対話型システム視覚的構築用クラスライブラリ：GhostHouse (I)

6R−1　—— 設計方針と概要 ——

杉本明　北村操代　中田秀男　川岸元彦　小島泰三
三菱電機株式会社　中央研究所

1．はじめに

EWSの普及と共に、グラフィカルユーザインタフェース（GUI）を実現するためのソフトウェア開発コストが増大している。これに対処するためXウインドウ上のOSF/Motif[1]ウイジェットをはじめ、各種GUI構築用ツールキットが提供されている。しかし、それらの学習は一般的なプログラマにとっては容易ではない。またプログラミングを必要とするため、ヒューマンインタフェースや応用領域の専門家がGUIの構築に直接参加できない。

このような問題に対処するため、我々はGUIを持つ対話型システムのソフトウェア開発の自動化の研究を進めている。我々のアプローチの特徴は、システムで利用するデータの定義から暫定的なGUIを持つ対話型システムを自動生成し、その後、画面上での修正という視覚的な手段により最終的な対話型システムを構築することにある。GhostHouseは、このようなアプローチを実現するためのソフトウェア部品として開発した、オブジェクト指向言語C++によるクラスライブラリである。

今回、GhostHouseを用いて、プラントや電力系統の監視操作インタフェース部や、帳票サブシステム等の対話ソフトウェアを視覚的に構築する開発支援環境の試作を行った。本稿ではGhostHouseのアプローチと全体の概要について述べ、関連報告［2〜4］で個々の特徴を示す。

2．GhostHouseの自動化アプローチ

図1に本稿で取り上げる対話型システムの構成を示す。図1のシステムでは、ユーザはEWSのスクリーン上に表示された監視画面や図面、テーブル、グラフなどを通してシステム内のデータにアクセスする。監視画面や図面等は、Xツールキットのウイジットや Xウィンドウの描画関数を直接使用したシンボルなど、多様なGUI部品により構成される。一方、システムの保持するデータも様々な格納形態を持つ。例えばリレーショナルデータベース内に格納されることもあれば、あるいはファイルを利用することもあり、さらには処理の高速化のため共有メモリ上に格納される場合もある。図上の対話処理ソフトウェアは、こうした様々なGUI部品とデータ間の橋渡しを行わなければならない。すなわち、データが変更されればGUI部品の表示を変更し、GUI部品に対する操作があれば、これに応じてデータの変更を行わなければならない。

GhostHouse試作の最終目的は、この対話処理ソフトウェアの開発を自動化することである。そのための手法として、我々は以下のような2つの段階からなる新しい枠組みを用いている。

1) 暫定的なGUIを持つ対話処理ソフトウェアのデータ定義からの自動生成

監視制御システムのような大規模システムの開発では、あらかじめシステムで用いるデータの型やサイズ、格納場所を定義する。我々のアプローチでは、まずこれらのデータ定義から、個々の対話システムの機能に応じた対話処理ソフトウェアを自動生成する。この時、画面上のGUI部品としては、定義されたデータの種類に応じたデフォルトGUI部品を用いる。この段階で、GUIは暫定的であるものの、実際に動作する対話処理ソフトウェアが作られる。

2) 画面上における対話形式の編集操作によるGUIのカスタマイズ

次にマウスを用いた対話的な編集操作により、暫定的なGUIを所望のGUIにカスタマイズする。編集操作としてはGUI部品の別のGUI部品への置換や、構造化、レイアウトの変更、メニュの設定といったことが可能である[5]。しかも、常に対話処理ソフトウェアとしての動作を確認しながら、最終的なGUIに作りかえていくことができる。

従来のGUIビルダではGUI部品を用いて最終的な画面を構成した後、プログラムによってGUI部品とシステム内のデータとの結合（関係付け）を行う必要があった。本方式では最初の自動生成の段階でデータとGUI部品との結合を行い、これを編集操作においても保つので、プログラミングすることなく所望のGUIを持つ対話システムの構築が行える。

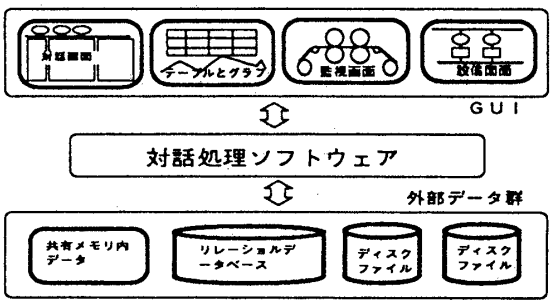

図1　対象とした対話型システム

GhostHouse: A Class Library for Developing UI Systems
Akira SUGIMOTO, Misayo KITAMURA, Hideo NAKATA, Motohiko KAWAGISHI, and Taizo KOJIMA
Mitsubishi Electric Corporation

3．クラスライブラリとしての枠組み

GHostHouseは前章で述べたアプローチを実現するために試作した、C＋＋言語によるクラスライブラリである。画面上の表示及び操作のインタフェースを担当するＧＵＩ部品としては、すでに多くの部品が開発、提供されている。またシステム内で保有するデータの格納形態は、監視制御システムとしての全体の仕様や、データ収集や実際の制御を担当する他のサブシステムの要求により決定される。したがってGhostHouseでは、既存のＧＵＩ部品と外部データとの橋渡しをするため個々のＧＵＩ部品やデータに取り付く、Ghostと呼ぶオブジェクトを中心にクラスを構成した。

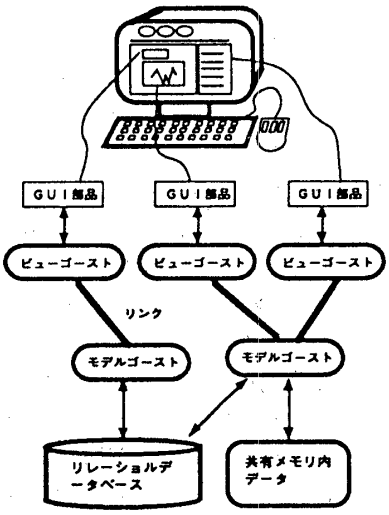

図2　GhostHouse の枠組み

図2にライブラリ構成上の枠組みを示す。モデルゴーストは特定の形式のデータを管理するオブジェクトである。一方、ビューゴーストは特定のＧＵＩ部品の操作を行う。GhostHouseでは、これらのゴーストが図のように個々のデータとＧＵＩ部品の背後に取り付き、以下のような機能を実現する。

(1)データとＧＵＩ部品間の相互作用の実現

ゴーストは基本的な機能として相互に結合するリンク機能を持つ。データの変更がモデルゴーストを通じて行われると、モデルゴーストは結合したビューゴーストによりデータ変更を通知する。ビューゴーストは通知を受けると、その受け持つＧＵＩ部品の所定の方式に従って表示を変更する。逆にユーザのＧＵＩ部品に対する操作は、ビューゴーストによりモデルゴーストに伝えられる。

(2)モデルゴーストによるビューゴーストの生成

前章で述べたデフォルトＧＵＩの生成は、モデルゴーストによりその管理するデータの表示、操作に適したビューゴーストを生成することで行われる。この時点でモデルゴーストとビューゴーストの間の結合が行われる。ビューゴーストは表示要求を受けた時点でＧＵＩ部品を生成し、以後、(1)の相互作用を実行する。

(3)ビューゴーストによるＧＵＩの修正

ビューゴーストにはＧＵＩを修正する機能も実現されている。GhostHouseでは特定のキーを押しながらのマウス操作は、ＧＵＩそのものの編集操作としてビューゴーストが処理する。編集操作は主にDrag＆Dropにより行われ、ＧＵＩ部品の配置や階層構造の変更といった操作に加え、データとの結合を切り換えるＧＵＩ部品の置換機能を持つ]。

(4)モデルゴーストによるデータの格納形式の仮想化

ビューゴーストとモデルゴーストは結合されると、あらかじめ決められた一様なプロトコルにより通信を行う。これを可能とするため、モデルゴーストの実現において、主記憶、ファイル、リレーショナルデータベースといったデータ格納形式の差異を吸収した[4]。その結果、ビューゴーストから見ると、データは一様なアクセス手段を実現したモデルゴーストにより仮想化される。一方、モデルゴーストから見ると、ＧＵＩ部品は所定のメソッドを持つビューゴーストにより仮想化される。

4．クラス構成

図3にGhostHouseの上位のクラスを示す。Ghostのサブクラスとして、前述したモデルゴーストに対応するGModelとビューゴーストに対応するGViewの他に、手続きやオブジェクトのメソッドに対応するGActionがある。GModelのサブクラスとしては、外部データに対応するGExtModelとプロセス内の変数に対応するGVarModelを用意している。一方、GViewにはOSF/Motifのウイジットを操作するGWidgetと、GhostHouseで独自に用意した任意図形部品を操作するGFigureがある。

5．おわりに

本報告ではＧＵＩを持つ対話型システムを画面上で視覚的に構築できる開発環境の実現に向けて開発したクラスライブラリGhostHouseについて、設計方針と概要を述べた。デフォルトＧＵＩを持つ対話処理プログラムを自動生成し、その後、視覚的手段によりインタフェースを修正していくことで、ＧＵＩの知識を全く持たなくても対話システムを構築していくことができる。関連報告［2～3］では、監視制御及び帳票システムに対する適用結果についてそれぞれ報告する。また［4]ではデータの実現方式の差異を吸収する仮想化方式について、具体的手法を述べる。

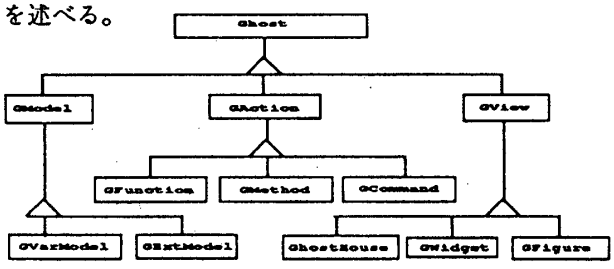

図3　主な上位クラス

[1]OSF:"OSF/Motif(TM) Programmer's Guide
[2]中田他：" 対話型システム視覚的構築用クラスライブラリ：GhostHouse（その２）　監視制御システムへの適用、本予稿集
[3]北村他：同（その３）　オンライン帳票ビルダへの適用、本予稿集
[4]川岸他：同（その４）　外部データアクセスの仮想化、本予稿集
[5]北村他：ＧＵＩ編集機能を持つクラスライブラリ GhostHouse（その２）第４５回情処全大5Q-02