南山大学 数理情報学部 情報通信学科 2007 年度卒業論文要旨集

オブジェクト指向を用いた組込みシステムタスクモデリングに関する研究

2004MT067　永東 丈寛，2004MT094　清水 隆則

指導教員　青山 幹雄

1. はじめに

現在，組込みシステムではミッションクリティカルなシステムが増加しており，リアルタイム性の保証のために厳格なタスクモデリングを行う必要がある．

本研究ではオブジェクト指向のモデリング方法に基づいた一貫性のあるタスクモデリング方法を提案する．さらに，タスク統合の統合基準とタスク設計の検証方法を提案することで，タスク設計の正当性が保証されるモデリングを実現する．

2. 組込み開発へのオブジェクト指向適用

2.1. 組込み開発へオブジェクト指向を適用する利点

オブジェクト指向を適用すると，モデルを構造，機能，振舞いの視点で捉えることで全体の理解が容易になる．さらに，UML を用いてモデルの表現を統一，標準化することで，モデルの利用者間の意思疎通が容易となり，一貫性のあるタスク設計が可能となる[2].

2.2. 組込み開発へオブジェクト指向を適用する問題点

組込み開発では，並行処理の単位であるタスクを厳格に設計する必要がある．しかしオブジェクト指向を組込み開発に適用する上で，タスクとオブジェクト間の粒度の差異によってモデル上のタスク表現が困難になる．

既存のアクティブオブジェクトを用いたタスク表現では，オブジェクトとタスクの 2 つが混在するため，モデルが複雑となる．この問題を解決するために，オブジェクトとタスクとにモデリングプロセスを分離し，それぞれのモデル表現を行う必要がある．

3. アプローチ

3.1. タスクモデリングのアプローチ

本研究では，オブジェクトとタスク間の粒度の差異を考慮し，タスク設計の正当性を保証するモデリング手法を提案する．提案する手法は以下の 6 つのプロセスから成る．

(1)　オブジェクトの抽出
(2)　オブジェクト単位のモデリング
(3)　タスク単位のモデリング
(4)　初期のタスク設計の検証
(5)　タスク統合
(6)　統合後のタスク設計の検証及び評価

3.2. モデル要素の抽象化

タスクモデリングプロセスにおいて論理レベルで統一した表現を行うために，モデル要素を抽象化する．

(1)　オブジェクトの振舞いの抽象化

オブジェクトの振舞いをモデル上で明示的に表す必要がある．そこで COMET で提案されるステレオタイプを用いた組込み制御系のクラスカテゴリ分類[1]に応じて抽象化を行い，分類する(図 1).

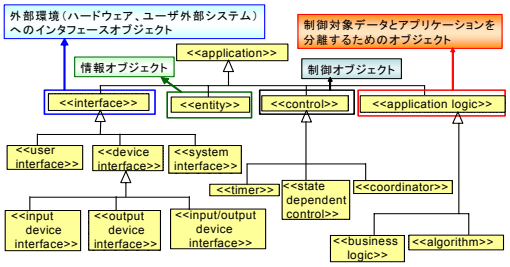

図 1 ステレオタイプによるオブジェクトの振舞いの分類

(2)　タスク間インタフェースの抽象化

RTOS のシステムコール[3]を介したタスク間インタフェースをモデル上で表現する必要がある．そのために図 2 に示す抽象化を行う．

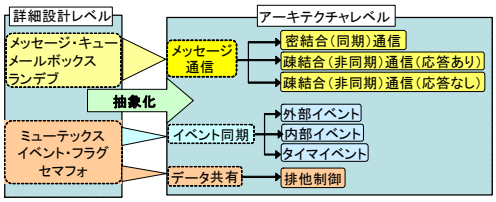

図 2　インタフェースの抽象化

4. タスク統合基準

4.1. 起動条件に基づくタスクの分類

タスクの振舞いに応じた統合基準を設定することで，体系的なタスク統合が可能となる．本研究ではタスクを図 3 に示すようなイベント起動，時間起動の 2 つの起動条件に基づいて分類し，各統合基準を設定する．

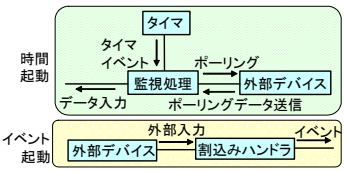

図 3　起動条件に基づくタスクの分類

4.2. イベント起動タスク統合基準

イベント起動タスクでは関連オブジェクト，データ型，起動・実行タイミングを統合基準の要素として扱い，以下のような統合基準を設定する．

(1) 起動，実行タイミング基準
どのシナリオでも各タスクの同時起動・実行が起きない．

(2) 関連オブジェクト基準
基準メッセージ出力先，入力元のオブジェクトが同一，またはステレオタイプが同一．

(3) データ型基準
起動メッセージ，出力メッセージのデータ型が同一．

4.3. 時間起動タスク統合基準

時間起動タスクでは起動周期，データ型，関連オブジェクトを統合基準の要素として扱い，以下のような統合基準を設定する．

(1) 起動周期時間基準
起動周期時間が同一，または起動周期タイミングが合う．

(2) データ型基準
ポーリングデータ，出力メッセージのデータ型が同一．

(3) 関連オブジェクト基準
ポーリング対象及びメッセージの出力先のオブジェクトが共通，またはステレオタイプが同一．

5. タスク設計の検証及び評価方法

5.1. タスク間通信の競合に基づく検証方法

組込みシステムへの非同期な外部イベントにより，タスク間通信の競合が発生する．システム処理中の競合発生の有無を検証し，タスク設計の正当性の保証が必要である．検証は以下のプロセスを用いて行う．

(1) タスク単位モデルから競合の可能性がある実行系列に含まれるタスクを抽出
(2) 抽出したタスク間の通信をシナリオに基づいたシーケンス図で表す
(3) 競合発生タイミングの有無を確認

5.2. オーバヘッドに基づく評価方法

タスク設計を評価するために，システムの処理におけるオーバヘッドを指標とした評価方法を用いる．本研究ではオーバヘッドとしてコンテキストスイッチングを用い，タスク統合前，統合後のタスク設計評価を行う．評価は以下のプロセスを用いて行う．

(1) タスク単位モデルからタスクを抽出
(2) 抽出したタスク間の通信をシナリオに基づいたシーケンス図で表す
(3) シーケンス図からコンテキストスイッチング回数を確認
(4) 統合前，統合後それぞれに(1)から(3)のプロセスを用いて，スイッチング回数の比較，タスク設計の評価

6. 提案手法によるタスクモデリングと検証

提案したタスクモデリング手法を具体的なシステムに対して適用し，提案手法の有効性を検証する．

6.1. モデリング対象システム

タスクモデリングの対象システムとして自動車のクルーズコントロールシステム(以下 CC と略記)を採用する．CC は設定速度で自動定速走行する速度制御装置である．ドライバが CC レバーを操作することで，車の加速，自動定速走行などの機能を利用する[1].

6.2. オブジェクトの抽出

システムに対する外部要求からユースケース記述を作成し，シナリオからモデリングに用いるオブジェクトを抽出する．作成したユースケース記述を表 1 に示す．

表 1 CC 制御ユースケース記述

CC 制御ユースケース	
アクタ	ドライバ
概要	ドライバが CC レバー，ブレーキ，エンジン外部入力装置をインタフェースとして入力を行うことで CC 制御を行う
事前条件	ドライバがエンジンを起動し，通常走行中.
記述	(1) ドライバが CC レバーを ACCEL に移動. CC は自動加速を開始. (以下省略)
事後条件	車が停止し，エンジンが切られている.

ユースケース記述を基に以下のオブジェクトを抽出した．

CC レバー，ブレーキ，エンジン，CC レバー割込みハンドラ ブレーキ監視，エンジン監視，ブレーキ監視タイマ エンジン監視タイマ，Acceleration，Cruiser Resumption，スロットル，スロットルアクチュエータ，CC 制御

6.3. オブジェクト単位からタスク単位のモデリング

ユースケース記述から抽出したオブジェクトを基にオブジェクト単位，タスク単位のモデルを段階的に作成する．

6.3.1. オブジェクト単位のモデリング

抽出したオブジェクト間のコミュニケーションを図 4 に示す．

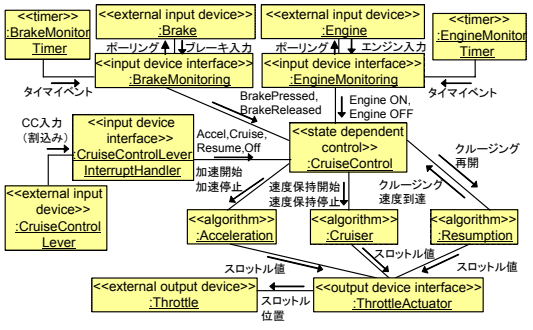

図 4 CC のオブジェクト間コミュニケーション図

ドライバが操作する CC レバー，ブレーキ，エンジン外部

入力装置によって起動する CruiseControlLever，Brake，Engine の 3 つのオブジェクトを起点とした CC の振舞いを表す．また，分類毎のステレオタイプをオブジェクトに表し，各オブジェクトの振舞いを明示する．

6.3.2. タスク単位のモデリング

タスク単位モデリングのために，オブジェクト単位モデルを用いてアクティブオブジェクトを決定する．図 4 を基に CC の機能を分析した結果，外部監視機能，制御機能，クルージングのための速度調整機能，スロットルへの出力機能が並行的な機能のまとまりとして妥当だと判断する．結果として以下のオブジェクトをアクティブオブジェクトに決定した．

CruiseControl，BrakeMonitoring，EngineMonitoring
Resumption，Acceleration，Cruiser
CruiseControlLeverInterruptHandler

抽出した各アクティブオブジェクトを起点とするタスクを分類し，タスク間のコミュニケーションを図 5 に示す．

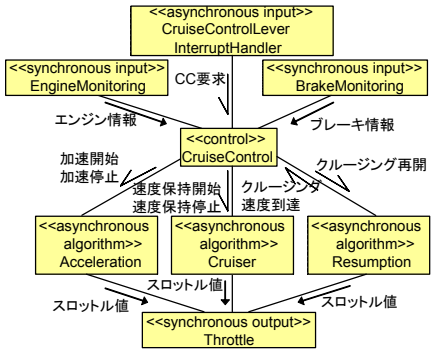

図 5 CC のタスク間コミュニケーション図

図 5 から，BrakeMonitoring タスク，EngineMonitoring タスクからの周期的な入力，CruiseControlLeverInterruptHandler タスクからの非同期な入力を確認できる．これら 3 つの外部入力のインタフェースタスクを起点とした処理の実行系列が存在することが分かる．

6.4. 初期のタスク設計検証

タスク設計を行ったシステムの正当性を検証する．図 5 に基づいて分析すると，CruiseControl タスクへの各監視タスクによる入力と Throttle タスクへの各 Algorithm タスクによる入力で競合の可能性が考えられる．

各タスクへの入力を分析した結果，通常シナリオにおける CC レバーと各周期監視のタスクから CruiseControl タスクへの入力で競合可能性が確認された．これらの競合は実装レベルにおけるセマフォの導入で入力の相互排他性を保証，もしくはメールボックスによるメッセージのキューイングによって入力順を規定することで解決可能である．ゆえに，システム処理を停止する競合発生の可能性は低い．

結果として想定される通常シナリオにおいて，ドライバが通常操作をする場合の正当性は保証される．

6.5. タスク統合

初期段階のタスク設計の検証によって正当性が保証されたため，統合可能タスクの選定を行う．タスク統合にあたり，イベント起動タスク，時間起動タスクを分類し，各振舞いのタスク統合を行う．コミュニケーション図からタスクの振舞いを分析した結果，図 6 に示すように各周期監視タスクの統合，そして各 Algorithm タスクの統合を行う．

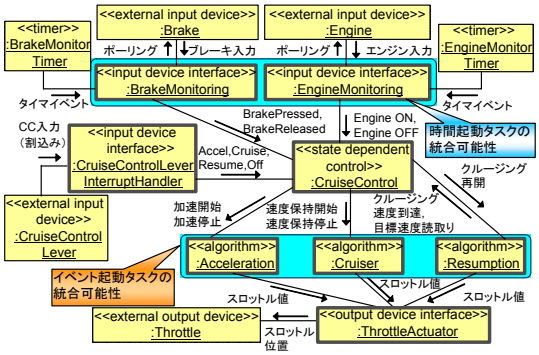

図 6 統合可能タスクの選定

(1) イベント起動タスクの統合

提案したタスク統合基準に従い，イベント起動タスクの統合可能性を分析する．

a) 関連オブジェクト

各タスク共に，入力元が CruiseControl オブジェクト，出力先が ThrottleActuator オブジェクトである．

b) データ型

入力データは共に CC 命令メッセージ，出力データは共にスロットル値であるため，データ型の統一は可能である．

c) 起動，実行タイミング

各 Algorithm タスクは CC レバーの各ポジションに対応して起動，実行をするため，同時実行は起こらない．

結果としてイベント起動タスクの統合は可能である．

(2) 時間起動タスクの統合

イベント起動タスクと同様に，統合基準に従って統合可能性を分析する．

a) 関連オブジェクト

ポーリング対象のステレオタイプが共通であるがオブジェクトが異なる．出力先は共に CruiseControl である．

b) データ型

BrakePressed，BrakeReleased，そして EnginON，EngineOFF を扱っている．扱うデータ数が等しく，データの種類も近似するため，データ型の共通性は高いといえる．

c) 起動周期時間

各監視の対象が CC の起動，停止に直結するクリティカルな部分であるため，それぞれ短い周期のポーリングが必要である．ゆえに，各起動周期時間を近似させることに問題はないと判断できる．

結果として，関連オブジェクトの基準は完全に満たしてい

ないが，データ型及び起動周期時間の基準を満たしているため，統合可能と判断する．

各タスクの統合が可能だと判断できたため，タスク統合を行う．統合後のタスク間コミュニケーションを図 7 に示す．

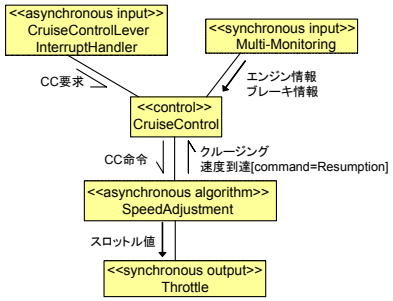

図 7 統合後のタスク間コミュニケーション図

6.6. 統合後のタスク設計の検証及び評価

(1) タスク設計の検証

タスク統合後のシステムの競合発生を検証する．図 8 を基に，CruiseControl タスクへの入力における競合の可能性があると判断した．

CruiseControl タスクへの入力を分析した結果，CruiseControlLeverInterruptHandler タスクと Multi-Monitoring タスクからの入力において競合の可能性が確認された．初期の検証と同様に，入力の順序付けを適切に行うタスク間インタフェースを用意することで解決可能だと考えられる．ゆえにシステムの正当性の保証は可能といえる．

(2) タスク設計の評価

タスク統合前と統合後のタスク設計におけるコンテキストスイッチング回数を調べ，その値を比較することでタスク設計の評価を行った．なお，起動頻度の違いから時間起動タスクとイベント起動タスクのスイッチングを分けて分析する．分析によるスイッチングの比較結果を表 2 に示す．なお，イベント起動の loop は Throttle への周期通信回数，時間起動の loop は CruiseControl への周期通信回数を表す．

表 2 コンテキストスイッチング回数の比較

タスク		コンテキストスイッチング回数	タスク数
(1)統合前	イベント起動	12+loop 回数×8	5
	時間起動	1+loop 回数×8	2
(2)統合後	イベント起動	10+loop 回数×8	3
	時間起動	0	1
(1)－(2)	イベント起動	2	2
	時間起動	1+loop 回数×8	1

統合の結果，コンテキストスイッチング回数は各タスク共に減少した．Algorithm タスクの統合で CC 命令イベント毎のスイッチングが軽減され，周期監視タスクの統合でタイマイベント毎のスイッチングがなくなったと考えられる．

結果として，タスク統合を行ったことでシステムのオーバヘッドが低下した．また，タスク数が全体で 3 つ減少したことからタスク設計の複雑さも軽減したと考えられる．

7. 評価と考察

(1)モデリングプロセス

オブジェクトによる詳細設計とタスクによる実行全体の流れをモデル毎に分離して表現するが可能となった．オブジェクト単位モデルでは細粒度の動的，静的なシステム設計が可能である．タスク単位モデルでは，並行動作する処理を中心としたモデル表現が可能である．

(2)タスク統合基準

タスクの起動条件に基づく統合基準を決定することで，体系的なタスク統合が可能となった．しかし，提案したタスク起動条件以外の分類が考えられるため，より厳密な統合基準を考える必要がある．

(3)タスク設計検証，評価

実行系列間のタスクの競合を指標とすることで，タスク設計の正当性を検証することが可能となった．タスク設計の評価ではコンテキストスイッチングを指標とし，実行におけるオーバヘッドを評価することで，アーキテクチャのレベルでのシステム評価が可能となった．これによって実装段階に移る前にプロトタイプ的にタスク設計の正当性を保証できるため，開発プロセスの向上につながる．

8. 今後の課題

(1) タスク統合基準の精度の向上

タスク統合基準ではイベント起動タスクと時間起動タスクに基づいた基準を提案したが，より詳細にタスクの振舞いを分類することで統合基準の精度を向上する必要がある．

(2) 検証における指標の追加

検証における指標の 1 つとしてタスク間通信の競合を用いたが，より効果的に設計の正当性を保証するためにはより多くの指標を用いて検証を行う必要がある．

9. まとめ

本研究では，オブジェクト指向モデリングに基づく統一的なタスクモデリング方法を提案した．オブジェクト指向を適用する上で問題となるタスクの表現を，オブジェクト単位，タスク単位に分離したモデリングを行うことで解消した．タスクモデリングの正当性を保証するためにタスク統合基準，タスク設計検証及び評価手法を提案し，その提案手法をクルーズコントロールシステムに適用して有効性を評価した．

参考文献

[1] H.Gomaa, Designing Concurrent, Distributed, and Real-Time Applications with UML, Addison Wesley, 2000.

[2] 渡辺 博之ほか，組み込み UML，翔泳社, 2002.

[3] Micro-ITRON4.0 仕様， http://www.ertl.jp/ITRON/SPEC/FILE/micro-402j.pdf.