2H−4　実行モデルを用いた実時間制御システムの動作監視

平井健治　杉本明　阿部茂
三菱電機(株)中央研究所

1　はじめに

制御システムなど実行時の動作が複雑なシステムを監視する場合には、膨大な監視情報を扱わなければならない。本稿では、制御システムの動作を抽象化した実行モデルを用いてシステムを監視するための手法について述べる。本手法では、監視対象のシステムから収集した情報を実時間で処理し、動作状態や時間制約の検査を行なえることが特徴である。

2　背景と目的

プラント制御システムなどの実時間制御システムでは、非同期に発生する事象を処理するため、システム中では複数の処理が並行実行される。このような並行システムの動作は、複雑かつ再現性がないため、トラブルが生じた時の原因の解析が困難になっている。

並行システムの動作解析やデバッグを行なうためには、実行中のプログラムの動作を監視し、必要な情報を収集・記録するモニタリング手法が用いられる。しかし、従来のモニタリング手法では、実行後の解析に主眼がおかれているため、監視対象の動作などについての情報を持たず、監視情報を単に記録するだけである。

従って、生産ラインの制御など連続運転されるようなシステムを対象にする場合には、膨大な量の監視情報を扱うことになり、その管理手法及び記憶領域が問題になる。

我々は、システム内の各処理を抽象的に表現した実行モデルを用いて制御システムを監視するための手法を提案する。本手法の特徴としては、

- 収集した監視情報を実行モデルを用いて処理するため、システムの動作状態を実時間で検査することができる
- 監視情報を実行モデルで処理するため、必要な情報だけを残すことができる（時間制約の満たされなかった処理の情報など）

3　監視系の構成

図 1に実行モデルを用いたシステム監視手法での処理の流れを示す。

Model- based Monitoring Technique for Real-Time Control Syetem
Kenji HIRAI, Akira SUGIMOTO, Shigeru ABE
Mitsubishi Electric Corp.

3.1　イベント収集部

並行システムの動作の監視は、各タスク内の基本的な操作の実行過程を対象として行なわれる [1]。監視系では、この基本的な操作の実行をイベント情報として収集する。本手法でも、システムの動作の監視はイベント情報を用いて行なう。

イベントの種類

1. タスク起動／終了、メッセージの送信／受信、デバイスアクセスなど
2. ユーザ定義のイベント
   (時間制約を検査するための検証点など)

イベント収集部は、各タスクのイベント情報を発生順にイベントキューに格納する。

3.2　状態検査部

実行モデルは、制御システム内の各処理の実行過程をペトリネットによって抽象的に表現したものである。状態検査部は、イベントキューから順に読み込んだイベント情報を実行モデルによって処理（ペトリネットでの状態遷移）し、制御システム内の各処理の実行過程の管理及び検査を行なう。

また、制御システムでは各処理の応答時間に制限があり、許容された時間内に実行が完了することが要求される。状態検査部では、このような時間制約の監視も行ない、システムが要求された通りに動作しているかを検査する。

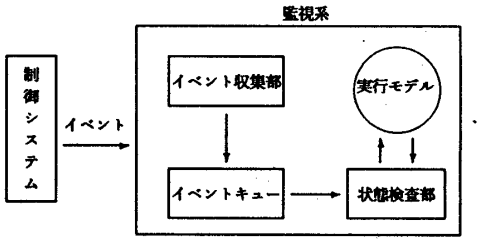

図 1:

4　実行モデルを用いた監視方式

4.1　イベント情報の管理

図 2に本研究で監視の対象とする制御システムをモデル化した図を示す。制御システムは、種々の優先度をもったタスク（A〜F）から構成される。システムは事象駆動型であり、制御対象の状態変化やオペレー

タ要求などの事象発生ごとに対応するトランザクション処理が実行される。各トランザクション処理は、1つのタスクで単独に処理されるか、または、他の必要なタスクを順次起動することによって処理される。トランザクション処理の起動要因となる事象を誘因と呼ぶ。

誘因となるイベントには次の2つがある。

- 周辺機器からの制御信号
- タイマーイベント

誘因は非同期に発生するため、システム内では複数のトランザクションが並行実行される（図では、誘因1によってタスクA、B、Cからなるトランザクション処理、誘因2によってタスクD、E、Fからなるトランザクション処理が各々起動されることを示す）。

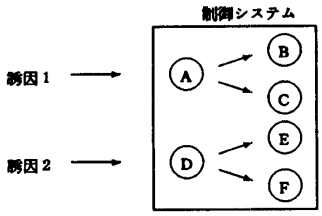

図 2:

監視系では、トランザクションを構成する各タスクから収集したイベント情報（基本的な操作の実行記録）によって以下の項目を検査する。

- 各トランザクション処理が正常に終了したか
  （イベント発生順序の検査）
- 各トランザクション処理の時間制約が満たされたか
  （イベント間の時間間隔を検査）

イベントの発生順序に関しては、各トランザクションの実行過程をペトリネットによって表現し、イベント発生ごとにペトリネット上での状態遷移を確認することによって行なう。このため、各トランザクションごとに実行モデルをもつ。

収集したイベント情報の系列には、複数のトランザクションのイベント情報が混在しているため、イベント情報を実行モデルで処理する場合には、その対応づけが問題になる。そこで、イベントと実行モデルとのマッピングは、各トランザクションの発生の起因となる誘因情報を用いて行なう。すなわち、実行時には、イベントに誘因情報を持たせ、誘因情報によって実行モデルを決定する。同じトランザクションに属するタスクに同じ誘因情報を持たせるため、タスクの起動時には誘因情報を起動側のタスクから起動される側のタスクに送信する（メッセージ送信時も）。これにより、誘因情報は、そのトランザクションで最初に起動されたタスクから順に関連タスクに送信される。

また、時間的な制約条件の検査は、各イベントに発生時刻に関する情報をもたせ、指定されたイベント間の経過時間を求めることによって行なう。

4.2 実行モデルの生成

実行モデルの生成には、次の2つの方法がある。

1. ソースコードから抽出した情報によるモデルの生成
2. タスク仕様からのモデルの生成

本研究では2の方針に従って実行モデルを生成する。図3に実行モデルの生成のために必要な情報を示す。タスク内のイベント情報には、タスク内で監視の対象となる操作（イベント）の発生順序を記述する。また、時間制約情報にはイベント間の時間制約を記述する。

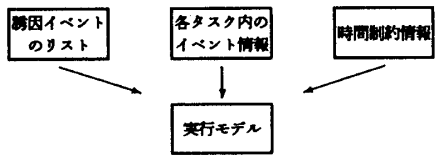

図 3:

実行モデルの生成は、次の手順で行なう。

1. 各タスクに対して記述したイベント情報から個々のタスク内の処理を表現するペトリネットを生成する。
2. 誘因とタスク間の起動関係及び、メッセージの送受信関係からペトリネットの合成を行ない、トランザクションごとに実行モデルを生成する。

図 4に3つのタスクA、B、Cから構成されるトランザクションに対する実行モデルの例を示す。A1〜A4、B1〜B2、C1は各タスク内のイベント情報であり、ペトリネットの発火条件となる。

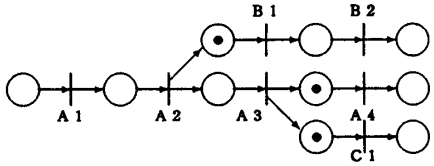

図 4:

5 おわりに

本稿では、制御システムを実行モデルを用いて監視する手法について述べた。現在、本手法による監視系を実現するため、プログラム情報からの実行モデルの生成について検討を行なっている。

参考文献

[1] R.J.LeBlanc and A.D.Robbins, "Event-Driven Monitering of Distributed Programs", Proc. of the 5th International Conference on Distributed Computing Systems, 1985,pp515-522