データベースシステム 94−20
（1993．7．23）

オブジェクト指向アプリケーション開発環境 -ActivePage[1]-

辻村 敏，宮部 義幸，楠見 雄規，井上 信治，南方 郁夫

松下電器産業株式会社 情報システム研究所

オブジェクト指向アプリケーション開発環境 ActivePage は、松下電器産業（株）と米国 Gain Technology 社とで共同開発したアプリケーション開発のための基本ソフトウェアであり、テキスト、グラフィックス、イメージ、アニメーションなどのマルチメディアオブジェクトを扱うことができる。また、これらのオブジェクトデータを保存するために、オブジェクト指向データベースを利用している。ActivePage では、開発者がクラスの定義を行なう機能を解放しておらず、あらかじめ OA アプリケーションを作成するために必要と思われるクラスを用意している。このため、一般のオブジェクト指向言語に比べアプリケーションの開発が容易である。

Object-Oriented Application Development Software -ActivePage-

Satoshi Tujimura, Yoshiyuki Miyabe, Yuki Kusumi, Shinji Inoue, and Ikuo Minakata

Matsushita Electric Industrial Co., Ltd
Information Systems Research Laboratory

The object-oriented application development environmant software “ActivePage” was jointly developed with Matsushita Electric Industrial Co., Ltd. and Gain Technology Inc.,. This software combines high-quality texts, graphics, images and animations to create multimedia applications. Moreover, ActivePage includes the object-oriented database to store these multimedia objects.

In ActivePage, application developers need not define the class themselves. Instead, a high-quality function for developping an OA application is already defined. This makes easier for developers to develop a new application than a general object-oriented language.

[1] 商標

1 はじめに

近年、オブジェクト指向データベースに関する研究が盛んに行なわれている。しかし、研究の多くはデータベースの構築技術や検索技術に関するものであり、オブジェクト指向データベースの利用技術に関する研究／報告は少ない。

我々は米国 Gain Technology 社と共同でオブジェクト指向データベースを利用した、オブジェクト指向マルチメディアソフト開発環境 ActivePage を開発した。ActivePage はテキスト、グラフィックス、イメージ、オーディオ、アニメーションなどをオブジェクトとして扱うことができる UNIX [2] ワークステーション上の開発環境である。更に、これらをオブジェクト指向データベースに保存することができるため、パーシステントオブジェクトを扱うプログラミングが可能である。

ActivePage には以下のような特徴がある。

- 広義のドキュメントをベースにしたアプリケーション作成に最適なクラスを実装し，アプリケーション開発効率の極めて高い，オブジェクト指向の開発環境である．
- ユーザインタフェース，ネットワーク，文書作成，表示・印刷，データ管理，オーディオ，アニメーション等，広範な機能をカバーする開発環境である．
- 操作性，データの互換性が保たれた，特定業務アプリ，汎用アプリ，マルチメディアアプリなど，様々な種類のアプリケーションが容易に構築できる開発環境である．

本稿では、まず、ActivePage の概要について述べ、次に ActivePage の特徴、特に一般のオブジェクト指向言語との違いついて説明する。そして最後に ActivePage を用いて作成されたアプリケーションの例について述べる。

2 ActivePage の概要

ActivePage はテキスト、グラフィックス、イメージ、オーディオ、アニメーションなどをオブジェクトとして扱うことができる。これらのオブジェクトはすべて、ActivePage が持つオブジェクト指向データベースに格納される。これにより、パーシステントオブジェクトを扱うことが可能なオブジェクト指向プログラミング環境を提供している。

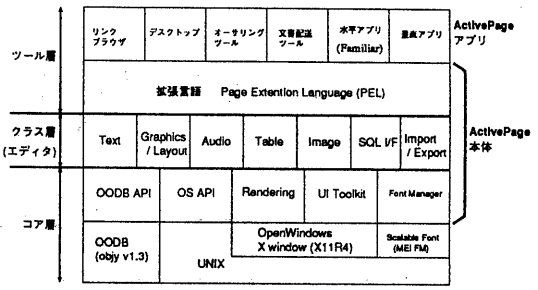

図 1: AvtivePage のソフトウェア構成

2.1 ActivePage の構成

ActivePage の構成を図 1 に示す。この図に基づいて、各部分の機能について述べる。

ツール層 ツール層は、後で述べる拡張言語 (PEL) インタプリタとそれを利用して開発された ActivePage アプリケーション群である。PEL は、クラス層で提供されるオブジェクトを操作するための言語であり、この言語を使用して、オブジェクトのメソッドに相当する「振舞い」を定義することができる。たとえば、電卓を例にとると、「ある数字のボタンが押されると、その数字をディスプレイパネルに表示する」という動作を PEL で記述し、数字のボタンに登録することで、ボタンの振舞いを定義することができる。

クラス層 クラス層は、ActivePage が提供するオブジェクト群を定義している層である。ActivePage が提供するテキスト、グラフィックスなどの各メディアに対して、それぞれクラスが定義されている。

コア層 コア層は、ActivePage が利用する UNIX オペレーティングシステム、オブジェクト指向データベース、X Window System[3]などの外部資源とのインタフェースを提供する層である。将来の移植性

[2] UNIX Systems Laboratories, Inc. の登録商標

[3] MIT の商標

を考え、外部資源をできるかぎり仮想化したインタフェースとなっている。

ActivePage で使用するオブジェクトおよび、そのオブジェクトに定義されたプログラムはすべて、コア層の OODB API を通して、オブジェクト指向データベースに格納される。

2.2 データ管理機構

ActivePage では、データの管理機構にオブジェクト指向データベースである Objectivity/DB[4]を使用している。ActivePage 上で作成されるデータはすべてこのデータベースに保存される。そのため、テキストのほかにグラフィックス、イメージ、オーディオなどのマルチメディアデータを扱うことが可能となる。

データ管理のためにオブジェクト指向データベースを用いることにより、UNIX のファイルシステムを使用するのと比べて以下の様な利点を得ることができる。

- **複雑な構造を持つ大容量データに対する高速なアクセスが可能である。**
  オブジェクトデータの管理を、すべてオブジェクト指向データベースに任せることができるため、複雑な構造を持つオブジェクトデータを大量に作成／保存しても高速なアクセスを行なうことが可能となる。

- **アプリケーション及びデータのバージョン管理が可能である。**
  特別なバージョン管理機構を作成しなくても、オブジェクト指向データベースの機能を使ったデータのバージョン管理が可能となる。

- **グループ環境におけるデータへの多彩なアクセスコントロールが可能である。**
  読みだし・書き込み許可のほかに文書の存在を知るための許可や削除許可など 8 種類の許可をページ単位で設定することができる。このため、一つの文書に対して、特定のページを特定の人だけが読むことができるというような設定が可能となる。

- **データ書き込み時のコンフリクトを防ぐ排他制御が可能である。**
  一つの文書を複数の人間で作成する場合、元文書に対して複数の人間が同時に編集を行うと、先に文書を保存した人の編集内容が、後から保存した人の文書で上書きされてしまい、消えてしまう事が起こる。これを防ぐために、CheckIn/CheckOut 機構を備えており、同時に一人の人間しか文書の編集/ 書き込みを行えないような排他制御が可能である。

- **ネットワーク上に分散したデータの統一的な管理が可能である。**
  Objectivity/DB の機能により、複数の WS に分散してデータベースファイルを置き、それらを統合した仮想データベースを作成することが可能である。これにより、共通に利用するアプリケーションやデータをサーバマシンに置き、個人のデータを各自のマシンに置くという構成が可能となる。

2.3 拡張言語 PEL

アプリケーションやデータは、クラス層で定義されている各種のオブジェクトの組合せによって作成されるが、これらのオブジェクトを操作するために拡張言語 PEL(activePage Extention Language) が用意されている。

PEL には次のような特徴がある。

約 20 種類の構文　PEL は、自然言語（英語）に近い、簡単な構文で構成されている。また、構文規則も約 20 種と少数である。このため、馴染みやすく、言語の習得が容易である。また、プログラミング経験のないユーザでも、短期間で使用できるようになる。

約 900 種類の関数群　簡単な構文規則に加えて、オブジェクトを操作するための豊富な関数が用意されている。このため、様々なニーズに対応するアプリケーションの開発が可能であり、ハイレベルなプログラマの要求にも答えることができる。

高機能なエディタ群　ActivePage のクラス層では、ボタンやリストなどの高機能なオブジェクト群や編

[4] Objectivity, Inc. の登録商標

集機能を持ったオブジェクト群を提供している。したがって、PEL を使って、これらを利用することにより、簡単にアプリケーションを開発することができる。

ハイパーメディア機能　すべてのオブジェクトに対して、PEL による動作を定義することができる。この特長を利用すれば、容易にハイパーメディア機能を有するアプリケーションの開発が可能である。

インタプリタにより動作　ActivePage では PEL プログラムをインタプリタによって、解釈実行する。このため、インタラクティブなアプリケーション開発が可能である。

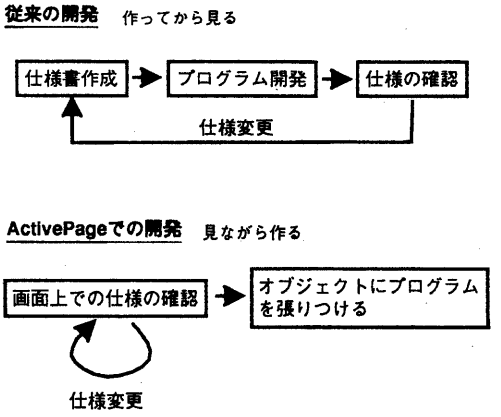

図 2: アプリケーション開発サイクル

2.4　アプリケーション開発ツール

アプリケーション開発を行なう際に、オブジェクトを操作するためのものとして、拡張言語 PEL の他に、対話的なアプリケーション開発ツールを提供している。従来のアプリケーション開発では、仕様書の作成、コーディング、実行という行程を行っていた。この方法では、仕様の変更が発生した場合、仕様書作成の段階まで遡って、作業をやり直す必要があった。

ActivePage では対話的に表示画面を作成することができる。また、ここで作成される画面がそのままのイメージで、グラフィカルユーザインターフェイスとして使用される。このため、仕様変更に対する手戻りが少なく、開発工数を大幅に削減することが可能である。

図 2 は従来のアプリケーション開発サイクルと ActivePage による開発サイクルとの違いを示した図である。

3　ActivePage におけるオブジェクト指向

3.1　オブジェクト指向アーキテクチャ

ActivePage ではオブジェクト指向のアーキテクチャを採用している。オブジェクト指向とは、問題の抽象化、モデル化を行うためにでてきた概念である。そこでは、解決しようとしている問題のなかで、物理的実態と概念的実態とを明らかにし、それをオブジェクトというもので表現する。

オブジェクトとは実態の持つデータと、そのデータを処理するための手続きの両方をもったものである。このため、プログラムのモジュール性が極めて高いものとなり、プログラムの保守性が良いと言われている。また、モジュール化されているため、オブジェクトの再利用が容易になり、開発の効率を高めることとなる。

一般のオブジェクト指向プログラミング環境では、本質が同じであると思われるオブジェクトをクラスとしてまとめ、クラス間の階層構造を決定するということが行われる。そして、この階層構造により、クラス間の属性やメソッド（データを処理するための手続き）の継承が行われる。したがって、クラスをどのように分類し階層化するかが、システムの持つ機能や開発効率、機能の拡張性の鍵をにぎっている。これは、適切なクラス階層を設計することが、オブジェクト指向プログラミングにおいて最も重要な作業であることを意味している。しかしながら、この作業を適切に行なうためには、オブジェクト指向に関する深い理解が必要であり、一般的には困難な場合が多い。

ActivePage では広い意味での OA 処理・文書処理などに適したクラスがあらかじめ実装されており、高機能なオブジェクトが用意されている。したがっ

て、アプリケーション開発者が、それぞれがもっている具体的な問題を解決するためには、既存のクラス定義に従ったオブジェクトを持ちいて、その問題を表現するだけで良い。これは、オブジェクト指向プログラミングにおいて、最も難しいクラス階層の設計を行う必要がないということであり、開発者の負担を軽くすることになる。

たとえば、ボタンのためのクラスを作る場合、ボタンは位置と大きさを持っているので、グラフィックスクラスのサブクラスとして作ることが考えられる。一方、ボタンにはボタンラベルが記入できるため、テキストを表示する機能が必要となり、テキストを扱うクラスのサブクラスである方が便利かも知れない。この他、ボタンはマウスボタンが押されたことを検知するメカニズムや色を反転させるメカニズムが必要であり、これらを上位クラスの機能として実現するのか、ボタンクラスのメソッドとして定義するのかが問題となる。

ActivePage では、クラス層にあらかじめボタンと言うクラスが定義されており、PEL を用いてこのクラスのオブジェクトを呼び出すだけで、ボタンの機能を使うことができる。このほか、禁則処理や組版機能を備えたテキストクラスや他のページオブジェクトをビューするためのビューワクラスなど、豊富なクラス定義がなされている。

図 3 に ActivePage においてあらかじめ定義されているオブジェクトの階層を示す。

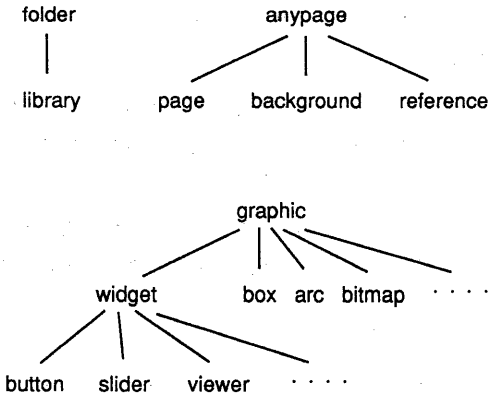

図 3: オブジェクトの階層関係

3.2 オブジェクトの親子関係

ActivePage のオブジェクトには、クラスの階層構造のほかにオブジェクト間の親子関係が存在する。この親子関係を図 4 に示す。

アプリケーション開発者にとって、オブジェクト間の関係はクラスの階層構造ではなく、この親子関係によって認識される。これは開発者がクラスを意識する必要がないことを意味している。実際、ActivePage を利用する上で開発者が理解しておかなければならないことは、オブジェクト自身が手続きを持っているということと、その手続きを呼び出すためにはメッセージをオブジェクトに対して送るということ、そしてオブジェクトの親子関係によってメッセージの配送が行なわれるということである。

ActivePage では、アプリケーションもデータも

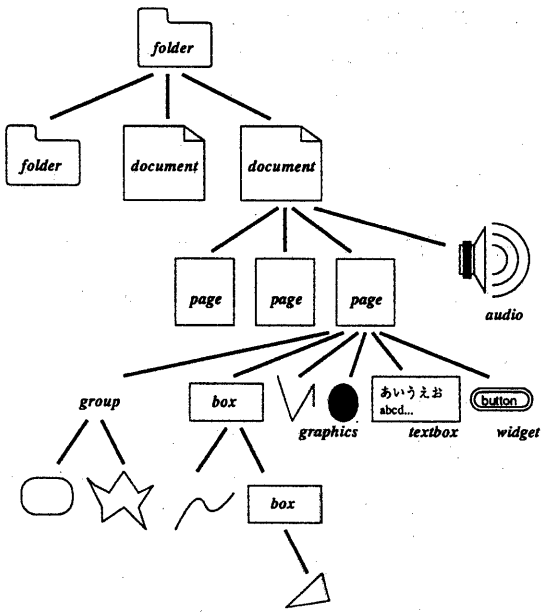

図 4: オブジェクトの親子関係

すべて文書という概念の上に作成される。文書の子供には複数のページが存在する。このページがアプリケーションにおける表示画面（グラフィックユーザインターフェイス：GUI）となる。

以下にアプリケーションの開発手順の概略を説明する。

1. アプリケーションの本体となる文書を作成し、その文書に名前をつける。この名前がアプリケーション名となる。
2. 文書にページを作成する。複数のパネルを使用するようなアプリケーションではパネルの数だけページを作成する。
   このページがアプリケーションのGUIとなる。
3. ボタンやグラフィックなどのオブジェクトをページ上に配置することにより、アプリケーションのためのユーザインターフェイスを作成する。これらのオブジェクトはページの子供となる。
4. 個々のオブジェクトやページに対して、処理（ハンドラ）を記述したプログラム（スクリプト）を設定する。
5. 文書のスクリプトとして、アプリケーションの起動／終了処理を定義する。この中で、アプリケーションが起動された時に初期画面をポップアップする処理も記述する。

以上のように、ActivePageでは、オブジェクトの存在する位置によって、親子関係が構成される。

3.3 仮想クラスとしてのオブジェクト

拡張言語PELによって、扱われるオブジェクトは、図3に示した個々のクラスのインスタンスである。

一般のオブジェクト指向言語では、インスタンスの属性を表すインスタンス変数とオブジェクトの振舞いを表すメソッドが存在する。このインスタンス変数やメソッドはクラス設計の時点でクラスの定義として作成されているため、これらを追加／変更するためには、クラス定義を変更する必要がある。

ActivePageのオブジェクトには、オブジェクトの属性を表すプロパティとオブジェクトの振舞いを表すスクリプトを定義することができる。これにより、仮想的なクラスを作成することが可能である。

3.3.1 プロパティ

プロパティにはオブジェクトに最初から定義されいるシステムプロパティとユーザが自由に定義できるユーザプロパティがある。このシステムプロパティがインスタンス変数にあたる。システムプロパティの値を変更することにより、オブジェクトの位置や大きさなどオブジェクトの外観を変更したり、オブジェクトが持つ処理手続きで扱うデータを変更したりすることができる。

一方、ユーザプロパティはオブジェクトごとに自由に定義することができる。これは、クラス定義を変更することなしオブジェクトを拡張できることを意味している。

例えば、複数のボタンのうち常に一つのボタンだけがONの状態になるようなボタングループを作成する場合、個々のボタンに対して、どのボタングループに属しているかの情報を持つためのプロパティを追加することにより、元のボタンオブジェクトを拡張したボタングループオブジェクトが作成できる。

3.3.2 スクリプト

ActivePageのオブジェクトには、プロパティの他に、オブジェクトごとのメソッドの集まりに相当するスクリプトを定義することができる。スクリプトはメッセージに対する処理を記述したハンドラの集まりである。ハンドラはクラスに定義されるメソッドととらえることができる。これはオブジェクトが独自のメソッドを持つことができることを意味している。

先のボタングループオブジェクトの例では、ボタンが押された時にそのボタンが属しているボタングループをプロパティの値から取り出し、同じボタングループに属しているボタンに対してOFFに設定するようなメッセージを送るというハンドラを定義すれば良い。

このように、プロパティやスクリプトは、同じクラスのインスタンスであるオブジェクトを、クラス定義を変更せずに別のオブジェクトとして拡張することができる枠組を提供している。これは、システムで定義されたクラスの上にオブジェクトのプロパティやスクリプトを定義することにより、仮想的な

クラスを作成することが可能であるということである。

言い替えるならば、ActivePageにおけるオブジェクトはインスタンスであると同時にサブクラスを持たない仮想クラスであるととらえることができる。この仮想クラスの機能が、開発者にクラスの作成を解放していないActivePageにおいて、既存のクラスで定義されている機能よりもさらに複雑な機能を、アプリケーションに付加することを可能にしている。

3.3.3 仮想クラスを利用したデータカプセル

以下に仮想クラスの利用例について述べる。

リレーショナルデータベース(RDB)で管理されている大量の数値データを不定量扱うようなアプリケーションの場合、データベースから取り出したデータをアプリケーション内でどう管理するかが問題となる。

C言語でこれを行う場合、データを保存するためのメモリー領域を確保し、その中で、開発者自身がデータの長さや区切りを意識しなければならない。また、そのデータに対してアクセスする関数などはデータがどの様に管理されているかを知っていなければならない。このため、扱うデータのフォーマットが少しでも変わるとプログラムの広い部分に影響が及ぶ。そこで我々は、ActivePageにおけるグラフィックオブジェクトをデータカプセルとして利用した。

RDBに保存されているデータをSQLを使って検索し、取り出す場合を考える。

取り出されるデータの数はRDBに保存されている内容によって、変化する。そこで、取り出すデータの種類によって、そのデータを保持するためのオブジェクトをあらかじめ作成しておき、このオブジェクトのプロパティとしてRDBから取り出したデータを保存する。オブジェクトには取り出したデータ群を処理するためのハンドラを用意しておき、外部からデータを利用する場合にはオブジェクトに対してメッセージを送ることにより処理が行われるようにする。

図5では、RDBから取り出した、データを"DATA1"という名前のデータカプセルに保存している。"DATA1"ではそのデータカプセルに保存さ

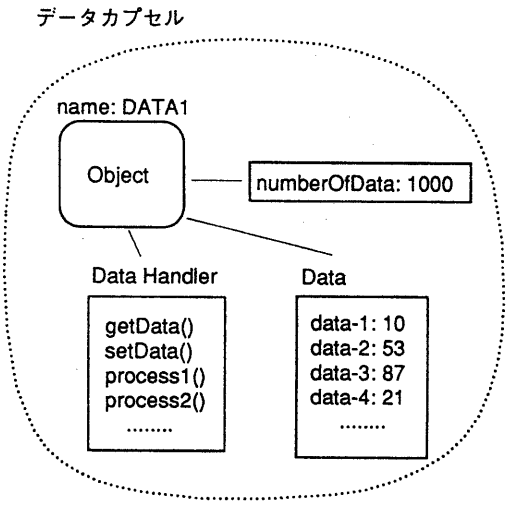

図5: データカプセルとしてオブジェクト

れているデータの数を"numberOfData"というプロパティの値として持ち、データそのものの値は"data-XXX"というプロパティの値として保存している。また、データを取り出したり、値を書き換えたりするためのハンドラが"Data Handler"として定義されている。

このように、データの数がわからないデータ群を、一つのオブジェクトとして取り扱うことが可能となる。また、データとそのデータを処理するためのプログラムが一箇所に管理されるため、データのフォーマットを変更した場合も、外部に対する影響が少ない。

以上のように、ActivePageでは、高機能なクラスが定義されているため、多くの部分では用意されているオブジェクトを組み合わせることにより、機能を実現できるが、さらに、複雑な機能を実現する枠組として、プロパティやスクリプトがある。つまり、ユーザプロパティやスクリプトを利用することにより、簡易的なクラス作成を行うことができ、あらかじめシステムで定義されているクラスの機能以上の機能を持ったオブジェクトの作成が可能である。

3.4 メッセージの配送

先に述べたように、ActivePageでは、一般のオブジェクト指向言語と違い、メッセージを処理するた

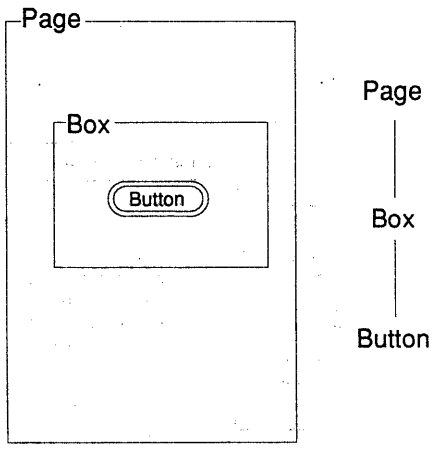

図 6: オブジェクトの親子関係の例

めのメソッドの継承関係も、クラス階層関係によって制御されるのではなく、オブジェクトの親子関係によって制御される。

つまり、ページの上に図 6 のようなオブジェクトが存在する場合、ボタンに対して送られたメッセージがボタンで処理されなかった場合、ボタンの上位クラスであるヴィジットに送られるのではなく、ボタンが画面上で含まれるボックス（このボックスがボタンの親オブジェクトになる）に送られるのである。更に、そこでも処理されなかった場合は、ボックスが含まれるページ、つまりボックスの親オブジェクトへメッセージが送られ、処理されることになる。

一般のオブジェクト指向言語では、メソッドの継承がクラスの階層関係によってなされる。また、この階層関係は画面に表示されるオブジェクトの関係と無縁なため、クラスの階層関係を把握している開発者でなければ、あるオブジェクトで処理されなかったメッセージが、どう配送されていくかを判断できない。

一方、ActivePage では、メッセージの配送関係が視覚的な関係により判断できるので、アプリケーション開発者はメッセージの配送経路を容易に知ることができる。そのため、メッセージを処理するためのハンドラをどのオブジェクトに定義するかの決定が容易となる。

クラスがあらかじめ定義されていることの他に、この意味においても、ActivePage ではオブジェクトのクラスを意識する必要がない。

3.5 アプリケーションとデータ

一般のオブジェクト指向言語やマルチメディアソフト開発環境の場合、データ管理という概念を明確に持っているものはほとんどない。このため、アプリケーションで扱われるデータのフォーマットやファイルの管理はアプリケーション開発者にまかされることになる。

ActivePage では、アプリケーションもデータもすべて文書オブジェクトとして管理されるため、データ管理をシステムにまかせることができる。

ワープロのような非定型データを扱うアプリケーションの作成を考えた場合、データとなる文書のファイル構造や管理方法を決定することは簡単ではない。

ActivePage では、データとなる文書を構成する要素はすべてオブジェクトであり、これはすべてアプリケーション同様 OODB に保存／管理されるため、データ管理機構の作成を容易に行なうことが可能である。

さらに、アプリケーションとデータの区別を明確にする必要がないため、データ自身にも処理プログラムを定義することができる。これはアプリケーションの機能の一部がデータにおいて実現することができることを意味しており、図 7 のように、機能を持ったデータの雛型を数種類作ることにより、ひとつのアプリケーションで、複数の違った処理を行なわせることが可能となる。

4 オフィスシステムへの応用

この章では、ActivePage を実際のオフィスシステムに応用した例を紹介する。

4.1 帳票データ入出力システム

従来メインフレームで行われてきた人事、経理システムなどいわゆるオフィスシステムを、ワークステーションで行うダウンサイジングが最近盛んである。この流れの中で、キャラクタベースで行われていたものが、ウィンドウシステムを使ったグラ

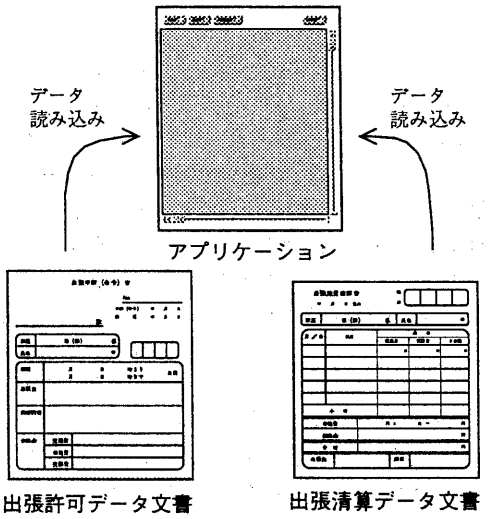

図 7: アプリケーションと雛型データの例

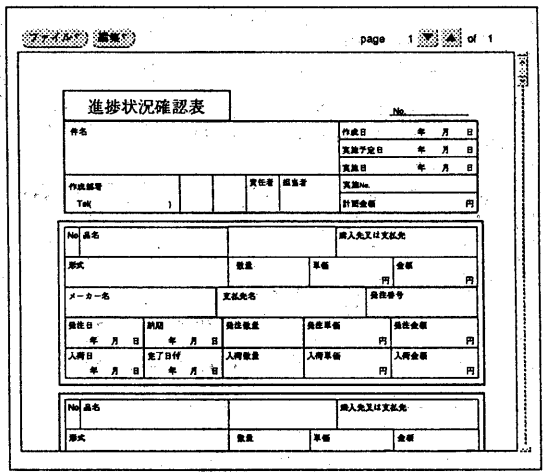

図 8: 帳票処理アプリケーションの例

フィックユーザインターフェイスを使用するシステムへと変わってきている。オブジェクト指向はこのグラフィックユーザインターフェイスの構築に適していると言われている。

ActivePage では、データ管理機構としてのオブジェクト指向データベースのほかに、リレーショナルデータベースとのインターフェイスを備えているため、これまで蓄積されてきたデータを無駄にすることなく、オフィスシステムの構築を行うことが可能である。

ActivePage のオフィスシステムへの応用例の一つとして、リレーショナルデータベースに必要なデータを格納し、ActivePage 上のアプリケーションでこれらのデータを取り出し、グラフィカルに表示したり、加工して、再び格納するといったことを行うものがある。

また、これらのアプリケーションでは最終的にデータを書き込んだ帳票を印刷する必要がある。ActivePage では、ディスプレイ、プリンタに対して共通の描画モデルを有しているため、画面上に表示されているページをそのまま印刷することが可能である。したがって、アプリケーション開発者は印刷部の開発を行う必要がない。これは、従来から使われてきた帳票を、データ文書のページとして開発するだけで、帳票の印刷を行うことができ、更に、アプリケーションを使用するユーザにとっても、従来の帳票と同じ形のインターフェイスでデータの入力が可能となり、新しいシステムへの移行が容易であると言うことである。

この点も、ActivePage が単なるユーザインターフェイス作成ツールと大きく違う所である。

図 8 にオフィスアプリケーションの一例の画面を示す。

4.2 部品ツールの組合せによる OA システム

ActivePage で使用されるデータの内容はすべてクラス層で定義されているため、ActivePage 上に作成されたアプリケーションでは自然にデータの互換性が保証されている。このため、個々のアプリケーションを別々に開発してもその上で扱われるテキストなどのオブジェクト群の移動や複写などのデータ交換が可能である。

また、View Link を作成するための viewer オブジェクトを用いると任意のツールで作成された文書（データ）の任意のページを view することが可能である。

このデータの互換性を利用した例として、グラフ作成や表作成、文書作成など小さな単位でのタスクを行うをアプリケーションを作成し、必要に応じて、これらを組み合わせて使用する OA システムが作成されている。

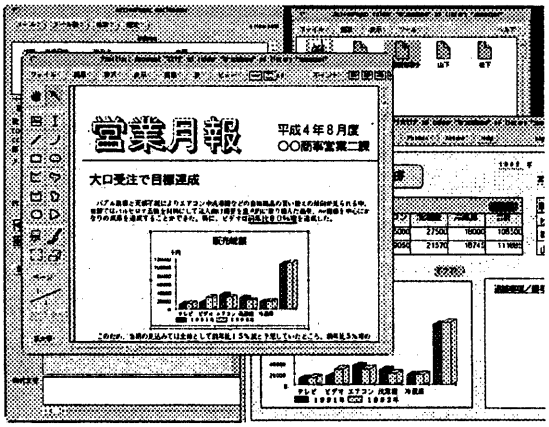

図 9: OA アプリケーション環境の例

このような小さな単位でのタスクを行うをアプリケーションを部品ツールと呼んでいる。部品ツールを組み合わせて OA システムを構築することにより、部品ツールの再利用による、開発工数の削減と、ユーザインターフェイスの統一が実現できる。

現在我々は ActivePage 上の部品ツールとして、文書作成ツール、グラフ作成ツール、計算式を扱える表作成ツール、文書配送ツール、入出力処理などの共通パネルなどの開発を行なっている。

これらを組み合わせた OA アプリケーション環境の例を図 9 に示す。

この図では画面中央の文書作成ツールで編集している「営業月報」の中から、右下の文書の中の「販売総額」のグラフを view している。また、この「販売総額」のグラフは、別に起動されているグラフ作成ツールによって、描画されたものである。

5 おわりに

本稿では、ActivePage の概要について述べ、次に ActivePage の特徴について説明した。そして最後に、ActivePage を使って作成したアプリケーションの例について述べた。このように ActivePage を用いると開発者は、クラス設計という最も難しい作業を行わずに、オブジェクト指向プログラミングの利点を利用することができる。

実際に ActivePage 上で開発されている２５のアプリケーションの開発者に調査を行った所、従来と比べて３倍から１０倍の開発効率の向上があったという結果が報告されている。

これは、1) ダイレクトマニピュレーションによるユーザインターフェイスの開発効率のよさ、2) 仕様変更に対して速やかかつ柔軟な対応が可能であること、3) あらかじめ機能の高いクラスがが定義されており、クラス設計が不要なこと、4) メッセージの配送がオブジェクトの親子関係によって決まり、配送経路が容易に判断できること、などの理由による。このことは、作成されたアプリケーションのユーザにとっても満足度の高いアプリケーションの作成が可能ということである。また、できあがったアプリケーションにおいて、操作性やデータ構造の統一などが自然に行われる事も ActivePage の特長である。

今後、動画や自然画などを扱う機能やアプリケーション開発ツールなどの開発、実行速度向上などが課題である。また、実際のアプリケーション開発を通して、開発手法に関するノウハウの蓄積を行っていく予定である。

参考文献

[1] G.Booch : Object oriented design with applications, Benjamin/Cummings (1991)

[2] A.Goldberg and D.Robson : Smalltalk-80 言語詳解 (1987)

[3] 田中 克巳 : オブジェクト指向データベースシステム- その背景と概念, bit 20, No.6, pp687-694 (1988)

[4] 宮部 義幸他 : オブジェクト指向アプリケーション開発環境 “ActivePage”, National Technical Report Vol.39 No.1 (1993)

[5] 堤竹 秀行他 : オブジェクトデータベースによるマルチメディア統合化, 情報処理学会技術研究会報告 DBS 86-4 (1991)