1N－10

オブジェクト指向開発アプローチ Crossover(3) - 実行確認系 -

前川佳春, 池田健次郎, 岸知二
NEC マイコンソフトウェア開発研究所

1 はじめに

我々は、オブジェクト指向開発アプローチ Crossover の検討を進めている [1][2]。

オブジェクト指向開発の中でもリアルタイムシステムのような実時間性を扱うシステムの開発では、設計段階における制御アーキテクチャの決定は、システムのふるまいや性能に大きな影響を与えるため重要な作業のひとつとなる。

本実行確認系は、詳細な製造を行う前に、制御アーキテクチャレベルの粗い設計モデルを作成し、従来、開発終了段階にならないと分からなかったシステムの全体的なふるまいや性能などをターゲット環境と独立した疑似環境上で早期に確認することで、開発の後戻りをなくすことをねらいとしている。

本稿では、本アプローチが基づく設計モデルの実行確認のための要件と方式について述べる。

2 制御アーキテクチャに起因する問題

制御アーキテクチャの設計に起因する代表的な問題を以下に示す。

- **外部イベントに対する応答性能**: 各タスクのプライオリティ、周期起床間隔、CPU 消費時間、通信方式 (同期 / 非同期) などの要因によりタスクの実行順序が決まり、その結果として応答時間が決まる [4]。
- **メッセージバッファのオーバフロー**: 同様の要因によりタスク間通信でボトルネックになるタスクがあるとメッセージバッファのオーバーフローが起こる。
- **タスク間でのデッドロック**: 例えば、プライオリティの低いタスクがリソースを掴んでいる状態でプライオリティの高いタスクが実行を開始し、同じリソースにアクセスしようとしてデッドロックが起こる場合など

上記のような問題を確認することが本実行確認系の目標である。

3 実行確認系の要件

実行確認系の要件を以下に示す。

- **ターゲット環境に近いプライオリティに基づいたタスクスケジューリングを疑似できること**: 実行確認系では、ターゲット環境としてリアルタイム OS を想定している。したがって、まずリアルタイム OS に近いプライオリティに基づいたタスクスケジューリングが疑似できる必要がある。このためには、タスクスケジューリングに関連する次のような要素を疑似できなければならない。
  - タスク (プライオリティ、タイムスライス、周期起床間隔)
  - タスク間通信 / 同期 (イベントフラグ、セマフォ、メールボックス)
  - 割り込み
- **実行確認系とターゲット環境とで設計モデルの変更が最小であること**: 実行確認系で作成した設計モデルのコードをできるだけ修正することなく、ターゲット環境上で実行できる必要がある。制御アーキテクチャに関わるコードに大きな差があるのでは、ターゲット環境

"Object-Oriented Development Approach: Crossover (3) - Simulator -", Yoshiharu MAEGAWA, Kenjiro IKEDA and Tomoji KISHI, NEC Corporation

上でのコードの修正と確認のためのコストが余計にかかってしまうからである。

- 設計モデルとコードを混在して実行できること: Crossover アプローチの主旨から設計モデルとコードを混在させて疑似できなければならない。

4 方式

4.1 並行動作の疑似

並行動作の単位となるタスクの疑似は、スタックフレーム切替方式 [3] とインクリメンタルリンカの関数フック機能を利用する。インクリメンタルリンカの関数フック機能により、(メンバ) 関数呼出しのタイミングでタスクの再スケジューリングを行う。これにより、タスク内部で明示的にタスク切替を行う必要がなく、よりリアルタイム OS に近いタスクスケジューリングを疑似できる。並行動作の単位は、タスククラスから継承したクラスのインスタンスに対応づける。周期起床は、スケジューラに論理時間を持たせることで実現する。

4.2 制御アーキテクチャ部品

実行確認系で、設計モデルのふるまいを確認するために、表 1に示すような基本的な制御アーキテクチャ (クラス) 部品を用意する。

表 1: 制御アーキテクチャ部品

クラス部品	説明
Task	並行動作を疑似
EventFlag	タスク間での同期を疑似
MailBox	タスク間での非同期通信を疑似
Semaphore	複数のタスク間で共有するリソースに対する同期 (排他) 制御を疑似
InterruptHandler	外部イベント用の割り込みハンドラを疑似

5 議論

前章で示した制御アーキテクチャ部品を用いて設計モデルを構築し、疑似実行することにより、制御アーキテクチャ設計段階で、ターゲット環境に近いモデルのふるまいやイベント応答性能を確認することができる。制御アーキテクチャ部品としては、H.Gomma[4] のタスクアーキテクチャの設計要素やリアルタイム OS の機能などから必要と考えられるものを選択した。

また、インクリメンタルリンカの関数フック機能は、未定義関数のデフォルト処理を行ったり、コード修正なしにインスタンス変数をモニタすることなどにも利用できるという利点がある。

オブジェクト指向の特性を生かすことで、以下のような効果も得られる。

- ターゲット環境上に実行確認系と同等の制御アーキテクチャ部品を用意することで、設計モデル自体はターゲット独立なコードとして作成できる。
- 制御アーキテクチャ部品を拡張 / カスタマイズすることで、様々なターゲット環境に対応することができる。

6 おわりに

現在、実行確認系のプロトタイプを開発中である。今後、制御アーキテクチャ部品を用いた設計モデルの構築、評価を通じ実行確認系の有効性を検証していく予定である。

参考文献

[1] 岸知二, 他: オブジェクト指向開発アプローチ Crossover (1) - 設計モデル -. 情報処理学会第 49 回全国大会 (1994).

[2] 野田夏子, 他: オブジェクト指向開発アプローチ Crossover (2) - 生成系 -. 情報処理学会第 49 回全国大会 (1994).

[3] TL ハンセン, 他:C++ アンサーブック. トッパン,(1992).

[4] Gomaa,H;:Software Design Methods for Concurrent and Real-Time Systems. Addison Wesley Publishing Company,(1993)