構造型MVCパターンとGUIビルダーへの応用

6N－5

三ッ井欽一

日本アイビーエム株式会社 東京基礎研究所

1 はじめに

Model-View-Controller(MVC)[2]は、グラフィカルユーザインターフェース（GUI）を実現する上での標準的な指針としてよく用いられている。MVCは、簡潔で抽象的なものであり、その考え方は非常に有効である反面、実際に具体的なGUIを作成する上では、モデルあるいはビューの設計はより複雑なものとなる。モデルとビューの間の関係は実際にはそれほど簡潔にならないことが多く、生産性や再利用性を上げるという観点ではより詳細化されたレベルでの枠組みを考えることが重要である。（例えばShanのMoDE[3]は同様の観点から、より詳細な枠組みを検討している。）

さて、近年の複雑なウィンドゥの入れ子構造を利用したGUIを実現するためのプログラミングは、MVCの考え方をしたとしても労力のかかるものとなっておりアプリケーション全体の作成工数に占める割合は依然として大きい。GUIビルダーは生産性の向上には貢献しているが、複雑なGUIを扱うには不十分な面がある。我々は、GUIビルダーがウィンドゥ間の関係が動的に変化するようなGUIの作成を扱いにくい点に注目し、そのような変化を扱うための「構造型MVC」を提案する。

2 GUIビルダーの問題点

GUIの作成をGUI部品の編集画面での直接操作にもとづいて行なうGUIビルダーが広く利用されるようになりGUIを持つアプリケーションの生産性は向上した。しかし、編集は基本的にウィンドゥ部品の静的な配置をもとにしているので、部品間の関係が動的に変化するようなGUI、例えば（１）サブウィンドゥの個数が動的に変化するもの、（２）サブウィンドゥのうち表示されるものが切り替わるもの、などを直接操作による編集のみで扱うことはできず、従来のテキストプログラミングによる制御を必要とした。

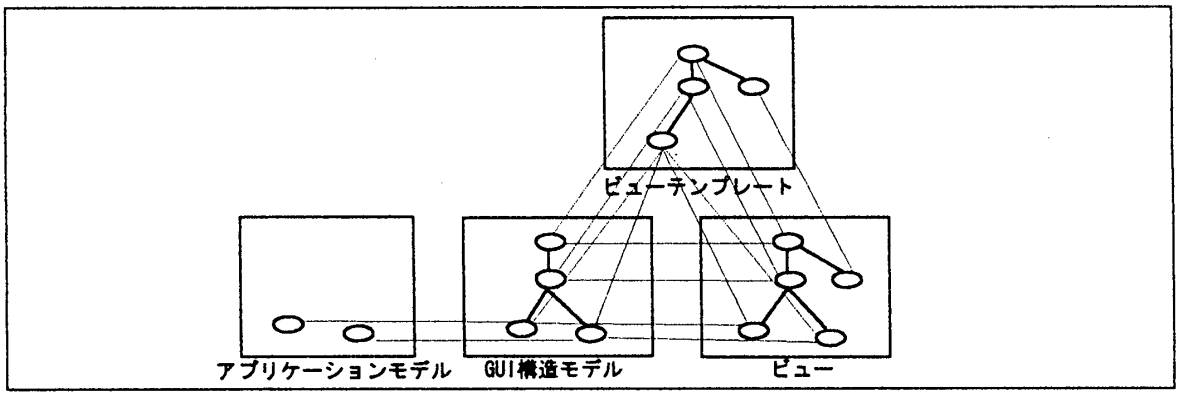
3 構造型MVC

MVCの利点は、表示される情報の論理的な表現と実際の表示方法および入力の処理を明確に分離し独立性を高めることにより、（１）それぞれの設計変更の他方への影響を最小限にできる、（２）単一のモデルに複数のビューを割り当てることができる、（３）それぞれの対応関係を実行時に動的に変更できる、ことである。そのような利点を保存し前節で述べたような構造を持ったGUIを扱えるようにMVCの枠組みを詳細化することがここでの目標になる。ここで、コントローラーは、ビューの構造の動的な変化とは直接関係が無いので以下では議論しないことを記しておく。さて、構造型MVCの直観的な説明はつぎのようなものになる。

a) 構造を持たない要素（例えば文字列型）については通常のMVC関係を設定する。モデルの状態の変化はビューに自動的に通知され表示の変化を起こす。

b) モデルにおける論理的な階層構造とビューにおけるウィンドゥの階層構造の間に対応関係がある場合、それぞれの階層中の節の間に対応関係を定義する。このとき、両者の階層構造は、完全に同形である必要はない。一般にビューにおける階層は余分な節を間に含む。部分構造の生成および消滅を含むモデル内での階層構造の変化は、ビューの階層構造の変化として自動的に通知され、対応するビューが生成および消滅を含む変化を起こす。（アルゴリズムの詳細は割愛する。）a)で述べた構造を持たない要素は、この階層構造の中に末端節として含まれる。

Structured MVC Pattern and Its Application to GUI Builders
Kin'ichi Mitsui
Tokyo Research Laboratory, IBM Research
1623-14 Shimotsuruma, Yamato-shi, Kanagawa 242, Japan



c) b)でいう論理的な階層構造とは、表示の構造の動的に変化する部分を抽象的に表現したものであり、純粋なアプリケーションモデルとしてのデータ構造を表しているとは限らない。従って、a)でいうモデルとb)でいうモデルは明確に区別すべきである。上図では、「アプリケーションモデル」と「GUI構造モデル」と呼び区別している。この区別により、GUI構造モデル中の構造を持たない末端節に対応するアプリケーションモデルをビューの階層構造の変更をせずに動的に取り替える操作が可能になる。

4 GUIビルダーへの組み込み

GUIビルダーの目的は、GUIの実現のために必要なテキストプログラミングを最小限にし、直観的な操作によりGUIを作成できるようにすることである。ここでは前に述べたGUIビルダーの問題点に対処するために構造型MVCの考え方をビルダーに融合する方法について述べる。

GUIビルダーは編集画面を持ち、そこでGUIの雛形が作成される。これを「ビューテンプレート」と呼ぶことにする。実行時にはビューテンプレートのコピーが作成されモデルと連結されて表示される。構造型MVCのビューにおけるウィンドゥの階層構造はビューテンプレートとしてGUIビルダーにより定義される。ビューテンプレートは普通はビジュアル部品の静的な配置に基づくが、ここでは動的なウィンドゥ間の変化を表現できるように拡張する。すなわち、ビューの階層構造中で動的に親子関係が変化をする節、これはいわゆるコンテナあるいはキャンバスと呼ばれるウィンドゥの入れ子を扱える部品に対応するが、その節において、子の定義は静的なものではなく、子の候補となる部分構造の集合を定義できるようにする。実際に表示される子または子の集合は実行時のモデルとの対応により決まる。この、GUI構造モデルとテンプレートの間の対応関係はGUIビルダー中での操作により定義される。上図は、モデル、テンプレート、ビューの間の関係を表現している。

5 まとめ

本研究の貢献は、動的なウィンドゥ間の関係の変化をMVCの枠組みで扱うための構造型MVCを提案し、従来の方法では複雑なプログラミングを必要とした部分を整理しより簡潔にプログラムできるようにしたことである。特にそのような変化を抽象的に扱うためのGUI構造モデルと呼ぶレイヤを導入した。これにより動的に変化する実際のGUIの実現方法とは独立に動的な変化のみを記述することができる。モデルとビューの間の階層構造の対応関係の維持には一般的な手続きを利用することができるので、アプリケーションごとにこれを記述する必要はない。これにより、GUIビルダーを用いてより複雑なGUIを効果的に作成できるようになる。

MVCは、より一般的にはObserverデザインパターン[1]としてGUI以外の場面でも有効であることが知られている。同様に構造的MVCで扱ったような必ずしも同形でない構造の間の対応関係の維持は、GUI以外の場面、例えばプログラムの段階的な拡張性のための技術として使える可能性があることを指摘しておく。

参考文献

[1] Gamma, E. et. al., "Design Patterns: Elements of Reusable Object-Oriented Software", Addison Wesley, 1994.

[2] Krasner, G. E. and Pope, S. T., "A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80," JOOP, 1(3), August/September 1986.

[3] Shan, Y. P., "MoDE: A UIMS for Smalltalk," ECOOP/OOPSLA, 1990.