分散監視制御システム向けフレームワークの改良について

4C－6

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

我々は、監視制御システムのソフトウェアを効率的に開発するために、アプリケーションフレームワークを開発している。そして、実システム開発に適用することで効果を上げてきた。しかし、開発効率をいっそう高めるためには、実システム開発時の問題点を解決するようにフレームワークに改良を加えて、成熟させていくことが必要である[1]。また、監視制御分野における多くのアプリケーションに適用することを検討しており、そのためにも現在のフレームワークに対して改良を加えることが必要である。本稿では、これらのフレームワーク改良方針について述べる。

2 フレームワークの改良点

2.1 フレームワークの適用時の問題点

我々が開発支援しているシステムは、GUIを用いて機器・設備等の状態を監視や状態の設定をするシステムである。変電所監視システムや水処理プラント監視システムなどの大規模な分散システムが対象の例である。

我々の開発したフレームワーク[2]では、アプリケーション群に共通する部分(フローズンスポット[3])として、監視制御システムの処理パターンとプロセス間通信機能を実装している。そしてアプリケーション固有の処理を監視対象オブジェクトの中で処理するようにしている。処理パターンとしてデータの取得/設定のインタフェースを定義しており、その詳細はアプリケーションごとに監視対象オブジェクトのサブクラスで実現する。さらに、監視対象オブジェクトを3つの層に分け、それらの連携をフレームワークで実装した。それらは、データサーバに対応するシステム層、コントローラや計測器などの機器層、そして個々の設備に対応する監視制御対象層である。これは、様々な監視形態に柔軟に対応するためであった。

Improvement of Framework for Distributed Supervisory Control Systems
Takahito NOZATO, Taizo KOJIMA,
Mitsubishi Electric Corp.

しかし、このフレームワークは監視対象オブジェクトの実装について、現場のプログラマに対して大きな自由度を与えていた。このために、オブジェクト指向プログラミングに不慣れなプログラマが開発すると、従来の実装方法に近いシステム層のデータサーバモデルのみを実装してしまった。この結果、我々の意図とは異なり非常に複雑なプログラムが作られてしまった。

2.2 適用の拡大

適用するアプリケーションを広げる際に問題となるのは、アプリケーションごとに監視対象データの扱いが異なっていることである。システムを作成する度にデータ処理部の実装を変更していては、フレームワークを利用する効果がなくなり、再利用性を向上させることもできなくなると予想される。例えば、同じ電力設備でも水処理プラント監視システムにおけるものと変電所監視システムにおけるものでは、データの扱われ方などが違うために、その定義からして異なっている。その結果、一見似たように見える処理でも実装部分は全く異なり、ソフトウェアを再利用することが難しい。

2.3 改良の方針

先に述べた問題点を解決するためには、以下の方針でフレームワークを改良することを検討した。

1. アプリケーションごとの可変部(ホットスポット)をサブクラスによって実現するだけでなく、可変な処理を外部定義データとして与えることでホットスポットを実現する。
2. アプリケーション毎に異なる設備データ定義の違いをフレームワークで吸収する。

1は、オブジェクト指向プログラミングに不慣れなプログラマに対しても、容易にフレームワークを利用させることを狙いとしている。監視対象設備モデルの作成が容易になれば、現場のプログラマが複雑なデータサーバモデルを作るような問題を避けることができる。また、2によってデータ定義の違いに起因するソフトウェア違いを減少させることもでき

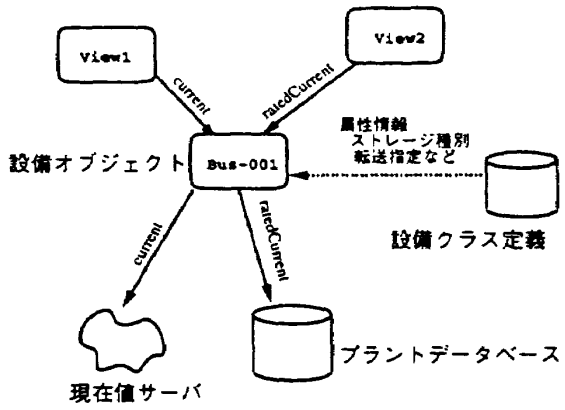

図 1: 設備クラスの利用

る。以上の改良を加えるために、設備クラス定義と呼ばれるデータを導入する。

3 設備クラス定義

設備クラス定義は、従来一つ一つの設備に割り当てられていた情報の中から共通している属性を抽出したものである。設備クラス定義はシステムに依存しないので、フレームワークが設備クラスを参照することでシステムの違いを吸収することができる。

次に設備クラス定義を用いたホットスポットの実現について、データ取得を例に説明する。図 1では、View1 は設備オブジェクト Bus-001 の属性 current の値を表示している[1]。設備オブジェクトは current の値を現在値サーバから取得し、 ratedCurrent の値をプラントデータベースから取得する。属性値をどのような手段で得るかは、後述する属性に付加されたストレージ種別によって決定している。

次にクラス定義の例を図 2に示す。クラスは、クラス名、継承元のクラス名、属性の定義からなる。クラス Bus は CompositeEquipment と呼ばれるクラスを継承している。属性定義は、属性の名前、データ型、属性修飾子からなる。例えば属性 status のデータ型は short、属性修飾子には cooked input と sporadic の 2 つが指定されている。

クラス定義の中でホットスポットの実現に大きな役割を持っているのは属性修飾子である。修飾子をどのように定めるかは、まだ検討中であるが、主要なものとして、ストレージ指定、転送指定、導出演算指定を用意している。ストレージ指定は、データ

[1] モデル母線-001 は分散オブジェクトであるので、この図におけるモデルは、分散処理を実装するオブジェクトなどの複数をまとめているものである。

```
class Bus : CompositeEquipment {
 short status:raw input,sporadic;
 float current:raw input,periodic;
 float maxCurrent: computed,
                   derived by max(current);
 float ratedCurrent:persistent;
 float ratedVoltage:persistent;
   ....
};
```

図 2: クラス定義の例

の取得 / 設定先を指定するものである。図の cooked input、 computed、 persistent がストレージ指定の例である。それぞれ、現在値サーバに問い合わせる、他の属性の値から計算する、データベースに問い合わせることによってデータを取得することを意味する。転送指定は、設備からのデータを GUI プログラムに転送する場合に定周期に行なう (periodic)、あるいは変化時にのみ行なう (sporadic) かを指定する。導出演算指定は、他の属性に対して演算処理をすることことを指定する。 maxCurrent は属性 current の最大値であることを意味している。これらの属性修飾子を利用することで、データの扱い方の違いなどのアプリケーションに依存する処理を、フレームワークのホットスポットを実際に変更することなく実装できる。

4 まとめ

フレームワーク適用時の問題点と適用分野を広げる場合に生じる問題点はどちらも監視対象のモデル化に起因している。その解決として設備クラス定義の利用について説明した。現在、設備クラス定義を用いるように改良を加えたフレームワークを使ってプロトタイプシステムを開発しており、今後さらに多くのアプリケーションへの適用する予定である。

参考文献

[1] 荒野他, ネットワーク管理フレームワークとその開発に関する一考察, 情処論文誌,Vol.38,No.6,1997

[2] 野里他, 分散型監視制御システム構築環境 (2), 第 52 情処全大,1996

[3] W. Pree,*Design Patterns for Object-Oriented Software Development*, ACM Press,1995