分散型監視制御システム構築環境(2)

1N－2

—フレームワーク—

野里貴仁 小島泰三
三菱電機株式会社 産業システム研究所

1 はじめに

本稿では，筆者らが開発している監視制御向けオブジェクト指向アプリケーションフレームワーク[1]の特長を述べる．最初に監視制御対象をオブジェクトとして捉えるために，筆者らが用いたモデル化手法について説明し，次に，本フレームワークで分散処理機能を実現するクラスライブラリ HolyGhost の非同期メッセージ処理の概要について説明する．

2 モデル化手法

本フレームワークでは，監視制御対象や監視用機器をオブジェクトとしてモデル化するために，ミラーワールド方式と呼ぶモデル化手法を導入した．図1にミラーワールド方式の全体図を示す．本モデル化手法は，エンドユーザに対し，理解の容易なデータ表現を提供することを目的としている．図中のモデルサーバと呼ばれるプログラム内では，監視対象である設備やコントローラ等がオブジェクトとして表現されており，クライアントである GUI プログラムからは，このモデルオブジェクトにメッセージを送ることにより，現在状態を取得したり制御を行うことになる．

本フレームワークにおけるモデル化の特長は，現実の世界を反転させる形でモデルサーバにおいてオブジェクト化していることにある．例えば，図1において，設備の現在状態は，設備 → コントローラ → 実時間データ収集サーバと伝播し，モデルサーバに到達する．そして，モデルサーバ内では，外界とは逆順で，それぞれに対応するモデル間をデータが伝播する．一方，GUI からの制御命令については，現在状態の場合とは逆の方向で命令が伝播する．GUI 側においては，対象とする実設備に対応するオブジェクトのみに注目すれば良く，対象システム毎に異なる伝送系システムやコントローラの構成を隠蔽できる．エンドユーザに対し，監視対象である設備をモデル化したオブジェクトを示すことにより，理解が容易な GUI を構築することが可能となる．さらにまた，システム構成の変更においても，ソフトウェアモジュールの入れ換えにより対処できるので，個々のソフトウェアの再利用性が高まるという効果もある．

3 非同期メッセージ処理

HolyGhost は分散処理機能を実現するための基本ライブラリであり，クライアントサーバ方式に基づく分散システムの構築を支援している．その特長は，オブジェクト間のメッセージ処理パタンをライブラリに収めて，非同期メッセージを処理していることと，メッセージ通信のフロー制御にある．

3.1 処理パタンのライブラリ化

HolyGhost では，システムの実時間性を確保するために，全てのプロセス間通信を非同期で動作させ，また，監視制御マンマシンシステムの基本処理については，オブジェクト間の非同期メッセージ交換を用いて実現している．

一般に返答を必要とするメッセージを非同期に処理する場合，その機構は複雑となり，そのプログラム開発は容易ではない．これに対し，本フレームワークでは，監視制御マンマシンシステム分野を適用範囲とし，応用プログラムの作成に必要な処理パタンを分析することにより，非同期にメッセージを扱う抽象クラスから構成されるライブラリを実現できた．このライブラリを用いることで，アプリケーション開発が容易になる．

3.2 フロー制御

HolyGhost では，分散型監視制御システム向けのメッセージフロー制御を，その抽象クラスに導入している．

モデルサーバは外界と GUI プログラムの間に位置しており，外界から受け取ったデータをマンマシンシステムで扱えるメッセージに変換して，GUI

---

an object-oriented framework for distributed supervisory control systems
Takahito NOZATO, Taizo KOJIMA,
Mitsubishi Electric Corp.

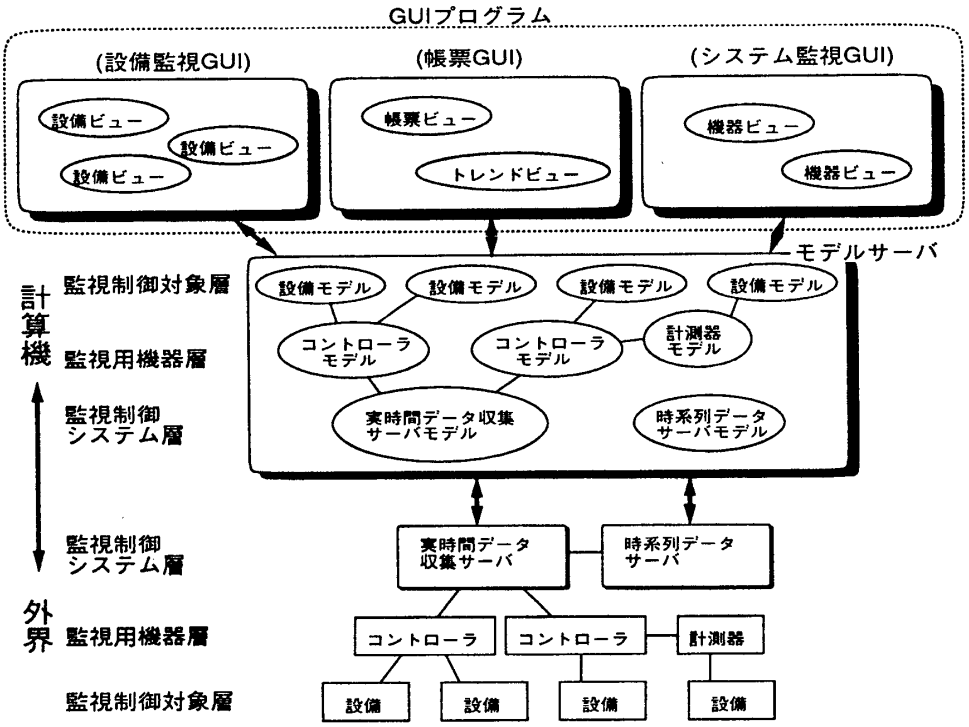

図 1: ミラーワールド方式

プログラムに送信しなければならない．通常，監視制御システムでは，制御機器の状態を一定周期 (数百ミリ秒から数秒単位) で画面上に表示する．表示をする度に，データサーバやコントローラに必要なデータを要求することは，通信量や処理速度の点から現実的ではない．そこで，画面の初期表示時点に必要なデータの要求を送信し，後は，データサーバやコントローラから周期的にデータ転送を行なう．

モデルサーバにおいて，非同期で送信されるメッセージは，送信先が受信するまで，一旦キューに貯められる．したがって，送信相手がプログラムエラーなどで読み出しを行なわなければ，キューが溢れてしまう．これを防ぐために，以下のようにして，メッセージの流量を制御している．

まず，データを二種類に分類する．一つは，全てのデータを通知しなければならないもの (時系列の変化や履歴の表示に使う) であり，もう一つは，最新データのみが重要であるもの (画面表示に使う) である．メッセージには，どちらの種類のデータを扱うかを示す印を付けておく．メッセージ送信時には，キュー内のメッセージの総量がある閾値以上であり，かつ，このメッセージが最新データのみを要求しているのなら，キュー内の対応する古いメッセージを削除し，新しいメッセージを追加する．

また，非同期に処理するために，画面切替時に前画面で参照していたデータが，しばしば流れる．この通信量を低減するために以下のようにしている．画面を構成するために必要なデータ要求の固まりを，セッションとしてグループ化し，これをサーバに登録する．データ受信時に通信パケットを調べ，該当するセッションがなければ，そのデータを不用と判断する．

本フレームワークにおけるフロー制御の仕組みは簡単であるが，実際のシステムに対して，効果的に作用している．

4 まとめ

監視制御システム向けアプリケーションフレームワークについて，本フレームワークで導入したモデル化手法であるミラーワールド方式と，プロセス間通信のフロー制御の仕組みについて述べた．

参考文献

[1] 小島他, 分散型監視制御システム構築環境 (1), 第 52 情処全大,1996

[2] Amjad Umar,*Distributed Computing*,Prentice Hall, 1993