1Q−2

Model-Interface-Deviceモデルによるアプリケーション構築

山本 学
日本アイ・ビー・エム株式会社 東京基礎研究所

1 はじめに

近年のワークステーションのアプリケーションは、ウインドウシステムの利用、他アプリケーションとの通信による連携化などにより、利用性は向上しているが、一方で入出力が複雑化し、開発が困難になりつつある。

これに対し、様々なツールが開発され、各機能ごとの開発支援が行なわれている。しかし、これらの機能を適切に結び付け、アプリケーションの構造を決定することは、依然として開発者の責任である。

このような問題を解決する方法として、様々な入出力を統一的に扱える構造を規定することが考えられる。この考えに基づいた構造モデルとしてMVCモデル[1, 2, 3]が提案され、その有用性は幾つかのシステムで実証されている。しかし、このモデルをXwindow System** PM*、Windows**などのようなウインドウシステムをベースとしたアプリケーションにそのまま適用することは幾つかの理由で適切ではない。また、分散アプリケーションに適用することは容易ではない。

本稿では、このようなMVCモデルの問題を解決するプログラム構造として、アプリケーションをModel、View、Controlerオブジェクトに分けるのではなく、Model、Interface、Deviceオブジェクトに分ける方法を提案する。

2 MVCモデルとその問題点

MVCモデルはプログラムを3つに分け、そのプログラムの管理すべきデータをModelオブジェクトが扱い、Viewオブジェクトがその Modelオブジェクトの内容をウインドウ上に表示しする。ユーザからの入力はControlerオブジェクトが受けとり、その入力に応じて、Modelオブジェクト、Viewオブジェクトにメッセージを送り、アプリケーションの動作を管理するというモデルである。(図1) このモデルを用いることにより、プログラムの構造化を行なうことができる。

MVCモデルは、各オブジェクトがウインドウシステムと密接な関係を持てるため、Controler、Viewオブジェクトと分けることに意味があるが、Xwindow SystemやPM、Windowsなどのようなウインドウシステムでは、ウインドウシステムと密接な関係を持つものは、コールバック関数やウインドウ関数であり、オブジェクトはこれらの関数を通してウインドウシステムと対話を行なわねばならない。また、Modelオブジェクトのデータに影響を及ぼさない処理(ウインドウの初期化やウインドウ状態の変更による再描画など)を記述しなければならず、MVCモデルを適用しても、それ以外の処理が必要となり、プログラムの構造を簡潔に記述できない。

さらに分散アプリケーションでは、1つのModelオブジェクトを複数のアプリケーションで共有することが必要であるが、MVCモデルでは、複数のコンピュータ上でControler、Viewオブジェクトを管理することは考慮されていない。

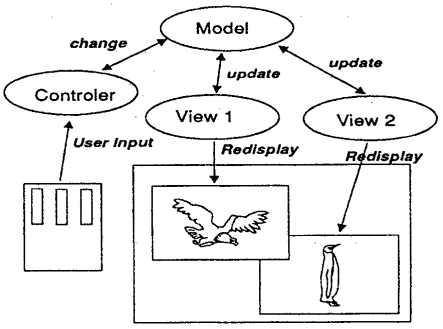

図1: MVCモデル

3 Model-Interface-Deviceモデル

3.1 Model、Interface、Deviceオブジェクト

前節で述べた問題点を考慮し、本稿ではウインドウシステムとの対話を含む入出力部分をControler、Viewオブジェクトではなく、Interface、Deviceオブジェクトに分けたModel-Interface-Deviceモデル(MIDモデル)を提案する。(図2)

Modelオブジェクトは、MVCモデルのModelオブジェクト同様にアプリケーションの本質的なデータを管理するオブジェクトである。

Deviceオブジェクトは、ウインドウシステムと密接に関係した処理だけを行ない、アプリケーションの本質的な処理であるModelの持つデータに対する処理は一切行なわない。例えば、XWindowの場合は、イベント処理のコールバック関数はDeviceオブジェクトが管理する。このオブジェクトはウインドウシステムを扱うのではなく、より一般的な外部インタフェースを扱うためのオブジェクトである。

Interfaceオブジェクトは、アプリケーションの動作を制御するオブジェクトである。これに大きく2つの機能がある。1つはDeviceオブジェクトからの入力の処理であり、1つはModelオブジェクトのアップデイトメッセージに対する処理である。

Interfaceオブジェクトにおける前者の役割は、ユーザ入力などに対しどのようにそのアプリケーションが動作するかを規定するものである。これらの入力イベントはDeviceオブジェクトまたは、その他の外部オブジェクトから送られてくる。後者の役割は、Modelオブジェクトの状態が変更された時、どのような処理を行なってユーザや外部に反映するかを規定する役割であり、実際は、Deviceオブジェクトにメッセージを送信する。

Application Development based on Model-Interface-Device Model
Gaku YAMAMOTO
Tokyo Research Lab. IBM Japan Ltd.

*PMはIBM社の商標です。
**Xwindow Systemは米国MITの商標です。Windowsは米国マイクロソフト社の商標です。

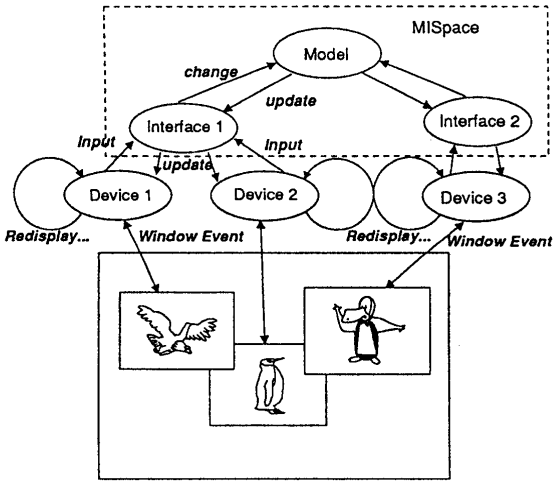

図 2: MID モデル

3.2 MI 空間

Interface オブジェクトと Device オブジェクトは、それぞれ密接な関係を持つが、Model オブジェクトと Interface オブジェクトはそうとは限らない。Model オブジェクトは自分のデータだけを管理するものであり、nterface オブジェクトとの関係は管理すべきではない。

このような情報を管理するために MI 空間 (Model - Interface 空間) を導入する。(図 2) MI 空間も 1 つのオブジェクトであり、これは 1 つの Model オブジェクトと複数の Interface オブジェクトの関係を管理し、Model オブジェクトの状態が変化した時に、各 Interface オブジェクトにそれを通知することである。

4 MID モデルの利用

実際のアプリケーションでは、アプリケーション全体で管理すべきデータがあり、複数のウインドウがそれぞれに局所的なデータを持ちながら、なんらかの依存関係を持って実行されている。この章では Model 、Interface 、Device オブジェクトの組合せ方法についてモードレスウインドウを用いたアプリケーションとグループウェアの例を挙げる。

[例 1:モードレスウインドウを用いたアプリケーション]

この例では、アプリケーションが全体で管理するデータの 1 部をモードレスウインドウを用いてエディットする場合を考える。この場合、MID モデルは図 3のように使われる。これは、2 つのウインドウが存在し、それぞれ別々のデータを持つ。しかし、これらのデータはなんらかの関係を持っており、一方が変更されたら他方も自動的に変更される。このために、互いのウインドウ間でメッセージ送受信を行なう。これは Interface オブジェクト間で行なわれる。

[例 2:グループウェア]

この例では、複数のホストでデータを共有するアプリケーションを考える。この場合、MID モデルは図 4のように使われる。共有されるデータは Model オブジェクトが管理する。MI 空間オブジェクトは Model オブジェクトと同じプロセス空間上に存在するが、ユーザ入出力は複数のホスト上で行なわれるため、それぞれのホスト上に Interface オブジェクトと Device オブジェクトは存在する。

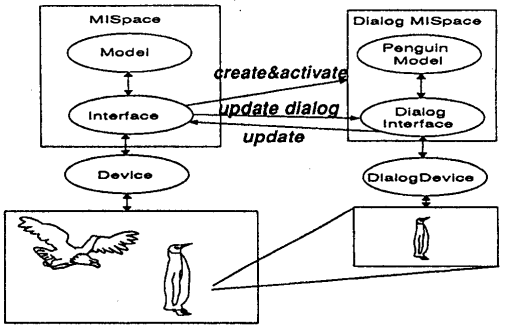

図 3: 例 3

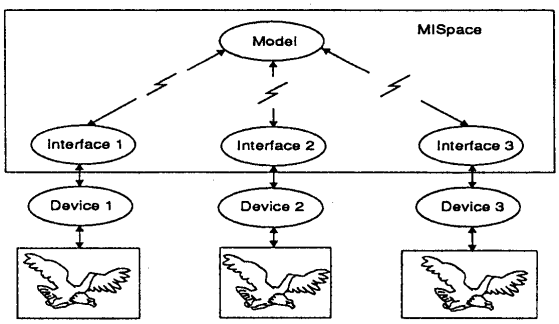

図 4: 例 4

5 おわりに

本稿では、ウインドウシステムをはじめとした複数の入出力を扱うアプリケーションの構造モデルとして Model-Interface-Device モデルを提案した。このモデルを利用することにより、アプリケーションプログラムの構造を規定できるため、開発効率の向上をはかれるとともに、プログラムの変更、デバッグを容易にすることができる。

しかし、Model オブジェクトの管理するデータを他のオブジェクトが参照する場合、現在のモデルでは問題が残る。これは、分散環境では特に問題がある。今後、この問題を考慮したモデルにする必要がある。

参考文献

[1] 春木: オブジェクト指向への招待、啓学出版、1989
[2] A.Goldberg: Smalltalk-80 The Interractive Programming Environment 、Addison-Wesley Publishing Company
[3] A.Goldberg 、D.Robson: Smalltalk-80 The Language and Its Implementation 、Addison-Wesley Publishing Company
[4] インターフェース 91.11
[5] L.J.Pinson 、R.S.Wiener: Applications of Object-Oriented Programming
[6] 山本、原: MVC モデルの分散システム開発への適用 第 42 回情報処理学会全国大会予稿 1991