アクティブデータベースを用いた監視制御システムの構築

4J－11

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

筆者らは、下水処理、トンネル監視などの公共プラントや電力系統の監視制御システムのソフトウェアの開発支援を行なっている。これらのソフトウェアは大規模なものであり、再利用性を高め、効率良く開発を行なうことが望まれている。しかしながら、実際の開発においては、ソフトウェアの再利用性は低く、効率良く開発されているとは言い難い。また、最近はシステムの低価格化の要求が強く、ソフトウェア開発の効率を向上させることが、よりいっそう必要となってきている。

本稿では、監視制御システムソフトウェアの開発効率を改善するための筆者らのアプローチについての述べる。まず、従来の監視制御システムソフトウェアの開発における問題点を述べ、その後に、筆者らが提案するソフトウェア構築方法について説明する。

2　監視制御システム構築の問題点

監視制御システムソフトウェアが実装している機能は、監視機能と制御機能である。監視機能とは、現在値や異常状態になったことをオペレータに伝えることであり、制御機能とは、設定値を変更したり、コントローラなどの設備を実際に動かすことである。一般に、このうちの監視機能の方に重点がおかれている場合が多い。本稿でも、監視機能の実現について考える。

監視機能のためには、入力データからシステムがどのような状態にあるかを判断することが必要である。多くの場合、システムの状態は入力データがある閾値を越えることや、特殊なイベントがある期間内に発生することなどから判定されている。入力データを処理し、その結果をオペレータに知らせるまでの一連の処理はほぼ共通である。よって、多くの部分を共通化することが可能であると考えられる。しかしながら、実際のソフトウェアの開発においては、入力処理部も、それ以後の処理もシステム（製品）ごとに実現している場合がほとんどである。このために、共通して用いることができるソフトウェアは少なく、再利用性は低いものであった。また、多くの入力処理では、システムに強く依存したデータファイルを必要としているので、設備データなどの本来システム間で共有できるデータでさえも、個別に用意せざるを得なかった。

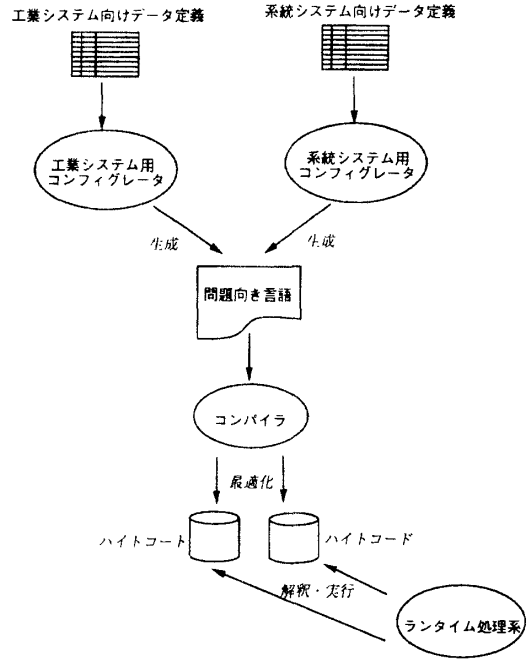

図 1: システム構築の流れ

3　問題向き言語

先に述べた問題を解決し、再利用性の高いソフトウェアを構築するために、各システムで異なる処理を問題向き言語で記述し、それをランタイム系で解釈・実行するという方式を提案する。図1にこの流れを示す。問題向き言語での記述は、コンフィグレータと呼ばれるツールを用いて生成する。コンフィグレータは、それぞれのシステム向けデータ定義やシステムで扱う設備間の依存情報を元に問題向き言語のプログラムを生成する。問題向き言語で書かれたプログラムはコンパイラによって最適化されたバイトコードに変換される。

我々の方式では、入力データの処理をどのようにするかを問題向き言語で表現しているが、これは中間言語に変換する計算機言語処理系と同様な構造である。このような言語処理系では、中間言語を用いることで多くの異

Buiding supervisary control systems with Active Database
Takahito NOZATO, Taizo KOJIMA,
Mitsubishi Electric Corp

なる機種への変換を容易にしている。我々の方式では問題向き言語で表現することで、それぞれの監視制御システムの違い、特に入力データ処理の違いを吸収する。このために、システムに依存した実装をできるだけ少なくさせ、ソフトウェアの再利用性を高めることができる。

ランタイム処理系における問題向き言語の解釈・実行の実現には、アクティブデータベース [1] の機構を用いている。ここでは問題向き言語をアクティブデータベースのルール記述として扱っている。監視制御システムではデータベース管理が必須であることと、ある条件に対して自動的に手続きが呼ばれるという特徴が、監視制御制御システムに適しているので、アクティブデータベース機構を導入した。

4 アクティブデータベース

アクティブデータベースとは、ある条件が満たされた時に、データベースに対するアクションを自動的に行なうものである。ここでのアクションは、データベースの操作だけではなく、新たなイベントの発生やアプリケーションプログラムへの操作も含んでいる。

[2] では、ECA(Event-Condition-Action) をルールの記述に用いている。我々も同様に ECA を用いるが、Condition に時制論理の表現を許す。監視制御システムの入力データ処理においては、二つのイベントが同時に起こる、あるいは、特定のイベントが連続して起こるなどを判定することが重要だからである。次に我々のルールの記述について説明する。

4.1 ルールの記述

我々のルール記述はイベントを記述の集まりである。イベント記述は次の形をしている。

```
イベント名{
        On イベント列
        Cond 条件式
        Body 処理記述
}
```

イベント列には、そのイベントを引き起こす原因となるイベントを列挙する。条件式には、イベントが発生する条件を記述する。ここには C 言語での条件式と監視制御のための特殊な条件式を使うことができる。ボディには、条件式が満たされた場合の処理を記述する[1]。

特殊な条件式としては、past(条件, 時間) と future(条件, 時間) を用意している。past は現在から”時間”前までの間に”条件”が満たされていた場合に真、future は現在から”時間”後までの間に”条件”が満たされれば真

[1] 現在、イベントの伝播を意味する raise_event のみが書ける。

```
FCB{
  On DI001;  /* 入力デバイス */
}

RY{
  On DI002;  /* 入力デバイス */
}

EV_Trip{
  On FCB,RY;
  Cond (FCB.value==0) && (RY.value==1) &&
       abs(FCB.time - RY.time) < T1 ;
  Body raise_event(1,FCB.time);
}

EV_Accident {
  On EV_Trip;
  Cond future(FCB.value==1,Tm1);
  Body raise_event(1,FCB.time+Tm1);
}
```

図 2: ルール記述例

となる。またイベントは value、flag、time の 3 つの属性を持っており”イベント名.属性名”とすることで参照される。

図 2 に我々のルール記述例を示す。EV_Trip は FCB と RY のイベントが同時 (T1 の時間差内) に発生し、その value が 0 と 1 である場合に起こる。EV_Accident は EV_Trip が発生した後、Tm1 以内に FCB のイベントが発生して、その value が 1 である場合に起こる。

5 まとめ

監視制御システム構築の問題点と我々が提案しているシステム構築方式について説明した。我々の方式では、問題向き言語を使って各システムの違いを吸収し、ソフトウェアの再利用性を高めることができる。ランタイム系ではアクティブデータベース機構を用いており、そのルール記述には時制論理の表現を導入して、イベントの時間的な側面を扱えるようにした。

参考文献

[1] 石川博. アクティブデータベース. 情報処理, 35(2):120–129, 1994

[2] Dennis R. McCarthy and Umeshwar Dayal. The Arcitecture of an Active Database Management System. In *Proceesings of the 1989 ACM SIGMOD Conference*, pages 215–224, 1989