長野大学紀要　第21巻第3号　10—25頁（298—313頁）1999

Visual Basic 活用のための「オブジェクト」思考
New Approach For Object-Oriented Programming

山　本　尚　志*
Naoshi Yamamoto

はじめに

オブジェクト指向は、1970年、ゼロックスパロアルト研究所で、smalltalk が誕生したことに始まるとされる。Smalltalk は、“幼児にもプログラミングできる言語”というコンセプトに基づき開発された言語であり、人が実世界の「事物」を、頭の中で理解できる最小単位として、オブジェクトという概念が、提案されたといわれている。

その後、オブジェクトの基礎概念の整理体系化も、オブジェクトプログラミングの方法論も拡充され、オブジェクトプログラミング言語も、C＋＋, Java, Visual C++, Visual C++, Visual J++、Visual Basic 等とラインアップも増え、これらを使いこなすためには、それなりの創造性技法が必要になる。

本稿では、オブジェクト指向の考え方を、活用するオブジェクト言語のクセを把握して、ユニークなオブジェクトプログラミング、Visual Basic によるアプリケーションプログラムを開発するための基本的な思考法を展開する。

この小論において、オブジェクト思考とは

“オブジェクト指向の考え方をオブジェクトプログラミングに
結びつける技術または行為である”

とする立場をとることにする。

そして、オブジェクト指向とオブジェクト思考に基づく、コンピュータソフトウェア開発に際して、スパイラルモデルを活用しながら、プロトタイプのソフトウェアを開発し、さらに次善のスパイラルモデルを設定しながら、さらに高品質プロトタイプのソフトウェア開発のための手がかりとする。このようなアプローチを、オブジェクトプログラミングのための、スパイラルアプローチということにする。この基本構造図は、第1図である。この構造図に基づいたスパイラルアプローチは、以下の手順

- A　スパイラルモデルによるオブジェクトプログラミングアプローチ
- B　オブジェクト指向の基礎
- C　Visual Basic Ver 6によるコンポーネント
- D　コンポーネントプログラミングのためのオブジェクトプログラミングのための思考法

に従う。この手順により、オブジェクトプログラムは、つぎからつぎへと高品質化される。また、ここで、活用するオブジェクトプログラミング言語は、Visual Basic Ver 6（以下、単にＶＢと記す）とする。

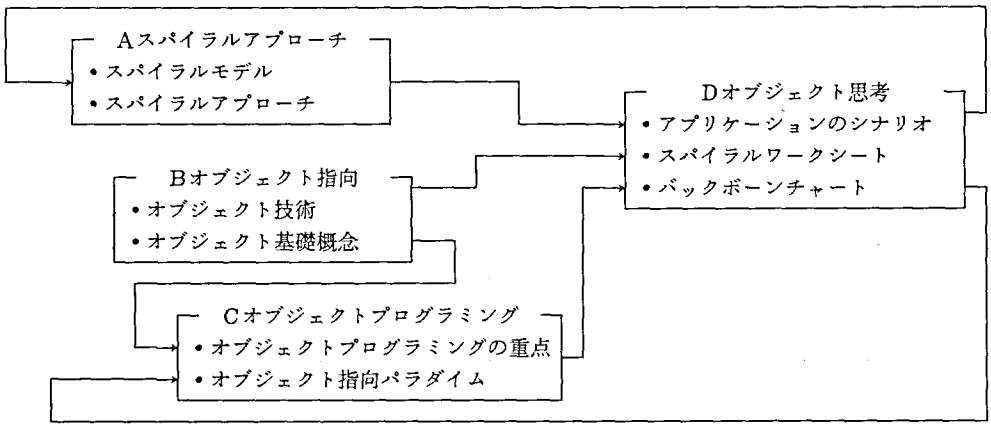
A．スパイラルアプローチ

コンピュータソフトウェアの例えば、素晴らしいゲームプログラムを実現するとき、何段階かのプロトタイプに仕分けて、開発する。

構想立案　　：当面実現したい状態
リスクマネジメント：上記構想を実現するための難点の分析と難点解消の方式検討
自己点検・客筋点検：実現内容の自己または客筋

*教授

図1　オブジェクト思考基本構造図



の評価

プロトタイプ実現　：上記の事項を反映するソフトウェアの開発

の4局面を納得のいく状態まで、上のように廻すアプローチをスパイラルアプローチと呼ぶ。

本稿では、このアプローチにより、アプリケーションプログラムの高品質化に適用する具体的な展開を試みる。

このスパイラルアプローチは、4局面を1サイクルにした意思決定過程を、段階的に展開するものである。これは、“もの”を作り上げる過程で、雑多な意思決定要因が、出たり入ったりすることを配慮した、受け皿である。

この展開の手がかりは、A－1図のようなスパイラルモデル基本図である。

この図は、

(1)　第i次実現状態の構想立案
(2)　(1)を実現するためのリスクマネジメント
(3)　(1)、(2)に基づく、評価、点検
(4)　第i次プロトタイプ・プログラムの開発
(5)　これで満足なら、完了
(6)　不満な、i＋1として(1)へ

このアプローチは、D章で、ワークシート、シナリオライチングと統合されて、具体的に展開される。

B．オブジェクト指向

オブジェクト指向は、オブジェクト指向技術に始まり、オブジェクト指向に関る基礎概念を把握し、オブジェクトプログラミングの伝統的基礎技術、カプセル化（Encapsulation)、多態性（Polymorphism)、継承（Inheritance）を点検し、この伝統的基礎技術を超えたオブジェクトパラダイムについて考察する。

B－1　オブジェクト指向基礎

オブジェクト指向は、オブジェクト技術の見直・点検から、方策を見出し、オブジェクトプログラミングのための基礎技術の整理体系化をおこなってきた。

B－1－1　オブジェクト技術

オブジェクト指向から、オブジェクト指向プログラミングを考える契機となるオブジェクト技術の五特性についての観察と考察から始めたい。

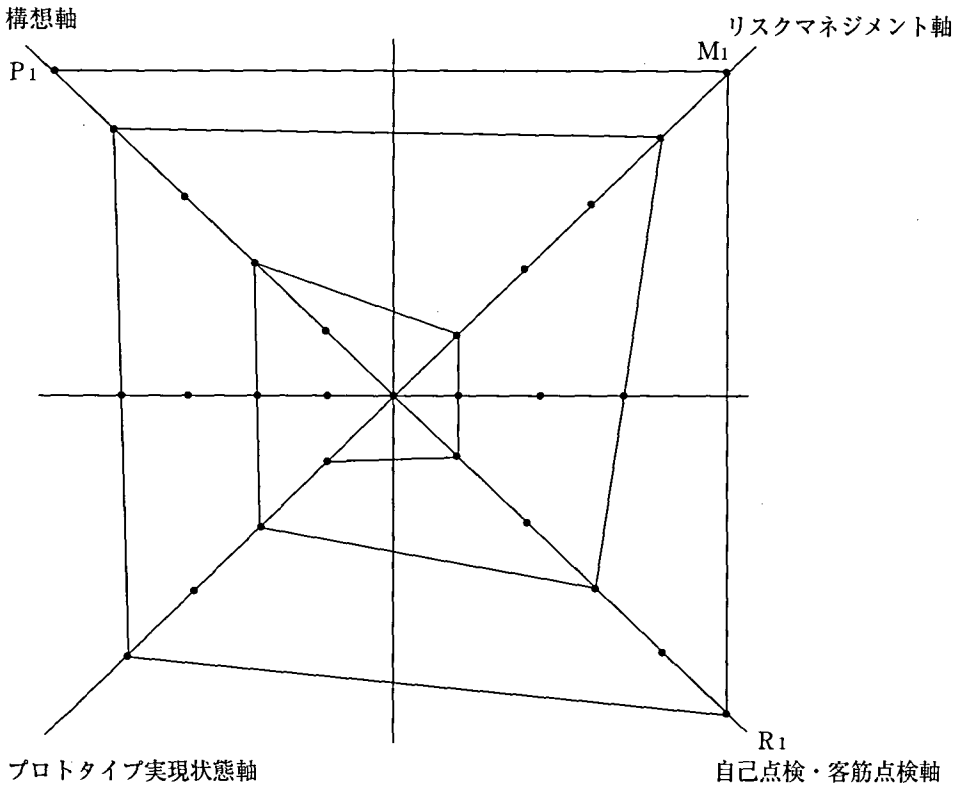
我々の周辺にある技術的成果は、すくなくとも、以下の五特性を備えている。

• モジュール方式

例えば、映像処理のシステムには、さまざまなものがあるが、ＰＣを主軸にしたものには、ＰＣ、デジタルビデオカメラ、またはデジタルカメラ、画像印刷用プリンタ、画像編集用ソフトのようにモジュールになっている。

• 実世界との一致

上のような映像システムには、デジタルビデオカメラ、またはデジタルカメラは、実世界の映像を記録し、ＰＣに映像を送り込む、画像編集用ソフトウェアで、解像度を調整したり、映像サイズの拡大収縮、色のメリハリ、音の調整など実世界のより良い反映のための機能を活用できる。さらに必要とあれば、画



A－1図　スパイラルモデル基本図

像印刷用プリンタにより印刷でき、実世界との一致の度合いをハードコピーにして確認できる。

• ゆるい結び付き

デジタルカメラ、デジタルビデオカメラは、いくつもの種類があり、これらのどのカメラも、ＰＣに接続可能である。これは、各カメラが、特定のＰＣでなければ、接続できないとしたものでなく、各ベンダーが。規格に準拠して、どのＰＣとでも接続可能なように、ゆるい結びつきを許すものとなっているからである。

• 増設の可能性

購入資金に制約があって、望ましい組み合わせになるように調達できないとすれば、デジタルカメラとＰＣだけで画像処理を楽しめばよい。ＰＣに Visual Basic Ver 6 でもあれば、プログラムを工夫して、画像処理プログラムを開発して、編集できるようにもなる。資金ができれば、増設可能である。

• 再利用

友人などの撮影した、映像を、あなたのＰＣ画像処理システムで処理できるし、あなたの映像を、だれかの画像処理システムで処理もできる。

要は、どのような映像オブジェクトでも、いろいろな場面で、再利用できるようにしてきているのが、オブジェクト技術である。

これら五特性をベースに、コンピュータのソフトウェア開発を目指す方策を、オブジェクト指向ということにし、オブジェクトプログラミングのための基礎技術、カプセル化、継承、多態性を実現する基礎概念を超えた、オブジェクトパラダイムへと整理体系化してきた。

B－1－2　オブジェクトの基礎概念の点検

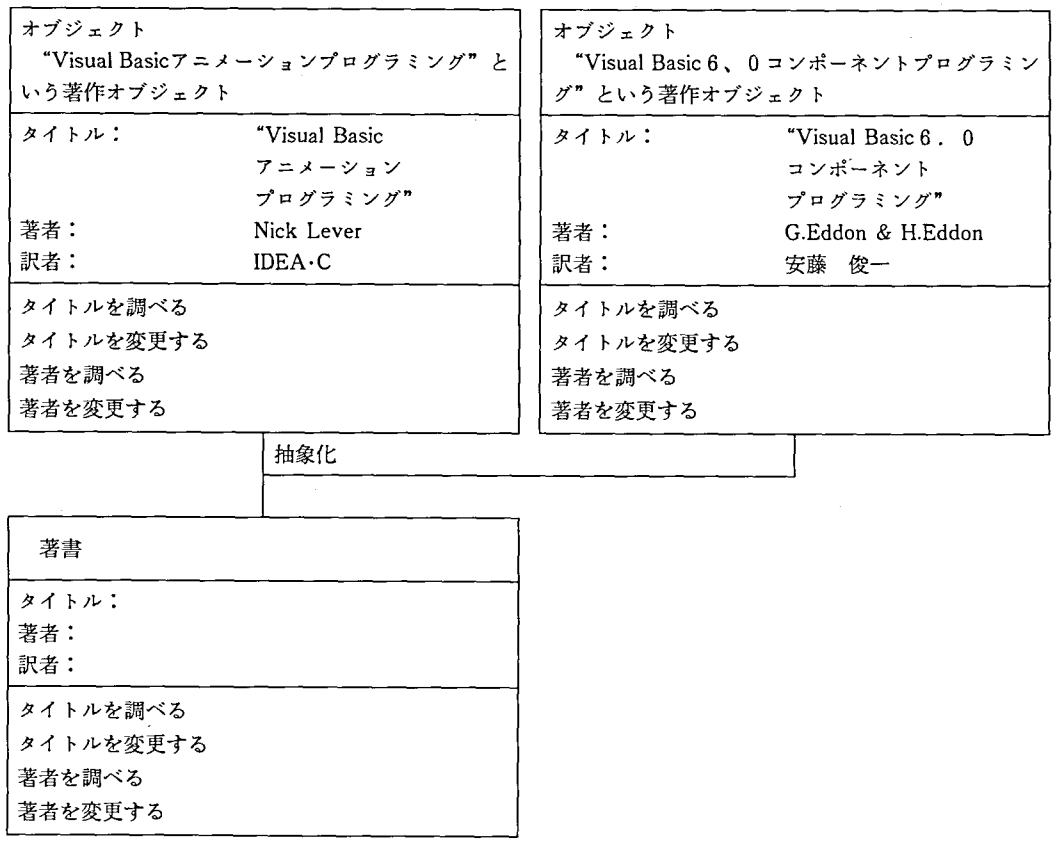
オブジェクト技術、オブジェクト指向、オブジェクトプログラミング、オブジェクト指向プログラム言語が、お互いに刺激し合うことにより、

オブジェクトアプリケーション開発の方法論が充実してきた。ここでは、オブジェクト指向に関る重点事項を整理しておく。

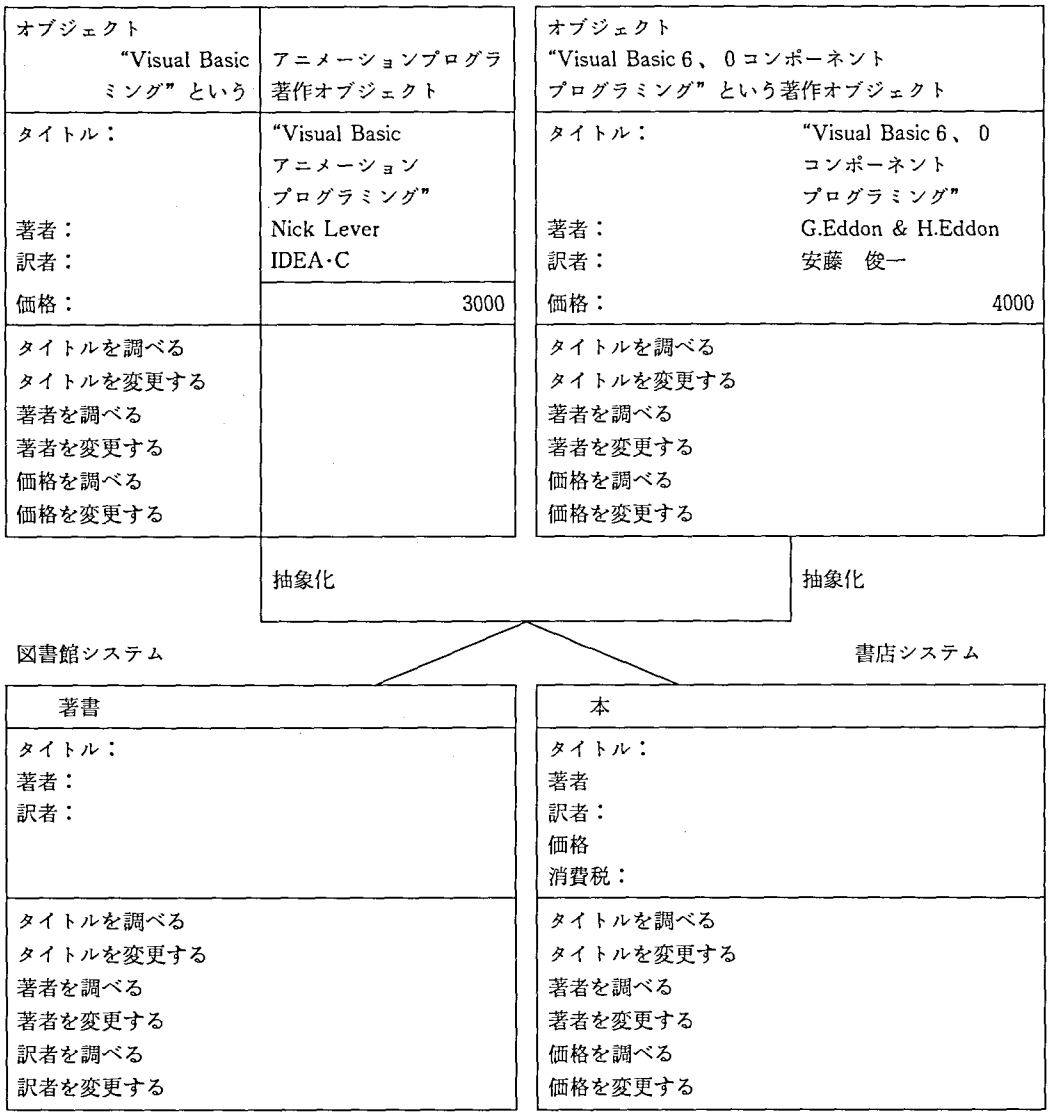
- オブジェクトの一般規定
  オブジェクトの一般規定は、「実世界に存在する事物」である。
  このオブジェクトの条件は、名前があり、名前がつけられる事物である。
- オブジェクトには、特性と行動がある
  例えば、経営システムというオブジェクトには、経営実体と一致することを示す特性、経営環境変化を先取りする戦略経営のための行動がある。
- オブジェクトは、目的、方針、目標、メソッドをもつ
  オブジェクトは
  "新経営戦略構想の実現に支障きたす異常情報を把握する戦略情報システムの強化形成"のような目的を持ち
  "向こう五年間戦略情報システムの再利用度を55%から、85%に増大する"などの方針
  "戦略情報システムのメンテナンスコストを50%削減する"などの目標と目標達成のメソッド（＝達成手段）をもつ

さらに、

- オブジェクトには、製造機がある
  クラスは、「オブジェクト」製造機である
  オブジェクトは、事物であるからなんらかの製造装置によって製造されることになる。製造されたオブジェクトは、インスタンス(instance)という。
- オブジェクトは、メンバー変数と値をもつ
  メンバー変数はオブジェクト内に、データを確保するための空間であり、値は、メンバー変数の中に出し入れされるデータである
- オブジェクトの構造



C－1－1図　オブジェクトからクラスの抽象化



図C－1－2　対象システム別クラス定義

is_a　関係
　一般化と特化の関係
　親子関係のような階層構造の確立
has_a　関係
　集約と分化の関係
　部品展開のような構成構造の確立
その他の関係、関連付けの関係もある。

こうした、概念整理のもとに、オブジェクトを決め、オブジェクトの構造を睨みながら、抽象化のプロセスを踏み、オブジェクトプログラミングの方法論をくみあげてきたのである。

C　オブジェクトプログラミング

オブジェクト指向の考え方をベースにして、特定のオブジェクト指向言語によりアプリケーションを開発する行為をオブジェクトプログラミングという。

本稿では、ＶＢがオブジェクト指向言語である。

C－1　オブジェクトプログラムの基本

オブジェクト指向の諸概念の整理とモデル化を行い、オブジェクトプログラミングの準備となる、オブジェクト形成の具体化を試みる。

- 抽象化とクラス定義

例えば、大学図書館を事例として、オブジェクトを決め、この抽象化により、クラスの定義の仕方を把握する。オブジェクトから、クラスを抽象化することを、データ抽象化という。データ抽象化の過程は、主として、オブジェクトが個々に保有している具体的な値を除去し、さらに、該当システムにとって、本質的でない属性の除去も行うことにある。

- 対象システムによるクラス定義の差

書店システムと図書館システムのクラス定義の仕方を実践し、対象システムによるクラス定義の違いを確認する。

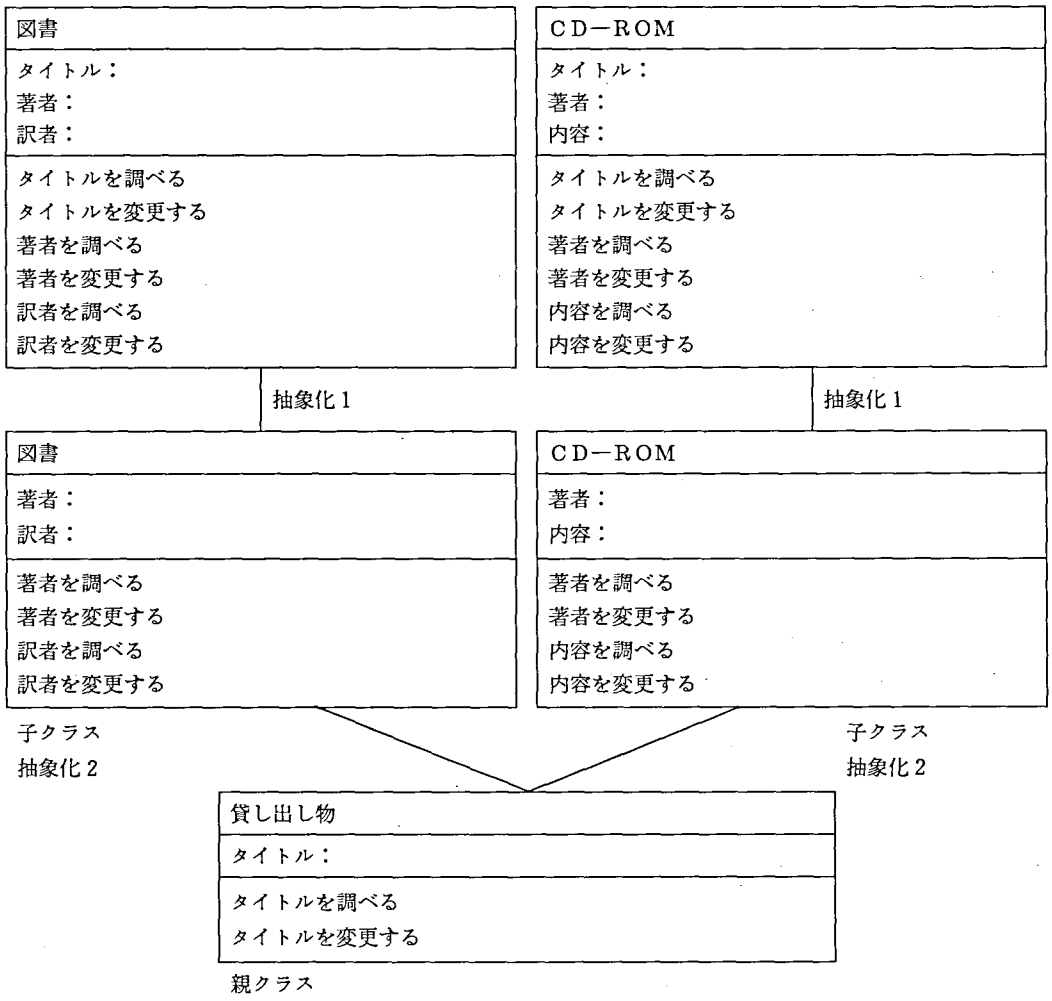
- スーパ抽象化プロセス

抽象化には、上述のデータ抽象化に加えて、スーパ抽象化がある。

スーパ抽象化は、該当システムから、複数のクラスが定義可能であり、このクラスの中で、同じ性質を持つクラスを、さらに抽象化することである。このことを確認するために、大学図書館システムの具体的な本とCD-ROMの場合を、例にする。そして、スーパ抽象化によって、親クラス、子クラスの階層構造が出来上がることも確認し、後に検討する継承の具体的な事例を与えるものである。

C－2　オブジェクトプログラミングの重点

オブジェクトプログラミングの重点は、カプセル化、継承、多態性である。カプセル化は、オブジェクトに実装されている詳細を非公開にするこ



図C－1－3　スーパ抽象化

とであり、継承は、既存の「オブジェクト」を再利用して、さらに特化したオブジェクトを作成できることであり、多態性は、利用するオブジェクトによって、複数の挙動を示すことである。

• カプセル化のプログラミング

オブジェクトプログラミングの文脈で、カプセル化とは、プログラムコードを関連するデータと結合して、オブジェクトとして定義することである。プログラムコードとデータを結合することを、実装という。このような実装は、定義されたオブジェクトだけのもので、他の関与するものではない。

また、オブジェクトには、インタフェースも装備され、インタフェースによって、オブジェクトとクライアントの間のアプリケーションや、他のオブジェクトが提供するサービスを使うことができることになる。ここでの、インタフェースは、ユーザインタフェースではなく、ソフトウェアの各構成要素がコミュニケーションできるための、プログラミング上のインタフェースである。

カプセル化のホンネとは、オブジェクト内部の実装に、変更、改良、拡張の必要が生じた場合に、このオブジェクトが公開しているインタフェースに影響がないようにすることである。

• 継承のプログラミング

継承とは、親子関係である。親の性質である、属性、操作、関係を、子はすべて受け継承する上に、独自の性質を追加したり、親の性質を再定義することができる。

コードの再利用は、オブジェクトプログラミングの最終ゴールである。継承では、アプリケーション内のコードの再利用を、他のオブジェクトから、機能を継承するオブジェクトを作成することによって実現している。

• ポリモーフィズム（多態性）のプログラミング

ポリモーフィズム（多態性）のプログラムは、ベースクラスに比較的、一般的なコードを書いておく。後になって、当初想定していなかった多種多様なコードを追加できるようにして、特化された転用オブジェクトが、適切に動作するようにしたものである。

環境が変化し、この変化に追随することで、結果的にポリモーフィズム（多態性）プログラムが実現するような埋め込みを、事前に行っておくことが肝要かも知れない。

C－3　VBによるオブジェクト指向パラダイム

オブジェクトプログラミングのパラダイムについて点検したい。パラダイムは、モデルであり、他と識別できるパターンがあり、物事を遂行するための、時代を反映した斬新な方式が備わっている場合に適用される概念である。

Visual Basic Ver 6では、オブジェクトプログラミングにおける再利用を増大する意図をもつ、コンポーネントプログラムが充実した。ここでコンポーネントとは、オブジェクトプログラムの要素であり、コントロールである。異論もあろうが、このことにより、Visual Basic Ver 6は、オブジェクト指向パラダイムを備えた、新ソフトウェアファクトリへの移行を企てるものである。

そして、コンポーネントプログラミング、オブジェクトリポジトリへの方策であり、再利用への取り組みが、強化されたのである。ここでの、オブジェクトリポジトリの意味は、オブジェクトを収める収納庫であり、外部からの要求に対して、調達したソフトウェアチップを提供したり、コンポーネント化された再利用可能なソフトウェアを格納し、管理し、検索し、保守する役割等をする。オブジェクトリポジトリを取り巻く、これからのソフトウェア工場のイメージは、図C－3である。以下この図の組み立ての中心概念を見直すことにする。

Visual Basic Ver 6では、オブジェクト指向のコンセプトが組み込まれ、再利用できるオブジェクト指向プログラムが、比較的簡単に作成できる多数のコンポーネントがある。

例えば、画像処理向けのコンポーネントは、以下のような事項が基本であり、画像処理用オブジェクトリポジトリに、格納される候補である。ここで、基本事例が、あると便利であるが、個々では省略する。

• スライダーの使い方
　ある決まった範囲にある値を、視覚的に設定してオブジェクトを動かす
• タブダイアログの活用

一つのフォームに複数のダイアログをセットして、クリックしたダイアログを前面に出す

- コモンダイアグラムの活用
  初期色の設定と選定色の取得、フォントデータの取得、プロパティによる印刷データの取得に基づき印刷機能を活用する
- イメージリストの活用
  イメージリストコントロールは、他のコントロールに、イメージを提供する機能である。Load Picture を使って、複数のイメージを設定できる。これら複数のイメージの集合は、コレクションになり List Images コレクションに読み込まれ、この各イメージには、1番からインデックスが付けられ、管理可能となる。
- ＯＬＥコンテナコントロールの活用
  ＯＬＥコンテナコントロールは、自分で作成したＶＢアプリケーションの中に、他のアプリケーションのオブジェクトを追加できるようにする。
- ＯＬＥオートメーションの活用
  ＯＬＥコンテナコントロールでは、例えば、Excel データを処理することはできても、この処理の途中で、あるセルの値が何であるか、知ることはできなかったが、ＯＬＥオートメーションを活用することによって

シート、グラフ、セル、セル範囲、文、段落等の情報を、それぞれのプロパティを使って知ることができる。これは、他人のプログラムを勝手に操作できることを意味している。これは、関連するオブジェクトを公開しているからで、このようなアプリケーションを、「ＯＬＥサーバ」という。

そして、公開されているオブジェクトを使う側のアプリケーションは、「クライアントアプリケーション」である。

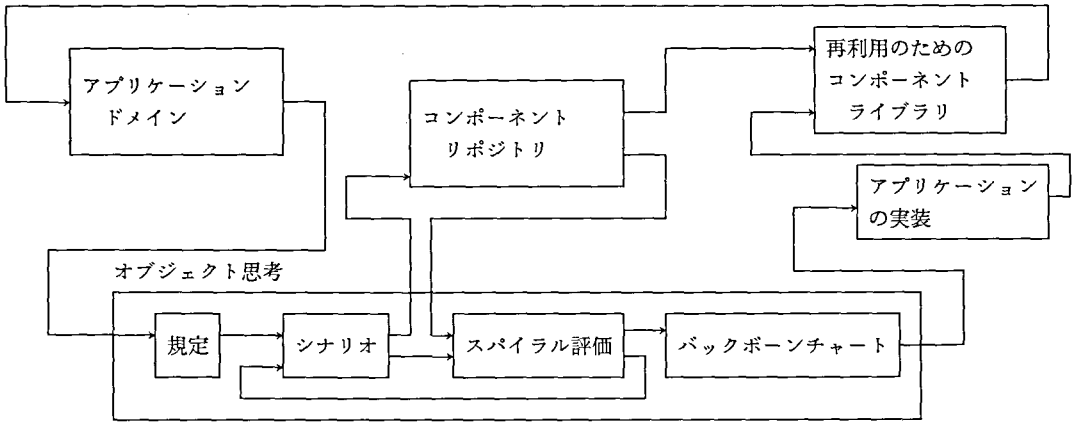
- ピクチャクリップコントロールの活用
  位置を指定して、画像を取り出したり、分割して画像を取り出す。さらに、画像のストレッチをする
- アニメーションコントロールを活用する
  aviファイルのオープン、Auto Play アプリケーションでファイルの自動再生、Play で指定して再生、アニメーションの配置等を可能にする
- ＡＰＩ関数で音楽を演奏する
  ＭＩＤＩ関数による音の操作の技術を取得し、音楽演奏する

これらのコンポーネントは、用例と共にオブジェクトリポジトリに格納され、特定のアプリケーション開発のためのオブジェクトライブラリに登録されていく。

こうしたオブジェクト指向パラダイムの進展は、先に指摘した、オブジェクト指向技術の５特性の再利用を最重点とする思想に根拠を置くものであると考えられる。

この、考えを整理すると、図Ｃ－３のような、近未来のアプリケーションソフトウェアハウスのイメージが発想される。

そして、後述するような、コンポーネントプロ



図Ｃ－３　アプリケーションソフトウェアハウス

主題：　モグラ叩きゲーム
　　プロシージャ制作のためのシナリオ

オブジェクト Command 1 Procedure Click スタートボタンをクリックしたら 　タイマーを動かし 　スタートボタンは非表示 区切る	オブジェクト Form Procedure Load フォームが開かれたとき （初期状態の設定） ゲーム回数を0にする ヒット回数を0にする 飛び出るモグラの番号を0にする モグラの絵を初期状態ではすべて潜ったモグラの絵にする 区切る	オブジェクト Picture 1 Procedure Click クリックされたPictur 1 のコントロール配列の要素が“顔を出したモグラ”であるとき、Picture 4の絵を代入して“頭を叩かれたモグラ”の絵にする そして、ヒット回数を加算する 区切る	オブジェクト Timer 1 Procedure Timer Intervalを1000にしたから、1秒経つと乱数に9.9を掛けて整数化すると0から9の数が得て、“顔を出したモグラ”の番号を求める For I = 0　to　9 　もし“顔をだしているモグラ”でなければ潜ったモグラの絵を代入 　そうでないなら“顔を出しているモグラ”の絵を代入 Next 区切る

フォームの配置手順

手順1　モグラは、一定時間1秒おきに、穴から出たり入ったりするが、出現する場所は、その都度変わるようにする。出現する場所として、コントロール配列を10配置する。この配置要領は、フォームに Picture Box を1個配置し、プロパティ Borde Style を「0－なし」に変更し、Picture Box のオブジェクト名は、Picture 1 のままにして、フォームのプロパティの Caption を「モグラ叩き」とする。

プロパティの背景色 Back Color は、白にする。

手順2　「Picture 1」の上でクリックして選択し、メニューから「編集（E)」－>「コピー（C)」として、クリップボードに取り込む。つづいて、「編集（E)」－>「貼り付け（P）コントロール配列を9個貼り付けるが、上下5個づつにする。

手順3　Picture Box のコントロール配列を上下5個づつ範囲指定して、メニューの「書式（O)」－>「整列（A)」－>「中央（M)」として、横一線に並べる。

手順4　ツールボックスから、ラインを選択して5個づつ上下に並べた間に、区切り線とする。プロパティを以下のように変更する。直線の太さを、15　直線の色をピンク。

手順5　モグラの状態図は、“潜っているモグラ”、“顔を出しているモグラ”、“コブを作られてもアカンベーしてるモグラ”の3図を用意して、順に Pictuer 2、Picture 3、Picture 4に格納しておく

図D－1－a　“モグラ叩き”シナリオ

グラミングを、プログラム化構想、リスクマネジメント、見直・点検、プログラミングとスパイラルに収束させながら、ソフトウェア品質を上げていく過程が、重視されてくる。

D　オブジェクト思考

このD章は、本稿の心臓部である。B章のオブジェクト指向の方策を理解し、C章のオブジェクトプログラミングの技術を活用して、アプリケーション高品質化のための、アプリケーション制作のためのシナリオライティング、スパイラルワークシート、バックボーンチャートの導入を展開する。

ここでのオブジェクト思考とは、以下の8つの手順に従う、オブジェクトプログラミングの開発準備をいう。

(1) オブジェクトプログラムの規定
(2) 第1次オブジェクト化のシナリオライティング
(3) 第1次プロトタイプのバックボーンチャート
　（バックボーンチャートは、D－3で導入する）
(4) スパイラルモデルの評価基準の設定
　• 構想立案の評価基準
　• リスクマネジメントの評価基準
　• ゲーム性の魅力度
　• 開発スピード
(5) 上記評価基準の第1次プロトタイプへ適用
(6) 第2次プロトタイプの品質水準の設定
(7) 第2次プロトタイプのシナリオライティング
(8) バックボーンチャート作成

ただし、第2次プロトタイプまでのアプローチでアプリケーションが完了する場合である。

D－1　アプリケーション制作のためのシナリオ

VBアプリケーションの開発では、大抵の場合オブジェクトは、ツールボックスからコピーして作成され、属性も、特別なものを除いて、プロパティボックスで設定できる。

問題は、プロシージャの制作である。

“モグラ叩き”のプロシージャ制作の例にして、プロシージャ制作シナリオを説明する。

第1次プロトタイプ“モグラ叩き”プログラムの事前準備として

　潜ったモグラの絵
　顔を出したモグラの絵
　頭をぶたれたモグラ絵

を、ビットマップファイルに格納する

プログラムの規定

はじめモグラは10個の穴、10個のピクチャボックスにモグラしておき、ピクチャボックスをクリックしたら、乱数を発生させ、適当な範囲の整数値なら、該当する穴から、顔をだしたモグラの絵がでるようにして、ヒットしたらヒット数を加算していく、ヒット数を競ったり、難易度別のゲームの楽しみ方等は、考えない。

図D－1－aは、上の第1次プロトタイプ“モグラ叩き”のプロシージャ制作シナリオである。

この場合のシナリオ創りの手順を示すと、

手順1　ゲームの舞台装置としてのフォームの配置を図の下段にまとめる

以下スタートから、動作させたい順番を意識して、左から右に書いていく。

手順2　「スタートボタンを押したらタイマーを動かす」、これだけでもよいが、オブジェクト指向プログラムを加味して、図のようにした。

手順3　ゲームの舞台となるフォームの開かれたときの初期状態の設定

手順4　モグラとのゲームのやり取りをする動き

手順5　タイマーによる、モグラの動きを発生させ、時間経過とともに、ゲームが進行するようにしている

このような、シナリオワークシートによると

• 全体展望がきく
• 改善個所が発見しやすい
• D－3のバックボーンチャートに、移行しやすい
• スパイラルチャートと見比べながら改良に挑める

D－2　スパイラルワークシート

ここでは、各段階のプロトタイプの構想立案、リスクマネジメント、判断、開発したプロトタイプの性能に関する、4局面の評価をし、各段階のプロトタイプの水準を把握することを考えたい。

ここでは、ゲームプログラム“モグラ叩き”を

プロトタイプ次数 スパイラル	第1次プロトタイプ		第2次プロトタイプ	
	評価事項	評価	評価事項	評価
構想	P1－1 メニューからスタート		P2－1 メニューからスタート	
	P1－2 1秒単位でモグラが出たり入ったりする		P2－2 1秒単位でモグラが出たり入ったりする	
			P2－3 ヒット回数を競う	
			P2－4 メニューからゲームの難易度を変えられる	
			P2－5 メニューから終了できる	
	評 価	9	評 価	5
リスクマネジメント	R1－1 スタート コマンドボタンの配置		R2－1 スタート コマンドボタンの配置	
	R1－2 Timer1－を配置しプロパティのintervalは1000		R2－2 Timer2を加え時間がきたら該当処理	
			R2－3 ヒットしたらモグラが舌を出し、ヒット点数に加算	
			R2－4 ダイアログ用のフォームモジュール追加	
			R2－5 終了	
	評 価	9	評 価	5
見直・点検	E1－1 至極当然		E2－1 至極当然	
	E1－2 スタートボタンをクリックしてもモグラは潜ったまま		E2－2 ゲームに難易度別に選択可能で対応力増大	
			E2－3 ヒット率の計算とゲームの反復試行可	
			E2－4 初級、中級、上級者別に、難易度選択可	
			E2－5 至極当然	
	評 価	9	評 価	5
開発	D1－1 開発コスト無視		2D－1 開発コスト無視	
	D1－2 初期設定		D2－2 二番目のタイマー2追加による振る舞いの弾力化	
			D2－3 フォームの複数化で管理の手間が少しかかる	
			D2－4 レベル選択プログラム開発追加	
			D1－5 開発コスト無視	
	評 価	5	評 価	3

表D－2

事例にして、スパイラルモデルによる、このゲーム品質改善過程を展開する。

第1次プロトタイプのシナリオをベースにして、4局面の評価をするための評価基準を設定すると以下のようになる。

構想の評価基準

ゲームの面白さの実現状態を評価基準とする。

全然面白くない	9
あまり面白くない	7
面白い	5
かなり面白い	3
極めて面白い	1

リスクマネジメントの評価基準

オブジェクト化に関る難度を基準にとる。

全然難しくない	9
あまり難しくない	7
難しい	5
かなり難しい	3
きわめて難しい	1

ゲームとしての出来映えの評価基準

ここでは、“モグラ叩き”ゲームの利用者の反復試行への誘惑等、ゲームとしての魅力度を評価基準とする。

全然魅力がない	9
あまり魅力がない	7
魅力がある	5
かなり魅力がある	3
極めて魅力がある	1

プログラムのスピード開発力

VBに組み込まれたコンポーネントを使えれば、考案する手間は省け、オブジェクトプログラミングはスピード開発可能であるが、考案する必要があればスケジュール化もままならず開発上の難点は大となる。そこで、考案の負荷の度合いを評価基準にとる。

極めてやさしい考案である	9
すこしの考案がある	7
普通の考案である	5
かなり難度の高い考案である	3
極めてたかい難度の考案である	1

これらの評価基準を、スパイラルモデルワークシート表D－2の左側の第1次プロトタイプに適用する。

第1次プロトタイプの構想の面白さの実現状態は、ゲームとしての体裁を整えた程度であるから、評価は、“全然面白くない”の9である。リスクマネジメントについては、オブジェクトの難度であるが、コマンドボタンの配置、タイマーの設定にかかわるものであるため、難度は低く、さしたるリスク減らしマネジメントではないので、評価は“全然むずかしくない”の9である。ゲームとしての魅力度も、“モグラ叩き”としての基本的な要素だけを組み込んだにすぎないので、評価は“極めてやさしい考案である”の9であるが、第2次以降のプロトタイプで、再利用が可能であるので、“普通の考案である”の5とする。これらの評価の結果を表D－2に記入する。図化すると、図D－2の一番外側の曲線、P1－M1－A1－T1が画ける。

以上の第1次プロトタイプの評価に基づき、改良に着手する。

第2次プロトタイプの規定（追加事項のみ記載）

第1次プロトタイプの規定内容に加えて、ゲームに難易度を追加し、繰り返し試行可能にする。

つぎに、このD－2表の右側に改善案を記入し、第2次プロトタイプのゲーム水準を設定しながら、上の評価基準を適用すると実現すべきゲーム内容は、“面白い”で5、オブジェクト化の難易度は、“難しい”の5、ゲームの魅力度は、“魅力がある”の5、この程度のアプリケーションとなると、かなりの力量がいるので、“かなり難度の高い考案”の3とする。

以上から、図D－2の内側の曲線、P2－M2－A2－T2が画かれる。

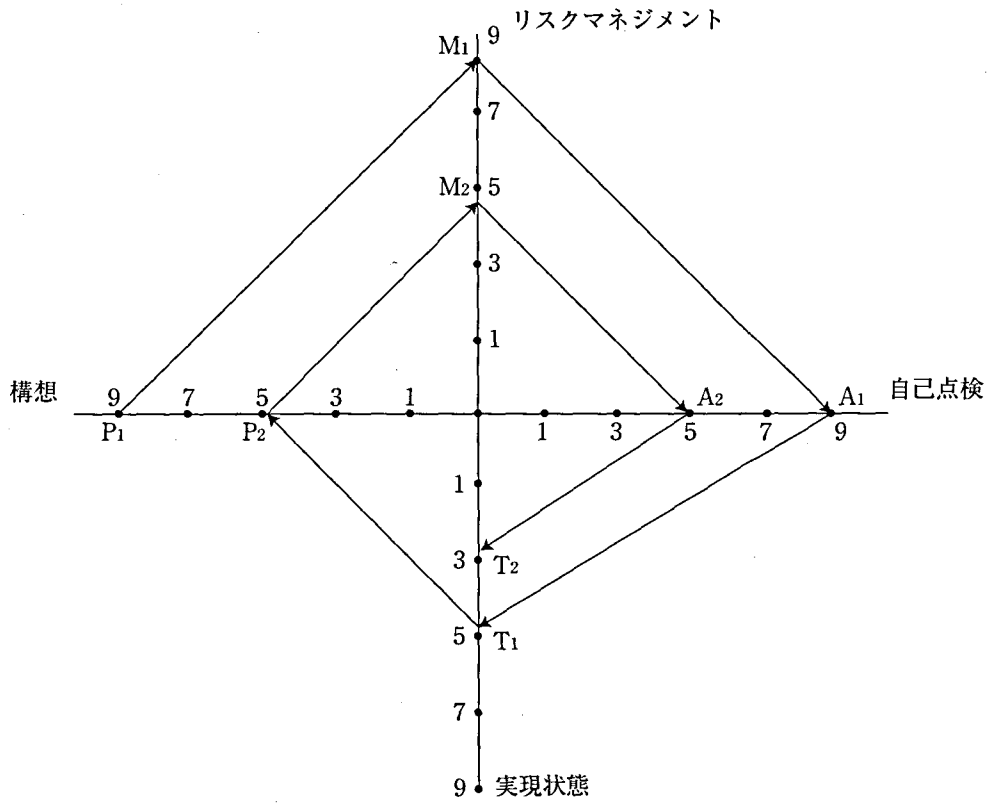
そして、改良水準のイメージP2－M2－A2－T2を参考にして、第1次プロトタイプを改良したシナリオ図D－2－b、第2次プロトタイプを策定する。

D－3　バックボーンチャート

VBで開発されたアプリケーションプログラムは、

オブジェクト　＋　イベントプロシージャ

で構成されている。オブジェクト設定ツール、属性設定のツール、イベントプロシージャ作成ツールの点検を行い、バックボーンチャートの導入を



図D－2 スパイラルモデル

試みる。

VBアプリケーション開発ツール

これからのオブジェクト指向パラダイムは、コンポーネント化が重要であり、特にVBにおけるオブジェクト指向プログラミングでは、コンポーネントプログラミングが主流であることをしてきした。そのために、VBの各コンポーネントの徹底理解はふかけつである。以下VBにおけるオブジェクト指向プログラミングの基本事項を点検すると、

- 手順1 ツールボックスの活用
  ツールボックスには、各種コントロールが用意され、ここからフォームにコピーすることで、オブジェクトが作成される
- 手順2 コントロールボックスの配置
  フォーム上に、テキストボックス、コマンドボックス等の、各種コントロールボックスを配置し、アプリケーションに必要なオブジェクトを決める
- 手順3 各コントロールボックスは、VBではプロパティとよぶ属性を設定することにより制御する
- 手順4 各コントロールボックスは、メソッドとよぶ手続きを使い動作を与える
- 手順5 その他外部のアプリケーションと処理のやりとりができる
- 手順6 外部のデータベースとの関連処理をするための機能がある
- 手順7 便利なデバッグ機能により、プロシージャを効率的に開発できる
- VBアプリケーション開発の基本過程

「画面に配置した、“挨拶”という名前がついたコマンドボタンを押したら、「今日は」とラベルトボックスに、表示される」

処理を考えてみよう

- ツールボックスには、クラスが準備されている
- フォームに、ツールボックスから、ラベルを適当なサイズでコピーすることにより、出力「オブジェクト」をつくる

オブジェクト Command 1 Procedure Click スタートボタンをクリックしたら 　タイマ１ーを動かし 　タイマ２ーを動かし 　スタートボタンは非表示 ゲーム回数を０にする ヒット回数を０にする 飛び出るモグラの番号を０にする モグラの絵を初期状態では10個の配列すべて“潜ったモグラ”の絵にする 区切る mgra GX Procedure Click 　終了指示 　区切る Form 2 の Command 1 ―Click 　自分自身のFormを消去する 　　Me.Hide 区切る	オブジェクト Form Procedure Load フォームが開かれたとき （初期状態の設定） モグラの絵を初期状態では10個の配列すべて“潜ったモグラ”の絵にする 区切る	オブジェクト Picture 1 Procedure Click クリックされたPictur 1 のコントロール 配列の要素が“顔を出したモグラ”であるとき、Picture 4 の絵を代入して“頭を叩かれたモグラ”の絵にする そして、ヒット回数を加算する 区切る	オブジェクト Timer 1 Procedure Timer Intervalを1000にしたから、１秒経つと乱数に9.9を掛けて整数化すると０から９の数が得て、“顔を出したモグラ”の番号を求める For I ＝0　to　9 　もし“顔をだしているモグラ”でなければ“潜ったモグラ”の絵を代入 　そうでないなら“顔を出しているモグラ”の絵を代入 　　ゲーム数加算 　End If Next 区切る Timer 2 Procedure Timer Intervalを3000にする ヒット率の計算し、メッセージボックスに「ゲームオーバー」と共に出力する ゲーム回数を０にする ヒット回数を０にする タイマー２をダブルクリックしたら何度もゲームが出来るように以下の処理をする タイマー１、タイマー２の動きを止め、配列すべて、“潜ったモグラ”の絵に戻しコマンド１ボタン１を再描画する 区切る	オブジェクト mgra GR Procedure Click 「ゲーム（G）」―>「レベルの設定（R）」 タイマー１、タイマー２の動きを止める コマンド１ボタン１を再描画する 配列すべて、“潜ったモグラ”の絵に戻し新たに追加したフォームForm 2 を表示 区切る Form 2 Option 1 Procedure Click レベル分けするオプションボタンをフレームに配置する ケース１ 　“初心者用のスピード”と出力 　インターバル1000に設定 ケース２ 　“中級者用のスピード”と出力 　インターバル500に設定 ケース１ 　“上級者用のスピード”と出力 　インターバル300に設定 区切る
オブジェクト　（グローバル変数の宣言） General Procedure Declarations			ゲーム回数、ヒット回数、ヒット率などを整数値として宣言	

フォームの配置手順

手順１　モグラは、一定時間１秒おきに、穴から出たり入ったりするが、出現する場所は、その都度変わるようにする。出現する場所として、コントロール配列を10配置する。この配置要領は、フォームにPicture Boxを１個配置し、プロパティBorde Styleを「０ーなし」に変更し、Picture Boxのオブジェクト名は、Picture１のままにして、フォームのプロパティのCaptionを「モグラ叩き」とする。

　プロパティの背景色Back Colorは、白にする。

手順２　「Picture１」の上でクリックして選択し、メニューから「編集（E）」―>「コピー（C）」として、クリップボードに取り込む。つづいて、「編集（E）」―>「貼り付け（P）」コントロール配列を９個貼り付けるが、上下５個づつにする。

手順３　Picture Boxのコントロール配列を上下５個づつ範囲指定して、メニューの「書式（O）」―>「整列（A）」―>「中央（M）」として、横一線に並べる。

手順４　ツールボックスから、ラインを選択して５個づつ上下に並べた間に、区切り線とする。プロパティを以下のように変更する。直線の太さを、15　直線の色をピンク。

手順５　モグラの状態図は、“潜っているモグラ”、“顔を出しているモグラ”、“コブを作られてもアカンベーしてるモグラ”の３図を用意して、順にPicture２、Picture３、Picture４に格納しておく

図Ｄ－１－ｂ　改良版　“モグラ叩き”シナリオ

- ラベルの名前を消し、フォントをサイズ20にし、中央揃えにし、背景色を白色にするようプロパティを設定し、出力オブジェクトに属性をセットする
- フォームに、ツールボックスから、コマンドボタンを適当なサイズでコピーする
- コマンドボタンに、例えば“挨拶”のように名前をつける（プロパティのcaptionを“挨拶”とする）このボタンを押せば、イベントプロシージャという、次のコードが働いてイベントが発生し

イベントプロシージャ1

```
Private Sub Command1-Chick( )
  Label1.Caption=“今日は”
End Sub
```

出力オブジェクトであるラベルに、“今日は”と表示される

- 二つめのコマンドボタンをフォームに配置して、終了と名づけ、押せば動作が終了するように、以下のコードを書く

イベントプロシージャ2

```
Private Sub Command1-Click( )
  End
End Sub
```

VBでは、外部からの情報刺激、つまりメッセージによって、行う動作をイベントという。上で、“挨拶”ボタンを押すことは、イベントであり、「今日は」と表示することは、この「“挨拶”ボタンを押す」というイベントに対応して、以下のイベントプロシージャ1が作動したことによる。

- イベントプロシージャ

VBのプロシージャは、プログラムの集合であり、従来のコンピュータ言語のサブルーチンや関数を、さらに統括的に扱えるように以下のような構造に体系化されていることに注目したい

プロシージャ

Sub	プロシージャ： 値を返さない（サブルーチン）
ジェネラル	プロシージャ： 全体に共通するコード記述
イベント	プロシージャ： イベント対応コード
関数	プロシージャ： 値を返す（関数）
プロパティ	プロシージャ： 独自のプロパティを開発

- バックボーンチャート

VBによる、オブジェクト指向プログラミングのための基礎的なデザイン方式としてバックボーンチャートの導入する。

図D－1－4は、前述のシンプルなアプリケーション“挨拶”プログラムのオブジェクト、プロパティ、プロシージャを抜き出して、オブジェクトを中心に、左プロパティ、右にプロシージャを配した図である。これを、背骨に見立て、バックボーンチャートと命名したものである。

VBによるアプリケーションは、オブジェクトを中心に、左プロパティ、右にプロシージャと、この3視点を押さえることで、基本的なプログラム要素は把握できることである。

モジュールごとの3視点を押さえて、アプリケーションは組み立てられていく。

シンプルなアプリケーション“挨拶”プログラムのオブジェクトは、2つである。

オブジェクトの組をオブジェクトコレクションという。

紙面の都合で改良版のバックボーンチャートは割愛した。

まとめ

アプリケーションの開発に際しては、オブジェクト指向の発展、オブジェクトプログラミングのためのコンポーネントプログラムの充実で、スピード開発が可能になってきた。

しかし、良きアプリケーションを開発するには

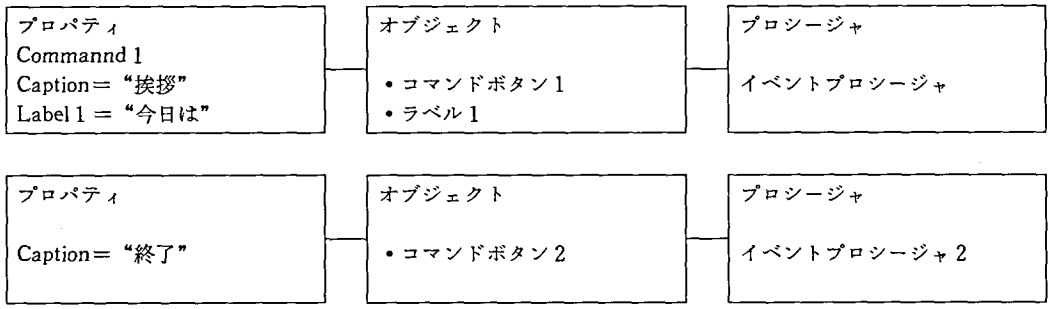
- ソフトウェアの実現状態の規定化
- 実現するためのシナリオライティング
- スパイラルモデルによる品質向上過程の手順化
- オブジェクトプログラミング最終準備としてのバックボーンチャート

等のオブジェクト思考体質形成が重要になってきたことに着目したい。

この小論は、プログラミング教育現場で、アプ

リケーション開発に際して、早い時期に、オブジェクト思考体質の基本を形成することが肝要であることを喚起するためにまとめられたものである。

(1999．9．27　受理)



図D－1－4　“挨拶”プログラムのバックボーンチャート

準備　潜っているモグラのイメージ
　　顔をだしているモグラのイメージ
　　頭を叩かれアカンベーをしているイメージ
　　を絵にして、ビットマップファイルに格納

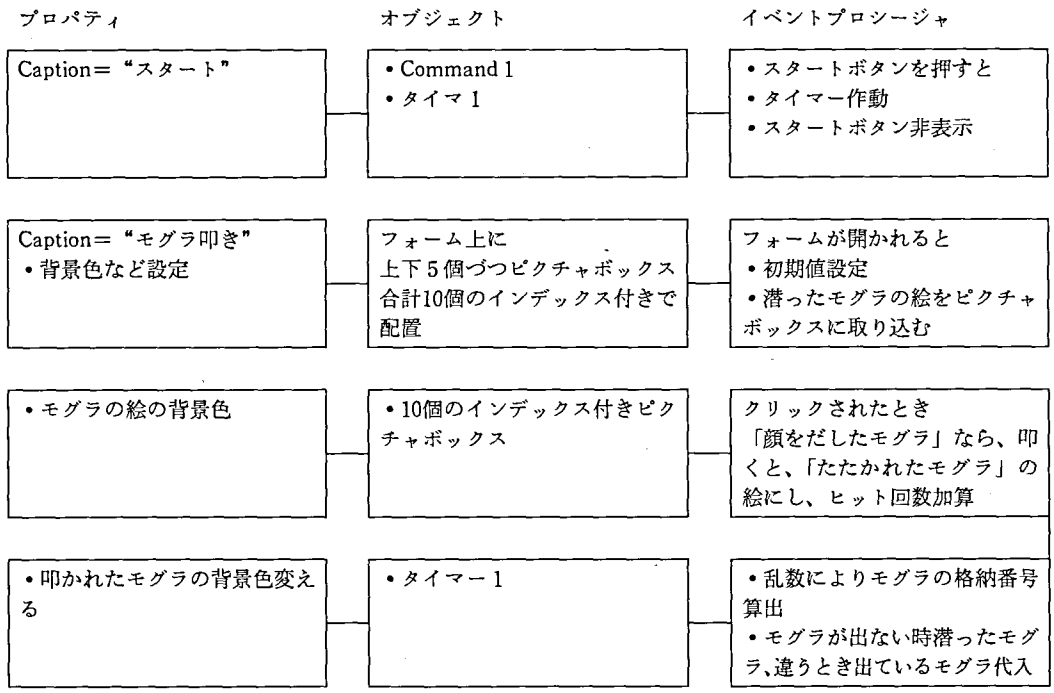


図　改良前の“モグラ叩き”のバックボーンチャート