.

p⇒r となる r がマークされていれば,

p と r に対して変換規則を適用する

p⇒r となる r がマークされていなければ,

r を Next の要素とする.

そうでないとき，p を Next の要素とする.

a2）InDeg(p)＝0 の場合

OutDeg(p)＜1 ならば，誤り.

OutDeg(p)＝1 ならば,

p をマークする.

p⇒r となる r を Next の要素とする.

OutDeg(p)＞1 ならば,

p をマークする.

p のレベルを 1 とする.

p⇒r となる r を Next の要素とする.

b）p が変換プロセスの場合

p をマークする.

q⇒p または p⇒q となるプロセスに対して,

q がマークされていれば，p と q に対して変換規則を適用する.

q がマークされていなければ，q を Next の要素とする.

c）p が出力プロセスの場合

c1）OutDeg(p)＞0 の場合

InDeg(p)＜1 ならば，誤り.

InDeg(p)＝1 ならば,

p⇒q となるすべての q がマークされていれば,

p をマークする.

レベル 1 の q が存在すれば，p のレベルを 1 とする.

p と q に対して変換規則を適用する.

r⇒p となる r がマークされていれば,

p と r に対して変換規則を適用する

r⇒p となる r がマークされていなければ,

r を Next の要素とする.

そうでないとき，p を Next の要素とする.

InDeg(p)＞1 ならば,

p⇒q となるすべての q がマークされていれば,

p をマークする.

p のレベルを 1 とする.

p と q に対して変換規則を適用する.

r⇒p となる r がマークされていれば,

p と r に対して変換規則を適用する

r⇒p となる r がマークされていなければ，
r を Next の要素とする．
そうでないとき，p を Next の要素とする．

c2）OutDeg(p)＝0 の場合
InDeg(p)＜1 ならば，誤り．
InDeg(p)＝1 ならば，
p をマークする．
r⇒p となる r を Next の要素とする．
InDeg(p)＞1 ならば，
p をマークする．
p のレベルを 1 とする．
r⇒p となる r を Next の要素とする．

(4) Next＝Current ならば
Next を φ とする．
Current の各要素 p について，以下を実行する．
p をマークする．
p のレベルを 1 とする．
q⇒p または，p⇒q となるプロセスについて
q がマークされていれば，p と q に対して変換規則を適用する．
q がマークされていなければ，q を Next の要素とする．
（アルゴリズム終わり）

［具体例］
以下のようなデータフロー図に基づいて，モジュール構造図を作成する．
外部境界 E ＝ {e_1，e_2，e_3}
プロセス p＝ {P_1，P_2，P_3，P_4，P_5，P_6，P_7}
データフロー関係
R ＝ {$e_1⇒P_1$，$P_1⇒P_2$，$P_2⇒P_3$，$P_3⇒P_4$，$P_4⇒P_5$，
$e_2⇒P_6$，$P_6⇒P_7$，$P_7⇒P_4$，$P_5⇒e_3$}
また，各プロセスの種別は，以下のようであるとする．
入力プロセス P_1，P_2，P_6，P_7
変換プロセス P_3，P_4
出力プロセス P_5
このデータフロー図を図 4−6 に示す．作成されるモジュール構造図を図 4−7 に示す．
［段階 0］
すべてのプロセスをノーマークとする．

［段階 1］
Next＝ {P_1，P_6，P_5}
Current＝Next
P_1，P_6，P_5 をマークする．
P_1 について，InDeg(P_1)＝0，OutDeg(P_1)＝1
P_1 をマークする．
P_2 を Next に追加する．
規則 4 を e_1 と P_1 に適用する．
P_6 について， InDeg(P_6)＝0，OutDeg(P_6)＝1
P_6 をマークする．
P_7 を Next に追加する．
規則 4 を e_2 と P_6 に適用する．
P_5 について，OutDeg(P_5)＝0，InDeg(P_5)＝1
P_5 をマークする．
P_4 を Next に追加する．
規則 5 を e_3 と P_5 に適用する．
［段階 2］
Next＝ {P_2，P_7，P_4}
Current＝Next
Next＝ φ
P_2 について，InDeg(P_2)＝1，OutDeg(P_2)＝1
$P_1⇒P_2$ となる P_1 がマークされているから，P_2 をマークする．
規則 2 を P_1 と P_2 に適用する．
$P_2⇒P_3$ となる P_3 がマークされていないから，P_3 を Next に追加する．
P_7 について，InDeg(P_7)＝1，OutDeg(P_7)＝1
P_7 をマークする．
$P_6⇒P_7$ となる P_7 がマークされているから，
規則 2 を P_6 と P_7 に適用する．
$P_7⇒P_4$ となる P_4 がマークされていないから，P_4 を Next に追加する．
P_4 について，InDeg(P_4)＝1，OutDeg(P_4)＝1
$P_4⇒P_5$ となる P_5 がマークされているから，P_4 をマークする．
P_4 のレベルを 1 とする．
規則 3 を P_4 と P_5 に適用する．
$P_7⇒P_4$ となる P_7 がマークされているから，
規則 1 c を P_4，P_7 に適用する．
［段階 3］

Next＝ $\{P_3, P_4\}$

Current＝Next

Next＝ϕ

P_3について，InDeg(P_3)＝1，OutDeg(P_3)＝1

　P_3をマークする．

　$P_2 \Rightarrow P_3$となるP_2がマークされているから，

　規則1aをP_2とP_3に適用する．

　$P_3 \Rightarrow P_4$となるP_4がマークされているから，

　規則1cをP_3とP_4に適用する．

P_4については，すべてのデータフローについて規則を適用したので，何もしない．

［段階4］Next＝ϕなので，終了．

（具体例終わり）

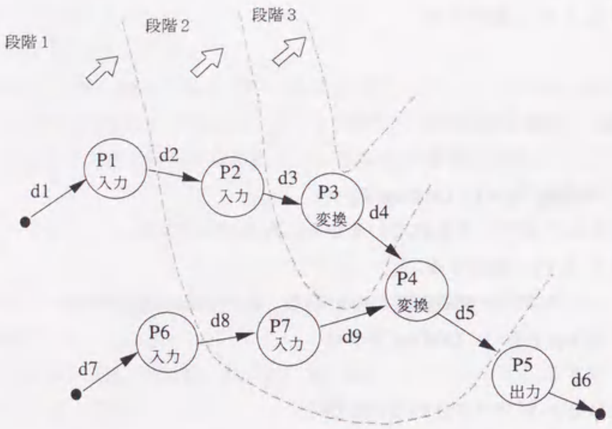


図4−6　データフロー図

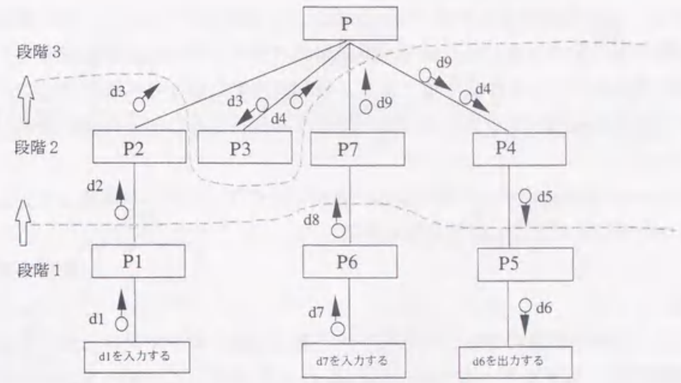


図4−7　データフロー図から作成されたモジュール構造図

4.5　モジュール構造図に基づくシナリオフロー図の作成法

以下では，構造化手法に基づくシナリオフロー図の作成手順を示し，構造化手法に基づいて3層クライアントサーバシステムを設計するための一つの指針を示す．なお，以下では，データフロー図およびモジュール構造図の作成手順の詳細については省略している．

［設計手順］構造化手法に基づくシナリオフロー図の設計

［ステップ1］データフロー図を作成する

構造化分析手法に従ってデータフロー図を作成する．また，データフロー図の作成では，第2章で述べた入出力データの依存関係に基づく手法を利用できる．

［ステップ2］データフロー図に基づいてモジュール構造図を作成する

構造化設計手法に基づいて，データフロー図からモジュール構造図を作成する．また，データフロー図からモジュール構造図への変換では，第3章で述べた手法を利用できる．

［ステップ3］モジュール構造図に基づいてシナリオフロー図の詳細を作成する

(1) モジュール構造図の各モジュールについて，モジュールが所属する層種別を判定する．ここで，モジュールの層種別は，表示層，機能層，データ層の3種類である．

モジュールが役割担当者に関する入出力データだけをパラメータとする場合，このモジュールの層種別は表示層である．モジュールが外部システムまたはデータストアに関する入

出力データだけをパラメータとする場合，このモジュールの層種別はデータ層である．モジュールが役割担当者，外部システム，データストアなど複数の入出力データを扱う場合，このモジュールの層種別は機能層である．

(2) モジュールの層種別にしたがって，各モジュールをシナリオフロー図の各層処理ブロックに配置する．このとき，モジュール構造図のパラメータの受渡しに従って，シナリオフロー図のデータフローを作成する．また，役割担当者に対する外部境界についての入出力データフローに基づいてシナリオフロー図のユーザインタフェース部の要求と応答を作成する．

(3) モジュール構造図のデータ層モジュールがアクセスするデータ実体とモジュールとのデータフローをシナリオフロー図に追加する．

（設計手順終わり）

［具体例］

以下では，図書館業務について構造化手法に基づいたシナリオフロー図の作成手順を示す． 図書館業務に関するデータフロー図を図 4−8 に示す．このデータフロー図に基づいて作成されたモジュール構造図を図 4−9 に示す．さらに，データフロー図とモジュール構造図から作成されたシナリオフロー図を図 4−10 に示す．

（具体例終わり）

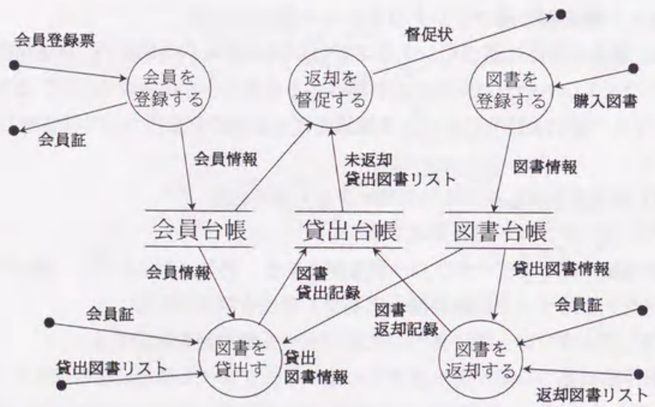


図 4−8　図書館業務のデータフロー図

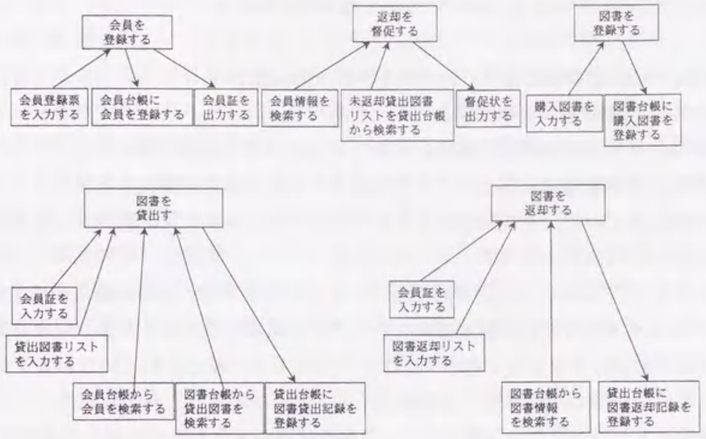


図 4−9　データフロー図から作成されたモジュール構造図

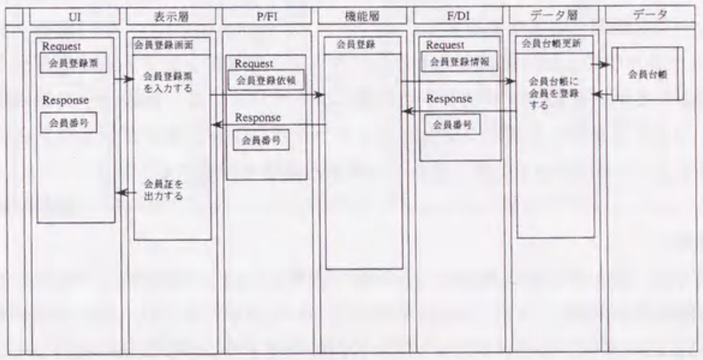


図 4−10　会員登録についてのシナリオフロー図

4.6 オブジェクトモデルからのシナリオフロー図の作成法

以下では，POOM 手法に基づくシナリオフロー図の作成手順を示し，POOM 手法に基づいて 3 層クライアントサーバシステムを設計するための一つの指針を示す．以下ではオブジェクトモデル図の作成手順の詳細については省略している．

［設計手順］POOM 手法に基づくシナリオフロー図の設計
［ステップ1］役割関係図を作成する
POOM手法に従って役割関係に着目したオブジェクトモデル図を作成する．
［ステップ2］役割関係図に基づいてオブジェクトモデル図を詳細化する
POOM手法に基づいて，役割関係図からオブジェクトのパターンに基づいてオブジェクトモデル図を作成する．
［ステップ3］オブジェクトモデル図に基づいてシナリオフロー図の詳細を作成する
(1) オブジェクトモデル図の機能層オブジェクトであるトランザクション型オブジェクトに着目する．トランザクション型オブジェクトのうちで一般化されていないオブジェクトがシナリオフロー図の作成単位である機能イベントに対応すると考えられる．したがってまず具体的なトランザクション型オブジェクトをオブジェクトモデル図から抽出する
(2) 抽出された各トランザクション型オブジェクトごとにシナリオフロー図を作成し，機能層処理として配置する．
(3) シナリオフロー図に配置したトランザクション型オブジェクトとオブジェクトモデル図で関連する表示層オブジェクトを抽出し，シナリオフロー図の表示層処理として配置する．このとき表示層オブジェクトとトランザクション型オブジェクトとの関係からP-Fインタフェースの要求と応答を作成する．
(4) シナリオフロー図に配置したトランザクション型オブジェクトとオブジェクトモデル図で関連するデータ層オブジェクトを抽出し，シナリオフロー図のデータ層処理として配置する．このときデータ層オブジェクトとトランザクション型オブジェクトとの関係からシナリオフロー図のF-Dインタフェースの要求と応答を作成する．

（設計手順終わり）

［具体例］

以下では，第 3 章 4 節の POOM 分析手順で説明した図 3－13 のオブジェクトモデル図における販売係の業務について POOM 手法に基づいたシナリオフロー図の作成手順を示す．図 3－13 のパターン化オブジェクトモデル図の販売トランザクションに対してシナリオフロー図を作成する．以下では設計手順のステップ 3 (1) までは図 3－13 で完了しているものとして，ステップ 3 (2) 以降について説明する．

まず図 3－13 からトランザクション型オブジェクトである「販売」オブジェクトを抽出し，シナリオフロー図の機能層処理「販売処理」を作成する．次に販売オブジェクトと関係する表示層オブジェクトとして「販売端末」オブジェクトを選択する．この「販売端末」

オブジェクトに基づいてシナリオフロー図の表示層処理「販売端末処理」を作成する．「販売端末処理」のユーザが「販売係」となる．データ層処理を作成するために，「販売」オブジェクトと関係するデータ層オブジェクトを抽出する．図 3－13 では，「販売伝票」オブジェクトが「販売」オブジェクトと関係するデータ層オブジェクトであるから，「販売伝票」をデータ層に配置する．このとき販売トランザクションの処理種別を検討すると「販売伝票」を登録するための更新処理であることが分かるから，データ層処理として「販売伝票登録処理」を作成する．また，このとき「販売伝票」と関係するデータ層オブジェクトとして「商品情報」と「顧客情報」があるので，「販売伝票登録処理」の延長でこれらのデータについても更新する必要があるかどうかを検討する．たとえば，販売伝票を登録する前に予め登録された顧客であるかどうかを審査する必要がある場合にはシナリオフロー図のデータ層に「顧客情報」も配置し，データ層処理として「顧客情報照会処理」を作成することになる．このようにして作成されたシナリオフロー図を図 4－11 に示す．また図 4－11 では必要に応じて発生する可能性のある例外情報を層間インタフェースに追加している．

（具体例終わり）

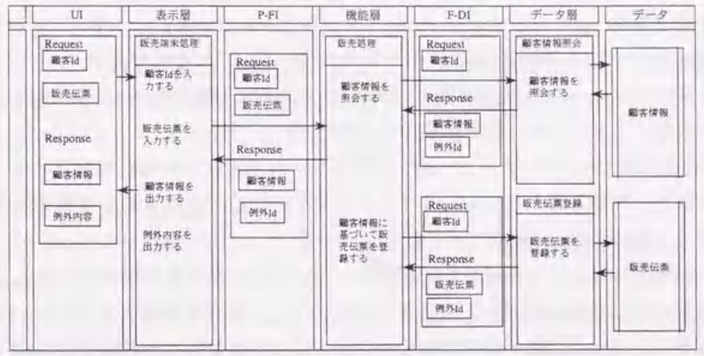


図 4－11　販売トランザクションに対するシナリオフロー図

4.7 まとめ

4.2 では，Bichler らが提案した Structured Hypermedia Development Technique(SHDT)手法における WWW ページの記述モデルに基づいてページフロー図を提案した．ページフロー図は 3 層アーキテクチャに従った WWW 情報システムのユーザインタフェースを記述するための図式である．SHDT を拡張したのは次の 2 点である．すなわち，シナリオフロー図の設計で必要となる機能イベントを明示するためにページ間遷移ラベルを追加したこととアプリケーションの結果によって動的に発生する条件分岐を表すページ間遷移を追加したことである．

4.3 ではシナリオフロー図を提案し 3 層アーキテクチャを構成する表示層，機能層，データ層間のインタフェースの設計法を明らかにした．この設計法では，シナリオフロー図と呼ぶ 3 層インタフェースの設計図式を考案し，各層モジュールの独立性の概念に基づいて，3 層クライアントサーバシステムにおける層間インタフェースの設計条件を明らかにした．さらに実際のインターネット系設備オペレーションシステムへの適用結果に基づき開発工数を約 35%削減できることを明らかにした．

4.4 では，モジュール構造図の生成方式を提案した．本方式で作成されたモジュール構造図の評価結果をまとめると以下のようになる．

(1) データフロー図に基づいて，詳細設計に必要なモジュール構造図の主要部分（約 70%）を自動生成できる．

(2) 複雑なデータフロー関係を持つデータフロー図に対しても適用できる．

(3) データやモジュール名などの設計情報をデータフロー図からモジュール構造図に誤りなく引き継ぐことができ，設計情報の再利用性を向上できる．

(4) 詳細設計段階では，データフロー図に現れていない初期化や例外処理などのモジュールの追加判断と，外部入出力モジュールやファイル入出力モジュールの必要性の判断を行うだけでよく，設計作業を効率化できる．

今後の課題としては，詳細設計まで考慮したモジュール構造を自動生成することやモジュール構造の評価基準に基づいて生成されたモジュール構造を最適化すること，クライアントサーバ型システムなどの分散型情報システムにおけるモジュール構造を生成することなどがある．

4.5 では，モジュール構造図に基づいて分散型情報システムの 3 層アーキテクチャを設計する手法を述べ，3 層アーキテクチャモデルに適した分散型情報システムを設計するための 3 層クライアントサーバシステム設計法を提案した．これにより，構造化手法に基づいて作成された既存システムから 3 層クライアントサーバシステムへの移行を容易化できると思われる．

4.6 では，オブジェクトモデル図に基づいて分散型情報システムの 3 層アーキテクチャを設計する手法を述べ，3 層アーキテクチャモデルに適した分散型情報システムを設計するための 3 層クライアントサーバシステム設計法を提案した．

第 5 章 WWW－データベース連携技術

5.1 まえがき

WWW が登場したことにより，WWW ブラウザをクライアントとするような新しい形態のデータベースアプリケーションが盛んに構築されるようになってきた．WWW とデータベース連携技術は，インターネット上の WWW 利用環境だけではなく，企業内の WWW を中心とするイントラネット利用環境でも重要な構成要素として着実に成長している．

データベースと WWW の連携方式では WWW サーバが規定する CGI（Common Gateway Interface）により，C 言語や UNIX の shell 等でアプリケーションを開発する必要がある．しかし，CGI によるゲートウェイアプリケーションを用いて WWW とデータベースを連携する場合，以下に示すような問題がある[1][2]．

(1) CGI によるプロセス起動が必要なこと，ならびに起動されるプロセスごとにデータベースのオープン／クローズが必要になるため，性能が低下する．

(2) WWW のプロトコル(HTTP)ではページ毎に通信の接続／切断が行われるため，複数ページに跨ってデータベースのトランザクション制御を行うことができない．

(3) C 言語や shell スクリプトを用いて業務処理ごとにデータベースと情報ページ間の連携処理の作成が必要であり開発効率が低下する．

本論文では，これらの問題を解決するために，HTML を拡張して SQL や簡易言語で記述されたプログラムを起動できる WebBASE スクリプト言語機能とセッション処理機能を持つ WebBASE を開発した[3][4][5][6][7][8][9][10][11]．本論文では，WebBASE ならびにその適用評価結果について述べる．

5.2 WebBASE のシステム構成

WebBASE のシステム構成を図 5－1 に示す．WebBASE は，一般の CERN や NCSA，Netscape Communication Server 等と同等の WWW サーバ機能を持ち，WWW ブラウザと HTTP で通信する．VGUIDE（Visual and General User Interface for Database Environments）と DBMS を搭載した DB サーバは，別マシンに配置することもでき，ネットワークからの侵入や処理の分散化にも対応できる．VGUIDE は NTT 研究所が開発したクライアントサーバ型のリレーショナルデータベースアプリケーション開発用のミドルウェアである[12][13][14]．WWW ブラウザと WWW サーバ間は，WebBASE アプリケーションプロセスの起動と通常の HTTP 通信処理を行う．WebBASE と VGUIDE の間では，VGUIDE プロトコルによる SQL 文の実行依頼／結果の取得，リモートプロシジャコール（RPC）／その結果の取得を行う．VGUIDE はクライアント／サーバ型の情報システムのためのミドルウェアで，TP モニタ機能により，高トラフィックの大規模システムにも適用できること，多様な DBMS およびベンダマシン環境に対応しており，アプリケーションの移植性が高いことなどの特徴を持つ．WebBASE では，VGUIDE を用いているので，DBMS に依存しないオープンな環境で WWW データベース連

携アプリケーションを開発できるだけでなく，TP モニタ機能により高速なデータベースアクセスを実現できる．

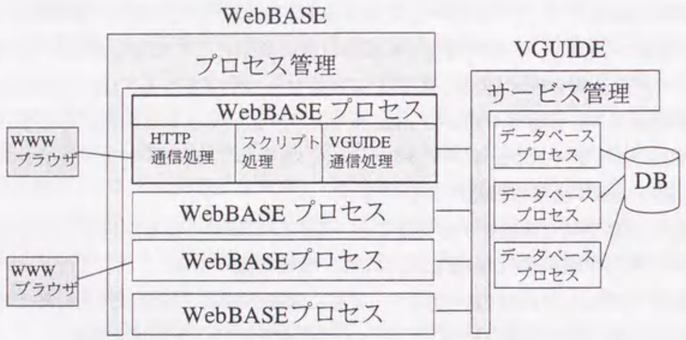


図 5－1　WebBASE のシステム構成

5.3　WebBASE の処理方式

WebBASE の基本的な処理方式を図 5－2 に従って説明する．まず，サービス開始前に，DB サーバ上で VGUIDE プロセスの常駐を開始し，データベースをオープンしておく．

(1) WWW ブラウザから，WebBASE アプリケーションプロセスの起動ならびに，WebBASE スクリプトファイル名と必要なパラメータを WebBASE に要求する．

(2) 起動された WebBASE アプリケーションプロセスは一意な識別子を生成する．これ以降，WWW ブラウザからの要求は，ポート番号と一意データを組み合わせた識別子に従って，サーバ上に常駐している WebBASE アプリケーションプロセスに通知される．

(3) 指定された WebBASE スクリプトファイルを読み込み，WWW ブラウザから渡されたパラメータを変数に代入する．

(4) スクリプトファイルの指定に従い，DB サーバ(VGUIDE)への接続と，データベースの検索を依頼する．

(5) データベースから検索結果として取得したデータを変数に代入する．ここで，検索結果はデータベースのカラム順に，スクリプトで指定されている変数 @1, @2･･･に格納される．また，WebBASE アプリケーションプロセスが常駐化されている間，変数の内容や DB サーバの排他状態が保持される．

(6) HTML ページを動的に生成して，識別子と共に HTTP に従って WWW ブラウザに送信する．

最後に，WebBASE スクリプトファイル中で EXITSESSION コマンドが指定された場合，セッションが終了し，WebBASE プロセスも終了する．

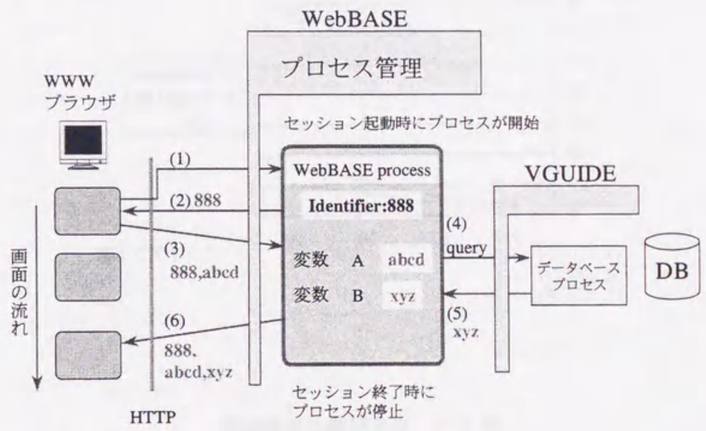


図 5－2　WebBASE の基本的な処理方式

以下では WebBASE の応答性能を評価するため，CGI，WebServer を用いて同じデータベースを検索し，応答速度を比較した結果を明らかにする．この比較では，同時走行クライアント数の増加に対する，応答速度の変化と CPU 使用率を測定した．

（1）実験条件

ツアーデータベースに対して，指定された予算の範囲で参加できるツアーを検索し，ツアーテーブルの内容を提示する．検索対象データベースはツアーテーブルと価格テーブルからなる．ツアーテーブルの各レコードは 7 カラムからなり 92 バイトである．価格テーブルの各レコードは 5 カラムからなり 44 バイトである．データベース中のレコード数は 30 万件である．3 台の SS20 を，クライアントマシン，WWW サーバマシン，データベースマシンとして用いた．これらの SS20 は，CPU 速度が 150MHz，メモリが 128Mbit，OS は Solaris2.5.1 である．

（2）比較結果

応答時間を比較した結果を図 5−5 に示す．この図から，WebServer 方式に較べて WebBASE では，平均して約 41%少ないことが分かる．また，CPU 使用率を比較した結果を図 5−6 に示す．この図から， WebServer 方式に較べて WebBASE では，平均して約 43%だけ CPU 使用率が少ないことが分かる．

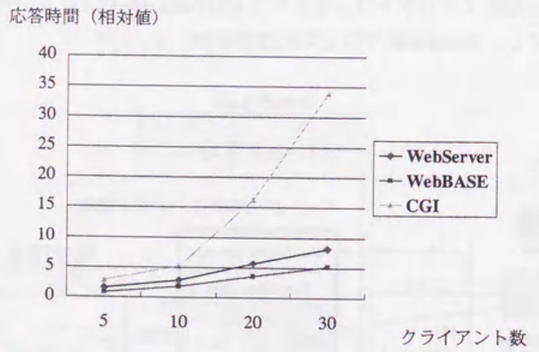


図 5−3　応答時間の比較結果

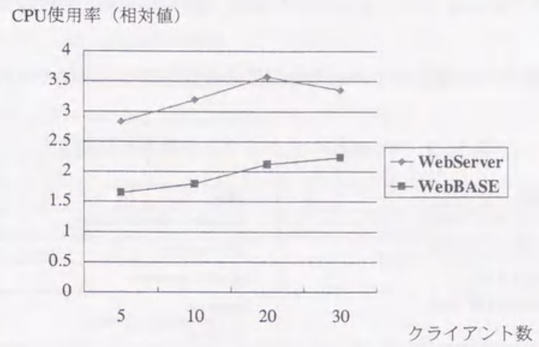


図 5−4　CPU 使用率の比較結果

5.4 WebBASE スクリプト

WebBASE スクリプトの構成要素には，(a)画面表示のための HTML 文 (b)データベースアクセス用の SQL 文 (c)VisualBasic の処理構文がある．データベースアクセス用の WebBASE スクリプトの基本的な構文には，表 5−1 に示した 12 種類がある．ここで，WebBASE スクリプトでは，通常の HTML 文(a)と区別するために，拡張部分(b)(c)の行頭にドット文字（'.'）を付けている．

図 5−5 に，指定した金額よりも多い給与の社員を検索するような WebBASE スクリプトの例を示す．

表 5−1　WebBASE スクリプトの基本的な構文

	機能	構文
1	データベースに接続	.DBConnect <database name>
2	データベースを切断	.DBDisconnect
3	SQL 文を実行	.Sql <SQLstatements>
4	検索結果を値で取得	.LoopFetch (*) <a sequence of arguments> .End Fetch
5	検索結果を文字列で取得	. <variable name> = DBgetcol(<column number>)
6	ファイルに検索結果を出力	.DBWriteFile <file name>
7	リモートプロシージャコール	.DBCpexec <CP name > [, <argument list>]
8	RPCの結果を取得	.Dbdownload
9	条件分岐文	.If <condition> Then <process block> .End If
10	代入文	.Let <variable name> = <value>
11	外部コマンドを起動	` <command name> `
12	セションを終了	.ExitSession

(*) < a sequence of arguments > の内容は以下のようである。
<display name>: @<column number> [, <display name>: @< column number>]
ここで、display name は情報ページのフィールド名である。検索結果はカラム番号（ column number ）に対応するフィールドの値として表示される。


```
<HEAD><TITLE>Employee database </TITLE></HEAD>
<BODY><H1>Query result of employee </H>
<HR>
Person whose salary is greater than @SAL
.DBConnect  "JINJIDB"
.Sql "SELECT ENO,ENAME,JOB,SAL,PHOTO FROM EMP" & DBCndNum("", "SAL" , ">", sal)
.Loopfetch
     Photograph of face :<img src="/kao/@5.jpg"><p>
     Employee number: @1<p>
     Name: @2<p>
     Role: @3<p>
     Salary: @4 ><p>
 <HR>
 .End Fetch
 .DBDisconnect
 .End
```

図 5−5　WebBASE スクリプトの例

WebBASE では，他言語で作成されたライブラリ関数を予め WebBASE アプリケーションプロセスにリンクしておき，動的に呼び出す DLL(Dynamic Link Library) 機能を提供している．この DLL 機能により，WebBASE スクリプトから CORBA オブジェクトのサービスを利用できる．WebBASE と CORBA を次のようにして連携できる．まず，IDL で定義された CORBA サービスに対して，WebBASE スクリプトで利用するための，DLL ライブラリを作成する．DLL ライブラリでは，CORBA オブジェクトの操作ごとに，これらの操作をラッピングするための DLL 関数を作成する．WebBASE スクリプトでは，利用するラッパ関数をスクリプトで宣言しておくことにより，WebBASEがアプリケーションプロセスの実行時に，DLL ライブラリでラッピングされたオブジェクト操作により CORBA オブジェクトのサービスを利用することができる．

たとえば，次のような IDL で定義された操作があるとする．

```
interface Game {
                long getTotal(in long player);
               }
```

このとき，getTotalという名前のラッピング関数をlibgameというDLL関数ライブラリに用意しておく．次に，このDLLを利用することをWebBASEスクリプトの先頭で宣言する．

```
.Declare Function getTotal Lib "libgreed"(ByVal p as long) as long
```

WebBASE スクリプトの実行部分では，以下のようにすれば，CORBA を用いて変数 s0 に値を設定することができる．

. s0 = getTotal (0)

以上述べたように，WebBASE では，DLL ライブラリを仲介させることにより CORBA オブジェクトのサービスを簡単に利用することができる．

WebBASE スクリプトの記述性を評価するため，CGI，WebServer を用いて同じ成績管理データベースを検索する問題に対して，記述量を比較した結果を以下に示す．

CGI 方式と WebBASE スクリプト，WebSerber 方式により記述した結果を図 5−6 に示す．この図から，WebBASE スクリプトでは，記述量が CGI 方式の約 30.6%になることが判明した．この理由は，WebBASE スクリプトでは，SQL 文を約 9.7%，機能記述を約 54.8%，HTML 記述を約 18%削減できているからである．また，ライブラリ方式の市販製品と比較すると，記述量を約 39%削減できたことがわかる．

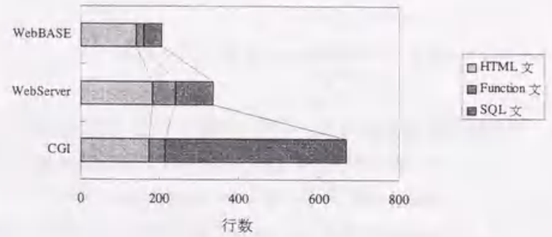


図 5−6　記述量の比較実験結果

5.5　WebBASE を用いたシステム構築事例

WebBASE が最初に適用されたシステムは，筑波にある金属材料技術研究所における物質スペクトル検索システム[15]である．このシステムは，1995 年 9 月からインターネット上で情報発信を始めた．物質スペクトル検索システムでは，さまざまな環境条件の下で測定された金属スペクトルデータを，データベースに格納しておき，インターネット上で検索しグラフ表示できる．金属スペクトルデータは，各金属物質のスペクトル波形を示す数値情報と測定条件を示す文字情報の組合せでデータベースに蓄積されている．しかし，従来の静的な HTML では，金属スペクトルデータごとに HTML ファイルを予め作成しておく必要がありスペクトルデータの追加や管理が困難でインターネット上での情報発信の大きな障害だった．このため，物質スペクトル検索システムでは，WebBASE を用いることにより，データベースに格納された数値情報から動的にグラフを作成し WWW ブラウザから容易に閲覧できるようにして，この問題を解決した．その後，次に示すような数多くのシステムがこれまでに開発されている．

（1）SGML データベースとの統合事例

SGML データベースを WWW ブラウザから検索できるようにした事例としては，新聞記事データベース[16]やソフトウェア情報管理システム[17]などがある．電子出版分野や CALS(Commerce At Light Speed) などでは，SGML を用いて文書情報や設計情報などを統一し，情報交換や情報共有の効率化が進められている．

新聞記事データベースシステムは，NTT が実施したマルチメディア共同実験の一貫として実現したシステムである．新聞社から毎日提供される SGML(Standard Generalized Markup Language)化された記事情報をデータベース内に蓄積しておき，マルチメディア共同実験網で検索利用できる．このシステムでは，SGML データベースの記事本文情報から記事のタイトルや作成日付，キーワードなどを抽出してリレーショナルデータベース上で記事管理情報として格納する．WWW ブラウザによるマルチメディア共同実験網からの新聞記事情報の検索では，WebBASE を用いてリレーショナルデータベースの記事管理情報を検索し，記事本文情報や写真などの情報を参照する．

新聞記事データベースでは，既存の SGML 情報から記事の管理情報を抽出してリレーショナルデータベースで管理する方式を用いた．これに対して，ソフトウェア情報管理システムでは，文書管理情報と文書内容を，それぞれ，リレーショナルデータベースと SGML データベースで予め区別して格納しておき，文書管理情報の検索はリレーショナルデータベースで行い，文書内容の検索は SGML データベースで行う方式を用いている．WebBASE スクリプトでは，検索対象ごとに利用するデータベースを切り替えるように指定するだけでよい．このシステムでは，共有情報の形式を意識することなく情報を登録したり検索，ダウンロードができる．

新聞記事データベースシステムやソフトウェア情報管理システムなどから分かるように，WebBASE を用いることにより，SGML データベースとリレーショナルデータベースを

WWW ブラウザからシームレスにアクセスするシステムを柔軟に構築できる．

（2）情報案内型の事例

情報案内型システムの例としては，インターネット上で新着ページ情報を検索できるＮＴＴDIRECTORY[18]や，フリーダイアル番号を業務種別分類やキーワードにより検索できるフリーダイアル検索システムがある[19]．

NTT DIRECTORY は，日本のホームページ約 20 万ページを管理する国内でも最大級のサーチエンジンである．NTT DIRECTORY の検索方法には，キーワード検索とジャンル別検索の 2 つがある．WebBASE はジャンル検索とホームページ情報の管理に適用されている．これに対して，キーワード検索には InfoBee 検索エンジンが適用されている[20]． NTT DIRECTORY の場合，定常的に毎日約 200 件のホームページ情報掲載の要求がある．このようなホームページ情報掲載作業を効率化するため，WebBASE を用いてディレクトリ情報をデータベース化するとともに，データベースから動的にホームページ情報を検索して WWW ブラウザに結果を提示する方式でジャンル検索を実現した．NTT DIRECTORY のジャンル検索では，ジャンル階層を柔軟に再構成できるようにするため，ジャンルとホームページを互いに独立するテーブルとしてリレーショナルデータベースで管理している．ジャンル検索の処理では，まず，指定されたジャンルに所属するホームページの識別子のリストを求め，次いで各識別子に対するホームページの情報を検索する． NTT DIRECTORY では，1 日 100 万ヒット以上のアクセスがあるため，アクセス負荷を軽減策として，検索結果をファイル化しておき，同一の検索要求に対してはデータベースを検索せずに，このファイルを用いて応答する．

フリーダイアル検索システムでは，NTT が提供するフリーダイアルの電話番号を格納した既存のデータベースから情報の一部をダウンロードして，ジャンル検索用のインデックスを付与した上でフリーダイアル検索用のデータベースとして格納している．この場合，プライバシー問題の発生を避けるため，検索用データベースに格納される電話番号はすべて顧客の許可を得るようにしている．また，フリーダイアルでは，電話料金が着信払いであることから，発進地域の限定サービスがある．このため，フリーダイアル検索システムでも局番指定やメニュ選択を用いた発進地域による絞込検索機能を提供する必要がある．検索用データベースに発進地域の完全な情報を格納してあるので， WebBASE スクリプトだけを追加変更するだけで，WWW ブラウザでの多様なユーザインタフェースを容易に実現できる．

（3）情報共有型の事例

情報共有型システムの例としては，経営管理システム[21]や技術資料データベース[22]等がある．情報共有型のシステムは，インターネット上で一般利用者が使うのではなく，企業内で利用するイントラネット型の情報システムである．

経営管理システムでは，既存のメインフレーム上に構築されたレガシーシステムのデータから必要な情報をそのまま利用してリレーショナルデータベースに定期的に格納している．また，業務アプリケーションを WebBASE により開発した．ユーザインタフェースは，WWW ブラウザと Excel を利用した．したがって，業務アプリケーションの追加や変更はレガシーシステムに手を加えることなく，WebBASE のスクリプトだけで対処でき，情報システムの保守作業を効率化できる．また，これまでは専用のクライアントマシンや専用ソフトがないとアクセスできなかった社内システムの情報に対して， WebBASE を用いた経営管理システムでは，WWW ブラウザや Excel という一般的な市販ソフトさえあれば，自由に参照できるようになった．

技術資料を社内で共有する場合，これまでは冊子の形で各事業所ごとに配布されており，必要なときに担当者がすぐに利用できないという問題があった．技術情報データベースでは，WWW ブラウザ上から技術情報を検索して提示することにより，この問題を解決している．技術情報の原本はパソコン上のデータベース管理ソフトで管理する．この理由は，技術情報の提供元との情報交換をスムーズにするためである．パソコン上の技術情報をワークステーションに転送して WWW 検索用のデータベースとして格納している．WebBASE を利用することにより，組織名，適用分野，技術分野，キーワード等の多様な検索を実現している．また，組織変更時にも HTML を変更するのではなく，データベースの内容だけを変更すればよいので維持管理が容易である．

（4）インターネットコマース型の事例

WebBASE を用いたインターネットコマースの例としては，インターネット上で実現されたホテル予約システム[23]，ISDN サービス申込み受付システム[24]，ゴルフ場予約システム[25]などがある．

ホテル予約システム「ロテル」は，パソコン通信会員がリアルタイムでホテル予約できる日本ではじめてのサービスで，1994 年にサービスを開始していた．その後，日本でのインターネットの急進に伴い，WebBASE を採用してインターネット上でのホテル予約サービスを開発した．パソコン通信の場合，会話形式で必要な情報を継続的にやりとりできるので，指定された処理に対して適切な情報を提示できる．WWW を用いたインターネット向けの予約サービスでもパソコン通信と同等の会話形式によるサービスを実現するために，ロテルでは，セッションを管理できる機能を持つ WebBASE を採用した．また，WWW サーバへのアクセス時に会員制ネットワーク側では会員情報等を暗号化して送信することにより，インターネットからの要求と会員制ネットワークからの要求とを区別する．この理由は，会員資格の有無を WWW サーバ側で自動的に判別して，予約に対するリスクを軽減するためである．

ISDN サービス申込み受付システムは，INS ネット 64 の仮申し込みを ISDN ホームページ上で受付けている．これは営業時間外でも INS ネット 64 の申し込みを受け付けて欲しいといった要望に答えたサービスの事例である．1996 年 4 月末にサービスを開始している．WWW ブラウザで入力された情報は，必要な情報項目が揃っていることが確認されると，データベースに登録された管理情報に従って，社内のネットワークを用いて担当支店に電

子メールで配送される．また，このシステムでは，WebBASE を載せたサーバマシンと，VGUIDE とデータベースを載せたサーバマシンとを独立させ，中間にファイアウォールを設けることにより，外部からのデータベースへの進入を防いでいる．

5.6 考察

5.6.1 適用可能性

WebBASE は適用事例で示したように，各種のインターネット，イントラネットに適用されており，適用範囲が広いことを実証している．

(1) 表示層

分散アプリケーションは，表示層，機能層，データ層に分けることができる．WebBASE を適用する場合，表示層は WWW ブラウザ，機能層は WebBASE，データ層はデータベース管理システムや文書管理システムなどを利用できる．このため，表示層として要求される機能が WWW ブラウザの機能の範囲に閉じている場合は，WebBASE だけで効率的にアプリケーションを開発できる．しかし，WWW ブラウザの機能を越える高度なユーザインタフェースが要求される場合には，Java などを用いて WWW ブラウザの機能を拡張するか，VisualBASIC などを用いて表示層を開発する必要がある．Java applet を用いて表示層インタフェースを拡充する場合，JDBC を利用する方式が考えられる．この場合，JDBC を用いて直接，DBMS をアクセスする方が WebBASE を仲介するよりも性能が良い可能性がある．しかし，表示層のプログラムが直接，データベースをアクセスするので拡張性の点で問題がある．また，表示層をデータベース層と独立にする場合は，WebBASE を用いてデータベースアクセスを機能層に隠蔽する方が良いと考えられる．

実際の業務アプリケーションの場合には，複雑な帳票形式を持つ入力インタフェースを実現する必要があり，WWW 上で実現しようとすると，ブラウザそのものを開発することになりかねないことがある．このような複雑な表示層インタフェースを持つ業務には，WebBASE を適用するのは避けるべきである．

(2) 再利用性

WebBASE では，HTML を拡張した WebBASE スクリプトによりアプリケーションを開発する．WebBASE スクリプトは，情報ページごとに作成する必要がある．このため，類似する処理がある場合，同じスクリプトファイルをコピーして，パラメータに相当する変数名を変更して再利用する必要があり規模が増加しやすいという問題があった．このため，以下のようなマクロ機能を WebBASE スクリプトに導入した．

.source "<File Name>"

このマクロ機能により，<File Name>で指定されたスクリプトファイルが，引用もとのスクリプトファイル内に動的に展開される．マクロ機能により，類似処理をモジュール化できるようになり開発規模を削減できるだけでなく，WebBASE スクリプトの再利用が容易になった．

(3) データ構造

WebBASE スクリプトでは，データ構造として，配列しか提供していない．構造体を提供していないのは，スクリプト言語を分かり易くするだけでなく，処理系の負担を軽減するためである．

構造体データを扱うアプリケーションを WebBASE で作成する場合，まず，構造体管理用のプログラムを個別に作成しておく．次いで，DLL 機能を用いて WebBASE スクリプトから構造体管理用のプログラムを呼び出すことにより構造体データを操作できる．このようにして，WebBASE ではデータ構造を拡張できる．

(4) ポート番号

WebBASE では，接続要求ごとにポート番号を発行しているため，ファイアウォールでポート番号を制限している利用者との通信ができないという問題があった．このため，WebBASE が発行するポート番号の範囲を予め設定しておき，その範囲内でポート番号を発行できる機能を実現した．

(5) アクセス制御

イントラネットでは，役割に応じてアクセスできる情報を適切に制御する必要がある場合がある．WebBASE では，マクロ機能を用いて，利用者の役割ごとにアクセスできる情報ページを制御できる．まず，役割モデルを作成しておく．利用者からアクセス権が設定された情報ページへのアクセスがあると，情報ページごとに挿入されているアクセス制御マクロでは，利用者の役割が情報ページが許可するアクセス権を持つかどうかを役割モデルを検索して判断する．もし，利用者の役割が情報ページが許可するアクセス権を持つ場合，利用者は情報ページを操作できる．これに対して，利用者の役割が情報ページが許可するアクセス権を持たない場合，利用者に対して情報ページは，提示されない．役割モデルを用いて情報ページへのアクセスを制御できるので組織が変更された場合でも役割モデルのデータを変更するだけで情報ページの内容を修正する必要はない．

5.6.2 適用上の留意事項

(1) アーキテクチャ設計

クライアント／サーバシステムでは，表示層，機能層，データ層からなる 3 層アーキテクチャを用いている．WWW を用いたイントラネットでは，表示層に WWW ブラウザを用いる．クライアント／サーバシステムでは，表示層をクライアント側でユーザインタフェースアプリケーションを開発する必要がある．しかし，入出力データ項目数が多数になる帳票画面の場合，HTML を対象にした WWW ブラウザでは実現できないことが多い．この場合，クライアント側で Java や VisualBASIC などを用いて表示層を実現するアプリケーションを開発する必要がある．これまでの WebBASE の適用事例では，データの入力系には従来の 3 層アーキテクチャを適用し，データの参照系には WWW を用いる統合的なアーキテ

クチャを用いている．

(2) 表示層の仕様確定

イントラネットの場合，HTML により容易に実行可能なユーザインタフェースをプロトタイピングできるので，表示層インタフェースが頻繁に変更され，表示層の仕様が確定し難いという問題がある．表示層のデータ項目が追加された場合，データモデルも変更を受けることがある．このように，データモデルが変更されてしまうと，他の情報ページにテーブル修正の変更が波及する可能性が高い．また，WWW ブラウザだけで実現できる表示層インタフェースには限界があるので，高度な表示層を実現する場合，プラグイン・アプリケーションの開発が必要になる．この問題を解決するには，表示層の仕様を早い段階で確定することが重要である．

(3) プロセス数

WebBASE では，同時に複数のプロセスを走行させることにより性能を向上できる．この場合，同時走行プロセスの個数を決定する必要がある．WebBASE のプロセス数を W，VGUIDE への検索要求のキューの長さを N，VGUIDE のプロセス数を V とする．このとき，WebBASE のプロセス数は W = N+V を満足する必要がある．この理由は，もし W > N+V ならデータベース処理が過負荷になるからである． W < N+V の場合，データベース資源を最大限に有効利用できない．実際には，WebBASE や VGUIDE が走行するハードウェア資源によって最適な値が異なるので，関係式 W = N+V を満足するように， V，W，N の値を調整する必要がある．

5.6.3 信頼性と生産性

(1) 信頼性

WebBASE を適用することにより，アプリケーション開発規模を削減でき，これまでよりもバグ数を削減できるとの評価を得た．また，WWW ブラウザを表示層として利用するので，設計工程での問題発生件数も減少することを確認した．

(2) 生産性

これまでの適用結果から，WebBASE により，データベースを検索し，WWW 上で表示するアプリケーションの製造工数を削減できることを確認した．しかし，仕様変更件数が多い場合，全体の開発工数が増加する可能性がある．これは，イントラネットでは，WWW ブラウザにより表示層のプロトタイピングが容易であるため，簡単にシステムが変更できると思われやすい．ところが，表示層の仕様変更がデータモデルの設計全体に影響を与えると，機能層のスクリプトを大幅に修正する必要があり，全体の開発工数は増加することになる．したがって，表示層の変更がデータモデルに与える影響が局所的なのか大域的なのかを見極めることが重要である．

5.6.4 関連技術

WWW－データベース連携ミドルウェア研究の課題には，クライアント側の課題とサーバ側の課題がある．クライアント側の問題は，柔軟なユーザインタフェースを効率的に構築することである．たとえば，ヘルプ機能により HTML query form を自動生成するアプリケーションジェネレータの研究がある[26]．ただし，この方法では生成できるデータベース操作がアプリケーションジェネレータが提供するヘルプ機能の範囲に限定されるという問題がある．これに対して，サーバ側の研究には，種々の異なるデータベース・サーバを柔軟に接続する研究や性能の高いアプリケーションを効率的に構築する研究がある．異なるデータベース・サーバを連携する研究には，Infomaster[27]，OO-SQL wrapper[28]，DataWeb[29]などがある．Infomaster では，知識ベースを用いて，ODBC や WWW ページなどの差異を吸収している．OO-SQL では，サーバ側に異なるデータベースの違いを吸収するラッパを置き，モバイルクライアント側からは共通インタフェースを用いてこれらのデータベースを利用できる．DataWeb では，マルチメディア情報の階層的なカタログをデータベースで管理しておき，CGI で記述された QueryServer により WWW ブラウザから段階的に検索できる．

WebBASE Script は，HTML と SQL のようなデータベース操作言語の差違を吸収する言語の一つである[30]．WWW－データベース連携アプリケーションの作成では，HTML 文，SQL 文などのデータベース操作の記述，制御構文を記述する必要がある[31]．このようなアプリケーションロジックの記述方式には，テンプレート方式と関数ライブラリ方式がある．以下では，テンプレート方式と関数ライブラリ方式について考察する．

(1) テンプレート方式

HTML 文，SQL 文，制御構文を，一つのスクリプト言語でまとめて記述する方式である．テンプレート方式の製品事例として，Microsoft 社の IDC(Internet Database Connector)[32]，O'Reilly 社の Cold Fusion [33]などがある．テンプレート方式の利点は，一つのスクリプト言語だけで Web データベース連携アプリケーションを作成できるので，従来のプログラミング言語を用いるよりははるかに生産性が高いことである．しかし，テンプレートで利用できる構文や変数に制約がある場合，必ずしも柔軟にアプリケーションロジックを記述できないという問題がある．このような制約の例として，IDC では，一つのテンプレートでは複数の SQL 文を記述できないこと，Cold Fusion では，複数の SQL 文を記述できるが IF ELSE 構文を使えないなどがある．また，IDC の場合，UNIX サーバの上で動作できないので，アプリケーションの可搬性に限界がある．

(2) 関数ライブラリ方式

HTML 文やデータベース操作文を既存のプログラミング言語の関数として用意することにより，従来のプログラミングの延長で Web データベース連携アプリケーションを作成する方式である．関数ライブラリ方式の製品事例として，Oracle 社の WebServer[34]や Netscape Communications 社の LiveWire Professional[35]などがある．WebServer では，Oracle のストアドプロシージャの記述言語である PL/SQL の関数ライブラリとして，HTML のタグを生成す

る HTF 関数と HTML の結果を出力する HTP 関数を提供する．LiveWire Professional では，Java script の外部関数として Open DataBase Connectivity(ODBC) とともに Informix, Sybase, Oracle などのデータベース操作インタフェースを提供する．関数ライブラリ方式の利点は，プログラミング言語をベースにしているので構文上の制約が少なく柔軟性が高いことである．しかし，アプリケーション開発の面では，HTML そのものではなく HTML を生成する関数を学習する必要があること，ベースとするプログラミング言語の学習が必要となることなどから，テンプレート方式に比較して生産性が高いとは言えない．とくに，WebServer の場合，データベース管理システムも Oracle だけに限定される．

5.7 まとめ

本章では，WWW 情報システム開発支援システム WebBASE について述べ，実験評価ならびに適用事例に基づき評価結果を示した．

WebBASE の適用効果をまとめると以下のようになる．

(1) 多様なイントラネットアプリケーション開発に適用し，有効性を確認した．

(2) CGI 方式に比較して約 6 倍の応答性能である．

(3) 開発規模を約 40%削減できる．

今後の課題には，WebBASE を用いた WWW 情報システムのコンポーネント化や，第 4 章で提案した WWW 情報システム設計方法論に基づく WebBASE アプリケーションの自動生成などがある．

第 6 章　適用実験に基づく評価

6.1 まえがき

第 2 章で提案した 3 層 WWW 情報システム開発法を，第 3 章の要求分析技術，第 4 章の設計技術，第 5 章の構築技術によって具体化した．本章では第 3 章以降で提案した各技術の有効性について実証的に評価した結果を明らかにする．

本章の構成は次のとおりである．6.2 では，モジュール構造図からのデータフロー図の復元方式の有効性を人手による復元結果と比較することにより評価する．6.3 では，POOM の有効性を評価するために実施した POOM と OMT との比較実験計画について述べる．まず 6.4 で POOM と OMT に対する個人による要求分析実験の結果を比較し POOM の有効性を明らかにする．次いで 6.5 で POOM と OMT に対する複数人からなるチームによる要求分析実験の結果を比較し POOM の有効性を明らかにする．6.6 では，データフロー図からのモジュール構造生成方式の有効性を人手による設計結果と比較することにより評価する．6.7 では，3 層システム設計法の有効性を実際のシステム開発への適用結果における層間インタフェースの共通化率に基づいて評価する．6.8 では，情報システム開発者へのアンケート調査に基づいて WebBASE による WWW 情報システム開発の有効性を従来型の情報システム開発と比較することにより評価する．

6.2 モジュール構造図に基づくデータフロー図の復元方式の有効性の評価

Stevens らの文献[1]で取り上げられている患者監視システムの例題に，3.3 節で提案したアルゴリズムを適用することにより，モジュール構造図に基づくデータフロー図の復元方式の有効性と限界を評価する[2]．このシステムに対するモジュール構造図を図 6−1 に示す．本アルゴリズムにより復元されたデータフロー図を図 6−2 に示す．

実験では，図 6−1 に示したデータフロー図から，モジュール構造図を人手で作成する課題を被験者に与えた．被験者は，構造化分析設計手法の経験者と未経験者からなる．被験者によるモジュール構造図からのデータフロー図の復元手順は次のようである．

(1) モジュール構造図を理解する．

(2) モジュール構造図に対応するデータフロー図を作成する．

(3) データフロー図の妥当性を検証する

(4) 必要があればデータフロー図を修正する

なお，各被験者は互いに独立してデータフロー図を作成した．

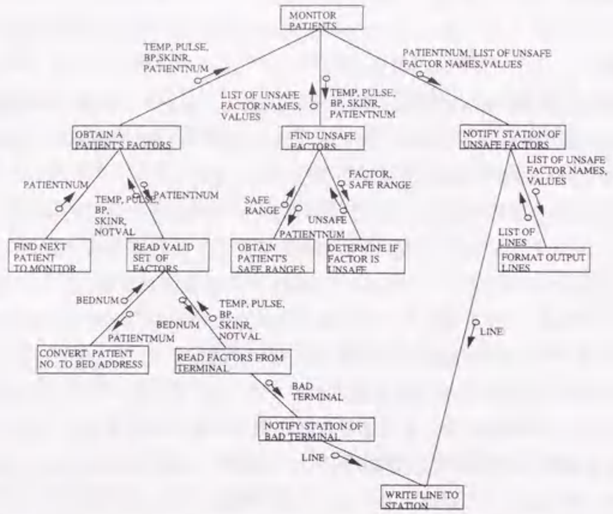


図 6−1　患者監視システムのモジュール構造図

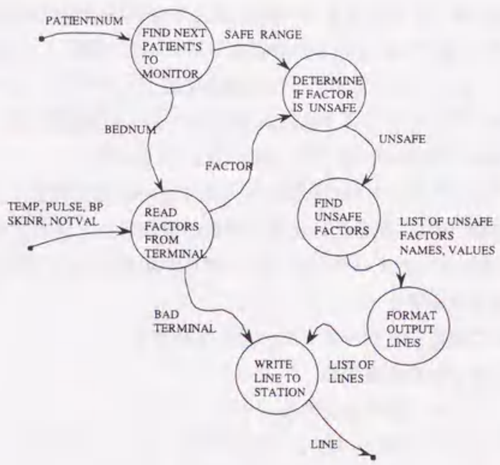


図 6−2　患者監視システムのデータフロー図

図 6−2 に示したデータフロー図を各被験者が作成したデータフロー図と比較した結果を図 6−3 と図 6−4 に示す．図 6−3 では，作成されたデータフローについて比較している．これに対して，図 6−4 では，作成されたプロセスについて比較している．

図 6−3 から，経験者が作成したデータフローについては，提案したアルゴリズムによりすべて自動生成されていることが分かる．未経験者の場合，制御データ，EOF（ファイルの終わりを示す制御コード）などをデータフローとして抽出している．この理由は，要求仕様を記述するデータフロー図と処理手順を記述するフローチャートとを混同しているためである．このような詳細なデータについては経験者および提案したアルゴリズムではデータフローとして抽出していない．また，初心者の場合，誤りを含んだデータフロー図を作成していた．たとえば，入力データフローや出力データフローを持たないプロセスを記述していた．提案したアルゴリズムでは，正しいデータフロー図を作成できるので，このような初歩的な誤りを予防できる．これに対して，提案したアルゴリズムには，以下のような限界もある．すなわち，提案したアルゴリズムでは，LINE や LIST OF LINES などの書式データや，NOTVAL や UNSAFE などの制御用データを抽出していたが，経験者はこれらをデータフローとして抽出していなかった．したがって，本アルゴリズムを適用する場合，これらのデータを手作業で削除する必要がある．

図 6−4 から，提案したアルゴリズムにより自動生成されたプロセスについては，すべて初心者も作成していることが分かる．しかし，初心者は，経験者や提案したアルゴリズムでは作成していない制御プロセスを作成していた．また，提案したアルゴリズムにより生成されたデータフロー図のプロセス数は，モジュール構造図のモジュール数の約 50%であった．これらの事実は，詳細レベルのモジュールに対するプロセスをデータフロー図から除去する上で提案したアルゴリズムが有効であることを示している．これに対して，提案したアルゴリズムには，以下のような限界もある．すなわち，提案したアルゴリズムが生成しているデータの入力制御や書式制御に関するプロセスを経験者は作成していないという問題がある．この原因は，上述したように，データの入力制御や書式制御に関するデータを入出力関係から削除できなかったためである．もし，これらのデータを入出力関係から削除することができれば，提案したアルゴリズムでもこれらのデータに対するプロセスを生成することはない．

モジュール構造図を理解し，データフロー図を作成するための時間については，経験者が平均約 47.5 分，初心者が平均約 52.5 分であった．また，データフロー図を検証し修正するための時間は，経験者が平均約 10.0 分，初心者が平均約 22.5 分であった．

提案したアルゴリズムを用いることにより，経験者および初心者に対して，モジュール構造図の理解とデータフロー図の作成時間を削減できる．結果として，モジュール構造図からのデータフロー図の復元工数を，経験者の場合には約 82.6%，初心者の場合には約 70%削減できる．

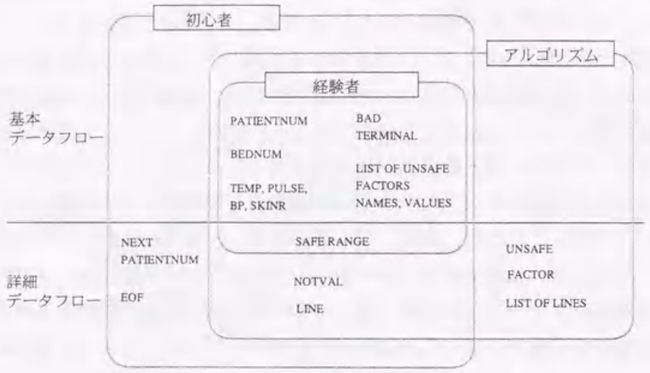


図 6−3　データフローの比較

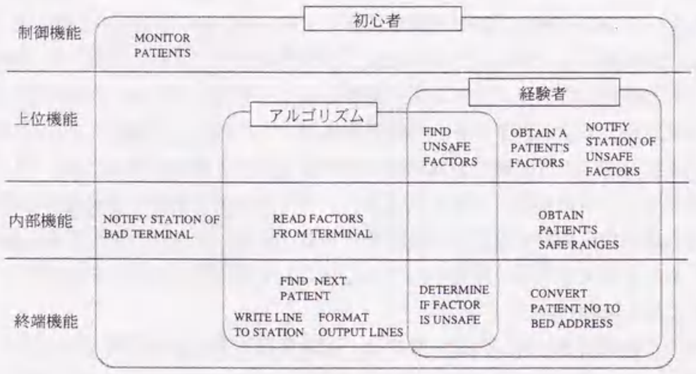


図 6−4　プロセスの比較

6.3　POOM の適用評価実験

これまで情報構造主導型オブジェクト指向分析方法論と利用構造主導型オブジェクト指向分析方法論とを実験的に比較評価する研究はほとんど行われてこなかった．本実験では，情報構造主導型方法論である OMT と，筆者らが提案している役割オブジェクト分析に基づく利用構造主導型方法論 POOM について，両者の有効性を実験的に評価する．ここでは，OMT と POOM による比較実験の目的と概要を述べる．6.4 では，個人分析実験の結果を比較する[3][4][5]．6.5 では，チーム分析実験の結果を比較する[6][7]．

本実験の趣旨は，利用構造主導型が情報構造主導型よりも有効であることを定性的に主張するのではなく，定量的に評価することにある．常識的には，着目点を規定した方法論の方が，着目点を規定しない方法論よりも作業効率が良くなると考えられる．しかし，着目点を規定したことにより解決できる問題の範囲が狭くなったり，解としてのオブジェクトモデルの質が低下する可能性があるだけでなく，効率の改善効果が定量的に分からないという問題がある．このような問題を解明するためには具体的で実証的な評価研究が必要である．

また，オブジェクトモデル作成上の主要な着目点としては，役割，機能，データの 3 つが考えられる．役割に着目するのが利用構造主導型方法論であり，データに着目するのが情報構造主導型方法論である．しかし，機能に着目してオブジェクトモデルを作成するのは，オブジェクト指向にはなじまないと思われる．したがって，本論文では，利用構造主導型の POOM と情報構造主導型の OMT とを比較することとした．

POOM は，利用構造主導型の方法論の一つであり，利用構造主導型の他の方法論との比較についても評価する必要がある．しかし，本論文では POOM と他の利用構造主導型との差異の比較評価については実験の対象外とした．この理由は，以下の 2 つである．

(1) もし，POOM と他の利用構造主導型との差がないとしても，今回の実験結果は，利用構造主導型と OMT との差異を比較する上で有用であると考えたこと

(2) もし，POOM と他の利用構造主導型との差がある場合でも，オブジェクトのパターン化の効果は役割オブジェクトモデルに着目することの効果よりも大きくはないと考えたこと

これまでに，情報構造主導型と利用構造主導型のオブジェクト指向分析設計手法の有効性を比較した研究には，Jacobson の研究[8]と Wirfs-Brock の研究[9][10]がある． Jacobson の研究は，定性的な比較論に留まっており，具体的，定量的な研究ではない．これに対して Wirfs-Brock は，ビール醸造プロセスの制御問題に対して作成された情報構造主導型と利用構造主導型のオブジェクトモデルについて，各種の設計メトリクスを用いて定量的に比較している．Wirfs-Brock は，Shlaer/ Mellor 手法の経験者 1 名と RDD 手法の経験者 1 名が同じ問題に対して作成したオブジェクトモデルを比較している．Wirfs-Brock は，オブジェクトあたりの属性数やメソッド数，ならびに，シナリオあたりのメッセージ数などについて比較した結果，情報構造主導型が利用構造主導型よりも個数が多いという意味で複雑にな

っていると指摘している．この理由として，Wirfs-Brock は，情報構造主導型では，制御オブジェクトが多数のデータオブジェクトを集中的に制御するようなオブジェクトモデルであるのに対して，利用構造主導型では，自律的な役割を持つオブジェクトが互いに協調する分散型のオブジェクトモデルが設計されているためであるとしている．

本論文では，制御分野ではなく企業情報システム分野を対象として利用構造主導型の POOM と情報構造主導型の OMT とを比較した．情報システム分野については，利用構造主導型であっても，データオブジェクトを扱う必要があり，必ずしも制御システム分野と同じ結果になるとは限らないため，情報システム分野を対象として情報構造主導型と利用構造主導型とを定量的に比較する必要があった．また，最近のクライアント／サーバ型の企業情報システムでは，表示層，機能層，データ層からなる 3 層アーキテクチャによるシステム設計が進められている．Wirfs-Brock が比較した事例では，表示層のオブジェクトについての設計を除外しているが，企業情報システムの場合，表示層を介して，互いに異なる役割を持った担当者が情報を操作することによって業務が進行するため，表示層まで含めて情報構造主導型と利用構造主導型とを定量的に比較する必要がある．さらに，Wirfs-Brock の評価の前提となった比較実験では，一つの問題に対して 2 人の被験者が情報構造主導型手法(Shlaer/Mellor)と利用構造主導型手法(Responsibility Driven Method)とを別々に適用しているだけであり，問題数，被験者数の点でも十分な比較評価であるとは言えない．また，設計段階のオブジェクトモデルを対象としており，要求分析段階のオブジェクトモデルについての評価にはそのままでは適用できない．

本論文では，以上述べた理由から，POOM と OMT を複数人を被験者として情報システム分野について複数の問題に対して定量的に比較した．この結果，すでに述べたように，分析効率，オブジェクトモデルの均質性，階層型アーキテクチャとの親和性，オブジェクトモデルの妥当性，習熟性などの点で POOM が OMT よりも優れていることを定量的に明らかにしている．これらはいずれも Wirfs-Brock の論文では明らかにされていなかった点である．とくに，OMT の場合，複数の被験者間で作成されるオブジェクトモデルの個人差が大きいことについてはこれまでのオブジェクト指向分析手法の研究では明らかにされていなかった．Wirfs-Brock の研究と本論文の違いをまとめて表 6−1 に示す．

表 6−1　Wirfs-Brock の研究と本論文の比較

	Wirfs-Brockの研究	本研究
比較対象手法	Shlaer/ Mellor手法 RDD手法	OMT手法 POOM手法
オブジェクトモデル	設計モデル	分析モデル
総被験者数	2名	24名
分析作業の形態	個人による分析	チームによる分析
対象分野	制御システム分野	情報システム分野

本論文では，被験者が共通に挙げた多数オブジェクトの個数を比較することにより，情報構造主導型の OMT よりも利用構造主導型の POOM の方がより均質性の高いオブジェクトモデルを作成できることを明らかにした．本実験では，オブジェクトモデルが階層アーキテクチャと対応しやすいことについて，直接的な評価は行っていない．しかし，オブジェクトモデルを構成する表示層オブジェクト，機能層オブジェクト，データ層オブジェクトの個数の比を比較した結果から，POOM の方が OMT よりもオブジェクトモデルの構成が安定しているという結果を得た．これまで，OMT は現実世界のオブジェクトを対象としているため，個人差が少ないのではないかと思われていたが，今回の結果からデータ層オブジェクトの構成比率に関する被験者間での分散が OMT では大きくなることが明らかになった．これは，POOM で用いた役割オブジェクトモデルのような振る舞いを最初に分析していく手法の方が，データ層のオブジェクトに対する制約を明示していくことができ，結果として，データ層のオブジェクトも均質化できるということだと考えられる．

6.3.1 分析ドキュメント

図 6−5 に示すように，本実験では，次の分析ドキュメントを作成することにより，OMT と POOM を比較した．したがって，オブジェクト候補一覧と役割オブジェクトモデル図の違いを除けば，OMT グループと POOM グループの作成ドキュメントはほぼ同じである．

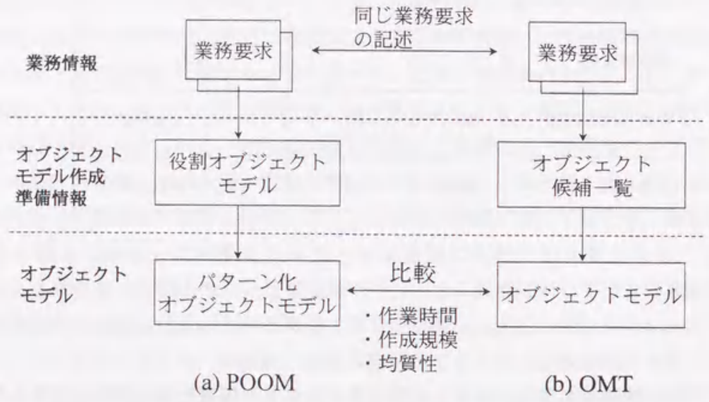


図 6−5　POOM と OMT の比較実験

(1) オブジェクトの抽出

OMT グループでは，業務記述に基づき，現実世界のオブジェクトを考慮してオブジェクトを抽出し，オブジェクト候補一覧を作成した．POOM グループでは，システムを利用する担当者の役割を考えて，役割オブジェクトとその関係を抽出し，役割オブジェクトモデル図を作成する．

(2) オブジェクトモデル図の作成

OMT グループでは，オブジェクト候補一覧に基づき，オブジェクトモデル図を作成する．POOM グループでは，役割オブジェクトモデル図とオブジェクトの分類に基づき，パターン化オブジェクトモデル図を作成する．

また，本実験の実施時に，POOM を支援するツールがなかった．このため，OMT グループ，POOM グループともワープロを用いて，分析ドキュメントを作成することにより，実験条件を揃えた．

6.3.2　実験プロジェクトの実行計画

分析実験では，分析手法の学習と，各手法を用いた分析モデルの作成を行う必要がある．具体的な実験手順を以下に示す．

［手順 1］手法の学習

約 3 時間の講習会により， OMT グループと POOM グループごとにそれぞれのオブジェクト指向分析手法を学習した．

［手順 2］オブジェクトモデルの個人による作成実験

個人分析問題 P-1 と P-2 についてオブジェクトモデルを作成する．分析対象とする問題は，書店会計業務問題（10 個の日本語文）と旅行予約業務問題（12 個の日本語文）である．

［手順 3］チームによるオブジェクトモデルの作成実験

POOM と OMT ごとに 3 人で 1 組の分析チームを構成し，チーム分析問題についてオブジェクトモデルを作成する．チームによる共同分析作業で対象とする問題は酒販売業務問題（87 個の日本語文）である．

また，分析実験を進める場合，被験者ごとに作業時間に差が出ると考えられることから，図 6−6 に示すように，分析問題ごとに週を割り当てたスケジュールとした．

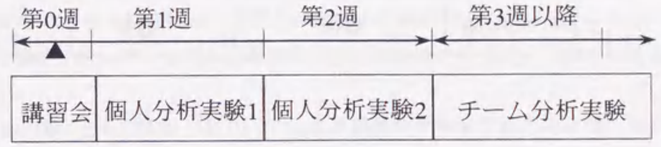


図 6−6　分析実験のスケジュール

6.3.3 プロジェクトチームの構成

OMT と POOM によるオブジェクト指向分析の効果を評価するため，被験者を OMT グループと POOM グループに分け，それぞれのグループの被験者が同じ問題について個別に分析した．被験者数は，各手法ごとに 12 名とし，合計 24 名である．今回の比較実験では，OMT の経験を持つ被験者が 6 名だった．被験者のソフトウェア開発経験やオブジェクト指向分析手法の経験が，グループごとに偏らないように，被験者を各グループにできるだけ均等に割り当てた．ソフトウェア開発経験年数の平均についての t 検定を実施したところ，各グループで有意な差がなかった（有意水準 5%）．実験プロジェクトの被験者構成を表 6−2 に示す．

表 6−2 実験プロジェクトの被験者構成

	OMT				POOM			
チーム	A	B	C	D	A	B	C	D
被験者数	3	3	3	3	3	3	3	3
OMTの経験	yes	no	no	no	yes	no	no	no
プログラム開発の経験年数の平均	4.33	7.33	6.67	5.67	6.0	5.33	10.3	4.67
グループの平均	6.0				6.6			

6.3.4 実験上の限界

(1) 実験期間

被験者のグループごとに OMT と POOM の講習会を実施してから，分析実験を行った．講習会の時間は，約 3 時間である．講習会の後，2 つの個人問題についてオブジェクトモデルを作成した．最後に，チーム問題について各被験者がチームでオブジェクトモデルを作成した．オブジェクト指向では，十分な訓練が必要であるとされている．今回の実験では，講習会と個人問題で約 2 週間をとっているが，OMT や POOM を十分に訓練した上での比較評価ではない．したがって，厳密にいえば，今回の実験結果は，短期的な訓練を行った場合の OMT と POOM の比較評価である．

(2) 実験方法

分析実験を実施する場合，同じ場所に被験者を集合させて実施する方法と，被験者が所属する機関で通常の業務環境の下で実施する方法とが考えられる．前者の場合，作業環境を統一できるが，同時に多数の被験者を集める環境を用意する必要があるだけでなく，非現実的な環境下での実験になるという問題がある．後者の場合，作業環境は被験者ごとに異なるが，実際の開発環境で現実的な実験データを収集できる．本実験では，長時間の分析作業になることが予想されること，ならびに，実環境下での分析データを収集できることから，実際の開発現場の環境で分析実験を進めた．ただし，被験者間で分析作業中に互いに情報交換しないように留意して実験を進めた．

(3) オブジェクト指向分析手法の選択

オブジェクトモデル作成上の主要な着目点としては，役割，機能，データの 3 つが考えられる．役割に着目するのが利用構造主導型方法論であり，データに着目するのが情報構造主導型方法論である．しかし，機能に着目してオブジェクトモデルを作成するのは，オブジェクト指向にはなじまないと思われる．したがって，本論文では，利用構造主導型の POOM と情報構造主導型の OMT とを比較することとした．

また，POOM は，利用構造主導型の方法論の一つであり，利用構造主導型の他の方法論との比較についても評価する必要がある．しかし，本論文では POOM と他の利用構造主導型との差異の比較評価については実験の対象外とした．この理由は，以下の 2 つである．

(a) もし，POOM と他の利用構造主導型との差がないとしても，今回の実験結果は，利用構造主導型と OMT との差異を比較する上で有用であると考えたこと

(b) もし，POOM と他の利用構造主導型との差がある場合でも，オブジェクトのパターン化の効果は役割オブジェクトモデルに着目することの効果よりも大きくはないと考えたこと

ただし，(a) (b) ともに仮説であり，これらの妥当性を検証するためには，さらに厳密な比較実験が必要である．

(4) 問題分野の選択

今回の実験では，情報システム分野を対象として，書店会計業務問題，旅行予約業務問題，酒販売業務問題を選定した．この点では，今回の実験結果は情報システム分野に限定される．したがって，制御システム系分野など情報システム以外の分野については，今回の実験結果がそのまま適用できない可能性がある．

(5) 実験範囲

今回の実験では，オブジェクトモデルの作成までを対象としている．したがって，設計，製造，試験まで含めたシステム開発全体に対するオブジェクト指向分析手法の比較評価ではない．

6.4 個人による分析実験の結果

6.4.1 分析データ

分析実験のデータを表6−3に示す．オブジェクトモデルを作成するために要した時間ならびにオブジェクトモデルにおけるオブジェクト数と関係数を調べた．また，オブジェクトモデルの均質性を評価するため，多数の被験者が共通に抽出したオブジェクトを何個記述しているかを多数オブジェクト数 COM として調べた．ここで，あるオブジェクト Obj が被験者グループ A において過半数の被験者によって抽出されているとき，Obj を被験者グループ A における多数オブジェクトであるといい，COM_A(Obj)で表す．このとき，グループ A の被験者 S が作成したオブジェクトモデル図 M の多数オブジェクト数とは，COM_A(Obj)を満たすような M に含まれるオブジェクト Obj の個数である．

また，分析実験データのうち，OMT-1,OMT-2,OMT-3 ならびに，POOM-1,POOM-2,POOM-3 の各被験者は，いずれも OMT の経験者である．これ以外の被験者はいずれも OMT の経験者ではない．

表6−3　分析実験　結果データ

被験者	[問題 P1] 書店会計問題						[問題 P2] 旅行代理店問題					
	OMD			OID			OMD			OID		
	T1	NO1	CO1	T2	NO2	CO2	T1	NO1	CO1	T2	NO2	CO2
OMT-1	180	13	6	90	13	6	130	18	11	120	16	11
OMT-2	120	13	11	80	8	8	90	22	12	60	17	11
OMT-3	200	23	10	50	15	9	200	19	11	120	15	9
OMT-4	240	10	7	200	14	7	330	14	1	160	19	3
OMT-5	510	18	7	280	22	8	150	19	11	260	15	10
OMT-6	720	9	4	500	12	6	600	19	5	120	20	7
OMT-7	300	12	10	90	17	4	1590	9	6	420	21	7
OMT-8	180	10	9	80	11	8	420	9	5	230	11	3
OMT-9	65	13	11	50	14	8	90	11	10	240	17	9
OMT-10	260	14	11	150	13	9	255	13	11	250	21	9
OMT-11	180	11	7	156	14	8	900	10	5	360	8	4
OMT-12	162	11	11	130	14	9	240	14	11	180	19	11
POOM-1	55	18	16	30	12	9	140	22	20	80	19	16
POOM-2	90	19	18	30	11	8	150	22	20	180	19	17
POOM-3	105	20	18	100	15	13	130	22	19	200	19	15
POOM-4	120	17	16	55	12	11	105	22	16	145	19	12
POOM-5	60	15	14	115	12	11	150	20	18	80	12	12
POOM-6	150	17	14	240	14	12	150	20	18	360	15	14
POOM-7	40	21	16	40	13	9	100	23	21	175	18	17
POOM-8	65	19	16	65	11	9	60	19	12	90	13	4
POOM-9	175	21	17	175	15	13	140	25	22	175	15	14
POOM-10	100	13	11	100	10	8	400	24	19	265	20	14
POOM-11	90	18	17	90	14	12	214	22	18	326	19	13
POOM-12	430	17	15	430	15	11	355	23	21	380	21	17

（表注）T1:オブジェクトモデル図作成時間（オブジェクト一覧，役割フロー作成時間を含む）単位は分である． NO1:オブジェクトモデル図に含まれるオブジェクト数．CO1: オブジェクトモデル図に含まれる多数オブジェクトの数． T2:オブジェクト交信図作成時間（分）． NO2:オブジェクト交信図に含まれるオブジェクト数．CO2: オブジェクト交信図に含まれる多数オブジェクトの数．

6.4.2　オブジェクト数と関係数の比較

［**観察 1**］作成されたオブジェクトモデル図のオブジェクト数は，POOM の方が OMT よりも多い（有意水準 5%）．

分析問題ごとの各グループのオブジェクト数の平均値を比較して図 6−7 に示す．POOM のオブジェクトモデル図のオブジェクト数の平均は，P-1 で約 17.9 個，P-2 では約 22 個となった．これに対して OMT のオブジェクトモデル図のオブジェクト数の平均は，P-1 では約 13.1 個，P-2 では約 14.8 個だった．したがって，約 37%から 49%程度 POOM の方が OMT よりもオブジェクト数が多いという結果となった．また，これらのオブジェクト数の平均値の間にはグループ間で有意な差が認められた．

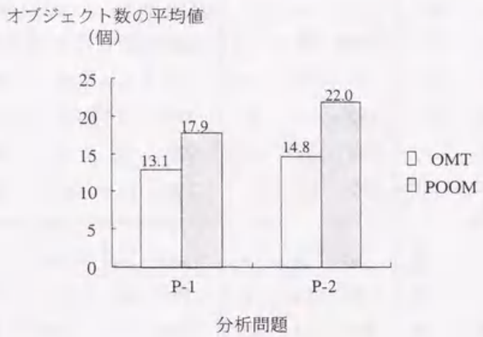


図 6−7　オブジェクト数の平均値

［**観察 2**］作成されたオブジェクトモデル図の多数オブジェクト数は，POOM の方が OMT よりも多い（有意水準 5%）．

POOM のオブジェクトモデル図の多数オブジェクト数の平均は，P-1 で約 15.7 個，P-2 では約 18.7 個となった．これに対して OMT のオブジェクトモデル図の多数オブジェクト数の平均は，P-1 では約 8.7 個，P-2 では約 8.3 個だった．したがって，約 2 倍程度 POOM の方が OMT よりも多数オブジェクト数が多いという結果となった．また，これらの多数オブジェクト数の平均値の間にはグループ間で有意な差が認められた．

［**観察 3**］作成されたオブジェクトモデル図の多数オブジェクト率は，POOM の方が OMT よりも高い（有意水準 5%）．

分析問題ごとの各グループの多数オブジェクト率の平均値を比較して図 6−8 に示す．ここで，被験者が作成したオブジェクトモデル図に含まれるオブジェクト数に対する多数オブジェクト数の比率を多数オブジェクト率という．POOM のオブジェクトモデル図の多数オブジェクト率の平均は，P-1 で約 87.7%，P-2 では約 84.6%となった．これに対して OMT のオブジェクトモデル図の多数オブジェクト率の平均は，P-1 では約 69.0%，P-2 では約 57.6%だった．したがって，それぞれ，18.7 ポイント，27 ポイントだけ POOM の方が OMT よりも多数オブジェクト率が高いという結果となった．また，これらの多数オブジェクト率の平均の間にはグループ間で有意な差が認められた．

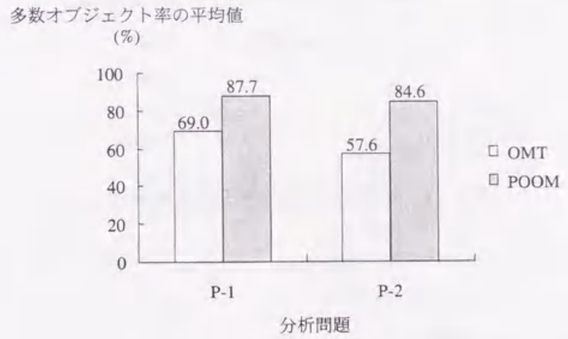


図 6−8　多数オブジェクト率の平均値

［**観察 4**］オブジェクトモデル図の関係数は，POOM の方が OMT よりも多い．

分析問題ごとの各グループのオブジェクトモデルにおける関係数の平均値を比較して図 6−9 に示す．ただし，P-2 では POOM の方が OMT よりも作成されたオブジェクトモデル図の関係数に有意な差が認められたが（有意水準 5%），P-1 では，POOM と OMT で関係数の平均値について有意な差はなかった．

［**観察 5**］オブジェクトモデルのオブジェクト数と関係数には正の相関がある．

分析問題ごとの各被験者のオブジェクト数と関係数の関係を図 6−10 に示す．P-1 の場合，オブジェクト数と関係数の間の相関係数は，OMT で 0.97，POOM で 0.95 である．P-2 の場合，オブジェクト数と関係数の間の相関係数は，OMT，POOM ともにで 0.82 である．したがって，オブジェクトモデルのオブジェクト数と関係数には，強い正の相関があるこ

とが明らかになった.

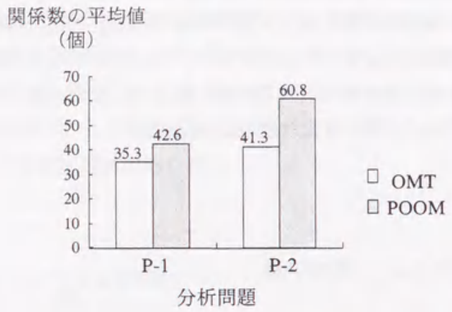


図 6−9　オブジェクトモデル図の関係数

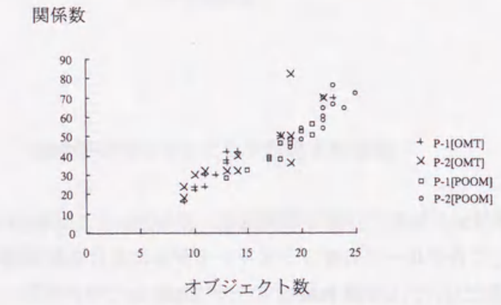


図 6−10　オブジェクト数と関係数の関係

6.4.3　作業時間分析

［**観察 6**］オブジェクトモデル図の作成時間は，POOM の方が OMT よりも少ない（有意水準 5%～10%）.

POOM のオブジェクトモデル図の作成時間の平均は，P-1 で約 120 分，P-2 では約 175 分となった. これに対して OMT のオブジェクトモデル図の作成時間の平均は, P-1 では約 260 分，P-2 では約 460 分だった. また，これらのオブジェクトモデル図の作成時間の平均値の間には，P-1 については有意水準 5%で，P-2 については有意水準 10%でグループ間に有意な差が認められた. 分析問題ごとのオブジェクトモデル作成時間の平均値の比較を図 6−11 に示す.

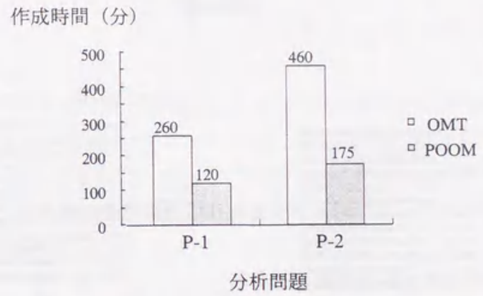


図 6−11　オブジェクトモデル作成時間の平均値

［**観察 7**］オブジェクト当たりの分析時間の個人差は， OMT では，16 倍から約 40 倍である. これに対して POOM では，約 5 倍から約 10 倍である.

問題ごとのオブジェクトモデル作成時間の最大値と最小値の比を比較して図 6−12 に示す. POOM では，P-1 よりも P-2 の方が分析時間の個人差が小さくなった. これに対して，OMT では，逆に， P-1 よりも P-2 の方が分析時間の個人差が大きくなった.

オブジェクトモデル分析時間で見ると，P-1 と P-2 のいずれの場合についても，POOM は OMT の約 40%の分析時間となることが判明した. このように POOM が，分析工数を削減できた要因は，POOM では業務担当者間の役割に基づいて分析することと，予めオブジェクトを類型化しているためである.

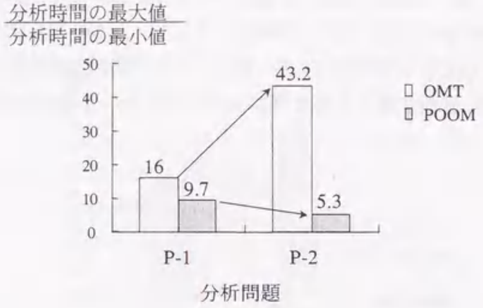


図 6－12　オブジェクト当たりの分析時間の個人差

6.4.4　階層型アーキテクチャとの親和性の分析

［**観察 8**］POOM の方が OMT よりも，オブジェクトモデルにおける表示層，機能層，データ層オブジェクトの比率が被験者間で安定している．

P-1 に対する被験者ごとの OMT のオブジェクト構成比を図 6－13(a)に，POOM のオブジェクト構成比を図 6－13(b)に示す．また，P-1 に対する被験者ごとの OMT のオブジェクト構成比を図 6－13(c)に，POOM のオブジェクト構成比を図 6－13(d)に示す．これらの図から POOM の方が OMT よりも，オブジェクトモデルにおける表示層，機能層，データ層の各オブジェクトの比率が被験者間で安定していることが分かる．また，各層オブジェクトの構成比率の分散について，問題ごとに，OMT と POOM の間で F 検定を実施したところ，P-1，P-2 のどちらについても，データ層，機能層，表示層のオブジェクト構成比率の分散が，OMT と POOM で有意な差が認められた（有意水準 5%）．

［**観察 9**］POOM の方が OMT よりも，オブジェクトモデルにおける表示層，機能層，データ層オブジェクトの比率が問題の影響を受けにくい．

図 6－13(a)と図 6－13(c)を比較すると，OMT の場合，問題によって，同じ被験者でも，オブジェクトの構成比が大きく変動することが分かる．これに対して，図 6－13(b)と図 6－13(d)を比較すれば，POOM の場合，P-1 よりも P-2 の方が表示層のオブジェクトが減って機能層，データ層のオブジェクトが増加しているものの，オブジェクトの構成比は安

定している．また，各層オブジェクトの構成比率の分散について，問題間で F 検定を実施したところ，以下のようになった．

（1）OMT では，データ層と機能層のオブジェクト構成比率に関する分散には有意な差がある（有意水準 5%）．また，OMT の表示層オブジェクトの構成比率に関する分散については，問題間で有意な差がないという結果になった．しかし，これは，OMT の場合，表示層オブジェクトの構成比率に関する分散が両方の問題とも大きいためである．

（2）POOM では，データ層と機能層のオブジェクト構成比率に関する分散が等しいことを棄却できない（有意水準 5%）．また，POOM の表示層オブジェクトの構成比率に関する分散については，問題間で有意な差があるという結果になった（有意水準 5%）．しかし，この原因は，表示層オブジェクトの構成比率に関する分散が P-1 よりも P-2 の方が減少したためである．

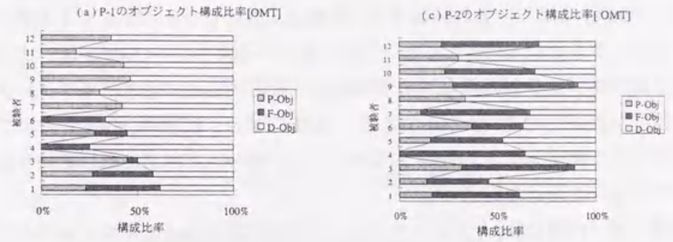


図 6－13　オブジェクト構成比

6.4.5 少数オブジェクト数の比較分析

以下では，オブジェクトモデルの作成で，過半数の被験者によって抽出されなかったオブジェクトについて分析する．ここで，あるオブジェクト Obj が被験者グループ A において過半数の被験者によって抽出されていないとき，Obj を被験者グループ A における少数オブジェクトであるといい，Min_A(Obj)であらわす．このとき，被験者グループ A の被験者 S が作成したオブジェクトモデル図 M の少数オブジェクト数とは，Min_A(Obj)を満たすようなMに含まれるオブジェクト Obj の個数である．

［**観察 10**］POOM の方が OMT よりも，オブジェクトモデルにおける少数オブジェクトの個数が少ない．

P-1 に対する OMT と POOM の少数オブジェクト数を図 6－14 に示す．また，P-2 に対する OMT と POOM の少数オブジェクト数を図 6－15 に示す．OMT では，一人の被験者だけが抽出したオブジェクトが P-1 と P-2 の平均で，少数オブジェクトの約 74.7%を占めている．これに対して POOM では，一人の被験者だけが抽出したオブジェクトが P-1 と P-2 の平均で，少数オブジェクトの約 62.9%だった．これからわかるように，OMT では，オブジェクト数の内，少数オブジェクトの割合が約 39%だが，その内の約 74.7%のオブジェクトは一人の被験者だけがあげたオブジェクトである．OMT の場合，被験者が作成したオブジェクトモデルに含まれるオブジェクトの約 30%がその被験者一人だけが抽出しており他の被験者は抽出していないことになる．

POOM では，約 14%が少数オブジェクトで，その内の約 62.9%のオブジェクトを異なる被験者が抽出している．したがって，POOM では，オブジェクトモデルに含まれるオブジェクトの約 9%だけを一人の被験者が抽出したことになる．これらからわかるように，少数オブジェクトは個人差が大きく問題分析で必須となるオブジェクトが少数オブジェクトの中に含まれているとは考え難い．また，POOM では，もともと少数オブジェクト数が 14%程度しかないので，オブジェクトモデル全体に影響を与えるとは考えられない．

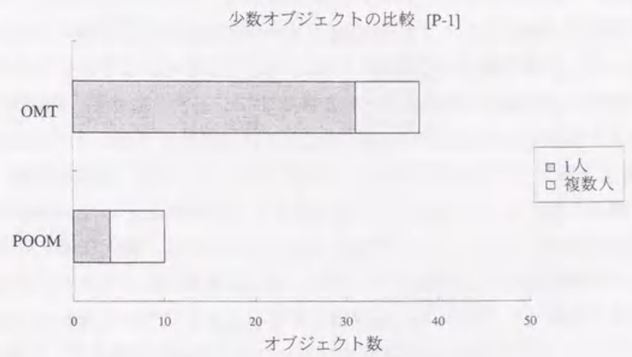


図 6－14　P-1 に対する少数オブジェクト数の比較

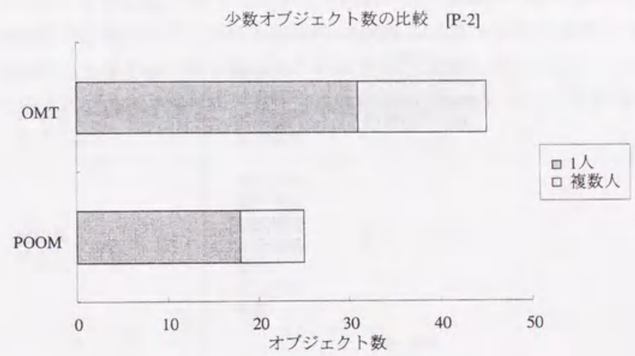


図 6－15　P-2 に対する少数オブジェクト数の比較

6.4.6 多数オブジェクトの比較分析

［観察 11］OMT の多数オブジェクトは POOM の多数オブジェクトに含まれる．

P-1，P-2 に対して OMT グループと POOM グループで抽出された多数オブジェクトを表示層，機能層，データ層の観点から比較して図 6－16 と図 6－17 にそれぞれ示す．

P-1 では，OMT と POOM 共通に抽出された多数オブジェクトは 9 個，OMT だけにある多数オブジェクトは顧客と納入伝票の 2 個，POOM だけの多数オブジェクトは 9 個である．顧客が POOM で多数オブジェクトになっていないのは，P-1 では，顧客が直接，書店の会計システムを利用しないので，役割オブジェクトとして抽出されていないためである．また，納入伝票が多数オブジェクトとして抽出されていないのは，納入伝票と売上伝票が同一であることが業務記述の中に明記されており，売上伝票だけをデータオブジェクトとして抽出したためである．したがって，本質的なオブジェクトについて見れば，P-1 では，OMT の多数オブジェクトは POOM の多数オブジェクトに含まれていることになる．POOM では，入力端末とトランザクションという 2 つの上位オブジェクトが多数オブジェクトとして抽出されている．これらの上位オブジェクトは問題分析で必ずしも必須とはいえないが，問題の構造を整理する上で重要なオブジェクトであり，多数オブジェクトとして抽出されることが望ましいと考えられる．

P-2 では，OMT と POOM 共通に抽出された多数オブジェクトは 12 個，OMT だけにある多数オブジェクトは 0 個，POOM だけの多数オブジェクトは 11 個である．したがって，P-2 の場合，OMT の多数オブジェクトは POOM の多数オブジェクトに完全に含まれている．P-1 で抽出されていなかった「顧客」が POOM の表示層オブジェクトとして抽出されているのは，旅行代理店システムを顧客が直接利用して旅行の予約や解約などを行う可能性があるためである．

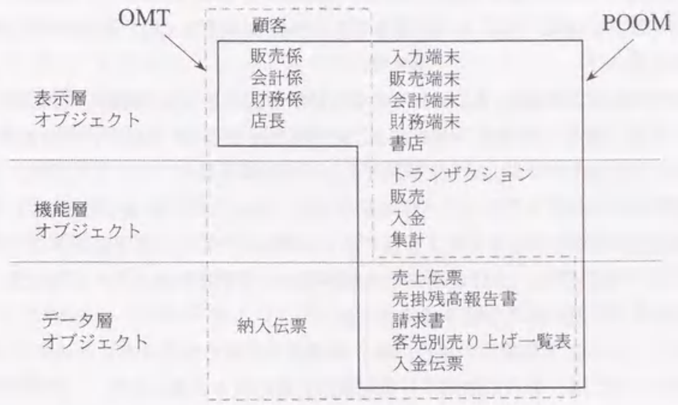


図 6－16　P-1 で抽出された多数オブジェクトの比較

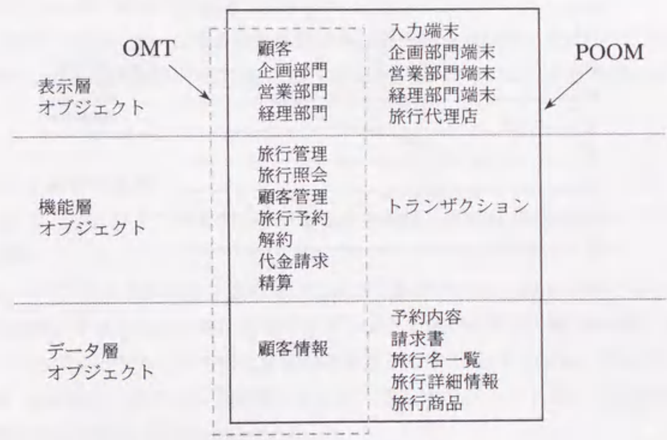


図 6－17　P-2 で抽出された多数オブジェクトの比較

6.4.7 OOA 経験と習熟度との関係

［**観察 12**］POOM では，2 回目の適用で OMT 未経験者と OMT 経験者の分析効率が同等になる．これに対して OMT では，2 回の適用では，OMT 経験者と OMT 未経験者の分析効率は差が縮まらない．

OMT と POOM による P-1，P-2 に対する分析時間の変化を OMT 経験の有無を区別して図 6－18 に示す．図 6－18 では，OMT/Exp，POOM/Exp が OMT 経験者の分析効率の平均を，OMT/Nov, POOM/Nov が OMT 未経験者の分析効率の平均を，それぞれ示す．ここで，分析効率は時間当たりのオブジェクト作成数を示す．図 6－18 から，POOM では，OMT 経験者と未経験者の分析効率の比が約 1 回目で約 1.25 倍あったが 2 回目では差がなくなった．これに対して，OMT では，OMT 経験者と未経験者の分析効率の比が約 1 回目で約 1.56 倍あったが 2 回目では差が拡大し約 3 倍になった．

この結果から，OMT 未経験者の場合，OMT 経験者と同等の分析効率になるまで OMT 手法に習熟するためには，さらに数回の分析経験が必要になると思われる．

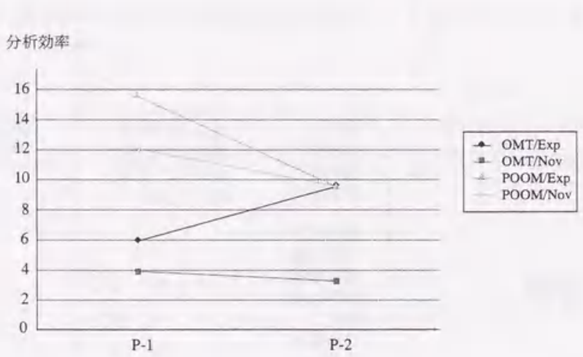


図 6－18　分析効率の比較

6.5 チームによる分析実験の結果

分析実験の結果から得られたデータを表 6－4 と表 6－5 に示す．各表では，OMT チームと POOM チームについて，オブジェクトモデルを作成するために要した時間ならびにオブジェクトモデルにおけるオブジェクト数や関係数を調べた．とくに，オブジェクト数については，表示層，機能層，データ層に対応するオブジェクトの個数についても調べた．また，オブジェクトモデルの均質性を評価するため，多数の被験者が共通に抽出したオブジェクトを何個記述しているかを多数オブジェクト数 COM として調べた．ここで，あるオブジェクト Obj が同じ分析手法グループで半数以上のチームによって抽出されているとき，Obj をその手法グループ G における多数オブジェクトであるといい，COM_G(Obj)で表す．このとき，手法グループ G のあるチームが作成したオブジェクトモデル図 D の多数オブジェクト数は COM_G(Obj)を満たす D 内のオブジェクト Obj の個数である．

6.5.1 分析時間

［**観察 1**］オブジェクトモデル図の作成時間は，POOM の方が OMT よりも少ない（有意水準 5%）．

POOM のオブジェクトモデル図作成時間の平均は，約 10.4 時間となった．これに対して OMT のオブジェクトモデル図作成時間の平均は，約 20.3 時間だった．したがって，POOM の分析時間は OMT の分析時間の約 50%であるという結果となった．また分析時間の平均値の間にはチーム間で統計的に有意な差が認められた．

また，OMT の場合，チーム間の分析時間の差は最大約 5 倍になった．これに対して，POOM では，チーム間の分析時間の差は約 2 倍であった．図 6－19 に，分析時間の比較結果を示す．

6.5.2 分析生産物の規模

［**観察 2**］オブジェクトモデル図のオブジェクト数は，POOM の方が OMT よりも多い（有意水準 5%）．

POOM のオブジェクトモデル図のオブジェクト数の平均は，約 52.5 個となった．これに対して OMT のオブジェクトモデル図のオブジェクト数の平均は，約 30.3 個だった．すなわち OMT のオブジェクトモデル図に含まれるオブジェクト数は，POOM の場合のオブジェクト数の約 60%程度であるという結果となった．またオブジェクト数の平均値の間にはチーム間で統計的に有意な差が認められた．

［**観察 3**］オブジェクトモデル図の関係数は，POOM の方が OMT よりも多い（有意水準 5%）．POOM のオブジェクトモデル図の関係数の平均は，約 78.8 個となった．これに対して OMT のオブジェクトモデル図の関係数の平均は，約 48.3 個だった．すなわち OMT のオブジェクトモデル図に含まれる関係数は，POOM の場合の関係数の約 60%程度であるという結果となった．また，関係数の平均値の間にはチーム間で統計的に有意な差が認められた．

表 6－4　OMT チームの分析実験結果

チーム	OMT-A	OMT-B	OMT-C	OMT-D	平均
オブジェクト抽出時間	60	900	180	180	330
オブジェクトモデル化時間	900	1200	1500	240	960
オブジェクト数	39	11	31	40	30.25
表示層オブジェクト数	5	0	5	5	3.75
機能層オブジェクト数	17	6	7	14	11
データ層オブジェクト数	17	5	19	21	15.5
多数オブジェクト数	7	1	7	6	5.25
表示層多数オブジェクト数	3	0	3	3	2.25
機能層多数オブジェクト数	0	0	0	0	0
データ層多数オブジェクト数	4	1	4	3	3
関係数	62	23	63	45	48.25
多数オブジェクト率	0.1795	0.0909	0.2258	0.15	0.1616
生産性	6.3125	0.9714	3.3571	12.75	3.6654

表 6－5　POOM チームの分析実験結果

チーム	POOM-A	POOM-B	POOM-C	POOM-D	平均
オブジェクト抽出時間	270	75	180	300	206.25
オブジェクトモデル化時間	300	480	300	600	420
オブジェクト数	68	43	45	54	52.5
表示層オブジェクト数	12	8	11	10	10.25
機能層オブジェクト数	20	15	12	16	15.75
データ層オブジェクト数	36	20	22	28	26.5
多数オブジェクト数	36	34	35	38	35.75
表示層多数オブジェクト数	11	7	11	10	9.75
機能層多数オブジェクト数	12	15	12	15	13.5
データ層多数オブジェクト数	13	12	12	13	12.5
関係数	103	67	71	74	78.75
多数オブジェクト率	0.5294	0.7907	0.7778	0.7037	0.7004
生産性	18.0	11.892	14.5	8.5353	12.575

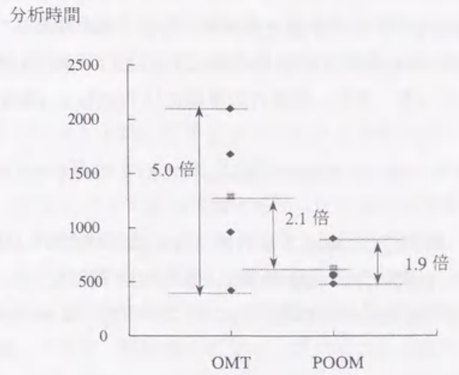


図 6－19　分析時間の比較

［**観察 4**］オブジェクトモデル図の生産性は，POOM の方が OMT よりも高い．（有意水準 10%）．

1時間当たりに作成されたオブジェクト数と関係数の合計を生産性として，POOM と OMT を比較した．POOM と OMT の生産性の平均は，それぞれ約 13.2 と約 5.8 となり，POOM が OMT の約 2 倍の生産性となった．また，生産性の平均には POOM と OMT で統計的に有意な差が認められた．

［**観察 5**］オブジェクトモデル図の生産性の分散は，POOM の方が OMT よりも小さい（有意水準 5%）．

生産性の分散については，OMT と POOM で統計的に有意な差が認められた．

また，OMT の場合，チーム間の分析生産性の差は最大約 13 倍になった．これに対して，POOM では，チーム間の分析生産性の差は最大約 2 倍であった．図 6－20 に，分析生産性の比較結果を示す．

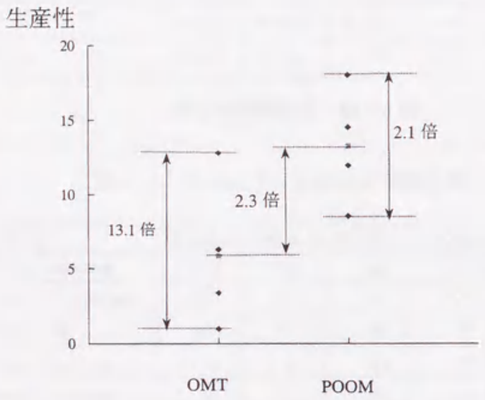


図 6－20　分析生産性の比較

6.5.3 分析プロセスの特性

オブジェクト抽出作業とオブジェクトモデル作成作業の工数分布を図 6－21 に示す．OMT では，オブジェクト抽出時間とオブジェクトモデル作成時間の比が約 1 対 3 となった．これに対して POOM では，オブジェクト抽出時間とオブジェクトモデル作成時間の比が約 1 対 2 となった．

［**観察 6**］抽出オブジェクト数に対するオブジェクトモデル図のオブジェクト数の比は，POOM の方が OMT よりも大きい．（有意水準 5%）．

図 6－22 に，オブジェクトの抽出作業とモデル化作業におけるオブジェクト数の関係を示す．この図から，POOM では，OMT に対して，抽出作業では，約 20 分の 1 のオブジェクトを抽出して，モデル化作業では約 1.5 倍のオブジェクトを作成したことが分かる．

OMT と POOM の抽出作業でのオブジェクト数の差が大きい理由は，OMT がオブジェクトを網羅的に抽出しておき，最終的に必要なオブジェクトに絞り込んでいく方式であるのに対して，POOM では，最初から必要な役割型オブジェクトだけをシナリオに基づいて抽出していくためであると考えられる．

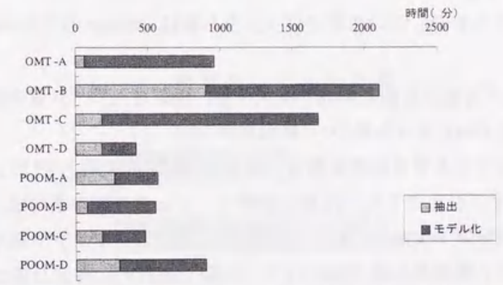


図 6－21　工数分布

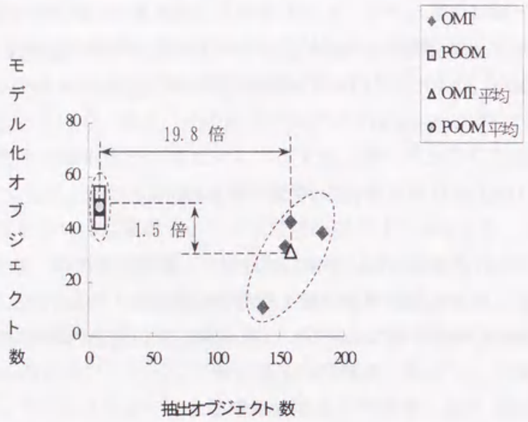


図 6－22　オブジェクト数の関係

6.5.4 分析生産物の特性

[**観察 7**]オブジェクトモデル図の多数オブジェクト数は，POOM の方が OMT よりも多い．（有意水準 5%）．

[**観察 8**] オブジェクトモデル図を構成する表示層，機能層，データ層の多数オブジェクト率は，POOM の方が OMT よりも多い．（有意水準 5%）．

POOM のオブジェクトモデル図の多数オブジェクト数の平均は，約 35.8 個となった．これに対してOMTのオブジェクトモデル図の多数オブジェクト数の平均は，約5.3個だった．したがって，約 7 倍程度 POOM の方が OMT よりも多数オブジェクト数が多い結果となった．多数オブジェクト数の平均値の間にはチーム間で統計的に有意な差が認められた．表示層，機能層，データ層の多数オブジェクト率の平均にも，チーム間で統計的に有意な差が認められた．

図 6－23 と図 6－24 に，OMT と POOM のチームごとの多数オブジェクト率を示す．OMT では，機能層に多数オブジェクトがないこと，全体的に多数オブジェクト率が低いことが分かる．これに対して，POOM では，表示層，機能層，データ層にわたって，各チームに多数オブジェクトが存在することが分かる．

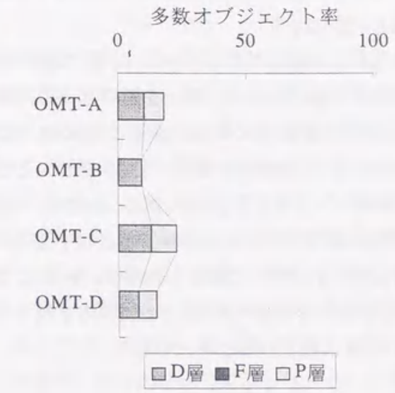


図 6－23　OMT の多数オブジェクト率

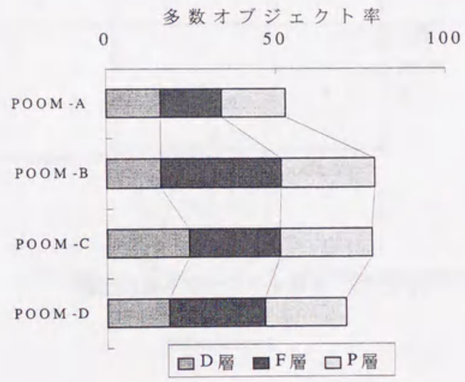


図 6－24　POOM の多数オブジェクト率

［観察 9］OMT の多数オブジェクトの約 77%が POOM の多数オブジェクトに含まれる．

OMT グループと POOM グループで抽出された多数オブジェクトを表示層，機能層，データ層の観点から比較して図 6−25 に示す．

OMT と POOM 共通に抽出された多数オブジェクトは 10 個である．OMT だけにある多数オブジェクトは，表示層の業者，経理担当と，データ層のセールス要員情報の 3 個である．POOM だけの多数オブジェクトは 27 個である．業者は POOM ではデータ層オブジェクトに業者情報として存在する．また，経理担当は，外部システムとしての経理システムを操作するので，本来，分析対象システムの範囲外である．POOM では，セールス要員情報は，受注情報や販売実績情報の属性中にセールスマン名として含まれる．したがって，分析対象システムで必要となるオブジェクトに限定した場合，POOM の他の多数オブジェクトの属性として存在するオブジェクトを含めれば， OMT の多数オブジェクトは，すべて POOM の多数オブジェクトに含まれていると考えられる．

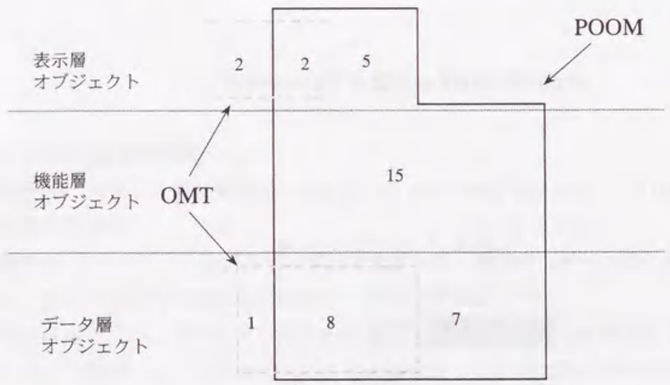


図 6−25　多数オブジェクト数の比較

6.5.5 アンケート評価

［観察 10］被験者のアンケートによる主観評価の点数は，POOM の方が OMT よりも高い．（有意水準 5%）．

実験分析後に，各被験者に対して，分析手法についてのアンケートを実施した．アンケートでは，以下の質問に対して，はいまたはいいえのどちらかで回答させた．

［質問 1］実験で用いた手法は分析を進める上で着想を刺激しましたか？
［質問 2］実験で用いた手法により分析する上で混乱しなかったですか？
［質問 3］実験で用いた手法は，アーキテクチャ設計との連続性が高いと思いますか？
［質問 4］実験で用いた手法により客観的な分析ができると思いますか？
［質問 5］実験で用いた手法によるモデル図は理解しやすいですか？
［質問 6］業務記述とモデル図のギャップが小さいと思いますか？

この結果を図 6−26 に示す．同図からわかるように，POOM では，これらのすべての質問に対して過半数の被験者が肯定的な回答を示した．これに対して，OMT では，質問 1 を除いて，過半数の被験者が否定的な回答を示した．とくに業務記述とモデル図のギャップが大きいと感じた被験者が OMT の場合には多かった．

また， POOM では，各被験者の肯定回答質問数の平均は約 4.17 である．これに対して，OMT では，各被験者の肯定回答質問数の平均は約 1.75 である．肯定回答質問数の平均には，POOM と OMT 間で統計的に有意な差が認められた．

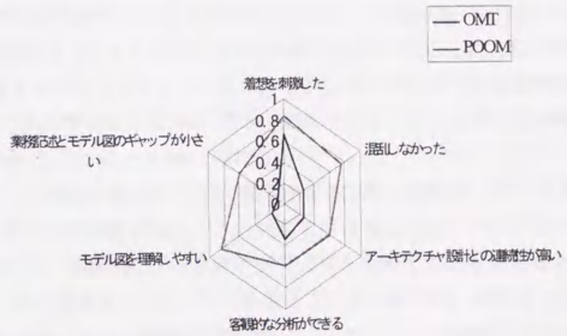


図 6−26　アンケート評価

6.5.6 議論

(1) POOM の有効性

これまでに示した評価基準に対する POOM と OMT の平均値に関する統計的な検定結果を表 6−7 にまとめて示す．この表から，POOM が OMT に比べて有効性の高いオブジェクト指向分析手法であること実証された．

また，生産性と多数オブジェクト率の関係を図 6−27 に示す．この図から分かるように，POOM によるオブジェクト指向分析では，OMT に比べ，より均質性の高いオブジェクトモデルを，より効率的に作成できることが明らかになった．

表 6−6　POOM と OMT の比較結果

評価項目	結果	有意水準
分析時間の平均	OMT > POOM	5%
オブジェクト数の平均	OMT < POOM	5%
関係数の平均	OMT < POOM	5%
多数オブジェクト数の平均	OMT < POOM	5%
生産性の平均	OMT < POOM	10%
オピニオン評価の平均	OMT < POOM	5%

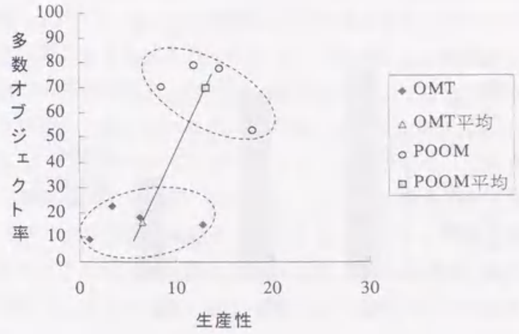


図 6−27　生産性と多数オブジェクト率の関係

(2)　POOM のパターン化の限界

各層ごとに多数オブジェクト率を比較した結果を図 6−28 に示す．この図からわかるように，POOM では，表示層，機能層では，POOM の多数オブジェクト率が高いにも係わらず，データ層の多数オブジェクト率が高くない．この理由は，(a) 表示層オブジェクトの中心となる役割型オブジェクトは担当者の業務を分析することにより自然に抽出できること (b) 機能層オブジェクトのトランザクション型オブジェクトは役割型オブジェクト間の関係から自然に抽出できること (c) データ層オブジェクトの抽出は役割間の関係だけでは完全に抽出できていないことが挙げられる．

したがって，現状の POOM では，問題が複雑になった場合，データ層オブジェクトの均質性が，低下する可能性がある．この問題を解決するには，データ層オブジェクトの分析手順を詳細化し，系統的にデータ層オブジェクトを抽出する方法を確立する必要がある．

しかし，データ層オブジェクトの多数オブジェクト率は，現状の POOM でも，OMT の約 2.4 倍だけ高い．以上の議論は，POOM をさらに完全な方法論にするための課題を考察したものであることに注意する必要がある．

また，問題が複雑になるにしたがって，問題業務の理解により多くの時間が必要になるため，パターン化によるモデル作成の効率化だけでは限界があると考えられる．これらについては，今後の課題である．

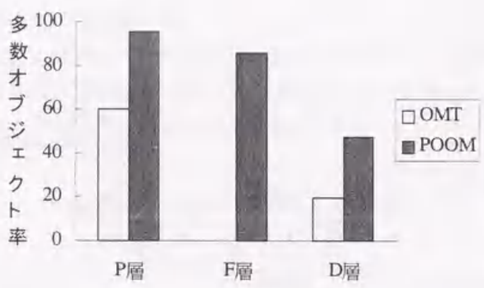


図 6−28　多数オブジェクト率の比較

(3) グループによる要求分析への適用性

グループによる要求分析のプラス要因としては，作業分担による 1 人当たりの作業負荷の軽減，知識の共有による生産物の品質の向上，多角的な視点による意思決定の信頼性の向上などが考えられる．逆に，グループによる要求分析のマイナス要因としては，コミュニケーション損失，迅速な意思決定が損なわれ易いこと，分析者間で表現や意味上の差違を統一する必要があることなどが挙げられる．今回の実験では，これらの要因の効果を詳細に評価することはできなかった．しかし，パターン化はオブジェクト分析モデルの表現を標準化する上で有効な考え方であり，今回の実験で POOM の方が OMT よりもグループ分析に適していることは実証できたと考える．

今後は，次の段階として，POOM を用いたグループによる要求分析のための方法論を明確にしていく必要がある．

6.6　データフロー図からのモジュール構造生成方式の有効性の評価

4.4 で提案したアルゴリズムの有効性を評価するため，NTT ソフトウェア研究所で開発したソフトウェア開発支援環境 SoftDA[10][11][12]のモジュール構造図作成支援機能を，本アルゴリズムを用いて実装した[13][14]．次に，本機能を用いて作成されるモジュール構造図を，データフロー図から手作業により作成されるモジュール構造図と比較評価した[15]．

モジュール構造図作成支援機能は，プロセス種別推定部，プロセス種別確認パネル部，モジュール構造図生成部から構成される[15]．プロセス種別推定部では，4.4.1 で提案したプロセス種別推定アルゴリズムを用いて，データフロー図のプロセスを入力，変換，出力に分類する．プロセス種別確認パネル部ではプロセスの分類結果を表形式でわかりやすく提示し，設計者によるプロセス種別の修正を可能にする．モジュール構造図生成部では，確認パネル部で指定されたプロセスの分類結果に基づき，4.4.2 で提案したモジュール構造生成アルゴリズムを用いて，データフロー図からモジュール構造図を生成する．

構造化分析手法を用いて人手によりデータフロー図からモジュール構造図を作成する場合と，提案したモジュール構造図作成支援機能を用いでモジュール構造図を作成する場合についてモジュール構造図の比較実験を行った．

実験ではメッセージ監視システム，口座管理システム，会議管理システムについてのデータフロー図からモジュール構造図を作成した．手作業で作成したモジュール構造図と自動生成したモジュール構造図について，モジュール数を比較した．手作業時のモジュール数を A，自動生成されたモジュール数を B とする．また手作業で作成したモジュールと同じ呼び出し関係を持つモジュールの個数を C とする．このとき，手作業モジュールの生成率を C/A で定義する．また自動生成モジュールの適合率を C/B で定義する．生成率が高ければ高いほど人手で作成したモジュールを自動的に生成できたことになる．逆に生成率が低いと人手で作成したモジュールを自動的に生成できていないことになる．人手で作成したモジュールをすべて自動生成したとしても，人手で作成していないモジュールまで自動生成する可能性がある．人手で作成していないモジュールを自動生成した場合，モジュール構造図が冗長になっている可能性がある．このため適合率では自動生成したモジュールのうち，人手で作成したモジュールと一致するモジュールの比率を表している．適合率が高ければ高いほど自動生成されたモジュールの冗長性が低いことになる．逆に適合率が低ければ自動生成されたモジュールの冗長性が高いことになる．

このモジュール構造図の比較結果を表 6−7 に示す．この表の生成率から，モジュール構造図作成支援機能により，人手で作成されるモジュールの約 70-80%を自動生成できることが分かる．したがって，モジュール構造図作成支援機能の有効性を実証できた．しかし，適合率は必ずしも高くない．したがって，提案したモジュール構造図作成支援機能には以下のような限界があり今後改良する必要がある．

(1) データフロー図では，システム要求を記述するため，初期化，後処理，例外処理などの詳細設計で必要となる処理がプロセスとして記述されていない．このため，このような

詳細な処理に対するモジュールをデータフロー図から自動生成できていない．したがって，これらの処理に対するモジュールを人手で追加する必要がある．

(2) 提案したモジュール構造図作成支援機能では，構造化設計手法に従って，システムの外部入出力やファイル入出力に対する単純な入出力モジュールを自動生成している．しかし，実際にはこのような単純な入出力処理をモジュールとして記述する場合は必ずしも多くない．これらの入出力処理は標準ライブラリを利用するのが普通である．したがって，このような単純な入出力処理をモジュールとして抽出するのは冗長であり最適化する必要がある．

表 6−7　モジュール構造図作成支援機能の適用結果

システム	メッセージ監視システム	口座管理システム	会議管理システム
手作業時のモジュール数 A	7	14	109
自動生成されたモジュール数 B	17	27	84
手作業モジュールを自動生成できた数 C	6	11	74
手作業モジュールの生成率 C/A	86%	79%	68%
自動生成モジュールの適合率 C/B	35%	41%	88%

6.7　3層システム設計法の有効性の評価

以下では，4.3で提案したシナリオフロー図に基づく3層システム設計法による層間インタフェースの共通化効果について，実際の3層システムの設計結果に基づいて評価した結果について述べる[16][17]．適用対象は次の3システムである．

（AP1）酒類販売管理システム

酒類販売管理システムは，酒類販売における酒類卸し売り販売事業者からの酒類の仕入れ，配送管理を支援するためのシステムである．

（AP2）ゴルフ場予約システム

ゴルフ場予約システムは，インターネット上で顧客がWWWを用いてゴルフ場のコースを検索したり，各コースの空き状況を検索して，コースを予約するシステムである．このシステムでは，顧客がインターネットからコースの検索･予約を行う検索系システムと，イントラネットで社内の担当者がコース情報を管理し予約情報を処理するための基幹系システムから構成される．

（AP3）通信サービス情報管理システム

通信サービス情報管理システムは，通信サービスの加入者情報および料金情報を管理するシステムである．

これらのシステムに対する3層システム設計の適用結果を表6−8に示す．

表6−8　3層システム設計の適用結果

	規模比	SFD数	シナリオ中の実体数	モジュール数			インタフェース数		
				表示層	機能層	データ層	共通化前	共通化後	共通化率
商品販売管理	1	55	3.7	103	40	53	142	99	70%
ゴルフ場予約	1.42	22	2.0	70	35	65	118	93	79%
通信サービス管理	194.8	495	6.12	495	283	891	2970	1980	67%

この表6－8から以下のことが分かる．

（1）各システムにおける表示層と機能層のモジュール数の比は，およそ2対1となっている．したがって，各機能層モジュールが平均すると2個の表示層モジュールから起動されることを示している．

（2）共通化によるインタフェース数の削減率は約21%～約33%であった．ここで100%から共通化率を引いた値をインタフェース削減率とする．したがって，シナリオフロー図を用いて層間インタフェースを明示的に設計することにより，類似するインタフェースの製造を抑止できるので，3層システム開発の効率を向上できる．

（3）1枚のシナリオフロー図に含まれる実体数と共通化率の関係を表6－8で調べると，シナリオ中の実体数が多くなる程，共通化率が小さいこと，したがってインタフェース削減率が大きくなっていることがわかる．この理由は，機能層モジュールが操作する実体数が多いほど，類似するF-Dインタフェースが増加するため，共通化の対象となるインタフェースが増えるためであると考えられる．

6.8 WebBASE によるシステム開発と従来の C/S システム開発との比較評価

WebBASE を用いた WWW とデータベース連携アプリケーション開発プロジェクトについて，(1) 開発言語とその開発規模 (2) 開発工数の分布 (3) クライアントサーバシステム開発との比較 (4) 開発上の問題点についてアンケート調査を実施した[18][19]．ここで，クライアントサーバシステム開発との比較では，開発期間，開発工数，開発規模，品質，仕様変更量，保守工数，教育工数の 7 項目について，イントラネット開発と比較した．また，アンケートの回答者は WebBASE を適用した実際のプロジェクトの開発責任者である．今回の調査対象プロジェクト数は 5 件である．また，開発対象システムの機能としては，電子掲示板，会議室管理，文書決裁，事例検索，設備予約，予約受付けなどである．したがって，情報共有型のイントラネットだけでなく業務システムやグループウェアなども含めた幅広い機能が WWW データベース連携機能で開発されるようになってきたことがわかる．以下では，調査結果に基づいてイントラネット開発の実態を見てみることにする．

（1）対象システム

各システムの開発規模と開発期間を以下に示す．

［S1］サービス受付システム：開発規模は約 16K ステップ，開発期間 8 ヶ月

［S2］ノウハウ共有システム：開発規模は約 8K ステップ，開発期間 2 ヶ月

［S3］電子会議システム：開発規模は約 50K ステップ，開発期間 2 ヶ月

［S4］事例検索システム：開発規模は約 21K ステップ，開発期間 2 ヶ月

(2) アンケート内容

アンケートでは，以下の質問に対して，従来のクライアントサーバシステム開発と比較して開発コスト，バグ数などの増減率で回答させた．

［質問 1］WebBASE で開発する場合と，クライアントサーバで開発する場合とを比較すると，開発期間はどれくらいでしたか？

［質問 2］WebBASE で開発する場合と，クライアントサーバで開発する場合とを比較すると，開発規模はどれくらいでしたか？

［質問 3］WebBASE で開発する場合と，クライアントサーバで開発する場合とを比較すると，問題発生件数はどれくらいでしたか？

［質問 4］WebBASE で開発する場合と，クライアントサーバで開発する場合とを比較すると，仕様変更件数はどれくらいでしたか？

［質問 5］WebBASE で開発する場合と，クライアントサーバで開発する場合とを比較すると，バグ発生件数はどれくらいでしたか？

［質問 6］WebBASE で開発する場合と，クライアントサーバで開発する場合とを比較すると，学習工数はどれくらいでしたか？

［質問 7］WebBASE で開発する場合と，クライアントサーバで開発する場合とを比較すると，修正工数はどれくらいでしたか？

［質問 8］WebBASE で開発する場合と，クライアントサーバで開発する場合とを比較すると，開発工数はどれくらいでしたか？

（3）アンケート結果

アンケート結果を図6－29に示す．

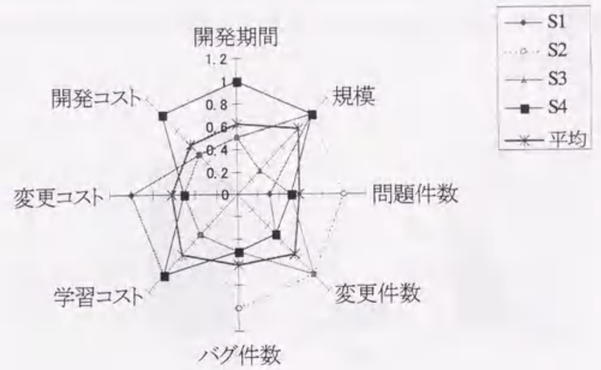


図6－29　アンケート結果

図6－29では，クライアント／サーバシステム開発とWebBASEによるイントラネット開発との比較結果の平均を示した．この結果，開発期間，開発規模，品質，保守工数，教育工数については，WebBASEによるイントラネットの場合，従来のクライアントサーバシステムに比べると，約40%程度削減できると考えられる．しかし，開発工数と仕様変更量については，差がない．この理由は，スクリプト言語によって開発規模は削減されるものの，仕様変更回数が増加するため最終的な規模は増加しないにも係わらず，開発工数は必ずしも削減されないためである．また，開発工数が小さくならないのに，開発期間が短縮されるのは，クライアントサーバシステム開発に比べると，WWW－データベース連携システム開発の方がWWWブラウザなどにより大きな単位でコンポーネント化が図られていること，スクリプト言語により開発が簡易化されていることなどから，比較的経験の浅い要員を同時に投入しやすく開発作業の並列化が大きくなるためではないかと推測される．しかし，この仮説の厳密な検証は今後の課題である．

（4）開発言語

まず，WWW－データベース連携システムの開発言語は，HTML, WebBASEスクリプト，Perl, shellスクリプトなどのスクリプトとC，Javaなどのプログラミング言語，そして，CP（WebBASEと通信するTPモニタVGUIDEのデータベースアプリケーション開発用4GL言語）であった．WWW－データベース連携システム開発における各言語の利用率の平均を図6－30に示す．この結果から，約65%がスクリプト言語で記述されていることが分かる．と同時に，約30%はCやJavaなどの3GLで記述されているという結果になった．また，図6－31には，プロジェクトごとの各言語の利用率を示した．これを見ると，言語の利用率はプロジェクトごとにかなり変化があることがわかる．たとえば，P3では，約85%がスクリプト言語で開発されているのに対して，P4では，スクリプト言語と3GLの割合がほぼ1対1になっている．

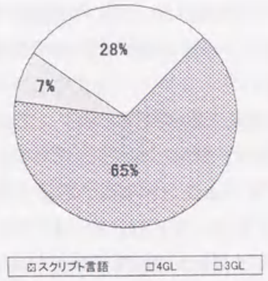


図6－30　言語の利用率の平均

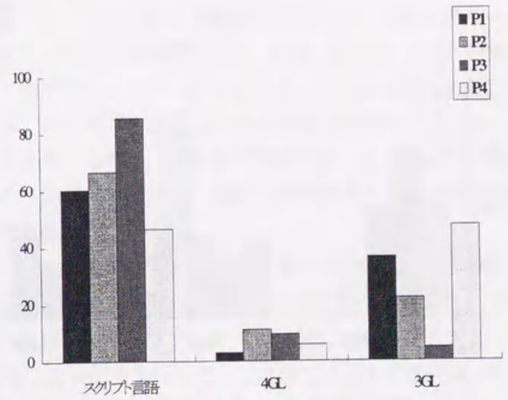


図6－31　プロジェクトごとの言語利用率

(5) 開発規模と開発期間

調査対象としたプロジェクトの開発規模は，約16Kステップから56Kステップだった．また，各プロジェクトの開発期間は2ヶ月から8ヶ月であった．しかし，開発規模と開発期間には相関関係はない．たとえば，2ヶ月のプロジェクトで約20Kステップを開発しているのに対して，8ヶ月のプロジェクトで約16Kステップしか開発していない．2ヶ月のプロジェクトの開発期間が短縮できたのはスクリプト言語の利用率が約90%と極めて高いためである．

(6) 開発工数の分布

図6－32にプロジェクトごとの開発工数の分布を示す．プロジェクトP1，P2，P3では，分析設計工程の比率が40%以上である．これに対して，プロジェクトP4，P5では，分析設計工程の比率が40%以下であった．プロジェクトP4では，製造工程の比率が高く，P5では，試験工程の比率が高くなっている．プロジェクトP1，P2，P3では，開発が計画通り完了したのに対して，プロジェクトP4，P5では開発が遅延した．この結果は，分析設計段階で十分な検討が行われて仕様を確定することがプロジェクトの成功にとって重要な鍵となることを示している．

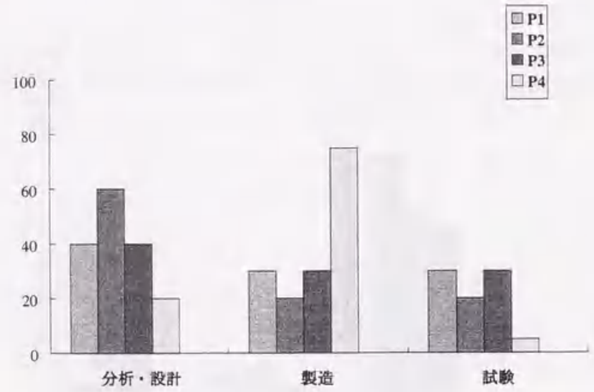


図6－32 プロジェクトごとの開発工数の分布

6.2 では 3.3 節で提案したデータフロー図の復元アルゴリズムを患者監視システムのモジュール構造図に適用して復元されたデータフロー図と，同じモジュール構造図から人手で復元したデータフロー図と比較して評価した．この実験評価では，被験者を経験者と未経験者に分けて自動生成された結果と比較した．この結果，初心者が作成したデータフロー図には設計レベルの詳細なデータフローや誤りのあるデータフローが含まれているものがあったのに対して，自動的に復元されたデータフロー図にはこのようなデータフローが含まれていないことから復元アルゴリズムの信頼性の高いことが実証された．また，モジュール構造図からのデータフロー図の復元工数を，経験者の場合には約 82.6%，初心者の場合には約 70%削減できていることから復元アルゴリズムにより生産性を向上できることを実証した．

6.3 では，POOM と OMT との比較実験の枠組みと 6.4 と 6.5 で実施した実験計画ならびに従来のオブジェクト指向分析実験との違いなどを示した．6.4 の実験では，個人による分析作業に関して OMT と POOM とを比較した．6.5 の実験では，複数の分析者が組になってオブジェクトモデルを作成する共同分析作業について実験した結果に基づいて比較を実施した．これらの結果から POOM の方が OMT よりも，オブジェクトモデルを効率よく作成できること，作成されたオブジェクトモデルの均質性が高いこと，階層型アーキテクチャに対応し易いことを明らかにした．このように，利用構造主導型の POOM が情報構造主導型の OMT に較べて優れた評価結果になった理由としては，利用構造主導型では，業務シナリオである役割フローからオブジェクトモデルを導出することと，作成されるべきオブジェクトモデル自体を類型化していることが挙げられる．また，POOM では，役割分析モデルとパターン化オブジェクトモデルによりオブジェクトモデルを利用構造に基づいて分析するので，今回の比較結果は，RDD や OOSE などの利用構造主導型のオブジェクト指向分析手法の有効性を示していると考えられる．本論文の実験では，オブジェクト指向分析に限定した実験により，POOM と OMT を比較した．設計や製造までも含めた総合的な比較実験については，今後の課題である．

6.6 では 4.3 で提案したシナリオフロー図に基づく 3 層システム設計法の効果を，実際にシナリオフロー図を適用したシステム開発の設計結果に基づいて評価した．層間インタフェースの共通化効果を調べた結果から，シナリオフロー図によるインタフェース数の削減率は約 21%～約 33%であることを明らかにした．したがってシナリオフロー図を用いることにより類似する層間インタフェースに対するモジュール開発を抑止できるので，3 層システムの開発効率を向上できることを実証した．

6.7 では 4.4 で提案したアルゴリズムにより生成されるモジュール構造図と，同じデータフロー図から手作業により作成されるモジュール構造図と比較して評価した．この結果，人手で作成されるモジュールの約 70-80%を自動生成できることから，モジュール構造図作

成支援機能の有効性を実証した．

6.8 では、WebBASE を用いたイントラネット・アプリケーション開発プロジェクトに関する調査とその分析結果についてのべた．この結果，イントラネット・アプリケーションの約 65%がスクリプト言語で作成されていることを明らかにした．また，この結果として，イントラネット・アプリケーション開発は，従来のクライアント／サーバシステム開発に比べ，開発期間，開発規模，品質などは約 40%程度削減できている．しかし，開発工数および仕様変更については，従来のクライアント／サーバシステム開発と同じだった．さらに，イントラネット・アプリケーション開発では，最新技術の教育，デファクト標準の選択，仕様の絞り込み，仕様の確定，仕様の変更管理，規模の見積もり，複数言語の選択が問題であることが明らかになった．また，今回の調査で対象としたプロジェクトは 1996 年に開発が開始されたプロジェクトである．したがって，開発担当者にとってもイントラネット開発ははじめてのケースが多いと思われる．今後，開発回数を経験してイントラネットの構築技術に習熟してくれば，ここで挙げた課題のいくつかについては解決されるだろう．また，イントラネット開発の生産性や品質などはさらに向上する可能性がある．しかし，そのためには，今後も課題解決に向けた開発データの蓄積と開発技術の確立が望まれる．

6 章で実施した実験評価の特徴を表 6−9 にまとめる．

表 6−9　実証評価の特徴

実証評価	評価法	評価対象作業	評価対象システム	主な評価尺度
データフロー図復元実験	自動生成の結果を分析者の結果と比較	個人作業	実験システム	作成時間 内容比較
モジュール構造図作成実験	自動生成の結果を分析者の結果と比較	個人作業	実験システム	自動生成率 適合率
オブジェクトモデル作成実験	異なる分析手法の適用結果を統計的に比較	個人作業 チーム作業	実験システム	分析モデルの均質性 分析時間，内容比較
シナリオフロー図フィールド調査	提案手法で開発されたプロジェクト生産物を調査	チーム作業	実システム	内容比較 共通化率
WWW情報システムフィールド調査	システム開発の担当者へのオピニオン調査	チーム作業	実システム	開発コスト，開発規模 問題件数

7.1 本研究のまとめ

本研究では，3 層アーキテクチャに基づく情報システムを実現するための系統的な開発方式に関する要素技術として，構造化システム分析技術，オブジェクト指向分析技術，分散システム設計技術，分散システム実行環境技術について，それぞれ，データフロー図ならびにモジュール構造図の自動生成方式の提案，役割分析に基づくオブジェクト指向分析方式の提案，3 階層クライアントサーバシステム設計方式の提案，スクリプトに基づく WWW ーデータベース連携方式の提案と，各技術についての実証的な評価を行った．本論文で得られた主な結果を以下に列挙する．

(1) データフロー図の自動生成ならびに復元方式（第 3 章）

3.2 では，データフローの入出力関係を表現する行列を論理ベクトルの集合と見なすことにより，論理ベクトルの大小関係に基づいてデータフロー図を一意的に生成する決定性アルゴリズムを考案した．このアルゴリズムの特徴は，以下の 3 点である．

[特徴 1] 入出力データの関係を満足する最小のデータフロー図を一意的に自動生成できる．

[特徴 2] 従来手法に較べ，効率的である．

[特徴 3] 任意の入出力データの関係に適用できる．

したがって，本方式は，任意の入出力データの関係に対してデータフロー図を効率よく自動生成できるので，実用性が高いことを示した．

3.3 では，モジュール構造図から入出力関係に基づいてデータフロー図を復元するアルゴリズムを提案した．大規模ソフトウェア開発ではソフトウェアの分析設計手法の経験者数が限定される．ところが，初心者の場合，誤りや冗長なデータフロー図を作成し易いという問題がある．提案したアルゴリズムを用いることにより，モジュール構造図から最適なデータフロー図を生成できるので，初心者でも標準的なデータフロー図を作成でき，ソフトウェアの保守工数を削減できる．また，提案したアルゴリズムの適用実験を行い，人手によるデータフロー図の復元作業に比べ，約 70%から 80%の作業工数を削減できることを明らかにした．

(2) 役割関係とオブジェクト分類に基づくオブジェクト分析方式（第 3 章）

3.4 では，情報システムのオブジェクト指向分析を容易化するパターン分類に基づくオブジェクト指向分析手法 POOM を提案した．POOM には，役割関係モデルとパターン化オブジェクトモデルの 2 つのオブジェクトモデルがある．役割関係モデルは，特定の分野に限定されてはいない．パターン化オブジェクトモデルでは，OMT の教科書[3]の ATM の例題に基づいて一般化した情報システム分野を対象とした 7 種類のオブジェクトパターンを提案している．役割関係モデルからこの 7 種類のオブジェクト間の関係を規則的に導出できることを明らかにした．POOM 手法では分析者によるオブジェクトモデルの変換手順を示し

ているため自動化の可能性を示唆しているが，変換アルゴリズムの考案までにはいたっていない．また POOM 手法では 7 種類のオブジェクトを表示層，機能層，データ層に分類しているため，3 層アーキテクチャに基づくシステム設計への連続性が高いという特徴がある．

(3) 3 階層クライアントサーバシステム設計方式（第 4 章）

4.2 では，Bichler らの SHDT 図式を拡張してページフロー図と呼ぶ WWW 情報システムのための情報ページ遷移図式を提案した．4.3 では，3 層アーキテクチャモデルに適した分散型情報システムを設計するための 3 層クライアントサーバシステム設計法を提案した．この設計法では，シナリオフロー図と呼ぶ 3 層インタフェースの設計図式を考案し，各層モジュールの独立性の概念に基づいて，3 層クライアントサーバシステムにおける層間インタフェースの設計条件を明らかにした．

4.4 ではデータフロー図からのモジュール構造図生成方式を提案した．モジュール構造図生成方式の効果をまとめると以下のようになる．

［効果 1］データフロー図に基づいて，詳細設計に必要なモジュール構造図の主要部分（約 70%）を自動生成できる．

［効果 2］複雑なデータフロー関係を持つデータフロー図に対しても適用できる．

［効果 3］データやモジュール名などの設計情報をデータフロー図からモジュール構造図に誤りなく引き継ぐことができ，設計情報の再利用性を向上できる．

［効果 4］詳細設計段階では，データフロー図に現れていない初期化や例外処理などのモジュールの追加判断と，外部入出力モジュールやファイル入出力モジュールの必要性の判断を行うだけでよく，設計作業を効率化できる．

4.5 では，モジュール構造図に基づいて分散型情報システムの 3 層アーキテクチャを設計する手法を述べ，3 層アーキテクチャモデルに適した分散型情報システムを設計するための 3 層クライアントサーバシステム設計法を提案した．これにより，構造化手法に基づいて作成された既存システムから 3 層クライアントサーバシステムへの移行方式を具体化した．この手法では，本論文で提案したアルゴリズムによりモジュール構造図からデータフロー図の復元過程とデータフロー図からモジュール構造図の生成過程までを自動化できる．モジュール構造図からシナリオフロー図への変換については，手順を具体化するだけにとどまっており変換アルゴリズムの考案までにはいたっていない．

4.6 では，3.4 で提案したオブジェクトモデル図に基づいて分散型情報システムの 3 層アーキテクチャを設計する手法を述べ，3 層アーキテクチャモデルに適した分散型情報システムを設計するための 3 層クライアントサーバシステム設計法を提案した．これにより 3 層アーキテクチャを用いた情報システム開発における分析工程と設計工程における生産物間の追跡性問題を解消するためのひとつの手法を具体化した．この設計手法では分析モデルからシナリオフロー図への変換手順を示しているため自動化の可能性を示唆しているが，変換アルゴリズムの考案までにはいたっていない．

(4) スクリプトに基づく WWW－データベース連携方式（第 5 章）

第 5 章では，イントラネットアプリケーション開発支援システム WebBASE について述べ，実験評価ならびに適用事例に基づき評価結果を示した．WebBASE の適用効果をまとめると以下のようになる．

［効果 1］多様なイントラネットアプリケーション開発に適用し，有効性を確認した．

［効果 2］CGI 方式に比較して約 6 倍の応答性能である．

［効果 3］開発規模を約 40%削減できる．

WebBASE は NTT ソフトウェアから 1996 年に販売を開始して以来，1999 年 12 までの類型販売数が約 740 本に達している．これからも分かるように WWW 情報システム構築環境として市場からも実用性を高く評価された．

(5) 開発方式の実証的な評価方式（第 6 章）

6.2 では 3.3 で提案したデータフロー図の復元アルゴリズムを患者監視システムのモジュール構造図に適用して復元されたデータフロー図と，同じモジュール構造図から人手で復元したデータフロー図と比較して評価した．この実験評価では，被験者を経験者と未経験者に分けて自動生成された結果と比較した．この結果，初心者が作成したデータフロー図には設計レベルの詳細なデータフローや誤りのあるデータフローが含まれているものがあったのに対して，自動的に復元されたデータフロー図にはこのようなデータフローが含まれていないことから復元アルゴリズムの信頼性の高いことが実証された．また，モジュール構造図からのデータフロー図の復元工数を，経験者の場合には約 82.6%，初心者の場合には約 70%削減できていることから復元アルゴリズムにより生産性を向上できることを実証した．

6.3 では，POOM と OMT との比較実験の枠組みと 6.4 と 6.5 で実施した実験計画ならびに従来のオブジェクト指向分析実験との違いなどを示した．6.4 の実験では，個人による分析作業に関して OMT と POOM とを比較した．6.5 の実験では，複数の分析者が組になってオブジェクトモデルを作成する共同分析作業について実験した結果に基づいて比較を実施した．これらの結果から POOM の方が OMT よりも，オブジェクトモデルを効率よく作成できること，作成されたオブジェクトモデルの均質性が高いこと，階層型アーキテクチャに対応し易いことを明らかにした．このように，利用構造主導型の POOM が情報構造主導型の OMT に較べて優れた評価結果になった理由としては，利用構造主導型では，業務シナリオである役割フローからオブジェクトモデルを導出することと，作成されるべきオブジェクトモデル自体を類型化していることが挙げられる．また，POOM では，役割分析モデルとパターン化オブジェクトモデルによりオブジェクトモデルを利用構造に基づいて分析するので，この結果は，RDD や OOSE などの利用構造主導型のオブジェクト指向分析手法の有効性を示していると考えられる．

6.6 では 4.3 で提案したシナリオフロー図に基づく 3 層システム設計法の効果を，実際にシナリオフロー図を適用したシステム開発の設計結果に基づいて評価した．層間インタフェースの共通化効果を調べた結果から，シナリオフロー図によるインタフェース数の削減率は約 21%～約 33%であることを明らかにした．したがってシナリオフロー図を用いることにより類似する層間インタフェースに対するモジュール開発を抑止できるので，3 層システムの開発効率を向上できることを実証した．

6.7 では 4.4 で提案したアルゴリズムにより生成されるモジュール構造図と，同じデータフロー図から手作業により作成されるモジュール構造図と比較して評価した．この結果，人手で作成されるモジュールの約 70-80%を自動生成できることから，モジュール構造図作成支援機能の有効性を実証した．

6.8 では，WebBASE を用いた WWW 情報システム開発プロジェクトに関する調査とその分析結果についてのべた．この結果，WWW 情報システムの約 65%がスクリプト言語で作成されていることを明らかにした．また，この結果として，WWW 情報システム開発は，従来のクライアント／サーバ・システム開発に比べ，開発期間，開発規模，品質などは約 40%程度削減できている．しかし，開発工数および仕様変更については，従来のクライアント／サーバ・システム開発と同じだった．さらに，WWW 情報システム開発では，最新技術の教育，デファクト標準の選択，仕様の絞り込み，仕様の確定，仕様の変更管理，規模の見積もり，複数言語の選択が問題であることが明らかになった．

以上が，本研究の要約である．本研究で提案した開発方式の自動化範囲をまとめて表 7－1 に示す．データフロー図とモジュール構造図については自動化方式を具体化した．オブジェクトモデル図，シナリオフロー図については具体的な変換手順を示すことにより自動化の見通しを得た．

また，WWW 情報システム構築環境については WebBASE として実用化することにより，NTT 内外をあわせて約 740 以上の WWW 情報システムに導入され，開発効率の向上に大きく貢献することができた．

本研究では，以上の結果により，3 層アーキテクチャに基づく WWW 情報システムを開発するための要求分析技術，設計技術，構築環境技術，開発方式の実証評価技術について，ソフトウェア工学分野における学術的な寄与並びに工学的な寄与ができたと考えられる．

表 7－1　本論文の開発方式の自動化範囲

作成方式	作成手順	記述規則	自動化
データフロー図の生成	アルゴリズム	最小化規則	自動化
データフロー図の復元	アルゴリズム	変換規則	自動化
モジュール構造図の生成	アルゴリズム	変換規則	自動化
オブジェクトモデルの作成	段階的な分析手順	変換規則	未
ページフロー図の作成	階層的な設計手順	未	未
シナリオフロー図の作成	層間インタフェース設計手順	設計指針	未
	構造化手法に基づく作成手順	変換規則	未
	オブジェクト分析に基づく作成手順	変換規則	未

7.2 今後の課題

モジュール構造図からのデータフロー図の復元方式の課題としては，データフロー図やモジュール構造図と入出力関係との一貫性の管理や，修正時の影響波及分析などがある．また，エンティティリレーションシップ図など他の要求分析図式と統合した要求分析のリバースエンジニアリング手法の検討も今後の課題である．さらに，3 層クライアント／サーバ型システムや 3 層イントラネットの開発が進展して資産として蓄積されるに従い，これらの分散型情報システムのリエンジニアリングに際して，3 層アーキテクチャで設計されたモジュールから，シナリオフロー図や，データフロー図を復元する方式が必要になると思われる．

本論文で提案した 7 種類のオブジェクト分類に属さないような分野へのパターン化オブジェクトモデルの適用性については未検証であり，今後の比較評価が必要である．しかし，本論文のオブジェクト分類では限界があると思われる場合でも適切なオブジェクト分類を導入することによりこの問題は解決できると思われる．設計や製造までも含めた総合的な比較実験や OMT 以外のオブジェクト指向分析手法と POOM との比較については，今後の課題である．また，役割関係に基づいてオブジェクトモデルを自動生成し，さらにシナリオフロー図まで自動生成する研究を進める必要がある．

3 層アーキテクチャに基づく設計技術の課題としては，ページフロー図やシナリオフロー図の等価性検証方式や最適化方式の確立や本手法に基づく CASE 環境の構築があげられる．

またモジュール構造図の自動生成法の課題としては，分散型情報システムにおける 3 層アーキテクチャを考慮したモジュール構造を各層ごとに自動生成するアルゴリズムを具体化する必要がある．

WWW－データベース連携技術の課題としては，WebBASE を用いた WWW 情報システム開発で必要となる共通性の高い処理のコンポーネント化や，ページフロー図やシナリオフロー図に基づく WebBASE アプリケーションの自動生成などがある．

分散型情報システムの分析・設計技術は，イントラネットやエクストラネットの拡大に伴い今後も発展していくと思われる．たとえば最近では XML を用いた企業間の情報連携サービスが急速に進展しつつあり，XML を前提とした情報システムの開発方式を早急に確立する必要がある．また，ERP，SCM，SFA など新しい情報システム向けのパッケージの企業への導入が進展している．今後は，個々の企業情報システムを開発するための技術だけではなく，これらのパッケージやグループウェアなどとの連携を含めて企業情報システム全体のアーキテクチャを踏まえた新たな分散型情報システムの開発方式の確立が望まれる．

さらにネットワーク上で多様なサービスが利用できるようになってくると，それらを選択し連携させるための WWW サービスが必要になる．たとえばネットワーク上に複数のデータベースやアプリケーションが存在する場合，サーバ上に配置されるアプリケーションには，業務アプリケーションを実行するものと，ネットワーク上のデータベースや業務アプリケーションの位置を検索するためのサービス・ロケータの 2 種が必要となる．このような WWW サービスの必要性は，従来の表示層，機能層，データ層からなる 3 層アーキテクチャにサービス連携層が追加された新たなアーキテクチャが必要になることを示している．サービス連携層は表示層と機能層の間に位置して，機能やデータの位置を提示したり，機能間の連携を必要とするトランザクションを制御したり，中継データを管理することになる．今後もこのような新しいアーキテクチャに応じたソフトウェア開発技術の研究として，サービス連携を前提とするシステム要求分析技術やその構築環境としてのスクリプト言語や連携トランザクション制御用ミドルウェアが必要になると思われる．

本研究は，3 層アーキテクチャに基づく WWW 情報システムの開発方式に関する研究であり，提案した様々な手法により開発効率を向上できることを実証した．しかし，一層の開発効率の向上を達成するためには，今後も上述したような課題の解決に向けた基礎的な研究と同時に実システム開発における開発データの蓄積と実証的な評価技術を確立していく必要がある．

謝辞

本研究をまとめるにあたり，種々の御指導，御鞭撻を賜りました，名古屋大学大学院工学研究科情報工学専攻教授 坂部俊樹博士に心から感謝の意を表します．また，本論文について，貴重な御討論，御助言をいただいた名古屋大学工学研究科長　稲垣康善博士，同教授　阿草清滋博士に深く感謝いたします．

本論文は，筆者が日本電信電話株式会社(NTT)ソフトウェア研究所および NTT マルチメディアシステム総合研究所並びに NTT 情報流通プラットフォーム研究所において行った研究実用化の成果をまとめたものであり，この間多くの方からご指導，御助言を賜った．心より御礼申し上げます．NTT ソフトウェア株式会社代表取締役社長　鶴保征城博士(元 NTT ソフトウェア研究所長)，同社　細谷僚一常務取締役（元 NTT ソフトウェア研究所長）同社　森道直取締役，同社　拝原正人取締役（元 NTT 情報システム本部情報処理技術部長)，NTT コミュニケーションウェア株式会社 長野宏宣取締役(元 NTT ソフトウェア研究所ソフトウェア開発技術研究部長)，同社　伊藤路夫運用管理部長（元 NTT 情報システム本部情報処理技術部担当部長）豊橋技術科学大学　磯田定宏教授（元 NTT ソフトウェア研究所基礎技術研究部長)，NTT ソフトウェア株式会社　名古屋支店　梅本栄治支店長　（元 NTT ソフトウェア研究所企画部長)，同社　吉田清 技術開発部長，NTT サイバーソリューション研究所 安原隆一 所長（元 NTT ソフトウェア研究所ソフトウェア技術研究部長)，NTT サービスインテグレーション基盤研究所　国立勉主幹研究員（元 NTT ソフトウェア研究所グループリーダ)，NTT コミュニケーションズ　NTT アメリカ　伊藤正樹担当部長（元 NTT ソフトウェア研究所グループリーダ)，NTT ソフトウェア株式会社　神谷芳樹担当部長には，研究の様々な段階で適切なご助言と種々の御指導を頂いた．それぞれの方々に深謝いたします．さらに，NTT 取締役 NTT サイバーコミュニケーション総合研究所　加藤邦紘所長(元 NTT マルチメディアシステム総合研究所長)，NTT 情報流通プラットフォーム研究所　伊土誠一 所長（元 NTT ソフトウェア研究所長)，NTT システムインテグレーション研究所　斎藤孝文プロジェクトリーダ（元 NTT 情報流通プラットフォーム研究所プロジェクトリーダ)，NTT 情報流通プラットフォーム研究所　畠中優行プロジェクトリーダには研究の機会を与えて頂くとともに日頃から御指導を頂いた．心から御礼申し上げます．第 3 章のデータフロー図自動生成方式の研究では NTT 情報流通プラットフォーム研究所鈴木英明主任研究員，太田賢治研究主任に，第 3 章および第 4 章のソフトウェア開発支援環境の研究実用化では NTT　ME 黒木宏明担当部長（元 NTT ソフトウェア研究所主任研究員)，NTT ソフトウェア斎直人担当課長(元 NTT ソフトウェア研究所主任研究員)，NTT 西日本　法人営業本部　ソリューションビジネス部　西永誠司担当課長（元 NTT ソフトウェア研究所研究主任）に，第 4 章のシナリオフロー図に基づく設計法の研究では，NTT 情報流通プラットフォーム研究所　忠海均主幹研究員，高田信一主任研究員，畑恵介研究主任に，第 5 章で述べた WebBASE の研究実用化では NTT 情報流通プラットフォーム研究所　川崎隆二主幹研究員，

元田敏浩主任研究員，黒川裕彦主任研究員，徳丸浩二主任研究員，第 6 章の開発方法論に関する実験評価法の開発では NTT コミュニケーションズ岡敦子担当課長（元 NTT ソフトウェア研究所研究主任）の諸氏に御協力を頂いた．ここに記して深く感謝いたします．

最後に，本研究を進める上で日頃から御協力，御議論して頂いた研究所の先輩，同僚諸氏に心から感謝いたします．

参考文献

第 1 章

[1] 長野宏宣，分散型情報システム開発方式の現状と課題，NTT R&D，Vol.45，No.8，pp.705-710, 1996.

[2] Faulk, S., “Software Requirements: A Tutorial,” in Software Engineering edited by Dorfman. M. and Thayer, R., IEEE Inc., 1997.

[3] 山本修一郎，忠海均，上野正巳，ドメイン指向要求分析技術：DREM，NTT R&D, Vol.45, No.8, pp.711-718, 1996.

[4] Garlan, D., “The Role of Software Architecture in Requirements Engineering,” IEEE, RE‘94, p.240, 1994.

[5] Mead, N., “The Role of Software Architecture in Requirements Engineering,” IEEE, RE‘94, p. 242, 1994.

[6] Schekaran, C., “The Role of Software Architecture in Requirements Engineering,” IEEE, RE‘94, pp. 245, 1994.

[7] 忠海均，高田信一，山本修一郎，リエンジニアリング技術：RELICS，NTT R&D, Vol.45, No.8, pp.739-744, 1996.

[8] Horowitz, E., “Migrating Software to the World Wide Web,” IEEE Software, Vol.15, No.3, pp.18- 21, 1998.

[9] Andleigh, P. and Gretzinger, M., “Distributed Object Oriented Data- Systems Design,” Prentice-Hall International, 1992.

[10] Donovan, J. J.,“ Business Re-engineering with Information Technology,” Prentice Hall, 1994.

[11] 青山幹雄，中所武司，向山博，コンポーネントウェア，共立出版，1998.

[12] Dennis, A., “Lessons form Three Years of Web Development,” Commun. ACM, Vol. 41, No.7, pp.112-113, 1998.

[13] Bieber, M. and Vitali, F., “Toward Dupport for Hypermedia on the World Wide Web,” IEEE Computer, Vol. 30, No.1, pp.62- 70, 1997.

[14] Balasubramanian, P., and Bashian, A., “Document Management and Web Technologies: Alice Marries the Mad Hatter,” Commun. ACM, Vol.41, No.7, pp,107-115, 1998.

[15] Isakowitz,T., Stohr, E. and Balasubramanian, P., “RMM: A methodology for structuring hypermedia design,” Commun. ACM, Vol.38, No.8, pp34-44, 1995.

[16] Schwabe, D., Rossi, G. and Barbosa, S.D.J.,“Systematic hypermedia application design with OOHDM,” in proc. for Hypertext‘96, 1996.

[17] 三喜英行，土屋正人，アルバート・リー，3 層クライアント／サーバ・システム構築技法，ソフト・リサーチ・センター，1996.

[18] 山本修一郎，川崎隆二，分散型情報システム構築基盤 VGUIDE の導入状況，NTT 技術

ジャーナル, Vol.8, No.8, pp.45-48, 1996.
[19] Adler,M., " An Algebra for Data Flow Diagram, Process Decomposition, " IEEE Trans. on Software Eng., Vol.14,No.2, pp.169 –183, 1988.
[20] 松本一教,本位田真一, 段階的ソフトウェアの生成と検証について, 人工知能学会研究会, SIG-FAI-9002, 1990.
[21] 鈴木英明, 高橋直久, プロセス分解代数に基づくデータフロー図の段階的詳細化について, 情報処理学会研究会報告, 91-SE-82, pp.71-78, 1991.
[22] Modell, M., "A Professional's Guide to Systems Analysis," McGraw-Hill, 1988.
[23] Stevens, W., Myers, G. and Constantine, L. " Structured design, " IBM Systems Journal, vol.13, no. 2, pp.115- 139, 1974.
[24] Myers, G., " Composite Structured Design, " Van Nostrand Rheinhold, 1978.
[25] Yourdon, E. and Constantine, L. " Structured Design, " Prentice-Hall,1979.
[26] Page-Jones,M., " The Practical Guide to Structured System Design, " Prentice-Hall/Yourdon Press, 1988.
[27] Tsai,J. and Ridge,J., " Intelligent Support for Specifications Transformation, " IEEE Software, Vol.5, No. 11, pp.28-35, 1988.
[28] Sharp, H., "KDA- A Tool for Automatic Design Evaluation and Refinement Using the Blackboard Model of Control, " 10-th ICSE, pp.407-416, 1988.
[29] Yamamoto, S. and Isoda, S., " A Transformational Algorithm for Generating Natural Structure Charts from Dataflow Diagram, " JCSE'92, pp.190-197, 1992.
[30] Benedusi, B., Cimitile, A., and De Carlini, U., " A reverse engineering methodology to reconstruct hierarchical data flow diagrams for software maintenance," Conf. On Software Maintenance, pp.180-189, 1989.
[31] Byrne, E., " Software Reverse Engineering: A CASE Study," SOFTWARE- PRACTICE AND EXPERIENCE, vol. 21, no. 12, pp. 1349-1364, 1991.
[32] O'Hare, A.B. and.Troan, E.W., "RE- Analyzer: From source code to structured analysis", IBM SYSTEMS JOURNAL, vol. 33, no. 1, pp. 110- 130, 1994.
[33] Yamamoto, S., "A Method for Generating Dataflow Diagram based on the Relationship between System Input and Output ," JCKBSE'94, pp. 14- 19, 1994.
[34] Yamamoto, S., "Reconstructing Dataflow Diagrams from Structure Charts based on Input and Output Relationship," IEICE trans. On Information and Systems, Vol.E78-D, No.9, pp.1118- 1126, 1995.
[35] Rumbaugh, J., et al., "Object- Oriented Modeling and Design, Prentice Hall," Englewood Cliffs, NJ, 1991.
[36] Cameron, J., "New ingredients for an object-oriented method, Object Magazine," Sep.-Oct., pp. 61- 67, 1992.

[37] Jacobson, I., "Object- Oriented Software Engineering, A Use CASE Driven Approach," ACM Press, Addison-Wesley, 1992.
[38] Rubin, K., and Goldberg, A., " Object Behavior Analysis," CACM, Vol.35, No.9, pp.48-62, 1992.
[39] Wegscheider, E., " Toward Code- Free Business Application Development," IEEE Computer, Vol. 30, No.3, pp.35- 43, 1997.
[40] Ning, J.Q., "A Component- Based Software Development Model," Compsac'96, pp.389-394, 1996.
[41] Philip, G.C., "Software design guidelines for event-driven programming", Journal of Sys. and Soft., Vol. 41, pp.79- 91, 1998.
[42] Hadjiefthymiades, S.P. and Martakos, D.I., "A Generic Framework for the Deployment of Structured Databases on the World Wide Web," Proc. Fifth Int'l World Wide Web Conf., May 1996, http://WWW5conf.inria.fr/.
[43] Genesereth, M.R., Keller, A.M., and Duschka, O.M., "Infomaster: An Information Integration System," SIGMOD'97, pp.539- 542, 1997.
[44] Wi Sae-Tung, Ohmori, T., and Hoshi, M., "An Information Integration Architecture for Mobile Users in WWW Environment," Trans. of Information Processing Society of Japan, vol.39, No.4, pp.888- 900, 1998.
[45] Miller, R.J., and Tsatalos, O.G., and Williams, J.H., "DataWeb: Customizable Database Publishing for the Web," IEEE Multimedia, vol.4. No.4, pp.14-21, 1997.
[46] Nguyen,T. and Srinivasan, "Accessing Relational Databases frOMThe World Wide Web," Proc. ACM SIGMOD Int'l Conf. On Management of Data, ACM Press, New York, pp.529-540, June 1996.
[47] WebServer, http://WWW.oracle.com/
[48] Live Wire, http://home.netscape.com
[49] IIS, http://WWW.microsoft.com
[50] WebSite, http://WWW.ora.com/
[51] Web.sql, http://WWW.sybase.com/
[52] Ousterhout, J., "Scripting: Higher- Level Programming for the 21st Century," IEEE Software, Vol. 31, No.3, pp. 23- 30, 1998.
[53] 畑田稔, 遠藤裕英, WWW-RDB 連携システムの開発, 情報処理学会論文誌, Vol.38, No.2, pp.349-358, 1997.
[54] 岡敦子, 山本修一郎, 磯田定宏, ソフトウェア開発実験に基づく構造化分析／設計手法の評価, 情報処理学会論文誌, Vol.34, No.12, pp.2543-2551, 1993.
[55] Jacobson, I., " A confused world of OOA and OOD," JOOP, Vol.8, No.9, pp.15-20, 1995.

[56] Wirfs-Brock, R., " Stereotyping: A technique for characterizing objects and their interactions, " Object Magazine, Vol.3, No. 4, pp.50-53, 1993.
[57] Wirfs-Brock, R., " How designs differ," ROAD, Vol.1, No.4, pp.51-56, 1994.
[58] Bichler, M.and Nusser, S.," SHDT– THE STRUCTURED WAY OF DEVELOPING WWW-SITES, " ECIS'96, pp.1093- 1101, 1996.

第 2 章

[1] Donovan, J. J., "Business Re-engineering with Information Technology," Prentice Hall, 1994.
[2] Wiederhold,G.,"Mediators in the Architecture of Future Information Systems," Computer, 25(3) :38- 49,1992
[3] Rumbaugh, J., et al., "Object- Oriented Modeling and Design, Prentice Hall," Englewood Cliffs, NJ, 1991.
[4] Jacobson, I., " A confused world of OOA and OOD," JOOP, Vol.8, No.9, pp.15-20, 1995.
[5] 山本修一郎，川崎隆二，分散型情報システム構築基盤 VGUIDE の導入状況，NTT 技術ジャーナル, Vol.8, No.8, pp.45-48, 1996.
[6] 山本修一郎，イントラネットを取り巻く状況と今後の展開，NTT 技術ジャーナル, Vol.8, No.10, pp.50-55, 1996.
[7] S.Yamamoto, H.Kurokawa, K.Tokumaru, S.Adachi, "WebBASE- An Intranet Application Development System," NTT REVIEW, Vol.8, No.4, pp.58-65, 1996.
[8] Yamamoto,S., and Tokumaru, K., "WebBASE: An intranet application development system," AMERICAN PROGRAMMER, vol.9, No.8, pp.27-35, 1996.
[9] 元田敏浩，徳丸浩二，WWW とデータベースサービスとの連携方式の検証，電子情報通信学会，知能ソフトウェア工学研究会 KBSE-7, pp.47-54 , 1995.
[10] Pressman, R., What a Tangled Web We Weave, IEEE Software, Vol.17, No.1, pp.18-21, 2000.
[11] 高田信一, 畑恵介, 山本修一郎, 3 層クライアント／サーバ型情報システムの設計手法, NTT 技術ジャーナル，Vol.8, No.1, pp.74- 77, 1996.
[12] 畑恵介，高田信一，山本修一郎，並行開発型分散情報システム開発技術 ParaDISE ，NTT R&D，Vol.45, No.8, pp.719-726, 1996.
[13] 畑恵介, 斎直人, 山本修一郎, ParaDISE: イントラネットアプリケーション設計方法論, NTT 技術ジャーナル, Vol.8, No.10, pp.64-67, 1996.
[14] Yamamoto, S., "Reconstructing Dataflow Diagrams from Structure Charts based on Input and Output Relationship," IEICE Trans. on Information and Systems, Vol.E78-D, No.9, pp.1118- 1126, 1995.
[15] 山本修一郎，黒木宏明，オブジェクトモデリング手法の実験評価，ソフトウェア科学会論文誌，コンピュータソフトウェア, Vol.16, No.4, pp.59-75, 1999.
[16] Yamamoto, S. and Isoda, S., "A Transformational Algorithm for Generating Natural Structure

Charts from Dataflow Diagram," JCSE'92, pp.190- 197, 1992.
[17] 元田敏浩，山本修一郎, WeBBASE, イントラネットアプリケーション構築環境， NTT 技術ジャーナル，Vol.8, No.10, pp.60-63, 1996.
[18] Yamamoto, S., Kawasaki,R., Motoda, T. and Tokumaru, K., "Internet/ Intranet Application Development System WebBASE and its Evaluation," IEICE Trans. on Information and Systems, Vol. E81-D, No.12, pp.1450-1457, 1998.
[19] 畑恵介，黒木宏明，山本修一郎，長岡満夫，ParaDISE 分析設計技法，電気通信協会，1998.
[20] Yourdon, E. and Constantine, L. " Structured Design, " Prentice-Hall,1979.
[21] Myers, G., " Composite Structured Design, " Van Nostrand Rheinhold, 1978.

第 3 章

[1] DeMarco,T., "Structured Analysis and System Specification, " Yourdon Press , 1978.
[2] Gane,C. and Sarson,T., "Structured Systems Analysis: tools and techniques, " Prentice- Hall ，1979.
[3] Yourdon,E., "Modern Structured Analysis, " Yourdon Press, 1989.
[4] Adler,M., " An Algebra for Data Flow Diagram, Process Decomposition, " IEEE Trans. on Software Eng., Vol.14,No.2, pp.169 –183, 1988.
[5] 山本修一郎,太田賢治,データフロー図の自動生成方式の提案,信学会論文誌Ｄ－Ｉ,Vol. J76-D-I, No.10, pp.504-513, 1993.
[6] Yamamoto, S. and Isoda, S., "SoftDA--- A Reuse Oriented Software Design System, " Compsac'86, pp.284- 290, 1986.
[7] 磯田定宏，山本修一郎，下村隆夫，黒木宏明：設計情報とコードの一体管理方式に基づくソフトウェア開発支援システム(SoftDA)"，NTT R&D, Vol.38, No.11, pp.1239-1248, 1989.
[8] Isoda,S., Shimomura,T., Yamamoto,S. and Kuroki,H., " Integrated CASE System:SoftDA, " NTT REVIEW, vol.2, no.2,(Mar., 1990) pp.52-61.
[9] 松本一教,本位田真一,段階的ソフトウェアの生成と検証について，人工知能学会研究会, SIG-FAI-9002, 1990.
[10] 鈴木英明，高橋直久,プロセス分解代数に基づくデータフロー図の段階的詳細化について，情報処理学会研究会報告, 91-SE-82, pp.71-78, 1991.
[11] Benedusi, B., Cimitile, A., and De Carlini, U., " A reverse engineering methodology to reconstruct hierarchical data flow diagrams for software maintenance," Conf. On Software Maintenance, pp.180-189, 1989.
[12] Byrne, E., " Software Reverse Engineering: A CASE Study," SOFTWARE- PRACTICE AND EXPERIENCE, vol. 21, no. 12, pp. 1349-1364, 1991.

[13] O'Hare, A.B. and.Troan, E.W., "RE- Analyzer: From source code to structured analysis", IBM SYSTEMS JOURNAL, vol. 33, no. 1, pp. 110- 130, 1994.
[14] Yamamoto, S., "A Method for Generating Dataflow Diagram based on the Relationship between System Input and Output ," JCKBSE'94, pp. 14- 19, 1994.
[15] Yamamoto, S., "Reconstructing Dataflow Diagrams from Structure Charts based on Input and Output Relationship," IEICE trans. On Information and Systems, Vol.E78-D, No.9, pp.1118- 1126, 1995.
[16] Monarchi, D., and Puhr, G., " A reserch typology for Object-Oriented Analysis and Design," CACM, Vol.35, No.9, pp.35-47, 1992.
[17] Jacobson, I., " A confused world of OOA and OOD," JOOP, Vol.8, No.9, pp.15-20, 1995.
[18] Rumbaugh, J., et al., "Object- Oriented Modeling and Design, Prentice Hall," Englewood Cliffs, NJ, 1991.
[19] Shlaer, S. and Mellor, S., " Object-Oriented Systems Analysis: Modeling the World in Data," Prentice Hall, 1988.
[20] Coleman, D. etal., " Object- Oriented Development: The Fusion Method," Prentice Hall, 1994.
[21] Rubin, K., and Goldberg, A., " Object Behavior Analysis," CACM, Vol.35, No.9, pp.48-62, 1992.
[22] Jacobson, I., "Object- Oriented Software Engineering, A Use CASE Driven Approach," ACM Press, Addison-Wesley, 1992.
[23] Wirfs-Brock, R., and Wiener,L., " Designing Object-Oriented Software," Prentice Hall, 1991.
[24] 山本修一郎，オブジェクトパターンに基づくオブジェクトモデリング手法の提案，電子情報通信学会，知能ソフトウェア工学研究会，KBSE 96-28, pp.17-22, 1997.
[25] 山本修一郎，黒木宏明，オブジェクトパターンに基づくオブジェクトモデリング手法の実験評価，電子情報通信学会，知能ソフトウェア工学研究会，KBSE 97-1, pp.1-6, 1997.
[26] 山本修一郎，黒木宏明，オブジェクトモデリング手法の実験評価，ソフトウェア科学会論文誌，，コンピュータソフトウェア, Vol.16, No.4, pp.59-75, 1999.
[27] Yamamoto, S., Kuroki, H., "Patterned versus conventional Object-Oriented Analysis Methods: A Group Project Experiment," IEICE Trans. on Information and Systems, Vol. E81-D, No.12, pp.1458-1465, 1998.
[28] 山本修一郎，黒木宏明，オブジェクトモデリング手法の実験評価，ソフトウェア科学会論文誌，コンピュータソフトウェア, Vol.16, No.4, pp.59-75, 1999.
[29] Yamamoto, S., "Comparative Study of Object-Oriented Analysis," ISFST'97, pp.37-43.
[30] Cameron, J., "New ingredients for an object-oriented method, Object Magazine," Sep.-Oct., pp. 61- 67, 1992.
[31] Schmidt, D., " Using Design Patterns to Develop Reusable Object- Oriented Communication

Software, " Commun. ACM., Vol.38, No.10, pp.65-74, 1995.
[32] Gamma, R., etal., " Design Patterns," Addison-Wesley, 1995.
[33] Modell, M., "A Professional's Guide to Systems Analysis," McGraw-Hill, 1988.

第 4 章

[1] Bichler, M.and Nusser, S.," SHDT – THE STRUCTURED WAY OF DEVELOPING WWW-SITES," ECIS'96, pp.1093- 1101, 1996.
[2] 畑恵介，斎直人，山本修一郎，ParaDISE: イントラネットアプリケーション設計方法論，NTT 技術ジャーナル, Vol.8, No.10, pp.64-67, 1996.
[3] 畑恵介，黒木宏明，山本修一郎，長岡満夫，ParaDISE 分析設計技法，電気通信協会，1998.
[4] 高田信一，畑恵介，山本修一郎，3 層クライアントサーバアーキテクチャに適応したシステム設計手法の提案，情報処理学会，情報システム研究会，56-3，pp.19-28，1995.
[5] 高田信一，畑恵介，山本修一郎, 3 層クライアント／サーバ型情報システムの設計手法，NTT 技術ジャーナル，Vol.8, No.1, pp.74- 77, 1996.
[6] 畑恵介，高田信一，山本修一郎，並行開発型分散情報システム開発技術 ParaDISE ，NTT R&D，Vol.45, No.8, pp.719-726, 1996.
[7] 畑恵介，高田信一，山本修一郎，3 層 C/S システム開発方法論の評価実験，情報処理学会，情報システム研究会，56-4，pp.29-36，1995.
[8]番井俊行，阿登弘和，畑恵介，インターネット系サービス用 OpS を開発，NTT 技術ジャーナル，Vol.9, No.10, pp.98-101, 1997.
[9] 石井孝監修，仲谷元編著，3 層クライアント／サーバ設計技法，共立出版,1998.
[10] Stevens, W., Myers, G. and Constantine, L. " Structured design, " IBM Systems Journal, vol.13, no. 2, pp.115- 139, 1974.
[11] Myers, G., " Composite Structured Design, " Van Nostrand Rheinhold, 1978.
[12] Yourdon, E. and Constantine, L. " Structured Design, " Prentice-Hall,1979.
[13] Page-Jones,M., " The Practical Guide to Structured System Design, " Prentice-Hall/Yourdon Press, 1988.
[14] Tsai,J. and Ridge,J., " Intelligent Support for Specifications Transformation, " IEEE Software, Vol.5, No. 11, pp.28-35, 1988.
[15] Sharp, H., "KDA- A Tool for Automatic Design Evaluation and Refinement Using the Blackboard Model of Control, " 10-th ICSE, pp.407-416, 1988.
[17] Yamamoto, S. and Isoda, S., " A Transformational Algorithm for Generating Natural Structure Charts from Dataflow Diagram, " JCSE'92, pp.190-197, 1992.

第 5 章
[1] 元田敏浩，徳丸浩二，WWW とデータベースサービスとの連携方式の検証，信学技報 KBSE-7, pp.47-54 , 1995.
[2] Motoda, T., Tokumaru, K., Kawasaki, Yamamoto, S., “Experimental Performance Analysis for Relational Database Applications using WWW,” ISFST-96, pp.159- 164, 1996.
[3] Tokumaru, K., Motoda, T., and Kurokawa, H., “Comparative Study of WWW Interfaces to Relational Databases,” NTT REVIEW, Vol.8, No.3, pp.74-79, 1996.
[4] S.Yamamoto, H.Kurokawa, K.Tokumaru, S.Adachi, “WebBASE- An Intranet Application Development System,” NTT REVIEW, Vol.8, No.4, pp.58-65, 1996.
[5] WWW-データベース連携システム構築法，日経 BP 社，1996.
[6] Yamamoto,S., and Tokumaru, K., “WebBASE: An intranet application development system,” AMERICAN PROGRAMMER, vol.9, No.8, pp.27-35, 1996.
[7] 元田敏浩，山本修一郎，WeBBASE，イントラネットアプリケーション構築環境， NTT 技術ジャーナル，Vol.8, No.10, pp.60-63, 1996.
[8] Yamamoto, S., “A Proposal for Integrating WWW and Relational Database,” JCKBSE’96, pp.218-221, 1996.
[9] Yamamoto, S., “Experiences on an Intranet Application Development System WebBASE,” . In Kowledge-based Software Engineering, edited by Navrat, P. and Ueno, H., IOS Press, 1998.
[10] Yamamoto, S., Kawasaki,R., Motoda, T. and Tokumaru, K., “Internet/ Intranet Application Development System WebBASE and its Evaluation,” IEICE Japan, to be appeared.
[11] WebBASE ホームページ, http://webbase.ntts.co.jp
[12] 黒川裕彦，徳丸浩二，元田敏浩，渡部一成，VGUIDE を用いたマルチメディア情報システムの構築, NTT 技術ジャーナル, Vol.7, No.12, pp.82- 85, 1995.
[13] S.Yamamoto, R.Kawasaki, M. Nagaoka, “VGUIDE: 4GL Application Platform for Large Distributed Information Systems,” COMPSAC‘96, pp.536-541, 1996.
[14] Yamamoto, S., Kawasaki,R., “Evaluation of a Client Server Application Development Tool: VGUIDE,” ECIS’96, pp.163- 170, 1996.
[15] K.Yoshihara and M.Yoshitake, “Construction of the Surface Analysis Network Database, Journal of Surface Analysis,” Vol.1, No.3, pp. 369- 373, 1995.
[16] 山本修一郎, 黒川裕彦, WebBASE を用いたニュースオンデマンドシステムの実現, NTT 技術ジャーナル, Vol.8, No.8, pp.16- 19, 1996.
[17] 忠海 均, ソフトウェア設計情報管理システムの開発, NTT 技術ジャーナル, Vol.9, No.1, pp.67- 69, 1997.
[18] NTT DIRECTORY: http://navi.ntt.co.jp
[19] 徳丸浩二，山本修一郎，WebBASE のマルチメディア・ディレクトリ・システムへの適用, NTT 技術ジャーナル, Vol.8, No.6, pp.58-60, 1996.

[20] 田中一男, InfoBee 検索エンジンを用いたディレクトリ検索サービス, NTT 技術ジャーナル, Vol.8, No.8, pp.24-27, 1996.
[21] 黒川裕彦, 伊藤光恭, 岩城勝博, VGUIDE と WWW を用いたダウンサイジング事例,NTT 技術ジャーナル, Vol.8, No.5, pp.54-57, 1996.
[22] 加藤保夫, 古谷博昭, 村上誠, WWW による技術情報の発信,NTT 技術ジャーナル, Vol.9, No.1, pp.89-91, 1997.
[23] 本田祐吉, ホテル予約システムの開発と今後の課題,NTT 技術ジャーナル, Vol.9, No.1, pp.84-88, 1997.
[24] 南 陽, 式場 薫, 西岡和彦, 本間健二, ISDN オンライン申し込みサービスの実現と今後の課題, NTT 技術ジャーナル, Vol.9, No.1, pp.76-79, 1997.
[25] 山路 薫, ゴルフ場予約システムの開発と今後の課題,NTT 技術ジャーナル, Vol.9, No.1, pp.80-83, 1997.
[26] Hadjiefthymiades, S.P. and Martakos, D.I., “A Generic Framework for the Deployment of Structured Databases on the World Wide Web,” Proc. Fifth Int‘l World Wide Web Conf., May 1996, http://WWW5conf.inria.fr/.
[27] Genesereth, M.R., Keller, A.M., and Duschka, O.M., “Infomaster: An Information Integration System,” SIGMOD‘97, pp.539- 542, 1997.
[28] Wi Sae-Tung, Ohmori, T., and Hoshi, M., “An Information Integration Architecture for Mobile Users in WWW Environment,” Trans. of Information Processing Society of Japan, vol.39, No.4, pp.888- 900, 1998.
[29] Miller, R.J., and Tsatalos, O.G., and Williams, J.H., “DataWeb: Customizable Database Publishing for the Web,” IEEE Multimedia, vol.4. No.4, pp.14-21, 1997.
[30] Nguyen,T. and Srinivasan, “Accessing Relational Databases from The World Wide Web,” Proc. ACM SIGMOD Int‘l Conf. On Management of Data, ACM Press, New York, pp.529-540, June 1996.
[31] Hettler, M., “Serving Up Data on the Web,” BYTE, September, pp.112-116, 1996.
[32] IIS, http://WWW.microsoft.com
[33] WebSite, http://WWW.ora.com/
[34] WebServer, http://WWW.oracle.com/
[35] Live Wire, http://home.netscape.com

第 6 章
[1] Stevens, W., Myers, G. and Constantine, L. “ Structured design, ” IBM Systems Journal, vol.13, no. 2, pp.115- 139, 1974.
[2] Yamamoto, S., “Reconstructing Dataflow Diagrams from Structure Charts based on Input and

Output Relationship," IEICE Trans. on Information and Systems, Vol.E78-D, No.9, pp.1118- 1126, 1995.
[3] Yamamoto, S., "Comparative Study of Object-Oriented Analysis," ISFST'97, pp.37- 43, 1997.
[4] 山本修一郎，黒木宏明，オブジェクトパターンに基づくオブジェクトモデリング手法の実験評価，電子情報通信学会，知能ソフトウェア工学研究会，KBSE 97-1, pp.1-6, 1997.
[5] 山本修一郎，黒木宏明，オブジェクトモデリング手法の実験評価，ソフトウェア科学会論文誌，コンピュータソフトウェア, Vol.16, No.4, pp.59-75, 1999.
[6]山本修一郎，黒木宏明，複数人によるオブジェクトモデル作成実験の比較評価，情報処理学会，ソフトウェア工学研究会，115-18，pp.125-132，1997.
[7] Yamamoto, S. and Kuroki, H., "Patterned versus conventional Object-Oriented Analysis Methods: A Group Project Experiment," IEICE Trans. on Information and Systems, Vol. E81-D, No.12, pp.1458-1465, 1998.
[8] Jacobson, I., " A confused world of OOA and OOD," JOOP, Vol.8, No.9, pp.15-20, 1995.
[9] Wirfs-Brock, R., " Stereotyping: A technique for characterizing objects and their interactions, " Object Magazine, Vol.3, No. 4, pp.50-53, 1993.
[10] Wirfs-Brock, R., " How designs differ," ROAD, Vol.1, No.4, pp.51-56, 1994.
[11] Yamamoto, S. and Isoda, S., "SoftDA--- A Reuse Oriented Software Design System, " Compsac'86, pp.284- 290, 1986.
[12] 磯田定宏，山本修一郎，下村隆夫，黒木宏明：設計情報とコードの一体管理方式に基づくソフトウェア開発支援システム(SoftDA)"，NTT R&D, Vol.38, No.11, pp.1239-1248, 1989.
[13] Isoda,S., Shimomura,T., Yamamoto,S. and Kuroki,H., " Integrated CASE System:SoftDA, " NTT REVIEW, vol.2, no.2,(Mar., 1990) pp.52-61.
[14] Isoda, S., Yamamoto, S., Kuroki, H. and Oka, A., "Evaluation and Introduction of the Structured Methodology and a CASE Tool," Journal of Sys. and Soft., Vol. 28, pp.49-58, 1995.
[15] Yamamoto, S. and Isoda, S., "A Transformational Algorithm for Generating Natural Structure Charts from Dataflow Diagram," JCSE'92, pp.190- 197, 1992.
[16] 畑恵介，高田信一，山本修一郎，並行開発型分散情報システム開発技術 ParaDISE，NTT R&D，Vol.45, No.8, pp.719-726, 1996.
[17] 畑恵介，高田信一，山本修一郎，3 層 C/S システム開発方法論の評価実験，情報処理学会，情報システム研究会，56-4，pp.29-36，1995.
[18] 山本修一郎，安原隆一，イントラネットアプリケーション開発の効率化について，電子情報通信学会，知能ソフトウェア工学研究会 SGC97-3, pp.27-34 , 1997
[19] Yamamoto, S., "Experiences on an Intranet Application Development System WebBASE," JCKBSE'98. In Knowledge-based Software Engineering, edited by Navrat, P. and Ueno, H., IOS Press, 1998.

本論文に関する原著論文著書ならびに特許

［学術論文］

[1] 山本修一郎，太田賢司，データフロー図の自動生成方式の提案，信学会，論文誌 D-I, Vol. J76-D-I, No. 10, pp.504-513, 1993.
[2] Yamamoto, S., "Reconstructing Dataflow Diagrams from Structure Charts based on Input and Output Relationship," IEICE Trans. on Information and Systems, Vol.E78-D, No.9, pp.1118- 1126, 1995.
[3] 山本修一郎，黒木宏明，オブジェクトモデリング手法の実験評価，ソフトウェア科学会論文誌，コンピュータソフトウェア, Vol.16, No.4, pp.59-75, 1999.
[4] Yamamoto, S. and Kuroki, H., "Patterned versus conventional Object-Oriented Analysis Methods: A Group Project Experiment," IEICE Trans. on Information and Systems, Vol. E81-D, No.12, pp.1458-1465, 1998.
[5] Yamamoto, S., Kawasaki,R., Motoda, T. and Tokumaru, K., "Internet/ Intranet Application Development System WebBASE and its Evaluation," IEICE Trans. on Information and Systems, Vol. E81-D, No.12, pp.1450-1457, 1998.
[6] Isoda, S., Yamamoto, S., Kuroki, H. and Oka, A., "Evaluation and Introduction of the Structured Methodology and a CASE Tool," Journal of Sys. and Soft., Vol. 28, pp.49-58, 1995.
[7] 岡敦子，山本修一郎，磯田定宏，ソフトウェア開発実験に基づく構造化分析／設計手法の評価，情報処理学会論文誌，Vol.34, No.12, pp.2543-2551, 1993.

［国際会議］

[8] Yamamoto, S., "A Method for Generating Dataflow Diagram based on the Relationship between System Input and Output ," JCKBSE'94, pp. 14- 19, 1994.
[9] Yamamoto, S. and Isoda, S., "A Transformational Algorithm for Generating Natural Structure Charts from Dataflow Diagram," JCSE'92, pp.190- 197, 1992.
[10] Yamamoto, S. and Isoda, S., "SoftDA--- A Reuse Oriented Software Design System, " Compsac'86, pp.284- 290, 1986.
[11] Yamamoto, S., "Comparative Study of Object-Oriented Analysis," ISFST'97, pp.37- 43, 1997.
[12] Yamamoto, S. and Kawasaki,R., "Evaluation of a Client Server Application Development Tool: VGUIDE," ECIS'96, pp.163- 170, 1996.
[13] Yamamoto, S., Kawasaki,R., Nagaoka, M., "VGUIDE: 4 GL Application Platform for Large Distributed Information Systems," Compsac'96, pp.536- 541, 1996.
[14] Motoda, T., Tokumaru, K., Kawasaki, Yamamoto, S., "Experimental Performance Analysis for Relational Database Applications using WWW," ISFST-96, pp.159- 164, 1996.
[15] Yamamoto, S., "A Proposal for Integrating WWW and Relational Database," JCKBSE'96,

pp.218- 221, 1996.
[16] Yamamoto, S., "Experiences on an Intranet Application Development System WebBASE," JCKBSE'98. In Knowledge-based Software Engineering, edited by Navrat, P. and Ueno, H., IOS Press, pp. 7- 15, 1998.

［著書］
[17] 畑恵介，黒木宏明，山本修一郎，長岡満夫，ParaDISE 分析設計技法，オーム社，1998
[18] 青山幹雄，中所武司，向山博編， コンポーネントウェア，共立出版，1998
[19] 西尾章治郎，岸野文郎，塚本昌彦，山本修一郎，石田亨，川田隆雄，岩波講座 マルチメディア情報学 第12巻 相互の理解， 岩波書店，1999

［NTT 機関論文誌等］
[20] 山本修一郎，川崎隆二，分散型情報システム構築基盤 VGUIDE の導入状況，NTT 技術ジャーナル, Vol.8, No.8, pp.45-48, 1996.
[21] 山本修一郎，イントラネットを取り巻く状況と今後の展開, NTT 技術ジャーナル, Vol.8, No.10, pp.50-55, 1996.
[22] 山本修一郎，忠海均，上野正巳，ドメイン指向要求分析技術：DREM，NTT R&D, Vol.45, No.8, pp.711-718, 1996.
[23] 忠海均，高田信一，山本修一郎，リエンジニアリング技術：RELICS，NTT R&D, Vol.45, No.8, pp.739-744, 1996.
[24] 磯田定宏，山本修一郎，下村隆夫，黒木宏明：設計情報とコードの一体管理方式に基づくソフトウェア開発支援システム(SoftDA)"，NTT R&D, Vol.38, No.11, pp.1239-1248, 1989.
[25] 高田信一, 畑恵介, 山本修一郎, 3 層クライアント／サーバ型情報システムの設計手法, NTT 技術ジャーナル，Vol.8, No.1, pp.74- 77, 1996.
[26] 畑恵介，高田信一，山本修一郎，並行開発型分散情報システム開発技術 ParaDISE ，NTT R&D，Vol.45, No.8, pp.719-726, 1996.
[27] 畑恵介, 斎直人, 山本修一郎, ParaDISE: イントラネットアプリケーション設計方法論, NTT 技術ジャーナル, Vol.8, No.10, pp.64-67, 1996.
[28] 山本修一郎, WebArc によるイントラネット・ソリューションの現状と課題，NTT 技術ジャーナル，Vol.9, No.8, pp.48-51, 1997.
[29] S.Yamamoto, H.Kurokawa, K.Tokumaru, S.Adachi, "WebBASE- An Intranet Application Development System," NTT REVIEW, Vol.8, No.4, pp.58-65, 1996.
[30] Yamamoto,S., and Tokumaru, K., "WebBASE: An intranet application development system," AMERICAN PROGRAMMER, vol.9, No.8, pp.27-35, 1996.
[31] Kawasaki,R. and Yamamoto, S., " 4GL Client Server Application Platform: VGUIDE," NTT REVIEW, Vol.8, No.3, pp.64- 73, 1996.

[32] 元田敏浩，山本修一郎, WeBBASE，イントラネットアプリケーション構築環境， NTT 技術ジャーナル，Vol.8, No.10, pp.60-63, 1996.
[33] 山本修一郎, 黒川裕彦, WebBASE を用いたニュースオンデマンドシステムの実現, NTT 技術ジャーナル, Vol.8, No.8, pp.16- 19, 1996.
[34] 徳丸浩二，山本修一郎，WebBASE のマルチメディア・ディレクトリ・システムへの適用, NTT 技術ジャーナル, Vol.8, No.6, pp.58-60, 1996.
[35] 川崎隆二，山本修一郎，イントラネットのアプリケーション・アーキテクチャ，NTT 技術ジャーナル，Vol.8, No.10, pp.56-59, 1996.

［研究会］
[36] 山本修一郎，太田賢司，データフロー図の自動生成方式の提案，電子情報通信学会，知能ソフトウェア工学研究会 KBSE92-19, pp.41-48 , 1992.
[37] 山本修一郎，システム入出力の依存関係に基づくデータフロー図の修正方式，電子情報通信学会，知能ソフトウェア工学研究会 KBSE93-16, pp.33-40 , 1993.
[38] 川崎隆二，黒川裕彦，山本修一郎，簡易言語による大規模分散型システム構築環境 VGUIDE の適用，情報処理学会，情報システム研究会，56-2，pp.9-18，1995.
[39] 高田信一，畑恵介，山本修一郎，3 層クライアントサーバアーキテクチャに適応したシステム設計手法の提案，情報処理学会，情報システム研究会，56-3，pp.19-28，1995.
[40] 畑恵介，高田信一，山本修一郎，3 層 C/S システム開発方法論の評価実験，情報処理学会，情報システム研究会，56-4，pp.29-36，1995
[41] 高田信一，山本修一郎，3 層クライアント／サーバシステム設計方法の提案，電子情報通信学会，知能ソフトウェア工学研究会 KBSE95-6, pp.39-46，1995.
[42] 瀧野修，足立真一，山本修一郎，業務情報とツール特性の分類に基づくイントラネット構成法，電子情報通信学会，知能ソフトウェア工学研究会 KBSE96-22, pp.9-14 , 1996.
[43] 西永誠司, 松岡寿延，山本修一郎，WWW を利用した事例検索型アプリケーションパッケージの開発，電子情報通信学会，知能ソフトウェア工学研究会 KBSE96-23，pp.15-20 , 1996.
[44] 山本修一郎，オブジェクトパターンに基づくオブジェクトモデリング手法の提案，電子情報通信学会，知能ソフトウェア工学研究会，KBSE 96-28, pp.17-22, 1997.
[45] 山本修一郎，黒木宏明，オブジェクトパターンに基づくオブジェクトモデリング手法の実験評価，電子情報通信学会，知能ソフトウェア工学研究会，KBSE 97-1, pp.1-6, 1997.
[46] 山本修一郎，安原隆一，イントラネットアプリケーション開発の効率化について，電子情報通信学会，知能ソフトウェア工学研究会 SGC97-3, pp.27-34 , 1997
[47] 神谷造，瀧野修，山本修一郎, WWW を用いたワークフロー管理システムに関する変更容易性の評価，電子情報通信学会，知能ソフトウェア工学研究会 KBSE97-13, pp.7-14 , 1997.
[48] 山本修一郎，黒木宏明，複数人によるオブジェクトモデル作成実験の比較評価，情報

処理学会，ソフトウェア工学研究会，115-18，pp.125-132，1997.

［特許］

[49] 特願平 4-47670 号 モジュール構造作成装置
[50] 特願平 4-113503 号 データフロー図作成装置
[51] 特願平 4-305485 号 データフロー図作成方法
[52] 特願平 5-140962 号 データフロー図作成方法
[53] 特願平 9-128604 号 ワークフロー制御方法及びその装置並びにワークフロー制御プログラムを記録した媒体
[54] 特願平 9-137092 号 ハイパーメディアアプリケーション構造作成方法
[55] 特願平 8-138789 号 組版言語文書を使用する情報提供システム
[56] 特願平 8-138789 号 情報提供システム