3F−4 ウィジェット操作のためのC++クラスライブラリの試作

北村 操代　中田 秀男　杉本 明
三菱電機株式会社 中央研究所

1 はじめに

近年、グラフィカルユーザインタフェース(GUI)ビルダの開発が進んでいる。GUIビルダにより画面作成は容易となった。さらにダイアログ操作や画面変化など、アプリケーション(AP)と統合化した段階でのインタフェースの動的側面の確認、修正もその場で行えれば、GUI開発期間の大幅な削減が期待できる。AP部分とGUI部分が密接に相互作用するソフトウエアにおいては、この統合化段階での確認と修正が繰り返されるからである。ところが、従来のGUIビルダでは、AP部分を統合化した段階でのインタフェース確認はその場では行えない。この確認のためにはGUI部分のソースコードの生成やAP部分との結合といった時間のかかる操作が必要という問題があった。

この問題を解決するため、筆者らはGUI部品そのものにビルダ機能を持たせる手法を採用した。これにより、AP実行時においてもGUI修正が可能となる。本稿では、C++言語により試作した、ビルダ機能を持つGUIクラスライブラリであるGhostHouseについて述べる。

2 GhostHouse

GhostHouseはXウィンドウシステムを対象としており、リソース管理やイベント管理はXt[1]、標準部品にはOSF/Motif[2]のウィジェットを利用している。

GhostHouseの特長は以下の通りである。

- 特別なエディタを必要とせず、AP実行時にDrag & Drop操作によりGUI部品の種類、属性、レイアウトを変更できる。
- ツールキットとビルダが一体化されているため、従来のようなGUIビルダに対するユーザ定義部品の登録が不要となる。
- ビルダ機能を抽象クラスに実現しているため、新たに追加した部品に対してもビルダ機能は自動的に継承される。

図1に実現の枠組みを示す。個々のウィジェットに対してこれを操作するオブジェクトが付加される。このオブジェクトをGhostと呼ぶ。通常時のユーザの操作はウィジェットを通して行われるが、特定のキー(Ctrlキー)を押した状態でのマウスイベントは付加されたGhostが取り扱い、ウィジェットの属性変更や状態の保存、後に述べるDrag & Drop操作やこれに伴うリペアレント(親の変更)などを行う。

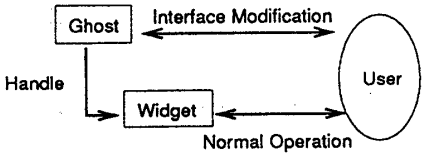

図1: Ghostクラスの役割

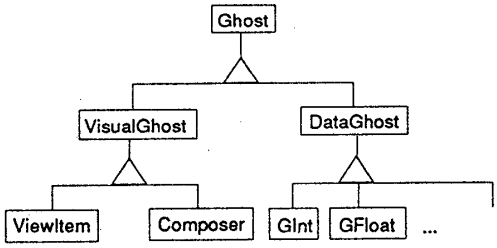

図2: 上位レベルのクラス階層図

3 GhostHouseのクラス階層

図2に、GhostHouseの上位レベルのクラス階層を示す。クラス設計においてはインタフェース部品としての機能を中心においた。従ってMotifの階層とは全く対応していない。Ghostは画面上の部品を操作するVisualGhostと、AP内のデータに対応するDataGhostに分かれる。VisualGhostは更に個々の表示要素(ViewItem)と、それらを画面上に構成するための部品(Composer)から成る。

図3はViewItemクラスの階層図である。細字で付加された名称は、それぞれのGhostに対応するMotifのウィジェット名である。対話部品(Interacter)としては単にイベントを送る場合とデータを送る場合との2通りがあり、Listenerは前者に、Mediatorは後者に対応する。Accessoryは画面の装飾や視点の変更などに用いられる部品である。図4はComposerクラスの階層図である。各種レイアウトにより部品を配置するBoardや、個々の部品をファイルに収納し、アイコン化して表示するGhostCabinet等がある。

GhostHouse: A C++ GUI Toolkit
Misayo KITAMURA, Hideo NAKATA, Akira SUGIMOTO
Mitsubishi Electric Corporation

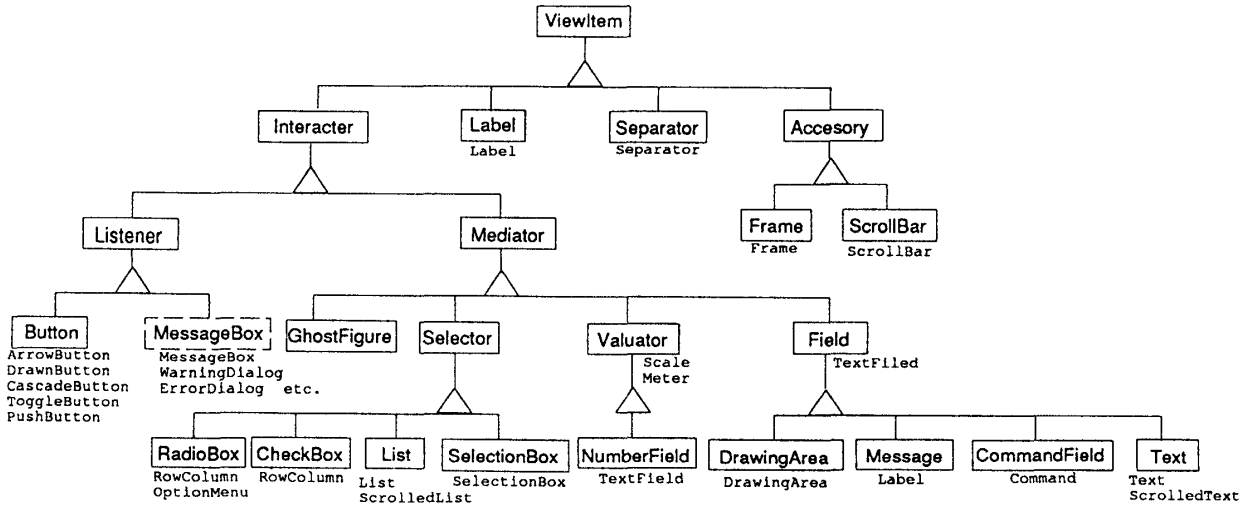

図 3: ViewItem クラスの階層図

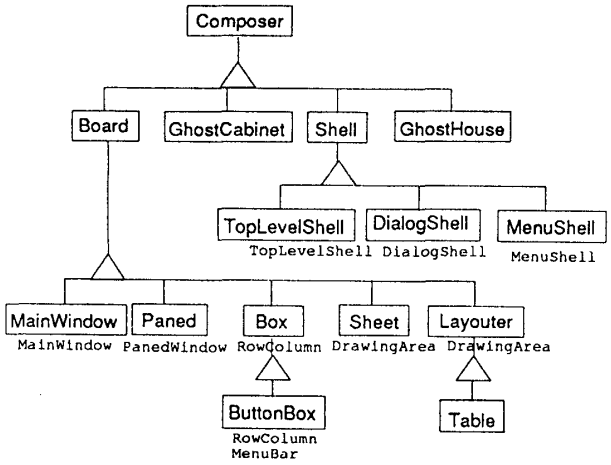

図 4: Composer クラスの階層図

4 Drag & Drop 操作

GhostHouse では、ドラッグドロップ操作をオブジェクト配置だけではなく、メニューやスクロールバーの設定にも用いている。あるオブジェクトをドラッグして別のオブジェクト上にドロップすると、その両者のオブジェクトクラスに応じた処理が行われる。この処理方法は個々のクラスで再定義されうるが、一般的なクラスにおける処理例を列挙したのが表 1である。

表中の番号はドロップ操作により引き起こされる以下の動作を示す: 1.AP との連結を保持したままでの部品の置換、2. 複合部品へのアイテムの挿入、3. ポップアップメニューの設定、4. ダイアログメニューの設定、5. 属性やアクセサリ (スクロールバーや枠) の設定、6. 両部品を Box に格納、7. 両部品のマージ。

表 1: Drop 操作が行われた際に実行される処理

対象	落すオブジェクト			
	Button	Selector	ButtonBox	Accessory
Button	1	4	3	—
Selector	2	1	6	5
ButtonBox	2	6	7	5

ドロップ操作においてウィジェットの種類は適切なものに変更される。例えば PushButton をチェックボックスに挿入すると、そのボタンは自動的に ToggleButton に変更される。また、アクセサリとして ScrollBar をドロップすると、例えば Text ウィジェットは ScrolledText に変更される。

5 おわりに

ウィジェット操作用の C++ クラスライブラリ GhostHouse を試作した。GUI ビルダ機能を持つツールキットであるため、AP 実行時においても GUI の修正が可能である。

本稿ではビルダ機能の実現に重点を置いたが、GhostHouse では図 1のような枠組を採用したため、ウィジェットの階層に捕らわれず柔軟に部品を拡張していくことができる。例えば現在、帳票テーブルなどの既存ウィジェットにない部品をゴースト側の機能定義により追加している。また、曲線や多角形などの長方形以外の境界を持つ図形部品も提供している。

参考文献

[1] J.McComack et al.: "X Toolkit Intrinsics", X11R4 online documentation

[2] OSF: "OSF/Motif™ Programmer's Guide"