1R－5

ラピッドプロトタイピングツールMuse(2) * MVCモデルを利用したオブジェクト指向開発

松澤 由香里† 山城 明宏† 神尾 広幸‡ 新田 恒雄‡
†株式会社 東芝 研究開発センター　‡株式会社 東芝 マルチメディア技術研究所

1 はじめに

GUIアプリケーションにおいて、ユーザインタフェースの変更・拡張は頻繁に起こり得る。ラピッドプロトタイピングツールMuse[1]では、ソフトウェアサイクルを考慮し、オブジェクト指向による開発を推進した。オブジェクト指向による開発では、オブジェクトの情報隠蔽、継承、多様性という3つの性質により保守・拡張が容易なシステムを構築できる。しかし、この利点を最大限に生かすためには、何をオブジェクトとし、どのようなクラス構造を構築するかという点が重要なポイントとなる。

そこで我々は、問題領域に特化した抽象的なクラス構造の枠組みをあらかじめ用意し、そこにあてはめる形でMuseのクラス構造を構築した。利用した枠組みは、GUIアプリケーションにおいて最適とされるMVCモデルである。

本稿では、MVCモデルを利用したMuseの具体的なクラス構造の構築方法について述べる。

2 MVCモデル

MVCモデルは、Model/View/Controller(MVC)の３種類からなるオブジェクト分類観点と、その関係を示したもので、Smalltalk-80において利用された枠組みである[2]。この３種類のオブジェクトとその関係を図1を用いて説明する。図1は1つのModelオブジェクトと3つのViewオブジェクト、3つのControllerオブジェクトを示している。Modelオブジェクトはあるデータ値を有し、このデータを3つのViewオブジェクトが様々な形式で画面表示する。また、各Viewオブジェクトは値を変化させるためのユーザ入力方式を定義するControllerオブジェクトを持っている。Controllerオブジェクトがユーザからの入力を受け付けると、Viewオブジェクトを通してModelオブジェクトに値の変更が通知される。Modelオブジェクトは所有しているデータに変更が生じると、依存関係にあるViewオブジェクト全てにデータの変更を通知する。各Viewオブジェクトは、自分の表示形式に合わせて変更されたデータを表示する。

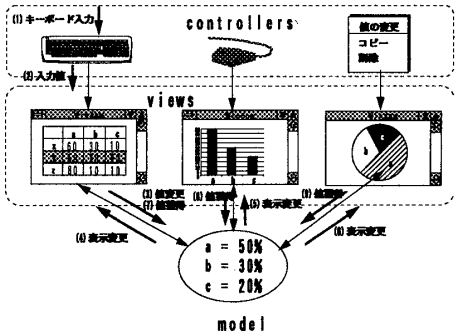


図1 MVCモデル

このMVCモデルに見られる特徴は次の2点にある。

1.Modelオブジェクトに割り付けた複数のViewオブジェクトを矛盾なく動作させることができる。(ModelとViewの依存関係)

2.ユーザ入力方式をControllerオブジェクトとして抽出することで、ユーザ入力方式を容易に変更できる。(ViewとControllerの独立関係)

上記2点の特徴により、MVCモデルを利用したGUIアプリケーションは、画面仕様の変更に強い拡張性のあるものとなる。

3 MuseへのMVCモデルの適用

Museのオブジェクト指向開発では、分析/設計手法にOMT法を、またプラットフォームとしてGUIクラスライブラリとOODBを用いている。実装言語はC++を用いた。これを前提にMuseにおけるMVCモデル適用のための具体的手順を示す。

手順１：Modelクラスの抽出

●永続データを抽出する。
●対多関連を目安に永続データをカプセル化し、Modelクラスとする。

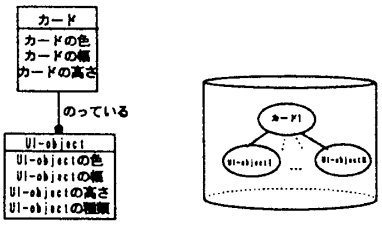
永続にすべきデータを抽出する。例えばユーザがMuse上で作成する画面(カード)仕様の色、種類、座標、画面遷移情報等が永続データである。

次にこれらの永続データをカプセル化してModelクラスとする。この目安として対多関連を利用する。

*Rapid Prototyping Tool Muse(2)
Object-Oriented Development using MVC Model

Yukari Matsuzawa†, Akihiro Yamashiro†,
Hiroyuki Kamio‡, Tsuneo Nitta‡
†Research&Development Center, Toshiba Corp.
‡Multimedia Eng. Lab., Toshiba Corp.

画面仕様を考えた時、図2(a)に示される対多関連が永続データ間に成り立つ。またこの時のOODB内のオブジェクトの格納状態を(b)に示す。



(a)画面仕様内　　(b)OODB内

図2　永続データの対多関連

以上によりModelクラスを抽出する。

手順2：Viewクラスの抽出

●Modelクラスに依存するViewクラスを抽出する。
●画面仕様に記述されるGUI部品をViewクラスとして抽出する。

Modelクラスの所有するデータを画面上にさまざまな仕様を用いて表示するものがViewクラスである。従ってまず、Museの画面仕様からModelクラスのデータを表示するViewクラスを抽出する。次にその他の画面仕様であるメニューバー、ウィンドウ、スクロールバーといったものをViewクラスとして抽出する。図3にMuseの画面仕様とModelクラス、Viewクラスの対応関係の一部を示す。

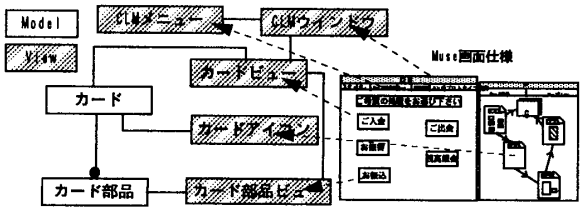


図3　画面仕様とModelクラス、Viewクラスの対応

手順3：Controllerクラスの抽出

●機能単位にサブシステム分割を行う。
●サブシステム毎にサブシステムの状態を管理するControllerクラスを追加する。

我々はControllerクラスの役割を2節で説明したよりもっと広義にとらえ、以下のように定義した。

Controllerクラスの定義：

●1サブシステムを管理するものであり、各サブシステム間のインタフェースの窓口となり得るものである。

サブシステム内の各機能は、それを管理するControllerクラスに依頼することで実行される。Controllerクラスの追加により機能実行の役割分担が明確化され、サブシステム間で一貫した機能の共有がはかれる。Controllerクラスを追加したMuseのMVCモデルによるクラス図を図4に示す。

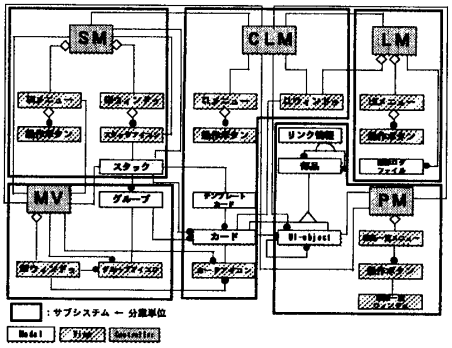


図4　MuseのMVCモデル

4　評価

図5はMuseの分析/設計/実装において追加されたクラス数とその内訳を示している。

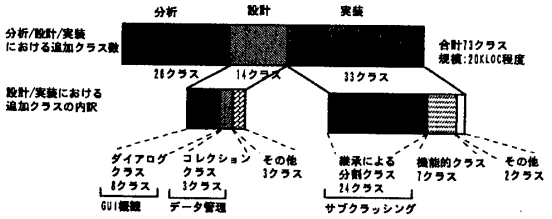


図5　分析/設計/実装時の追加クラス数

図5では、分析時に問題領域のクラス抽出、設計時にGUI概観（ダイアログクラス）とデータ管理（コレクションクラス）のクラスが追加されている。実装時には、設計時のクラスをサブクラッシングしたクラスが追加される。即ち、分析時に構築したMVCモデルによるクラス構造は設計/実装を通して維持され、クラス構造が安定していることがわかる。

5　おわりに

オブジェクト指向開発において、クラス構造にMVCモデルという枠組みを利用するための手順を示し、その有効性を評価した。今後は、利用するクラスライブラリの構造に着目した分析/設計の方法を検討していく。

参考文献

[1] 神尾他"ラピッドプロトタイピングツールMuse(1)" 第52回情報処理学会全国大会, 1996. 3
[2] Erich Gamma他著 (本位田真一/吉田和樹監訳) デザインパターン, SOFTBANK BOOKS, 1995. 10