Vol.2010-SE-167 No.14
2010/3/18

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]

サービス指向アーキテクチャ(SOA)に基づく車載ソフトウェアの協調制御におけるタイミング制約の記述と評価方法を提案する．SOA を車載ソフトウェアへ適用する上で，サービス協調の End-to-End のタイミング制約を満たす設計が必要である．本稿ではサービス協調の時間特性に着目し，End-to-End のサービス協調実行のタイミング設計方法を提案する．End-to-End のサービス協調シナリオをコンテキストとして定義し，インタフェースに時間特性を拡張してコントラクトとして定義して，End-to-End のタイミング制約記述を実現する．OCL に基づいてタイミング制約を形式記述し，モデルベースの制約検証方法を示す．制約検証で用いる WCET 検証ツールのプロトタイプを作成し，提案方法をリモートセキュリティサービスへ適用し，その有効性を示す．

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

Takehiro Nagato[1] and Noboru Nakamichi[2] and Mikio Aoyama[2] and Yosuke Sato[3] and Akihito Iwai[3]

We propose a design method for timing of cooperative control for service-oriented automotive software. To apply SOA (Service-Oriented Architecture) to automotive software, we have to assure end-to-end timing constraints along with cooperation of services. We propose an assuring method for end-to-end timing constraints by focusing on the timing properties. We define the end-to-end scenario of cooperation of services as a context, and the timing properties as a contract of an interface. We define the contract with an extension of OCL, and validate end-to-end timing constraints based on the model. We developed ContractValidator, a tool of WCET validation, and evaluated the effectiveness of the proposed method by application to the remote security service.

1. はじめに

近年，車載ソフトウェアの大規模化により機能間の相互作用が複雑化しており，ソフトウェアの再利用や拡張が困難である．また，異なるベンダのソフトウェア間や車内外のネットワーク間のインタフェースやプラットフォームが異なるため，協調制御が困難である．これらの問題を解決するために，サービス指向アーキテクチャ(SOA: Service-Oriented Architecture)に基づく車載ソフトウェアプラットフォームが研究されている[1][7]が，サービス間の協調実行タイミングに対する制約保証が必要である．

本稿では，時間制約定義を付加したサービスモデルによるモデル駆動タイミング設計方法と，モデルベース制約検証方法を提案する．時間特性に基づき拡張したサービスインタフェースをモデル化し，サービス協調のタイミング制約情報を付加したサービスコントラクトモデルを定義する．サービスコントラクトモデルに基づくモデルベース制約検証方法を提案し，設計レベルでのタイミング制約保証の実現を図る．

2. 車載ソフトウェアへの SOA の適用課題

サービス指向プラットフォームにおけるサービス協調のタイミング制約のモデルを図 1 に示す．自動車制御のためのサービス協調は，センサ入力を受けるサービス(Provided Service)，入力に応じた制御処理を実行するサービス(Control Service)，制御結果毎にアクチュエータへの出力を行うサービス(Required Service)によって行われる．そのため，センサ入力からアクチュエータ出力までの End-to-End のサービス協調の厳密なタイミング制約保証が必要である[8][9].

また，自動車制御のためのサービスの振舞いとして，時間駆動処理，イベント駆動処理が挙げられ，各処理固有の時間特性が存在する．時間駆動処理における時間特性の例として，起動周期，周期ジッタ，1 周期毎の WCET (Worst-Case Execution Time)が挙げられる．また，イベント駆動処理における時間特性の例として，メッセージ到着間隔に対するジッタや，時間駆動処理と同様に WCET が挙げられる[3]. 厳密なタイミング制約保証を実現するためには，これらの自動車制御におけるサービスの周期的，非周期的な処理から抽出できる時間特性に基づいたタイミング制約の設計と検証が必要である．

以上から，サービス協調のタイミング制約を厳密に保証するためにはサービスの振舞いに基づく時間特性に着目した End-to-End のタイミング設計，検証が必要である．

[1] 南山大学大学院 数理情報研究科
Graduate School of Mathematical Sciences and Information Engineering, Nanzan University
[2] 南山大学 情報理工学部 ソフトウェア工学科
Department of Software Engineering, Nanzan University
[3] 株式会社デンソー
DENSO CORPORATION

しかし，従来のサービス指向に基づく開発ではタイミング制約のような非機能特性に基づく設計を行う開発方法は確立していないため，厳密なタイミング制約保証が困難である．

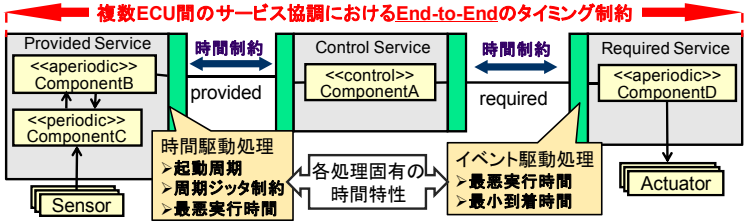

図 1　サービス協調のタイミング制約モデル

3. 関連研究

車載ソフトウェアの実行シナリオ毎の制約情報を定義可能なインタフェース(AI: Analytic Interface)を持つコンポーネントモデルに基づいて，WCET を分析，評価する方法が提案されている[6]．提案されるコンポーネントモデルでは，システム実行のシナリオ毎のソフトウェアの振舞いに関わる制約をインタフェースのメタ情報として定義している．このインタフェース定義に基づいて，コンポーネントへの入力に対応するシナリオの制約情報を抽出し，推論フレームワークを用いた制約分析を行い，WCET 評価を実現している．しかし，サービス指向に基づいた制約分析は実現していない

また，QoS に基づく拡張インタフェース情報をモデル化するためのモデル記述言語として，QoSCL(QoS Constraint Language)が提案されている[5]．QoSCL は，QoS に関わる制約情報を記述するために UML2.0 のメタモデルを拡張したモデル記述言語である．QoSCL の拡張要素を以下に示す．

(1) Dimension

Operation 要素を継承した要素であり，QoS に関わる制約定義を行うために用いられる特性である．遅延時間や実行時間等の時間制約値を定義可能である．

(2) ContractType

Interface を継承しており，Contract を定義するための Dimension の集合の型である．サービスインタフェースの抽象定義部分に相当し，制約定義を行う型として用いる．

(3) Contract

ContractType で定義する Dimension の集合の型に基づく制約を記述する要素である．

(4) QoSComponent

インタフェースに ContractType を定義するコンポーネントである．シンタックス定義に Contract で記述される制約情報を付加した拡張インタフェースを定義している．

4. 問題解決へのアプローチ

問題解決へのアプローチを以下に示す(図 2).

(1) モデル駆動開発に基づくタイミング設計

拡張インタフェースのタイミング制約情報を段階的にモデル化することで，サービスのコンテキスト毎の End-to-End のタイミング設計が可能と考える．また段階的なモデルの詳細化によって，論理レベルから一貫してサービス協調のタイミング制約を保証し，モデルベースの制約検証を行うことが可能と考える．

(2) サービスインタフェースの拡張

インタフェースにタイミング制約情報を付加する[4]ことで，インタフェースの Syntax 定義に基づいてコンテキストを分類し，コンテキスト毎の End-to-End のサービス実行におけるタイミング制約を定義できる．そのため，インタフェースのタイミング制約による拡張に基づいてタイミング設計を行うことが可能と考える．

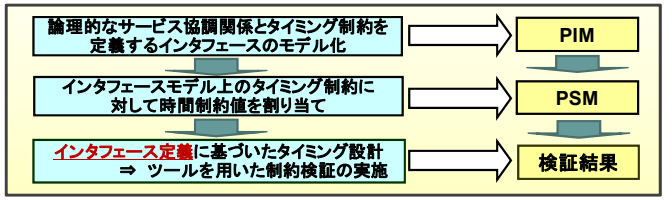

図 2　モデル駆動開発に基づくタイミング設計

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プロセスは，以下の 2 つのプロセスで構成される(図 3).

(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 のタイミング制約の検証を行う．

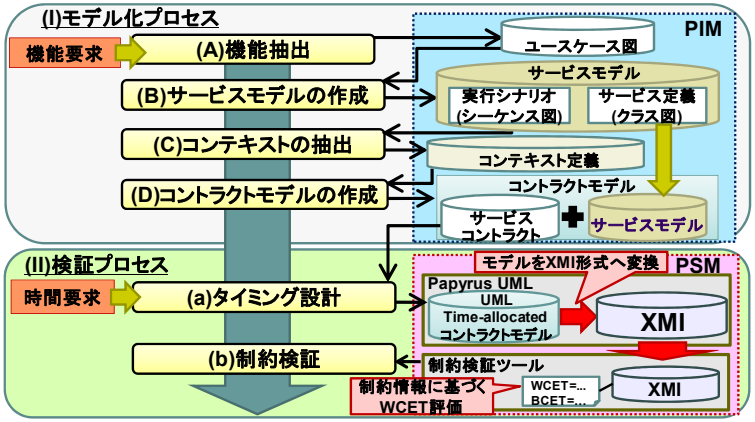

図 3　タイミング設計プロセス

5.2　モデル化プロセス

5.2.1 モデル化プロセスの工程

図 4 に示す以下のモデル化プロセスの工程を実行する．

(A) 機能抽出

システムへの要求から必要な機能を抽出し，モデル化する．抽出した機能はユースケース図でモデル化し，ユースケース毎のシナリオをユースケース記述としてモデル化する．機能抽出以降のプロセスにおけるサービスモデルの作成やコンテキスト抽出は，1 つのユースケース単位に行う．

(B) サービスモデルの作成

機能抽出によってモデル化したユースケース毎のサービス構成をクラス図でモデル化し，ユースケース記述に従ったシナリオをシーケンス図としてモデル化する．

(C) コンテキストの抽出

センサ入力からアクチュエータ出力までの End-to-End の実行系列をシーケンス図から抽出する．

(D) コントラクトモデルの作成

各サービスの拡張インタフェースの制約情報をモデル化する．拡張インタフェースが定義する制約情報として，タイミング制約に関わる時間特性と，時間特性を用いて OCL (Object Constraint Language)[13]に基づき形式記述した時間制約式を定義する．これらの情報を持つ拡張インタフェースをコントラクトとしてモデル化する．

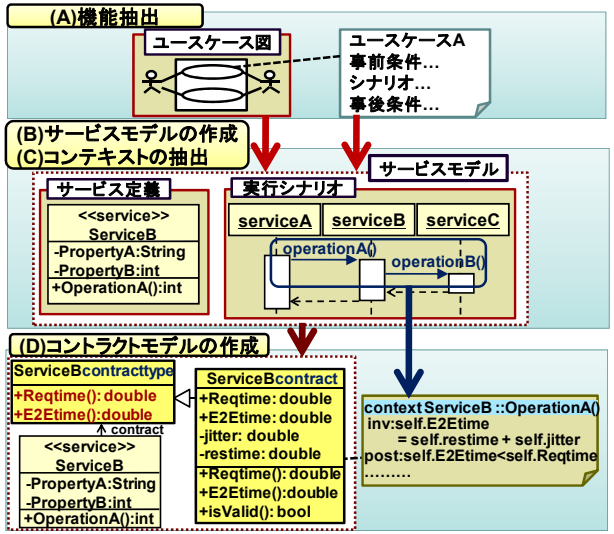

図 4　モデル化プロセスの実行工程

5.2.2 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．

5.2.3 コンテキスト定義

車載ソフトウェアとしてのサービスの振舞いは，センサから入力されたデータの取得，入力毎の制御処理の実行，制御結果毎のアクチュエータへの出力という流れで構成される．つまり，センサ入力毎にアクチュエータ出力までのサービス協調の実行系列が存在する．

そこで，センサ入力という実行条件に基づき，センサ入力からアクチュエータ出力までという車載システム内部の End-to-End の実行系列をコンテキストとして定義し，タイミング制約の保証を行うシナリオとして利用する．

5.2.4 コントラクトモデル定義

QoSCL を拡張したコントラクト記述方法に基づき，コントラクトモデルを以下のモ

デル要素で構成する．

(1) ServiceContractType

ContractType をサービス協調の特性に基づいて拡張することで，時間特性の集合の型定義を可能とする．

コントラクトを持つサービスが拡張インタフェース上で定義する End-to-End の実行時間制約に関わる時間特性を定義している．End-to-End の実行時間制約に関わる時間特性として，End-to-End のサービス実行に対して要求される時間を表す Reqtime と，タイミング設計時に割り当てる実行時間を表す E2Etime が挙げられる．

(2) ServiceContract

Contract をサービス協調の特性に基づいて拡張することで，End-to-End のサービス実行における振舞いに基づき抽出される時間特性の定義が可能である．

定義する時間特性として，ServiceContractType で型定義した Reqtime と E2Etime，サービスの振舞いに基づき抽出した時間特性(ジッタや応答時間等)がある．Reqtime と E2Etime はサービスが拡張インタフェース上で定義する制約情報である．一方，サービスの振舞いに基づき抽出した時間特性は Reqtime と E2Etime を記述するために利用されるため，ServiceContract のみで定義される．

以上のコントラクトモデルで定義した時間特性を用いて，OCL に基づく形式記述によってタイミング制約を定義する．制約定義は事前条件，事後条件，不変条件で構成し，時間特性の制約値割り当てやサービス毎の厳密な制約定義を実現する．

5.3 検証プロセス

検証プロセスでは，タイミング設計プロセスで制約値割り当てを行ったコントラクトモデルに基づき，ツールによる制約検証を行うことで，タイミング設計の妥当性確認を行う．検証プロセスでは，以下の 2 つのプロセスを実行する(図 5).

(1) 制約値の割り当て

時間要求に従って，コントラクトモデル上の制約記述に対してタイミング制約値を割り当てることで，厳密なタイミング制約情報をコントラクトモデルに付加する．割り当てるタイミング制約値は，End-to-End の実行時間である E2Etime を構成する各時間特性の制約値と，サービス毎の実行時間制約である Reqtime に対する制約値である．

(2) コントラクトモデルの変換

モデルエディタ上で制約値を割り当てたコントラクトモデルを記述し，制約検証ツールの入力形式である XMI へ変換する．また，コントラクトモデルの情報に基づきコンテキストを XML で定義する．モデル変換を行うツールとして，Papyrus UML[10]を利用した．

(3) タイミング設計の妥当性確認

変換した各モデル定義ファイルを制約検証ツールへ入力し，WCET に基づく制約検証を行う．制約検証の結果から，タイミング設計の妥当性を確認する．

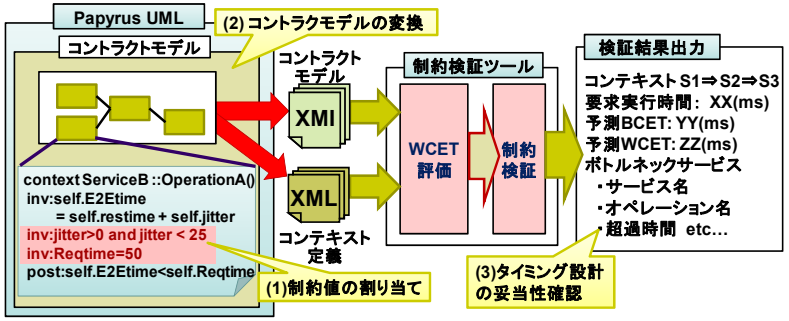

図 5 検証プロセスの実行工程

6. ContractValidator の実装

制約検証プロセスにおける WCET 検証に基づくタイミング設計の妥当性確認を支援するツール ContractValidator を，Eclipse 上で Java を用いて実装した．

6.1 ContractValidator のアーキテクチャ

図 6 に ContractValidator のアーキテクチャを示す．

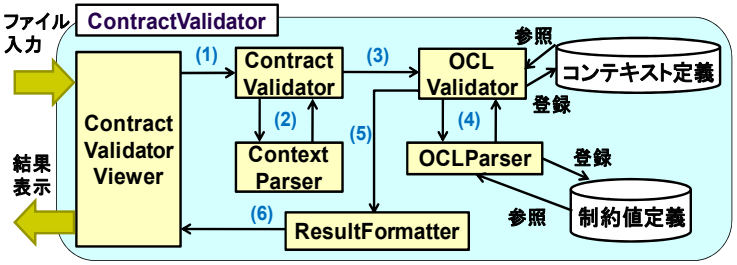

図 6 ContractValidator のアーキテクチャ

上述の ContractValidator のアーキテクチャに基づく実行の流れを以下に示す．

(1) ユーザ操作イベントに基づくデータ受け渡し

ContractValidatorViewer が提供する GUI 上で，ユーザが XMI 形式のコントラクトモデルと，XML 形式のコンテキスト定義のファイルを指定する．

次に，検証開始のボタンをユーザが押下することでイベントリスナがイベントを検知し，各ファイルのデータが ContractValidator へ受け渡される．

(2) コンテキスト分析
受け取ったコンテキスト定義ファイルデータの DOM から，コンテキスト定義要素を抽出するために，ContextParser へファイルデータを受け渡し，抽出処理を行う．
(3) コントラクト及びコンテキスト定義のデータ受け渡し
抽出したコンテキスト定義要素とコントラクトモデルの DOM を OCLValidator へ受け渡す．
(4) 制約記述検証
コンテキスト定義要素に基づき，OCLParser に単一コンテキスト毎の制約検証を実行させる．コンテキスト定義に従って，コントラクトモデル DOM から検証対象コンテキストのサービスのコントラクトを抽出し，制約定義に従った制約値の情報登録と制約情報に基づく WCET 計算を行う．
WCET に基づく制約検証を行い，コンテキスト毎のボトルネックや WCET，BCET 等の検証結果を抽出する．
(5) 検証結果受け渡し
全コンテキストの検証完了後，ResultFormatter へ検証結果データを受け渡す．
(6) 出力形式への変換
受け取った検証結果データを，GUI に出力可能な形式に変換する．そして，ContractValidatorViewer へデータを受け渡すことで，GUI 上に検証結果を出力する．

6.2 実行環境と実装規模

表 1 と表 2 に ContractValidator の実行環境と実装規模を示す．

表 1 実行環境

OS	Windows Vista SP2
Java 実行環境	JRE1.6.0_07
Eclipse	3.5.0
XML パーサ	JAXP1.4

表 2 実装規模

クラス数	8 個
LOC	1040 行

6.3 ContractValidator を用いた制約検証方法

ContractValidator は，指定されたコンテキスト定義(図 7b)に従った実行系列毎に，コントラクトモデル(図 7a)の XMI ファイルに記述される制約式を解析する検証を行い(図 7c)，タイミング制約値に基づく WCET を計算する．
検証後，コントラクトモデルで定義される制約記述を表示する(図 7d)．また，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，検証結果として表示する(図 7e)．検証結果によって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．

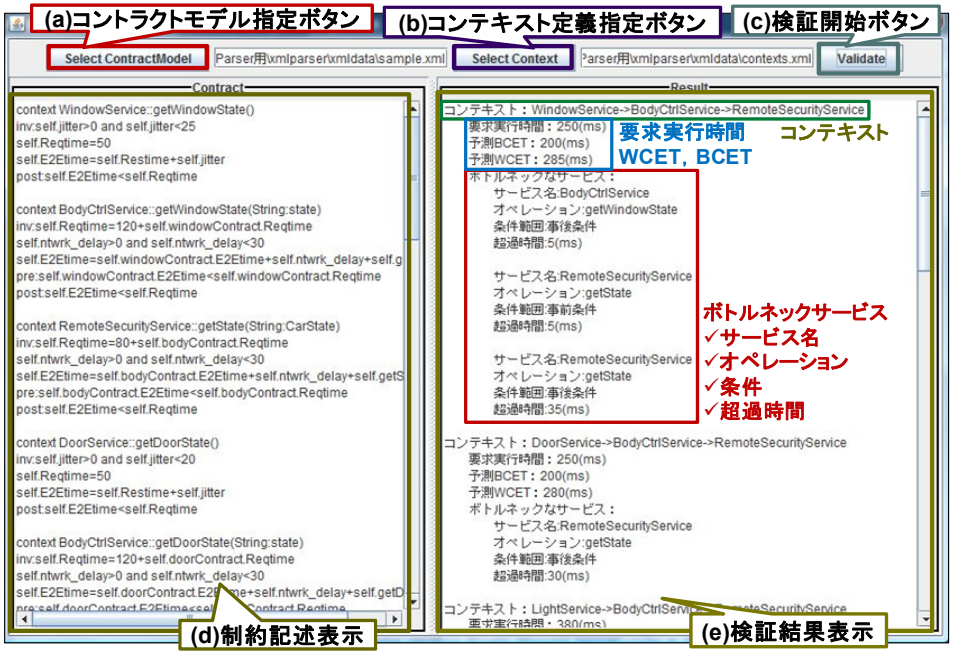

図 7 ContractValidator のユーザインタフェース

7. 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．

7.1 適用対象システム

提案するタイミング設計方法の妥当性を評価するために，Lexus のリモートセキュリティサービス[12]に適用した．リモートセキュリティは，ドアロックの閉め忘れや，ハザードランプの消し忘れ等をした場合にユーザへの通知を行ううっかり防止機能と，車両が盗難にあった場合の車両位置の特定やエンジン停止，ステアリングロック等を遠隔的に実現する盗難防止機能を実現するサービスである．ユーザへの通知や車両の遠隔操作はサポートセンターを介することで実現する．

7.2 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，以下のコンテキストを抽出した(図 8).

(1) うっかり通知のコンテキスト

End-to-End のサービス協調の実行系列として，各車両状態の取得を起点としたサービス協調に着目する．この実行系列では，各車両状態を取得後，BodyCtrlService による状態データの統合が行われ，RemoteSecurityService に送信される．その後，RemoteSecurityService がサポートセンターに車両状態を送信することで，車内のサービス協調は終了する．以上から，各車両状態取得オペレーションの開始からサポートセンターへの車両状態送信オペレーション実行完了までの End-to-End のサービス協調の流れを，コンテキストとして抽出した．

(2) リモート操作のコンテキスト

リモート操作における End-to-End のサービス協調の実行系列は，サポートセンターが RemoteSecurityService に操作命令を送信する処理を起点として発生する．操作命令受信後，RemoteSecurityService が操作命令通知を BodyCtrlService に送信し，操作命令情報に従って各車両制御へ実行系列が分岐する．これらの分岐した実行系列毎の処理が完了するまでの End-to-End のサービス協調の流れを，コンテキストとして抽出した．

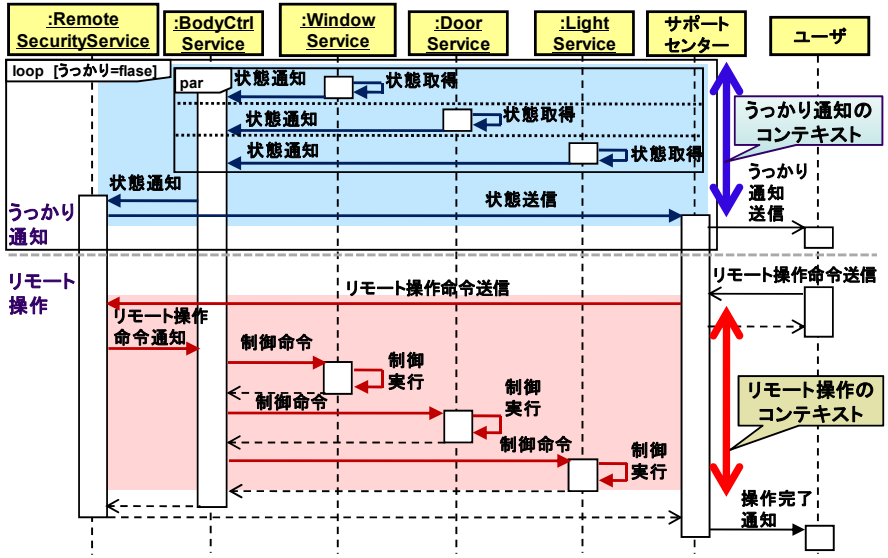

図 8 リモートセキュリティの実行シナリオとコンテキスト

うっかり通知のコンテキストでは，周期的な車両状態取得処理を行う WindowService，DoorService，LightService が存在する．これらの時間特性として，周期処理に関わる応答時間，周期処理のジッタを定義する．また，非周期処理かつ一方向のメッセージングを行う BodyCtrlService と RemoteSecurityService には，時間特性としてサービス実行時間とネットワーク遅延時間を定義する．

リモート操作のコンテキストでは，ユーザ操作命令の通知を起点とした一方向のメッセージングを全サービスが実行する．そのため，各サービスは実行時間とネットワーク遅延時間を時間特性として定義する．

以上，抽出したコンテキスト毎に，サービスの振舞いに基づく時間特性を抽出し，制約定義を行うためのコントラクトモデルを作成した(図 9). コントラクトモデルでは，サービスコントラクト間の依存関係をロールによって定義し，時間特性間の制約値の参照を可能としている．時間特性間の制約値参照によって，制約記述式における End-to-End の実行時間制約定義を実現する．

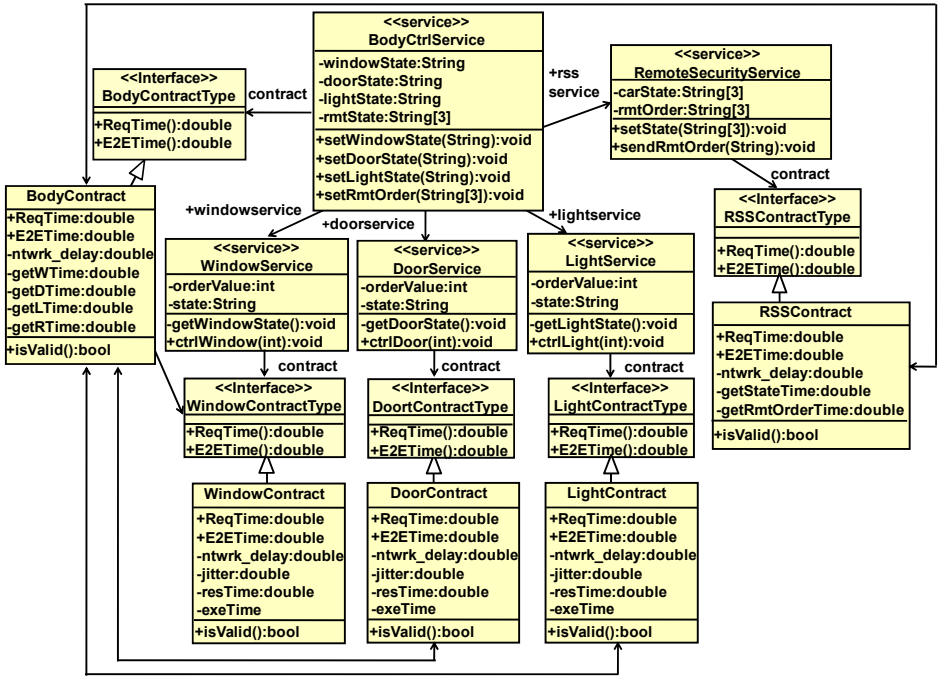

図 9 リモートセキュリティのコントラクトモデル

7.3 制約検証とボトルネックの発見

コントラクトモデルで定義した時間特性を用いて，OCL に基づく制約記述を行い，各コンテキストの End-to-End の実行時間制約を定義する．図 10 にうっかり通知のコンテキストのタイミング制約記述を示す．コンテキストの時間要求に基づく実行時間制約を Reqtime，End-to-End の実行時間を E2Etime として定義し，各時間特性の制約値を割り当てた．

```
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 10　うっかり通知の制約記述

制約値割り当てによるタイミング設計を行った制約記述式を定義するコントラクトモデルを XMI 形式に変換し, ContractValidator へ入力することで, WCET を評価し，制約検証を行った．制約検証結果を表 3 に示す．

制約検証の結果，うっかり通知のコンテキストにおいて，窓状態通知の実行時間制約 250(ms)に対する WCET の超過時間 35(ms)，ドア状態通知の実行時間制約 250(ms)に対する WCET の超過時間 30(ms)を特定した．各超過時間に対して，RemoteSecurityService の超過時間の割合が高いことから，制約値割り当ての変更が必要なボトルネックサービスとして発見した．

同様に，リモート操作のコンテキストにおいて，ドア制御の実行時間制約 150(ms)に対する WCET の超過時間 5(ms)，ライト制御の実行時間制約に対する WCET の超過時間 10(ms)を特定した．各超過時間に対して，BodyCtrlService の超過時間の割合が高いことから，制約値割り当ての変更が必要なボトルネックとして発見した．

表 3　リモートセキュリティの制約検証結果

機能 / 検証結果	うっかり通知			リモート操作		
コンテキスト	1:W-B-RS	2:D-B-RS	3:L-B-RS	4:RS-B-W	5:RS-B-D	6:RS-B-L
Reqtime (ms)	250	250	500	150	150	150
BCET (ms)	200	200	310	95	105	110
WCET (ms)	**285**	**280**	400	145	**155**	**160**
ボトルネック (サービス, 超過時間)	B 5(ms)	RS 30(ms)	なし	B 20(ms)	B 20(ms)	B 20(ms)
	RS 35(ms)				D 5(ms)	L 10(ms)

7.4 提案方法の評価

(1) 制約記述能力の評価

コントラクトモデルの記述可能要素として，サービスの構文定義と制約定義がある．構文定義として，サービスのオペレーションと内部のプロパティが記述可能である．

また，制約定義の記述能力として，コントラクトモデルで定義した時間特性に基づく実行時間定義と実行時間制約定義が可能である．周期ジッタ等の可変な時間特性についても，形式的な記述に従って変化範囲を記述可能である．さらに，コントラクト間のロールを記述することで，コンテキストを定義するためのサービス間の依存関係を定義できる．

(2) 制約検証能力の評価

制約検証で対象としているシナリオは，センサ入力からアクチュエータ出力までの End-to-End のサービス協調の実行系列として定義されるコンテキストである．コンテキストは 1 つのセンサ入力から派生する実行パスの単一経路として考えられる．このような，ある入力に対する網羅的な実行系列に基づく制約検証によって，入力毎の End-to-End のタイミング制約を厳密に保証できる．

しかし，各実行系列が独立的に実行される場合はコンテキストに基づく検証が有効であるが，複数実行系列間の同期等，依存関係が存在する場合は，コンテキスト間の依存関係をモデル化した上での制約検証が必要となる．

8. 考察

(1) タイミング設計に対する考察

制約検証のシナリオにおいて，複数コンテキスト間で発生するタイミング競合を検証する必要がある．そのため，各コンテキストの起点となるセンサ入力のタイミングを分析し，同時並行して発生するコンテキストを特定することで，依存関係に基づく制約情報をモデル化する必要がある．

(2) 関連研究との比較に基づく考察

a) インタフェース定義要素の比較[6]

文献[6]で提案されるコンポーネントモデル上のインタフェースの制約定義は単一コンポーネントへの入出力に基づく実行シナリオを対象としている．本稿では，単一サービスではなく，複数サービス間の End-to-End の協調実行シナリオを対象とした制約定義を行うことで，End-to-End の実行時間制約検証を可能としている．

b) 制約記述言語のモデル記述能力の比較[5]

文献[5]の QoSCL では，サービスの特性に基づいた記述形式が定義されていない．本稿では，サービスの構文定義と End-to-End の制約定義に関わるモデル要素を拡張し，サービスの拡張インタフェース情報を定義している．

c) 車載ソフトウェア開発の抽象レベルにおける位置づけ[2][11]

本稿で提案するコントラクトモデルを文献[2][11]の車載ソフトウェア開発抽象モデルと比較して考察する．図 11 に抽象モデルにおけるタイミング設計のモデル要素の位置づけを示す．文献[2][11]が定義する車載ソフトウェア開発の抽象レベルでは，システムの要求分析，設計までの機能的な視点で上位の 3 階層を定義している．本稿で提案するタイミング設計では，論理レベルのサービスの振舞いから時間特性を定義し，制約情報をモデル化しているため，これらの 3 階層へマッピングできるが，下位の実装レベルまで対応していない．

タイミング設計を実装レベルへ対応するためには，コントラクトモデルが定義する時間特性をハードウェアトポロジの視点から拡張し，実行時間制約の定義を厳密化する必要がある．今後は，定義すべき拡張情報の分析に基づき，コントラクトモデルを基盤とした設計レベルから実装レベルへの段階的なモデル化方法を考える必要がある．

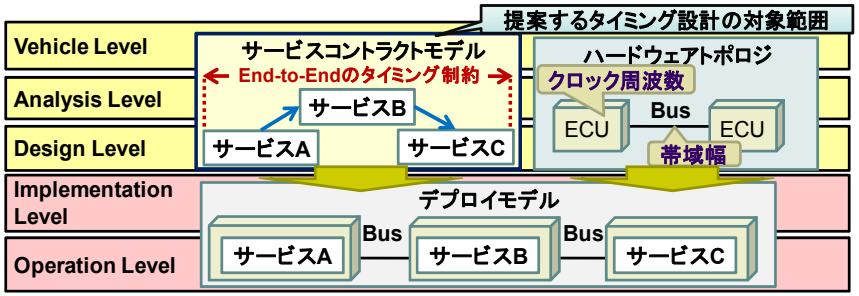

図 11 車載ソフトウェア開発プロセスにおける位置づけ

9. 今後の課題

時間特性を抽出するための車載サービスの振舞いパターンを詳細化し，厳密な時間特性を定義する必要がある．また，本稿で定義したコンテキストは静的な End-to-End の実行系列であり，動的なサービスの振舞いにおける制約定義が困難である．そのため，動的なサービスの振舞いをモデル化するためのコンテキスト定義の拡張が必要である．さらに，コンテキスト拡張に対応可能な ContractValidator の拡張が必要である．

10. まとめ

拡張インタフェースモデルを用いたタイミング設計方法と制約検証方法を提案した．車載サービスの振舞いに基づく時間特性を定義し，時間特性を用いたタイミング制約を記述するコントラクトモデルを提案した．また，コントラクトモデルの制約情報に基づき，サービス協調のコンテキスト毎の WCET を算出し，WCET 検証を実現するツール ContractValidator を実装した．リモートセキュリティを例題として，コントラクトモデルによる制約情報の記述能力と，ContractValidator を用いた制約検証能力を評価し，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) ATESST (Advancing Traffic Efficiency and Safety through Software Technology), EAST ADL2.0 Specification Version 2.0, 2008
3) AUTOSAR (AUTomotive Open System ARchitecture), http://www.autosar.org/.
4) A. Beugnard, et al., Making Components Contract Aware, IEEE Computer, vol. 32, No. 7, Jul. 1999, pp. 38-45.
5) S. Gerard, et al., Model Driven Engineering for Distributed Real-time Embedded Systems, Hermes Science, 2006.
6) J. E. Kim, et al., Extracting, Specifying and Predicting Software System Properties in Component Based Real-Time Embedded Software Development, Proc. IEEE ICSE '09, May 2009, pp. 28-38.
7) I. H. Krüger, et al., Service-Based Software Development for Automotive Applications, Proc. of Convergence 2004, No. 2004-21-0040 CTEA, Oct. 2004.
8) 永東 丈寛，ほか，協調するサービスの時間特性のモデル化方法，ソフトウェアエンジニアリング最前線 2009 (情報処理学会ソフトウェアエンジニアリングシンポジウム 2009 論文集)，近代科学社, Sep. 2009, p. 194.
9) 永東 丈寛，ほか，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, 2009 年度南山大学大学院数理情報研究科修士論文(OJL 報告書), 2010.
10) Papyrus UML, ver. 1.12, 2009, http://www.papyrusuml.org/.
11) TIMMO (TIMing MOdel), TADL: Timing Augmented Description Language Version 2, 2009.
12) トヨタ自動車, Lexus LS 電子技術マニュアル, 2006.
13) J. Warmer, et al., Object Modeling with the OCL, Springer, 2002.