サービス指向車載ソフトウェアの協調制御における タイミング設計方法の提案と評価

永東 丈寛[†1]　中道 上[†2]　青山 幹雄[†2]　佐藤 洋介[†3]　岩井 明史[†3]

南山大学大学院 数理情報研究科[†1]　南山大学 情報理工学部 ソフトウェア工学科[†2]　株式会社デンソー[†3]

1. はじめに

サービス指向アーキテクチャ(SOA: Service-Oriented Architecture)に基づく車載プラットフォームが研究されている[1]が，サービス間の協調のタイミングを厳密に保証する必要がある．

本稿ではタイミング制約に基づくサービスインタフェースの拡張に着目し，サービス協調のタイミング設計，検証方法を提案する．制約検証の支援ツールを実装し，有効性を評価する．

2. 車載ソフトウェアへの SOA の適用課題

サービス指向車載プラットフォームでは，センサ入力からアクチュエータ出力までの End-to-End のサービス協調における厳密なタイミング制約保証が必要である．

3. 関連研究

実行シナリオ毎の制約を記述可能なインタフェースを持つコンポーネントを組み合わせ，WCET を評価する方法が提案されている[2].

また，QoS に関わる制約情報のモデル記述を行う言語である QoSCL(QoS Constraint Language)が提案されている[3].

4. 問題解決へのアプローチ

タイミング制約を形式的に記述できるサービスインタフェースの拡張を提案し，End-to-End のサービス協調の制約を分析可能な拡張インタフェースを定義する．

さらに，モデル駆動開発に基づき，拡張インタフェースの制約の段階的な定義を可能とすることで，End-to-End のサービス協調のタイミング制約保証を実現する設計，検証方法を提案する．

5. モデル駆動タイミング設計方法

5.1. タイミング設計プロセス

提案するタイミング設計は，(I)拡張インタフェースのタイミング制約をモデル化するモデル化プロセスと，(II)モデルに基づきタイミング制約検証を行う検証プロセスから成る(図 1).

以下，各プロセス実行の流れを示す．

(I) モデル化プロセス

A) 機能抽出

システムへの機能要求をユースケースとして記述し，ユースケース毎にシナリオを記述する．

B) サービスモデルの作成

制約定義を付加するサービスインタフェースとサービス協調の実行シナリオをモデル化する．

C) コンテキストの抽出

タイミング制約保証が必要な実行系列を，シーケンス図からコンテキストとして抽出する．

D) コントラクトモデルの作成

抽出したコンテキストとサービスモデルに基づき，各サービスの拡張インタフェースに制約を定義する．

(II) 検証プロセス

a) タイミング設計

サービスの振舞いから抽出される時間要求に基づいて決定するタイミング制約値を，コントラクトモデルの制約定義に割り当てる．

b) 制約検証

コントラクトモデルに割り当てた制約値に基づきコンテキスト毎の WCET を算出し，End-to-End のタイミング制約に対する検証を行う．

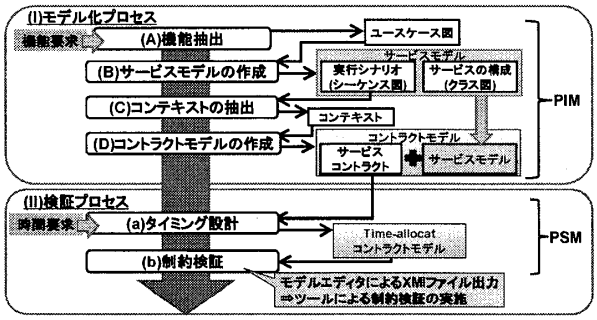

図 1 : タイミング設計プロセス

5.2. サービスコントラクトのモデル化

モデル化プロセスの流れと，モデル間の関係を図 2 に示す．

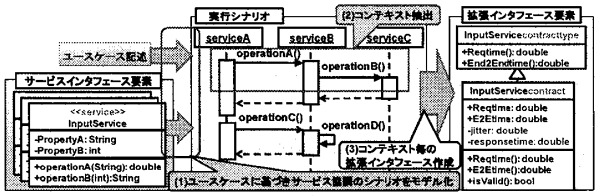

図 2 : コントラクトモデル作成プロセス

Design Method for Timing of Cooperative Control for Service-Oriented Automotive Software.

†1 Takehiro Nagato, Graduate School of Mathematical Sciences and Information Engineering, Nanzan University.

†2 Noboru Nakamichi, Mikio Aoyama, Department of Software Engineering, Nanzan University.

†3 Yosuke Sato, Akihito Iwai, DENSO CORPORATION.

5.2.1. サービスモデル定義

サービスインタフェースの構文要素(サービス名，オペレーション，属性)と，サービス協調の実行シナリオをサービスモデルとして定義する．

5.2.2. コンテキスト定義

外部イベントによるトリガ(センサ)を起点とするサービス協調の出力(アクチュエータ)までの実行系列をコンテキストとして定義する．コンテキストは，サービス協調の実行シナリオから End-to-End の実行系列として抽出する．

5.2.3. コントラクトモデル定義

抽出したコンテキスト中のサービスの拡張インタフェースをコントラクトとして定義する．

コントラクトでは，コンテキスト中のサービスの振舞いから抽出される時間特性を用いて，OCL (Object Constraint Language)に基づく QoSCL の形式で End-to-End の制約を記述する．

5.3. 制約検証ツール ContractValidator

制約検証を支援する ContractValidator を Java で実装した．モデルエディタで定義したコントラクトモデルを XMI 形式に変換し，End-to-End の実行時間制約に対する WCET を検証する．ContractValidator は，指定されたコンテキスト定義(図 3b)に従った実行系列毎に，コントラクトモデル(図 3a)の XMI ファイルに記述される制約式を解析し，タイミング制約値に基づく WCET を計算する．WCET に基づき各コンテキストの End-to-End の実行時間制約を満たさないサービスを抽出し，エディタに表示することで(図 3c)，ボトルネックを可視化し，タイミング設計の妥当性確認を支援する．

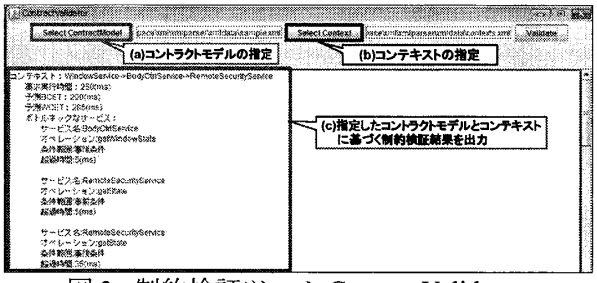

図 3 : 制約検証ツール ContractValidator

6. リモートセキュリティへの適用と評価

提案方法を自動車のリモートセキュリティへ適用し，有効性を評価した．

6.1. サービスのモデル化とタイミング制約定義

リモートセキュリティを実現する一連のサービスとサービス協調のシナリオをモデル化し，制約保証が必要なコンテキストを抽出した(図 4)．

窓，ドア，ライトの状態検知を起点としたうっかり通知と，ユーザの操作を起点としたリモート操作のコンテキストを抽出した．

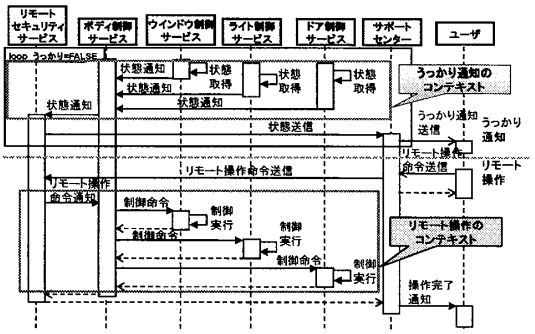

図 4 : リモートセキュリティの実行シナリオ

抽出したうっかり通知のコンテキストのタイミング制約記述を図 5 に示す．

```
context WindowService :: getWindowState()  --WindowServiceの制約記述
  inv : self.jitter > 0 and self.jitter < 25
  inv : self.Reqtime = 50
  inv : self..E2Etime = self.Restime + self.jitter
  post : self.E2Etime < self.Reqtime
context BodyCtrlService :: getWindowState(String:state)  --BodyCtrlServiceの制約記述
  inv : self.Reqtime = 120 + self.windowContract.Reqtime
  inv : self.ntwrk_delay > 0 and self.ntwrk_delay < 30
  inv : self.E2Etime = self.windowContract.E2Etime + self.ntwrk_delay + self.getWtime
  pre : self.windowContract.E2Etime < self.windowContract.Reqtime
  post : self.E2Etime < self.Reqtime
context RemoteSecurityService::getState(String:CarState)  --RSSServiceの制約記述
  inv : self.Reqtime = 80 + self.bodyContract.Reqtime
  inv : self.ntwrk_delay > 0 and self.ntwrk_delay < 30
  inv : self.E2Etime = self.bodyContract.E2Etime + self.ntwrk_delay + self.getStatetime
  pre : self.bodyContract.E2Etime < self.bodyContract.Reqtime
  post : self.E2Etime < self.Reqtime
```

図 5 : うっかり通知の制約記述

6.2. 制約検証とボトルネックの発見

図 5 の例では，ネットワーク遅延 ntwrk_delay と周期ジッタ jitter が最大となる実行時間を WCET として算出した．ボディ制御サービス，リモートセキュリティサービスの各 WCET が実行時間制約を超過することを発見した．

6.3. 評価

提案したタイミング設計方法により，車載サービスの時間特性に基づき End-to-End の実行時間制約を厳密に定義可能となった．

さらに，ContractValidator によって実行系列毎の WCET が予測可能となり，タイミング設計のボトルネックの発見が容易になった．以上から，提案方法の制約保証への有効性を確認できた．

7. まとめ

制約定義可能な拡張インタフェースに基づくタイミング設計，検証方法を提案し，それに基づく制約検証ツール ContractValidator を実装した．提案方法を例題へ適用し，有効性を示した．

参考文献

[1] 青山幹雄, ほか, 車載ソフトウェアのサービスプラットフォームのモデルとアーキテクチャ, 自動車技術会, Oct. 2008, No.97-08, pp. 21-26.

[2] J. E. Kim, et al., Extracting, Specifying and Predicting Software System Properties in Component Based Real-Time Embedded Software Development, Proc. ICSE'09, May. 2009, pp. 28-38.

[3] S. Gerard, et al., Model Driven Engineering for Distributed Real-time Embedded Systems, Hermes Science, 2006.