対話型システム視覚的構築用クラスライブラリ：GhostHouse(IV)
-外部データアクセスの仮想化-

6R−4

川岸元彦 小島泰三 杉本明
三菱電機株式会社 中央研究所

1 はじめに

電力系統や工業プラントなどの大規模制御システムでは,種々多様なデータが処理され、また，扱うべき外部データの種類も豊富である．本稿では，このような分野においてソフトウェア再利用性を向上させる、オブジェクト指向方式による外部データアクセスの仮想化手法について述べる。また、GhostHouse[1] における実現についても説明する。

2 外部データアクセスの仮想化

筆者らは、電力系統や工業プラントなどの大規模制御システムのソフトウェア開発支援の研究を行っている。そして、現在、オブジェクト指向方式における外部データアクセスの仮想化について検討中である。本章では、まず適用対象のシステムについて簡単に示し、次にデータアクセス仮想化の概要を説明する。

図1に監視制御システムの構成例を示す。図のシステムは，電力系統設備などを遠隔制御し、また、その保守を支援するものである。図において、制御対象の装置は入出力装置を介して主計算機に接続しており、また、複数の計算機がLANやWANを介して接続している。そして、グラフィック画面を用いた対話型プログラムが各ワークステーション(WS)上で動作し，装置の制御や装置属性の保守操作が行われる。

大規模制御システムにおいて特徴的なことは、ある単一の設備に関するデータであっても、一様な形式では格納されないということである。例えば、装置の現在状態のように動的なデータは、主記憶データベース(以下、MDBと記す)に，また，変更の少ないデータは，リレーショナルデータベース(以下、RDBと記す)等に格納される。また，同じデータに対するアクセスであっても，通信形態を考慮し，異なるアクセス手段が用いられることもある。

しかしながら、ソフトウェアの再利用性を高め、また、拡張性の高いシステムの構築を可能とするためには、設備単位にオブジェクトを定義できることが望ましい。このためには、オブジェクトの個々の属性値のアクセス方法の違いを隠蔽することが必要である。

このようなアクセス方法の違いの隠蔽は、オブジェクト指向方式ではメソッドによりなされる。しかし、この隠蔽のためのメソッドの実現は容易ではない。例えば、あるクラスのサブクラスに属性を追加したとすると、継承したアクセスメソッドやオブジェクトの保存、回復を行なうメソッド等をサブクラスで再定義する必要がある。しかも、筆者らの想定している大規模システムの開発では、クラス基本部、データ格納部、更に、各応用プログラムの開発が複数のプログラマにより分担して、同時並行に行なわれる。したがって、クラスの属性の定義と、実際のその属性値の格納先の設計が独立して行なわれることが望ましい。

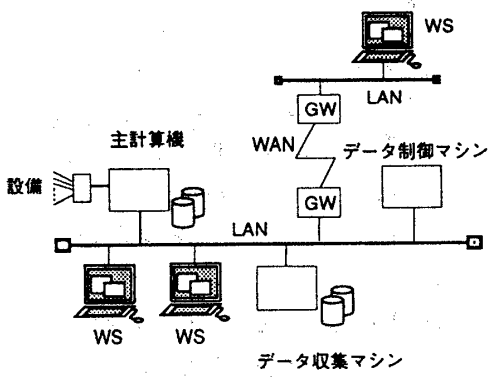

図 1: 監視制御システムの構成例

そこで、本稿では図2のような設備を表すクラスの実現において、仮想オブジェクトと実オブジェクトに分離する手法を提案する。仮想オブジェクトは、応用プログラム内でアクセス方式の違いを隠蔽したインタフェースとして働き、設備としての継承関係を持つ。また、実オブジェクトは実際のデータを管理し、格納方式に基づいたクラスの継承階層が構成される。

そして現在、GUIライブラリ GhostHouse 上に本機能を組み込み、実際のシステム開発に適用することにより、その有効性を検証している。

3 実現方式

本章では、GhostHouse における仮想オブジェクトの実現方式について述べる。GhostHouseでは、仮想オブジェクトはモデルゴーストのサブクラスとして実現されている。

GhostHouse(IV):A Class Library for Developing UI System
Motohiko KAWAGISHI, Taizo KOJIMA, and
Akira SUGIMOTO
Mitsubishi Electric Corporation

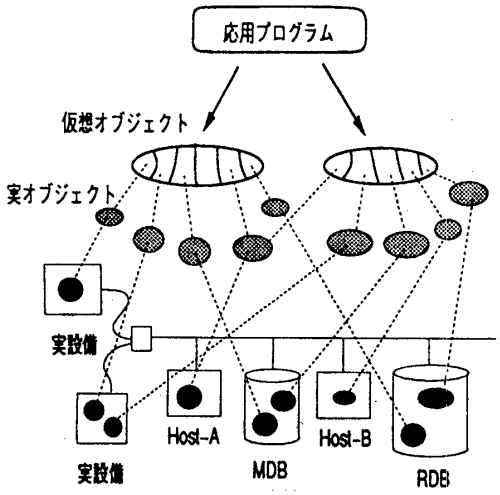

図 2: データアクセスの仮想化

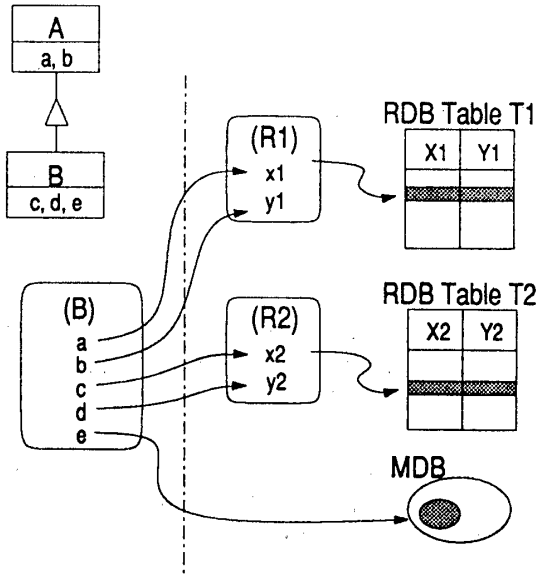

図 3: 属性間リンク

3.1 属性間リンクによるデータアクセス

仮想オブジェクトの属性と実オブジェクトの属性は属性間リンクにより結合される。図3に仮想オブジェクトと実オブジェクト間の構造を示す。この図では、仮想オブジェクトとしてクラスA、そのサブクラスであるクラスB があり、クラスA は属性a、bを、クラスBは属性c、d、e をそれぞれ持っているものと仮定している。そして、クラスB においては属性a、b はRDB のTable1 に、属性c、d はRDB のTable2 に、e はMDB にそれぞれ格納されているものとする。ここで、図の左下にあるようなクラスB のインスタンスB'を考える。すると、図ような属性間リンクがはられ、データアクセス時にはそのリンクをたどって実際のデータがアクセスされる。例えば、B の属性a はテーブルT1 のレコードであるR1 のx1 に、属性b は、y1 をそれぞれアクセスすることになる。

3.2 属性情報による関係の定義

仮想オブジェクト内の属性と、対応する実オブジェクト及びその属性の関係は、仮想オブジェクトのクラス定義時に属性情報として与えられる。例えば、図3の例では、属性情報a、b はクラスA に、属性情報c、d、eはクラスB にそれぞれ格納されている。もし、クラスA のa の格納先がクラスB のa の格納先と異なる場合は、B についてa の属性情報を与えることも可能である。

3.3 オブジェクトIDによる復元

通常 C++ 言語では、ポインタを用いてリンク構造を実現する。ポインタは実際はアドレスであり、そのプログラムの実行中でのみ有効である。従って、リンク構造を保存するには、なんらかの機構が必要である。仮想オブジェクトと実オブジェクトとの対応はオブジェクトID (OID) により行なわれる。仮想オブジェクトの生成においては構築子にOID を指定する。生成された仮想オブジェクトは、属性情報とOID を用いて図3のようなリンク構造を復元させる。そのため、実オブジェクトはOID を用いた問い合わせに対して、そのデータのアドレスを返すように実現されている。例えば、RDB のテーブルオブジェクトはOID に対応するレコードをRDB から取りだし、その中の属性に対応するデータに対するアドレスを返す。

4 おわりに

現在，データアクセス機構を GhostHouse に組み込んでいる。そして、電力系統の設備データベース保守支援システムの開発に適用することにより，その有効性を確認する予定である。なお，今後の課題として，仮想オブジェクトと実オブジェクトの結合に属性間リンクを用いることによるメモリ効率の問題が残されている。このためより効率の良い結合の実現について検討してゆくつもりである。

参考文献

[1] 杉本他. 対話型システム視覚的構築用クラスライブラリ:GhostHouse(I). 第46回 情処全大.