ソフトウェア工学 96−23
（1994. 1. 21）

オブジェクト指向プログラミングによるデスクトップの開発

是津 耕司

日本IBM 先進ソフトウエア製品開発

Abstract

本稿では、ウィンドウシステムにもとづくGUIアプリケーションの1つであるデスクトップの開発を通して、オブジェクト指向プログラミングを用いたアプリケーション開発について考察する。

オブジェクト指向プログラミングによる開発では、従来のプログラミング・パラダイムに比べて、アプリケーション全体の見通しが非常によくなり、より柔軟性に富んだものになること、デザインとプログラム・コードの間の隔たりが少なくなるなるなどの特徴がうかがえる。

The study of developing desktop application by object oriented programming

Kohji Zettsu

Advanced S/W Development, IBM Japan
1623-14, Shimotsuruma, Yamato-shi, Kanagawa-ken, 242, Japan

Abstract

This paper explains the advantage of developing the GUI application - 'desktop' application environment by object-oriented programming paradigm.

In object-oriented programming paradigm, we find out that the application structure becomes more clear and flexible, and also the design specification and programming code get more closer to each other.

1　はじめに

近年、PC・WSにおけるウィンドウシステムの発展により、GUI(Graphical User Interface)を中心としたアプリケーションが数多く登場してきている。これらGUIを駆使したアプリケーションの大きな利点は、視覚に訴えることにより、ユーザーがアプリケーションと直観的なやりとりを行えることである。

デスクトップと呼ばれるアプリケーション環境は、こうしたGUIアプリケーションのなかの1つである。デスクトップでは、様々なプログラムやデータが1つのウィンドウ上に並べられており、ユーザーはこれらを自由に選択したり起動したり移動したりして自分の望む処理を遂行することができる。

これらデスクトップ上のプログラムやデータセットは、それぞれが固有なデータや手続きを持ち、これらはユーザーの操作によって自由に変化する。このような環境を、1つのアプリケーション・グローバルなデータ領域とそれに対する手続きを定義した従来のプログラミング・パラダイムで実現しようとすると、データと手続きが複雑に絡み合い、アプリケーション全体が繁雑で柔軟性のないものになってしまいがちであった。

そこで本稿では、新しいプログラミング・パラダイムとして注目されているオブジェクト指向プログラミングを用いて、従来のプログラミング・パラダイムにあった複雑さを取り除き、よりわかりやすく柔軟性に富んだデスクトップをいかにして構築していったらよいかを考察する。

2　オブジェクト指向プログラミングとは

オブジェクト指向とは、人間の思考過程における「まとまり」すなわち抽象化された概念をオブジェクトとして表現し、問題をオブジェクトの関係として記述するための手法である。従って、従来のノイマン・ロジックを基調としたプログラミング・パラダイムに見られた、逐次的な一連の手続きによる問題解決手法に比べ、より人間の思考過程に近い形で問題を表現することができる。

オブジェクト指向プログラミングでは、データと手続きを1つにまとめてオブジェクトとして表現する。オブジェクトは、私的なデータや手続きを内部に隠蔽し外部からの不用意なアクセスを防ぐとともに、これら内部情報に対する操作を外部に公開することでオブジェクトへのインタフェースを提供する。この情報隠蔽の仕組みにより、オブジェクトはその仮想的な実体と現実の実装とを分離することができる。従って、問題は仮想的な実体で表現し、実行モジュールは、問題の表現を変えることなく、それぞれの実装系に合わせて作ることができる。

さらに、様々なオブジェクトに共通なデータや手続きを1つのオブジェクト・クラスにまとめ、このクラスを基底として導出したクラスの中で各オブジェクトに固有なデータや手続きを再定義することによって、様々な特徴を持ったオブジェクトを容易に作成することができる。

3　デスクトップの構成

今回開発したデスクトップを以下に簡単に説明しよう。

デスクトップ上での主な作業は、様々な文書データの登録・管理である。図1に実際の様子を示す。

デスクトップは大きく3つの構成要素に分けられる。

アイコン

文書を表す。アイコンには、

- 文書データ
- 管理項目データ
- 管理項目の編集処理

が含まれる。

アイコンボックス
複数のアイコンをまとめてグループ化する。アイコンをアイコンボックス間でドラッグ&ドロップすることにより、グループに含むアイコンを変更することができる。

ベース
デスクトップの最も基盤になる部分。

これらの構成要素の中で、アイコンとアイコンボックスをオブジェクトとして表現し、ベースの上でデスクトップ環境を構築する。

4 オブジェクトの定義

ここでは、デスクトップを構成するオブジェクトについて、それぞれのオブジェクトに必要なデータと手続きを簡単に定義する。

4.1 親子関係

アイコンとアイコンボックスの間にオブジェクトの親子関係を定義する。

アイコンはそれが含まれるアイコンボックスの子であり、ここではこのアイコンボックスをアイコンの**親アイコンボックス**と呼ぶ。親アイコンボックスは自身に含まれるアイコンを管理する。

親と子は一対多の関係であり、アイコンは生成から消滅までの間、必ずどれかのアイコンボックスの子として存在する。オブジェクトの親子関係はアイコンのアイコンボックスへの追加や削除の際に変化する。

4.2 アイコン

4.2.1 構成に関する定義

- アイコンを構成するウィンドウの情報 (face, label...)
- 各構成要素のアイコン上での位置 (face_geometry, label_geometry...)
- 表面に貼るビットマップやラベル
- アイコンの生成の手続き (new())
- アイコンの消滅の手続き (delete())
- 再描画手続き (redraw())

アイコンは論理的な1つのオブジェクトとしての表現であり、実際はウィンドウやビットマップといったより低レベルの要素がいくつかまとまって構成されている。これらの構成要素はオブジェクト内ですべて管理され、オブジェクト外部からは隠蔽されている。

4.2.2 操作に関する定義

選択

- 選択状態
- 選択と選択解除の手続き (select(), deselect())

移動

- アイコンの現在位置 (geometry)
- 移動の手続き (move())

起動

- 起動状態
- アイコンに定義された処理の起動と起動解除の手続き (activate(), deactivate())

アイコンは自分に対する操作を検出し判断して、それぞれの操作にたいする処理を行なう。

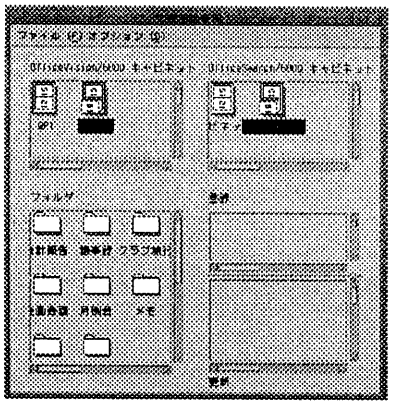

図 1: デスクトップ

4.2.3 親子関係に関する定義

- 親アイコンボックス (parent)
- 親アイコンボックスの変更手続き (change_parent())

アイコンボックスとの親子関係を表す。

4.3 アイコンボックス

4.3.1 構成に関する定義

アイコンと同様に、アイコンボックスを構成するウィンドウや、生成・消滅に関する手続きなどを定義する。

4.4 アイコンの管理に関する定義

- 現在含まれているアイコンのリスト (entry_list)
- 現在選択されているアイコンのリスト (select_list)
- 現在起動されているアイコンのリスト (activate_list)
- アイコンボックスへの追加の手続き (add())
- アイコンボックスからの削除の手続き (remove())
- アイコンの再配置 (align())
- アイコンの全選択と全選択解除の手続き (select_all(), deselect_all())
- アイコンが選択または選択解除されたことの通知を受ける手続き (notify_select(), notify_deselect())
- アイコンが起動または起動解除されたことの通知を受ける手続き (notify_activate(), notify_deactivate())

アイコンは自分が選択されたり起動されたりすると、そのことを親アイコンボックスに通知する (親アイコンボックスの手続きを呼ぶ)。これによって、アイコンボックスは現在自分に含まれているアイコンの状態を管理することができる。

4.5 オブジェクト・クラスとしての定義

先に挙げたアイコンやアイコンボックスの定義を基底クラスとし、これらから導出されたオブジェクトでそれぞれに固有な部分を再定義することにより、さまざまな特徴を持ったアイコンやアイコンボックスを容易に作成することができる。

4.5.1 アイコンの再定義

それぞれのアイコンに定義されたデータやプログラムに関する固有の情報や手続きを導出されたオブジェクトで定義する。

例えば、今回のデスクトップでは、

- 文章・イメージデータ
- 管理項目
- 管理項目の編集処理

といったものである。

4.5.2 アイコンボックスの再定義

- アイコンのグループ化のポリシー
  追加や削除の手続きがこの情報をもとに、アイコンを選択的に追加・削除することでアイコンを目的にしたがってグループ化することができる。

5 デスクトップの構築

デスクトップは、アイコンやアイコンボックスといったオブジェクトを組み合わせて構築される。ここではその一例として、ドラッグ&ドロップによるアイコンのアイコンボックス間の移動処理を見てみよう。

5.1 ドラッグ

ドラッグは、アイコン上でマウス・ボタンを押したままポインタが動き始めることによって開始される。アイコンはこのイベントを検出すると自分がドラッグされると判断し、ドラッグの準備を行なう。

ドラッグ中、アイコンはポインタの位置までアイコンの構成要素すべてを移動させる(ICON.move())。結果としてアイコン上でのポインタの位置を変えずにアイコンを移動することになり、逆に見れば、ポインタの動きに合わせてアイコンが移動したことになる。

ドラッグ中にマウス・ボタンが離されると、アイコンはドラッグが終了したと判断し、続いてドロップ処理に入る。

この様子を図2に示す。

5.2 ドロップ

まず始めにアイコンは、自分がドロップされた位置に何かオブジェクトがあるか調べる。

アイコンボックスが見つかると、アイコンはアイコンボックスへの自分自身の追加手続きを呼びだす(ICONBOX.add())。アイコンを受け入れる側のアイコンボックスは、このアイコンの親アイコンボックスを自分に設定し(ICON.change_parent())、追加処理を行う。また同時に、アイコンは以前に属していた親アイコンボックスに対し自分自身のアイコンボックスからの削除を通知する。これを受けて、親アイコンボックスはこのアイコンをエントリーから削除する(ICONBOX.remove())。

また、ドロップされた位置になにもオブジェクトが見つからなかった場合、見つかってもそのオブジェクトが自分を入力として受けとらない場合には、ドロップされた位置にとどまるだけで、その先の処理は行なわれない。

この様子を図3に示す。

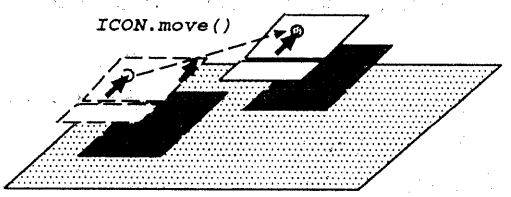

図 2: ドラッグ

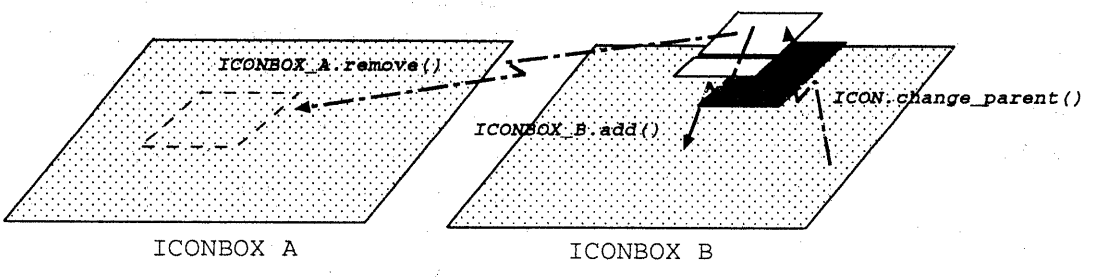

図 3: ドロップ

6 まとめ

本稿では、オブジェクト指向プログラミングによりデスクトップの構築を実際に行なった。開発を通じて得られた特徴をまとめると以下のようになる。

オブジェクト指向プログラミングによる開発の最も優れた点は、アプリケーション・デザインと実際のプログラミングとの間の隔たりが少ないことである。

デスクトップの開発においては、まず目的のデスクトップからオブジェクトになり得るものを探し出す。まとめて管理されるべきデータの集まりとそれらに対する手続きがオブジェクトになる。例えばアイコンなどは、それ自身がある特定のデータやプログラムを表すものであり、デスクトップ上のオブジェクトとして非常にわかりやすいものである。

オブジェクトの定義では、オブジェクトが持つデータと手続きを実際に定義する。これらには、オブジェクト内での閉じた処理のために内部に隠蔽すべきものと、外部に対してオブジェクトにアクセスするためのインタフェースとして公開すべきものがある。

プログラミングの観点から見た特徴として、デザイン・スペックとプログラムのコード（特にヘッダーファイル内のクラス定義）がオブジェクトの定義のレベルではほぼ等しい言える。このことは、特にオブジェクトの外部に対するインタフェースの定義に関していうことができる。例えば、4「オブジェクトの定義」で挙げたオブジェクトの定義は、それぞれがそのままオブジェクト内のデータあるいは手続きとして実装することができる。この特徴は、オブジェクト指向プログラミングが人間の直観的な思考をよりよく反映し、『考えながら書く』ことができるプログラミングであるということを表している。

さらにもう1つ、オブジェクト指向プログラ

ミングによる開発の優れた点は、オブジェクトが提供する機能のインタフェースの組合せによって、目的とするアプリケーションが構築できることである。特にデスクトップのような、機能やデータの追加・削除が任意に行なわれたり、提供された機能やデータを自由に組み合わせて作業を行なうようなアプリケーションでは、これらをオブジェクトの組合せとして表現することによって、非常に柔軟性に富んだ環境を容易に作ることができる。

今回開発したデスクトップは比較的小規模のものあったが、オブジェクト指向プログラミングによる開発では、コード・デザインにおいて、従来のプログラミング・パラダイムに比べアプリケーション全体の見通しがよく、さらに大規模なものに関しても規模の差を感じない開発が行なえると考えられる。

参考文献

[1] 春木良且：オブジェクト指向への招待, 啓学出版, 1991.

[2] Bjarne Stroustrup 著／斉藤信男訳：プログラミング言語 C++, トッパン, 1992.

[3] 木下凌一：X-Window Ver.11 プログラミング, 日刊工業新聞社, 1992.

[4] 木下凌一他：X-Window OSF/Motif, 日刊工業新聞社, 1992.