情報処理学会論文誌 Vol.57 No.8 1703–1717 (Aug. 2016)

組み込み制御ソフトウェア開発のための Simulink モデルから UML モデルへの変換ツール

黒木 裕太[1,†1,a)] 田中 亨祐[1,b)] 兪 明連[1,c)] 横山 孝典[1,d)]

受付日 2015年11月19日, 採録日 2016年5月17日

概要：本論文では，組み込み制御ソフトウェアの開発効率向上を目的に，制御ロジックを記述した Simulink モデルをソフトウェア設計に適した UML モデルに変換するツールを提案する．本モデル変換ツールは制御ロジックの構造を表す階層の Simulink モデルを入力とし，状態遷移や条件分岐に応じて実行する処理を切り替えられる UML モデルを生成する．具体的には，制御ロジックの構造を記述した階層の Simulink モデルを入力とし，クラス図，オブジェクト図，シーケンス図，およびアクティビティ図を出力する．そして，複数の Simulink モデルに対して適用実験を行い，本モデル変換ツールの有用性を確認した．

キーワード：組み込み制御ソフトウェア，モデルベース開発，UML，モデル変換

A Simulink to UML Model Transformation Tool for Embedded Control Software Design

Yuta Kuroki[1,†1,a)] Kosuke Tanaka[1,b)] Myungryun Yoo[1,c)] Takanori Yokoyama[1,d)]

Received: November 19, 2015, Accepted: May 17, 2016

Abstract: The paper presents a tool to transform Simulink models to UML models. The model transformation tool analyzes the data flows and control flows of Simulink models and generates UML models with efficient control flows. The tool inputs a structure-layer Simulink model and outputs class diagrams, object diagrams, sequence diagrams and activity diagrams. We have applied the tool to a number of Simulink models and have confirmed its usefulness for embedded control software design.

Keywords: embedded control software, model-based design, UML, model transformation

1. はじめに

自動車や家電製品に用いられる組み込み制御ソフトウェアの開発量は増大しており，開発効率の向上が重要な課題となっている．開発効率を向上させる手法として，MATLAB/Simulink [1] 等の制御系 CAD/CAE ツールを用いたモデルベース開発が注目されている．しかし制御モデル設計用のツールは制御ロジックの設計には適しているが，ソフトウェア設計に十分な機能を提供していないといわれている．たとえば Sangiovanni-Vincentelli らは，制御系 CAD/CAE ツールでは機能モデルとアーキテクチャモデルを分離して記述することができない，タスクやリソース等のモデルが定義されていない等の問題を指摘している [2]．制御系 CAD/CAE ツールは制御設計にのみ用いて，ソフトウェア設計にはソフトウェアを対象としたモデリング言語である UML を用いるべきと考える．そのためには，制御系 CAD/CAE ツールを用いて作成した制御モデルをソフトウェア設計に適した UML モデルに効率良く変換できるツールが必要である．

Simulink モデルから UML モデルへの変換については，すでに様々な変換ツールが提案されている．Ramos-Hernandez らは Simulink モデルを UML モデルのクラス

[1] 東京都市大学
Tokyo City University, Setagaya, Tokyo 158–8557, Japan
[†1] 現在，株式会社シンカーミクセル
Presently with Xincor miXell Co., Ltd.
[a)] yuta_kuroki@xon.jp
[b)] g1581515@tcu.ac.jp
[c)] myoo@tcu.ac.jp
[d)] tyoko@tcu.ac.jp

図に自動変換するツールを開発している [3], [4]．Müller-Glaser らは Simulink モデルを UML モデルのクラス図とコミュニケーション図に自動変換するツールを開発している [5], [6]．Sjöstedt らは Simulink モデルを UML モデルの複合構造図とアクティビティ図に自動変換するツールを開発している [7]．しかしこれらは Simulink モデルの要素 1 つ 1 つを UML モデルの要素に変換しており，クラスの粒度が小さいうえ，まとめて再利用されることが多い複数のクラスを集約やコンポジション表記することもないため，必ずしも再利用性の高い UML モデルとはならない．

そこで我々は，入力値，出力値，観測値，測定値，目標値，制御パラメータ等の制御上重要な意味を持つデータ（物理量）に着目して階層化して得た制御ロジックの構造を表す Simulink モデルから，再利用性の高いクラスからなる UML モデルを生成するモデル変換ツールを提案した [8]．本モデル変換ツールは Subsystem ブロックからなる Simulink モデルを UML モデルのクラス図，オブジェクト図，シーケンス図に変換する．ただし，このツールは条件分岐を含まないデータフローの解析のみを行っており，状態遷移や条件分岐を表すブロックを含む Simulink モデルは対象としていない．しかし実際には，制御ロジックの構造を表す階層の Simulink モデルに状態遷移や条件分岐を表すブロックが含まれることも多く，それらに対して単純なデータフロー解析のみを行ったのでは，条件や状態によっては不要な値をつねに計算する効率の良くない UML モデルが生成されてしまう．

本研究の目的は，状態遷移や条件分岐を表すブロックを含む Simulink モデルのデータフローと制御フローを解析し，無駄な計算を排除した効率の良い UML モデルに変換可能なモデル変換ツールを開発することである．これにより，より広範囲の Simulink モデルを扱うことが可能になる．

以下本論文では，まず 2 章で本研究で扱う変換対象のモデルについて述べる．3 章ではデータフローの選択や条件分岐を表すブロックを含む Simulink モデルを対象としたデータフロー解析や制御フロー解析を必要とする変換について述べる．4 章ではまとめて再利用される複数クラスに対応するためのコンポジションを考慮した変換について説明する．5 章では大規模なモデルを扱う場合に必要となるデータフローが分岐や合流を含む場合の変換について述べる．さらに，6 章でモデル変換ツールの実装，7 章で適用実験，8 章で関連研究との比較について述べる．最後は 9 章で本論文をまとめる．

2. 変換対象のモデル

2.1 変換対象とする Simulink モデル

Simulink モデルは，一般に階層化したモデルとして記述する．MATLAB のユーザ団体である MathWorks Automotive Advisory Board（MAAB）は，制御モデル記述のためのガイドライン [9] を規定している．ガイドラインに記載されている J-MAAB の階層構造 TypeA を**図 1** に示す．J-MAAB の階層構造 TypeA は，トップレイヤ，トリガレイヤ，構造レイヤ，データフローレイヤの 4 階層から構成される．トップレイヤでは，モデルの入出力を記述し，トリガレイヤでは演算のタイミングを記述する．構造レイヤでは制御ロジックの構造を記述し，データフローレイヤではデータの流れを記述する．

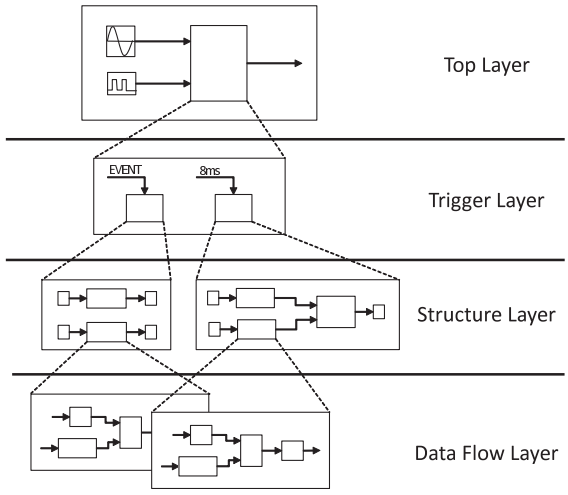

図 1 J-MAAB の階層構造 TypeA
Fig. 1 J-MAAB model architecture Type A.

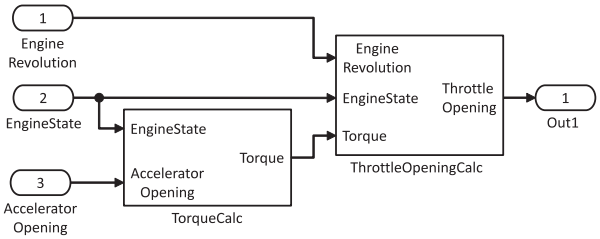

図 2 スロットル制御の Simulink モデル
Fig. 2 Simulink model of throttle controller.

我々がすでに提案したモデル変換ツール [8] が対象とした Simulink モデルは構造レイヤに対応する．すなわち，制御上重要なデータのみが構造レイヤに現れるようにし，構造レイヤの Simulink モデルを UML モデルに変換することで，再利用に適したクラス構成を持つ UML モデルを生成できる．構造レイヤの例として，スロットル制御の Simulink モデルを**図 2** に示す．図 2 の Simulink モデルは，エンジン回転数（Engine Revolution），エンジン状態（Engine State），アクセル開度（Accelerator Opening）の 3 つの Inport ブロックと，トルク算出（Torque Calculation），スロットル開度算出（Throttle Opening Calculation）の 2 つの Subsystem ブロック，Out1 の 1 つの Outport ブロックからなる．まず，トルク算出はエンジン状態とアクセル開度からトルク（Torque）を算出する．次に，スロットル

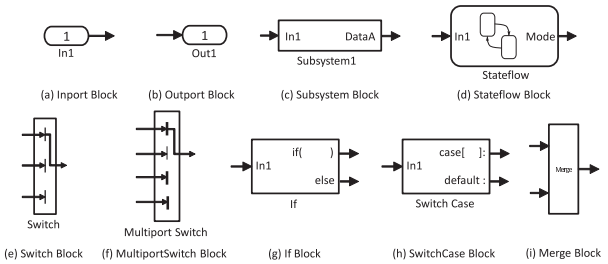

図 3 対象とするブロック
Fig. 3 Target blocks.

開度算出はエンジン回転数とエンジン状態とトルクからスロットル開度（Throttle Opening）を算出する．そして，スロットル開度を Out1 に出力する．

MAAB のガイドラインでは各階層で使用できるブロックが規定されているが，構造レイヤで使用可能なのは，これまで対象としてきた Subsystem ブロック，Inport ブロック，Outport ブロックのほか，すべての階層で許可されている SwitchCase ブロック等の状態遷移や条件分岐を表すブロックも使用できる．

そこで我々は，状態遷移や条件分岐を含む構造レイヤの Simulink モデルを変換可能とする．具体的には**図 3** に示す Inport ブロック，Outport ブロック，Subsystem ブロック，Stateflow ブロック，Switch ブロック，MultiportSwitch ブロック，If ブロック，SwitchCase ブロック，Merge ブロックの 9 種類のブロックからなる Simulink モデルを対象とする．Switch ブロックおよび MultiportSwitch ブロックは Version 2.0 では使用可能であったが，Version 3.0 からは除かれている．しかし，現実には広く使用されているため対象に含めている．なお，最近公開されたガイドライン Version 4.0 [10] ではモデルの階層構造はルールではなく考え方になっているが，サブ機能レイヤ，制御レイヤ，選択レイヤが変換対象の階層に対応すると考えている．

2.2 生成する UML モデル

モデル変換ツールは時間駆動オブジェクト指向ソフトウェア開発法 [11] に基づいた UML モデルを生成する．具体的には，**図 4** で表現されるデザインパターンに従った UML モデルを生成する．本デザインパターンは，構造レイヤ中の 1 つの制御機能全体を表すコントローラクラス（Controller）と，その制御機能に含まれる制御上重要なデータ（物理量）を表すデータ値オブジェクトクラス（ValueObject）からなる．データ値オブジェクトクラスは，データの読み出しを行う get メソッドと，データの更新を行う update メソッドを持つ．コントローラクラスは，データ値オブジェクトクラスの update メソッドを呼び出すための exec メソッドを持つ．

コントローラクラスとデータ値オブジェクトクラス間はコンポジションの関係がある．データ値オブジェクトクラス間は，データの参照を意味する Consume 関連を持つ．これらのクラスは抽象クラスであり，モデル変換ツールではこれらのクラスを継承して具体的な制御処理を実行するクラスを生成する．

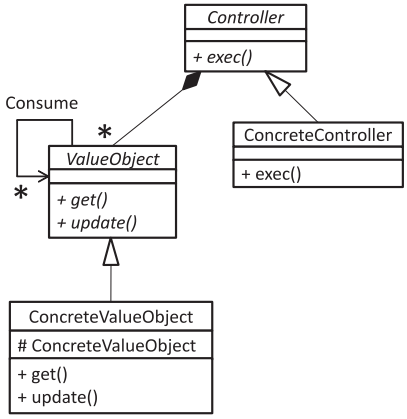

図 4 デザインパターン
Fig. 4 Design pattern.

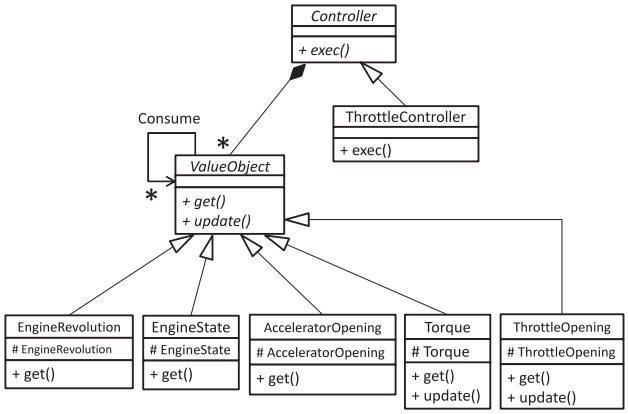

図 5 図 2 の Simulink モデルに対応するクラス図
Fig. 5 Class diagram corresponding to the Simulink Model shown by Fig. 2.

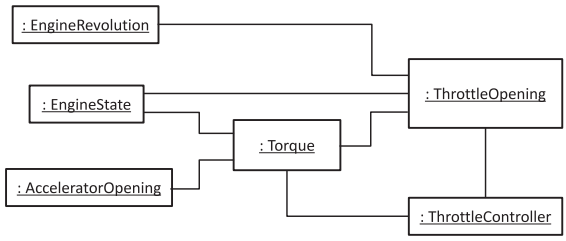

図 6 図 2 の Simulink モデルに対応するオブジェクト図
Fig. 6 Object diagram corresponding to the Simulink Model shown by Fig. 2.

本デザインパターンに従った UML モデルの例として，図 2 の Simulink モデルに対応した UML モデルのクラス図，オブジェクト図，シーケンス図，アクティビティ図を**図 5**，**図 6**，**図 7**，**図 8** に示す．

図 5 のクラス図に示すように，エンジン回転数，エンジン状態，アクセル開度，トルク，スロットル開度クラスは，

データ値オブジェクトクラスのサブクラスである．エンジン回転数，エンジン状態，アクセル開度クラスは，Inport ブロックのデータ名に対応しており，内部データを表す属性とデータの読み出しを行う get メソッドのみを持つ．また，トルク，スロットル開度クラスは Subsystem ブロックの出力データ名に対応しており，内部データを表す属性とデータの読み出しを行う get メソッドとデータの更新を行う update メソッドの両方を持つ．また，コントローラ抽象クラスのサブクラスであるスロットル制御クラス（Throttle Controller）は，図 2 の Simulink モデル全体に対応し，トルク，スロットル開度クラスの update メソッドを呼び出すための exec メソッドを持つ．

また，図 6 のオブジェクト図に示すように，図 5 中のクラス図のエンジン回転数，エンジン状態，アクセル開度，トルク，スロットル開度，トルク制御のクラスに対応したオブジェクトを生成し，データ間の参照関係のあるオブジェクト間をリンクで結ぶ．また，update メソッドを持つオブジェクトであるトルク，スロットル開度とスロットル制御間のリンクを生成し，それがスロットル制御を構成するオブジェクトであることを表す．

図 7 のシーケンス図に示すように，スロットル制御オブジェクトの exec メソッドがトルクオブジェクトの update メソッドとスロットル開度オブジェクトの update メソッドを呼び出す．

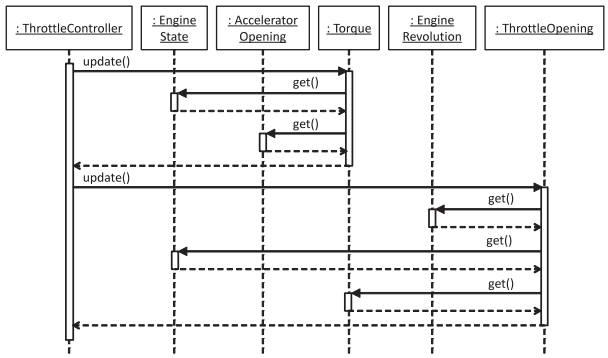

図 7 図 2 の Simulink モデルに対応するシーケンス図

Fig. 7 Sequence diagram corresponding to the Simulink Model shown by Fig. 2.

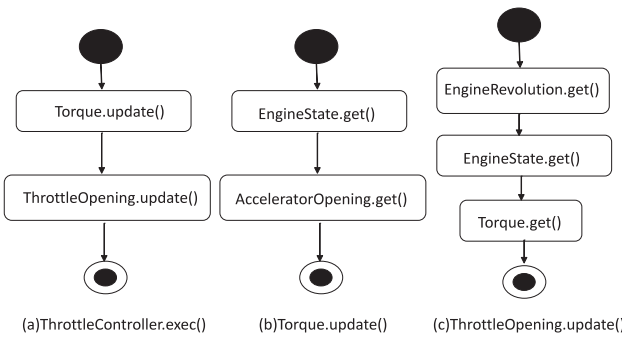

図 8 図 2 の Simulink モデルに対応するアクティビティ図

Fig. 8 Activity diagram corresponding to the Simulink Model shown by Fig. 2.

図 8 のアクティビティ図に示すように，スロットル制御クラスの exec メソッドで，トルクとスロットル開度の update メソッドを呼び出す．トルクとスロットル開度の update メソッドは，それぞれ必要な値の get メソッドを呼び出す．

3. データフローや制御フローに関する変換

3.1 データフロー・制御フロー解析の課題

図 9 の Simulink モデルを従来の変換方法 [8] により変換すると**図 10** のシーケンス図が生成される．

図 9 の Simulink モデルは，Switch ブロックによるデータフローの選択を含むパターンの例である．この Simulink モデルは，Stateflow ブロックであるモード算出（ModeCalc）が出力する選択信号（値は 0 か 1）により，Subsystem ブロックである設定速度算出（SetSpeedCalc）のデータまたは車速算出（CarSpeedCalc）のデータの一方を Out1 ブロックに出力するモデルである．Mode がしきい値（threshold）0.5 よりも大きい場合（1 の場合）は，設定速度算出のデータを Out1 に出力する．また，Mode がしきい値 0.5 よりも小さい場合（0 の場合）は，車速算出のデータを Out1 に出力する．

図 10 のシーケンス図は，クルーズ制御オブジェクト（CruiseController）がモードオブジェクト（Mode）の update メソッドを呼び出し，次に設定速度オブジェクト

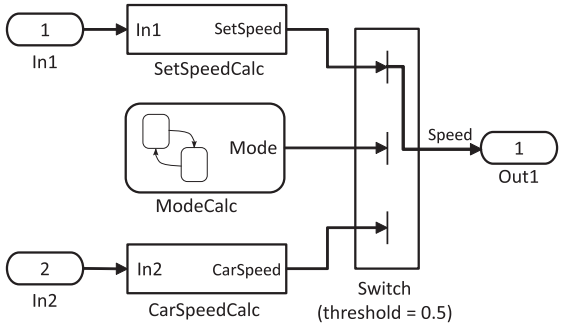

図 9 Switch ブロックを用いた Simulink モデルの例

Fig. 9 Example Simulink Model with a Switch Block.

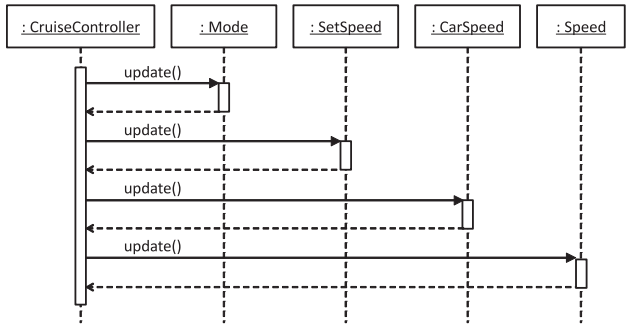

図 10 データフロー・制御フロー解析の課題

Fig. 10 Problem of data flow analysis and control flow analysis.

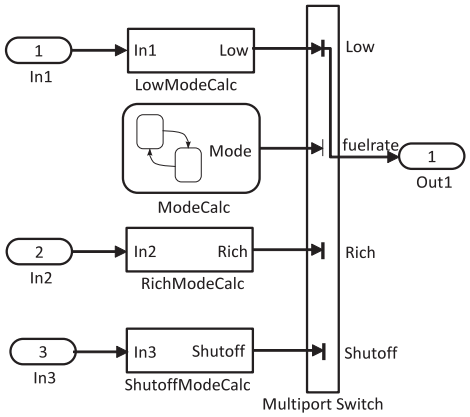

図 11 MutiportSwitch ブロックを用いた Simulink モデルの例
Fig. 11 Example Simulink Model with a MultiportSwitch Block.

(SetSpeed) と車速オブジェクト (CarSpeed) の update メソッドを呼び出す．最後に，速度オブジェクト (Speed) の update メソッドを呼び出し，速度オブジェクトが設定速度オブジェクトの値または車速オブジェクトの値のうち，どちらか一方の値を取得する．すなわち，モードの値によらずつねに設定速度と車速の両方を算出している．無駄な計算処理を行わないためには，条件を考慮したデータフロー解析と制御フロー解析を行う必要がある．

3.2 Simulink モデルのパターン

図 3 のブロックを用いて構成される状態遷移や条件分岐を含む Simulink モデルは，図 3 の (e)〜(h) に対応した，Switch ブロックを用いたパターン，MultiportSwitch ブロックを用いたパターン，If ブロックを用いたパターン，および SwitchCase ブロックを用いたパターンの 4 種類のパターンに分類できる．前者 2 つはデータフローが選択を含むパターンで，後者 2 つは制御フローが条件分岐を含むパターンである．Switch ブロックを用いたパターンについては 3.1 節で説明したため，ここでは他の 3 つのパターンについて説明する．

MultiportSwitch ブロックによるデータフローの選択を含むパターンの例を**図 11** に示す．この Simulink モデルは，Stateflow ブロックであるモード算出が出力する選択信号（値は Low，Rich，Shutoff のいずれか）により，Subsystem ブロックである Low モード算出（LowModeCalc）または Rich モード算出（RichModeCalc），Shutoff モード算出（ShutoffModeCalc）のうち 1 つのデータを Out1 ブロックに出力するモデルである．Mode が Low の場合は，Low モード算出のデータを，Mode が Rich の場合は Rich モード算出のデータを，Mode が Shutoff の場合は Shutoff モード算出のデータを Out1 へ出力する．

If ブロックを用いたパターンは，条件分岐を表す If ブロックとデータフローを合成する Merge ブロックを使用する．**図 12** は，図 9 の Simulink モデルと同じ制御ロジックを If ブロックを用いて表現した例である．この Simulink モデルは，Stateflow ブロックであるモード算出が出力する選択信号をもとに If ブロックが持つ条件式によって算出する Subsystem ブロック（設定速度算出または車速算出）を切り替えるモデルである．Mode が 0.5 よりも大きい場合（1 の場合）は，設定速度算出を実行しその出力データを Out1 に出力する．また，Mode が 0.5 よりも小さい場合（0 の場合）は，車速算出を実行しその出力データを Out1 に出力する．

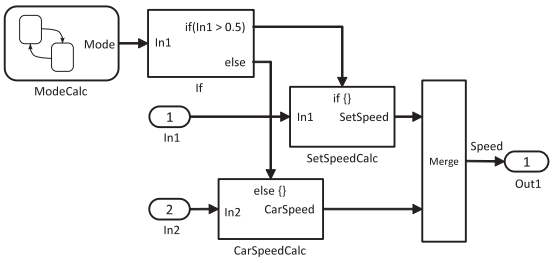

図 12 If ブロックを用いた Simulink モデルの例
Fig. 12 Example Simulink Model with a If Block.

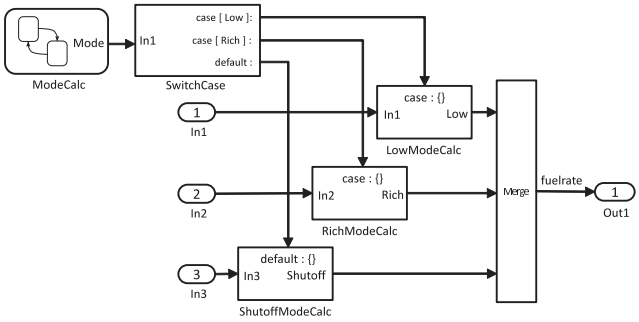

図 13 SwitchCase ブロックを用いた Simulink モデルの例
Fig. 13 Example Simulink Model with a SwitchCase Block.

SwitchCase ブロックを用いたパターンは，条件分岐を表す SwitchCase ブロックとデータフローを合成する Merge ブロックを使用する．**図 13** は，図 11 の Simulink モデルと同じ制御ロジックを SwitchCase ブロックを用いて表現した例である．この Simulink モデルは，Stateflow ブロックであるモード算出が出力する選択信号（値は Low，Rich，Shutoff のいずれか）をもとに SwitchCase ブロックが持つ条件式によって算出する Subsystem ブロック（Low モード算出または Rich モード算出，Shutoff モード算出）を切り替えるモデルである．Mode が Low の場合は Low モード算出を実行し，その出力データを Out1 に出力する．また，Mode が Rich の場合は Rich モード算出のデータを実行し，その出力データを Out1 に出力する．Mode が Shutoff の場合は Shutoff モード算出のデータを実行し，その出力データを Out1 に出力する．

If ブロックや SwitchCase ブロックを用いたパターンは条件分岐のある制御フローが陽に記述されているため，そ

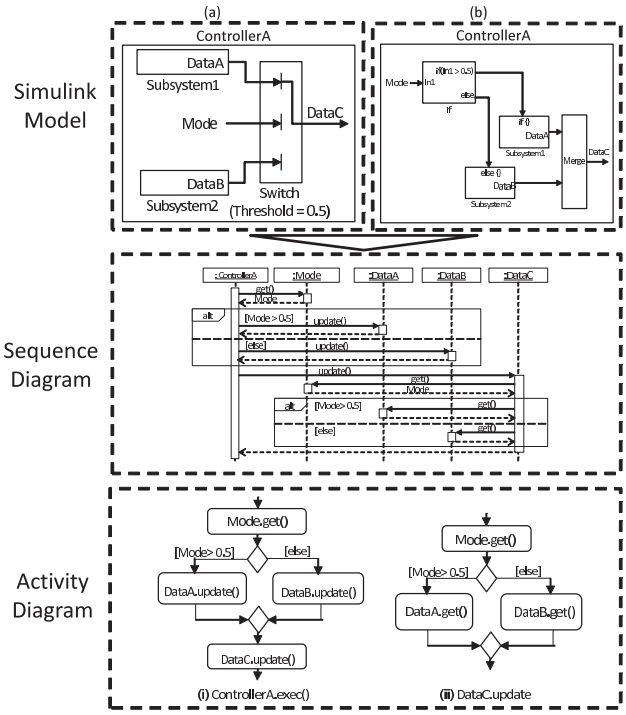

図 14 選択のあるデータフローや条件分岐のある制御フローに関する変換ルール

Fig. 14 Transformation rules for selectional data flows and conditional control flows.

のまま条件分岐をともなうソフトウェアモデルに変換できる．一方，データフローの選択を行う Switch ブロックや MultiportSiwtch ブロックを用いたパターンは，データフローを解析して条件分岐を含む制御フローを持つソフトウェアモデルに変換する必要がある．

3.3 変換規則

選択のあるデータフローや条件分岐のある制御フローに関する変換ルールを**図 14** を用いて説明する．図 14 の (a) は，Switch ブロックを含む Simulink モデル，(b) は，If ブロックを含む Simulink モデルである．両者は同じ形のシーケンス図とアクティビティ図に変換される．

図 14 (a) の Simulink モデルをシーケンス図に変換するルールは，以下のとおりである．まず全体オブジェクト（ControllerA）が Stateflow ブロックに対応したオブジェクト（Mode）の get メソッドを呼び出すメッセージを生成する．次に，条件分岐を表す Alternative フラグメント（図中の alt）を生成し，Switch ブロックが持つ条件ごとに Subsystem ブロックに対応したオブジェクト（DataA，DataB）の update メソッドを呼び出すメッセージを順に生成する．条件式は，Stateflow ブロックの状態値と Switch ブロックのしきい値を比較する条件式（Mode > 0.5）に変換してシーケンス図に記述する．最後に，Switch ブロックに対応したオブジェクト（DataC）の update メソッドを呼び出すメッセージを生成する．このオブジェクト（DataC）の update メソッドのメッセージ生成では，まず Stateflow ブロックに対応したオブジェクト（Mode）の get メソッドを呼び出すメッセージを生成する．次に，条件分岐を表す Alternative フラグメントを生成し，Switch ブロックが持つ条件（Mode > 0.5）ごとに Subsystem ブロックに対応したオブジェクト（DataA，DataB）の get メソッドを呼び出すメッセージを順に生成する．

図 14 (a) の Simulink モデルをアクティビティ図に変換するルールは，全体を表すクラス（ControllerA）の exec メソッドと Switch ブロックに対応するクラス（DataC）の update メソッドのアクティビティを表すアクティビティ図を生成する．まず全体を表すクラス（ControllerA）の exec メソッドは Stateflow ブロックに対応するクラス（Mode）の get メソッドを呼び出す．次に条件分岐を表すデシジョンノードを生成し，Switch ブロックが持つ条件ごとに Subsystem ブロックに対応するクラス（DataA，DataB）の update メソッドを呼び出す．このとき Stateflow ブロックの状態値と Switch ブロックのしきい値を比較する条件式（Mode > 0.5）を生成してアクティビティ図に記述する．最後に，Switch ブロックに対応するクラスの update メソッドを呼び出す．Switch ブロックに対応するクラス（DataC）の update メソッドは，Stateflow ブロックに対応するクラス（Mode）の get メソッドを呼び出す．次にデシジョンノードを生成し，Switch ブロックが持つ条件ごとに Subsystem ブロックに対応するクラス（DataA，DataB）の get メソッドを呼び出す．このとき，全体を表すクラスの条件式と同じ条件式をアクティビティ図に記述する．

図 14 (b) の If ブロックを含む Simulink モデルをシーケンス図とアクティビティ図に変換するルールにおいても，(a) の変換ルールと同様の方法で生成するが，条件式は If ブロック内の条件式をそのままシーケンス図に記述する．

MultiportSwitch ブロックや SwitchCase ブロックを用いた Simulink モデルにおいても，同じ考え方を用いたルールで UML モデルに変換することができる．

3.4 変換例

図 9 の Simulink モデルを UML モデルに変換すると，**図 15** に示すクラス図，**図 16** に示すオブジェクト図，**図 17** に示すシーケンス図，**図 18** に示すアクティビティ図を出力する．

図 15 のクラス図に示すように，全体を表すクルーズ制御クラス（CruiseController）や，設定速度クラス（SetSpeed），車速クラス（CarSpeed），速度クラス（Speed）等のデータ値オブジェクトクラスが生成されている．

図 16 のオブジェクト図に示すように，図 15 の各具象クラスに対応したオブジェクトが生成され，Simulink モデルのラインに対応したリンクで結ばれている．クルーズ制御オブジェクトと設定速度，車速，速度の各オブジェクト

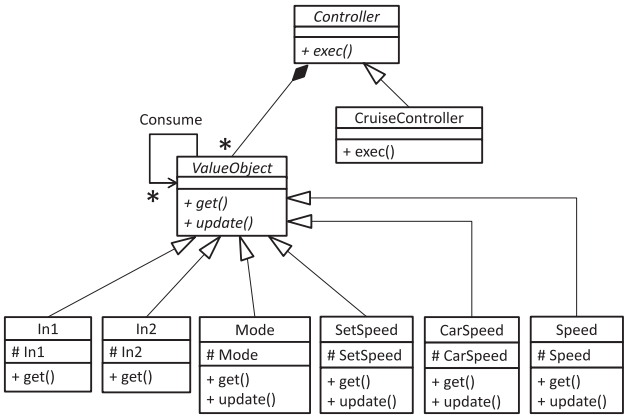

図 15 図 9 の Simulink モデルに対応するクラス図
Fig. 15 Class diagram corresponding to the Simulink Model shown by Fig. 9.

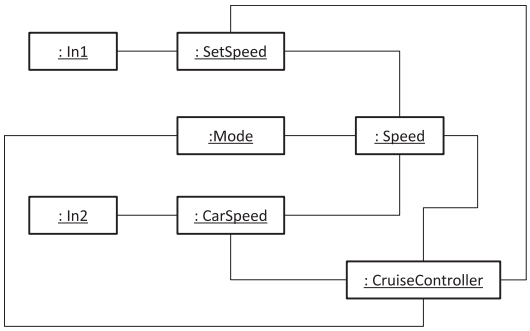

図 16 図 9 の Simulink モデルに対応するオブジェクト図
Fig. 16 Object diagram corresponding to the Simulink Model shown by Fig. 9.

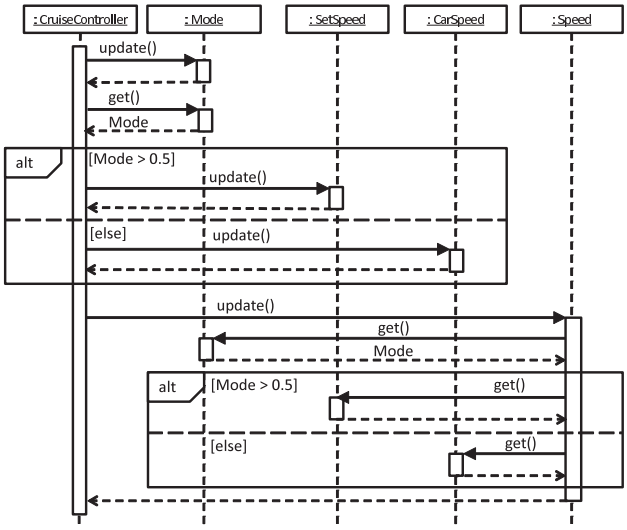

図 17 図 9 の Simulink モデルに対応するシーケンス図
Fig. 17 Sequence diagram corresponding to the Simulink Model shown by Fig. 9.

もリンクで結ばれている．

図 17 のシーケンス図に示すように，クルーズ制御オブジェクト（CruiseController）が取得した状態値に基づいて設定速度オブジェクト（SetSpeed）または車速オブジェクト（CarSpeed）の処理を実行する．そして，速度オブジェクト（Speed）を更新して設定速度オブジェクト（SetSpeed）または車速オブジェクト（CarSpeed）の実行結果を取得する．

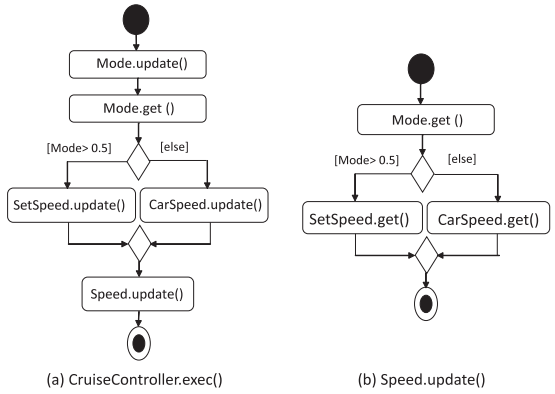

図 18 図 9 の Simulink モデルに対応するアクティビティ図
Fig. 18 Activity diagram corresponding to the Simulink Model shown by Fig. 9.

図 18 のアクティビティ図に示すように，クルーズ制御クラスの exec メソッドは取得した状態値に基づいて設定速度クラスまたは車速クラスの更新を実行した後，速度クラスの更新を実行する．速度クラスの update メソッドは，取得した状態値に基づいて設定速度クラスまたは車速クラスの実行結果を取得する．

4. コンポジションを考慮した変換

4.1 変換規則

データフローの選択や条件分岐を含む場合，同じデータの値を複数のクラスを用いて算出することになる．同一データに関する複数のクラスはまとめて再利用されることが多いため，コンポジション構造に変換する．コンポジションを考慮した変換ルールを**図 19** を用いて説明する．図 19 の (a) は Switch ブロックを含む Simulink モデル，(b) は If ブロックを含む Simulink モデルである．両者は同じ形のクラス図，シーケンス図，アクティビティ図に変換される．

図 19 (a) の Simulink モデルをクラス図に変換するルールは，以下のとおりである．まず 2 つの Subsystem ブロックの出力データ名（DataA）をクラス名とするクラスを生成する．そして，各 Subsystem ブロック（Subsystem1, Subsystem2）に対応したクラスをそれぞれ生成する．そのクラスの名前は，“出力データ名_ブロック名”（DataA_Subsystem1, DataA_Subsystem2）とする．変換後のクラスは，内部データを表す属性，データの読み出しを行う get メソッド，データの更新を行う update メソッドを持つ．最後に，出力データ名に対応したクラスと各 Subsystem ブロックに対応したクラス間をコンポジションの関係とする．

図 19 (a) の Simulink モデルをシーケンス図に変換する

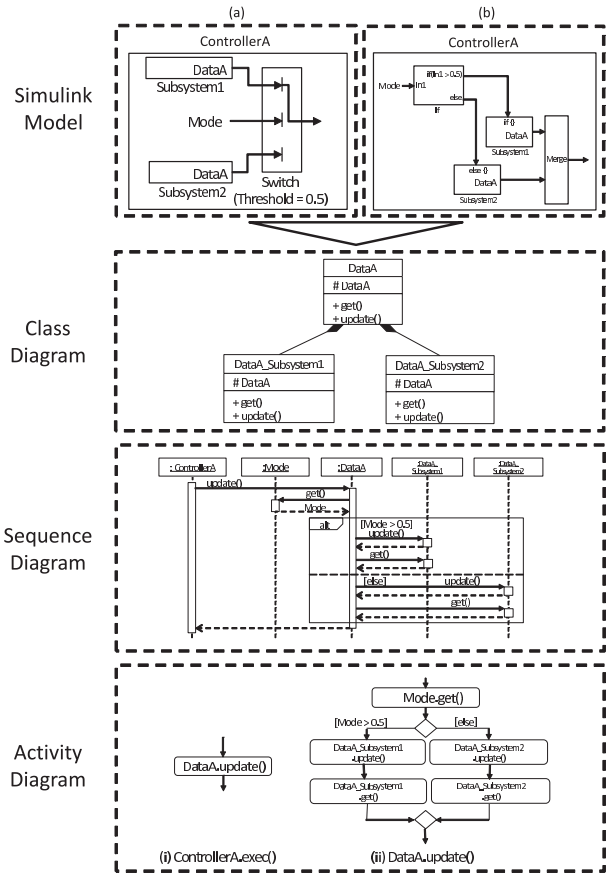

図 19 コンポジションを考慮した変換ルール
Fig. 19 Transformation rules for composition.

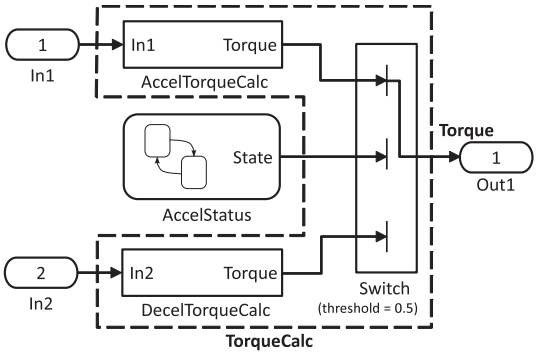

図 20 トルク算出の Simulink モデル
Fig. 20 Simulink model for torque calculation.

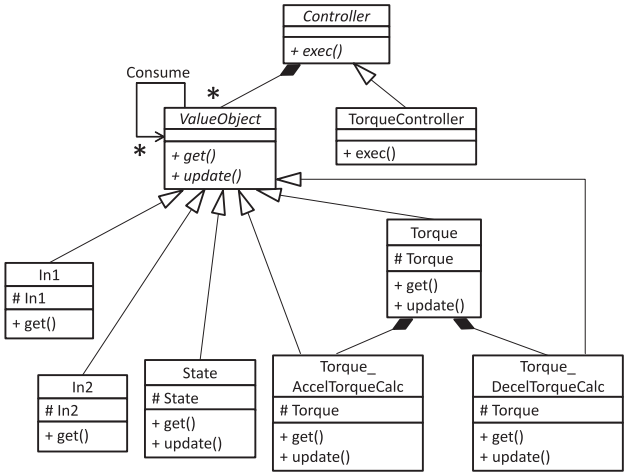

図 21 図 20 の Simulink モデルに対応するクラス図
Fig. 21 Class diagram corresponding to the Simulink Model shown by Fig. 20.

ルールは，以下のとおりである．まず全体オブジェクト（ControllerA）が 3 つのブロック全体を表すオブジェクト（DataA）の upadate メソッドを呼び出すメッセージを生成する．次に，そのオブジェクトが Stateflow ブロック（Mode）に対応したオブジェクトの get メソッドを呼び出すメッセージを生成する．そして，条件分岐を表す Alternative フラグメントを生成し，Switch ブロックが持つ条件ごとに Subsystem ブロックに対応したオブジェクト（DataA_Subsystem1，DataA_Subsystem2）の update メソッドを呼び出すメッセージと get メソッドを呼び出すメッセージをそれぞれ生成する．また，Stateflow ブロックの状態値と Switch ブロックのしきい値を比較する条件式（Mode > 0.5）に変換してシーケンス図に記述する．

図 19 (a) の Simulink モデルをアクティビティ図に変換するルールは，全体を表すクラス（ControllerA）の exec メソッドと Switch ブロックに対応するクラス（DataC）の update メソッドのアクティビティを表すアクティビティ図を生成する．まず全体を表すクラス（ControllerA）の exec メソッドは，3 つのブロック全体を表すクラス（DataA）の update メソッドを呼び出す．3 つのブロック全体を表すクラス（DataA）の update メソッドは，Stateflow ブロックに対応するクラス（Mode）の get メソッドを呼び出す．次に条件分岐を表すデシジョンノードを生成し，Switch ブロックが持つ条件ごとに Subsystem ブロックに対応するクラス（DataA_Subsystem1, DataA_Subsystem2）の update メソッドと get メソッドを呼び出す．このとき Stateflow ブロックの状態値と Switch ブロックのしきい値を比較する条件式（Mode > 0.5）を生成してアクティビティ図に記述する．

図 19 (b) の If ブロックを含む Simulink モデルをクラス図，シーケンス図，アクティビティ図に変換するルールにおいても (a) の変換ルールと同様の方法で生成するが，条件式は If ブロック内の条件式をそのままシーケンス図・アクティビティ図に記述する．If ブロックや SwitchCase ブロックを用いた Simulink モデルにおいても，同じ考え方を用いたルールで UML モデルに変換することができる．

4.2 変換例

図 20 の Simulink モデルは，2 つの Subsystem ブロックおよび Switch ブロックの出力データ名が同じトルク（Torque）である．図 20 の Simulink モデルを UML モデルに変換すると，**図 21** に示すクラス図，**図 22** に示すオブジェクト図，**図 23** に示すシーケンス図，**図 24** に示す

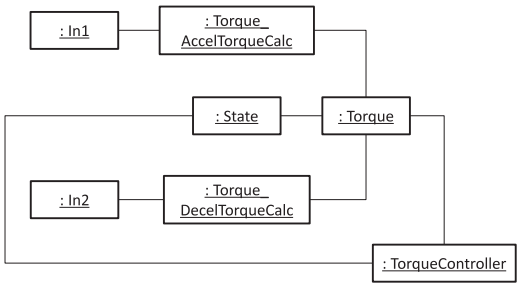

図 22 図 20 の Simulink モデルに対応するオブジェクト図
Fig. 22 Object diagram corresponding to the Simulink Model shown by Fig. 20.

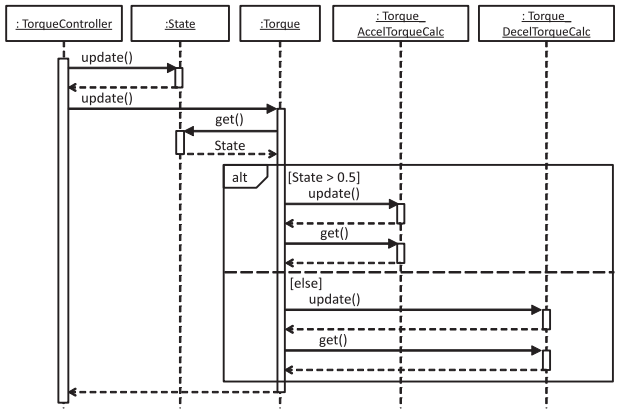

図 23 図 20 の Simulink モデルに対応するシーケンス図
Fig. 23 Sequence diagram corresponding to the Simulink Model shown by Fig. 20.

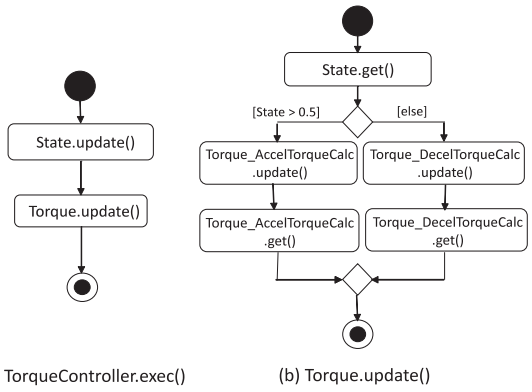

図 24 図 20 の Simulink モデルに対応するアクティビティ図
Fig. 24 Activity diagram corresponding to the Simulink Model shown by Fig. 20.

アクティビティ図が得られる．

図 21 のクラス図に示すように，加速時トルク算出（AccelTorqueCalc），減速時トルク算出（DecelTorqueCalc）および Switch ブロックはまとめてトルクを表すクラス（Torque）で表現され，その下位クラスとして AccelTorqueCalc ブロックに対応した加速時トルク算出クラス（Torque_AccelTorqueCalc）および DecelTorqueCalc ブロックに対応した減速時トルク算出クラス（Torque_DecelTorqueCalc）がコンポジションの関係で表現されている．

図 22 のオブジェクト図に示すように，図 21 の各具象クラスに対応したオブジェクトが Simulink モデルのラインに対応したリンクで結ばれている．

図 23 のシーケンス図において，トルク制御オブジェクト（TorqueController）の exec メソッドはトルクオブジェクト（Torque）が取得した状態値に基づいて加速時トルク算出オブジェクト（Torque_AccelTorqueCalc）または減速時トルク算出オブジェクト（Torque_DecelTorqueCalc）の処理を実行し，実行結果を取得する．

図 24 のアクティビティ図において，トルク制御クラス（TorqueController）の exec メソッドは加減速状態クラス（State）とトルククラスの更新を実行する．トルククラスの update メソッドは，取得した状態値に基づいて加速時トルク算出クラスまたは減速時トルク算出クラスの更新を実行し，実行結果を取得する．

5. データフローが分岐や合流を含む場合の変換

5.1 変換規則

3 章および 4 章では，データフローの選択を行う Switch ブロックや MultiportSwitch ブロックの入力や，条件分岐のための If ブロックや SwitchCase ブロックの出力に直接接続される Subsystem が 1 つのみの場合の変換方法について説明した．しかし，実際の Simulink モデルは多くの Subsystem ブロックからなり，それらを結ぶデータフローが分岐や合流を含むのが普通である．

データフローの分岐や合流を含む Simulink モデルを振舞いモデルに変換するルールを**図 25** を用いて説明する．構造を表す UML モデルであるクラス図やオブジェクト図については，データフローの分岐や合流を含む場合も特に変換方法は変わらない．そこでここでは振舞いを表すシーケンス図，アクティビティ図のみを示す．

図 25 の Simulink モデルにおける破線ラインは，複数の Subsystem ブロックを直列または並列に接続できること，直列接続と並列接続を組み合わせた Subsystem ブロック群でもよいことを表す．(a) は Switch ブロックにつながる Subsystem ブロックが条件ごとに独立している場合，(b) は両方の条件につながる Subsystem ブロック（DataX）が存在する場合である．

図 25 (a) の Simulink モデルにおいて，Stateflow ブロックが出力する選択信号がしきい値 0.5 よりも大きい場合は，Subsystem3 ブロック～Subsystem1 ブロックのみ，しきい値 0.5 よりも小さい場合は，Subsystem4 ブロック～Subsystem2 ブロックのみの計算を実行すればよい．

図 25 (a) の Simulink モデルをシーケンス図に変換するルールは以下のとおりである．まず全体オブジェクト（ControllerA）が Stateflow ブロックに対応したオブジェ

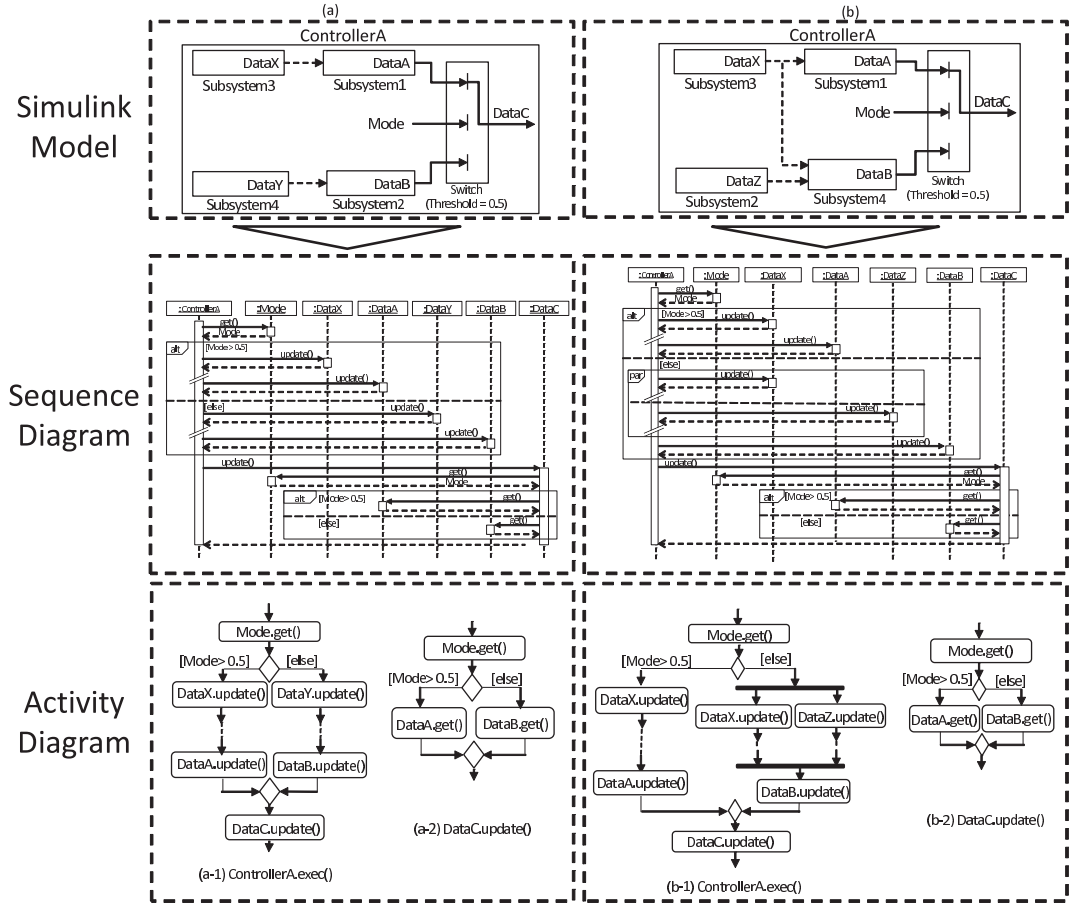

図 25 振舞いモデルへの変換ルール
Fig. 25 Transformation rules for Behavior Models.

クト（Mode）の get メソッドを呼び出すメッセージを生成する．次に，条件ごとに，Switch ブロックにつながる Subsystem ブロック（Mode > 0.5 の場合は DataX〜DataA，そうでない場合は DataY〜DataB）をすべて見つけ出す．そして，条件分岐を表す Alternative フラグメントを生成し，条件ごとに，入力である Inport ブロックに最も近い Subsystem ブロックから順に Subsystem ブロックに対応したオブジェクトの update メソッドを呼び出すメッセージを生成する．最後に，Switch ブロックに対応したオブジェクト（DataC）の update メソッドを呼び出すメッセージを生成する．この update メソッドは，まず Stateflow ブロックに対応したオブジェクト（Mode）の get メソッドを呼び出すメッセージを生成する．次に，条件分岐を表す Alternative フラグメントを生成し，Switch ブロックが持つ条件ごとに Subsystem ブロックに対応したオブジェクト（DataA または DataB）の get メソッドを呼び出すメッセージを順に生成する．

図 25 (a) の Simulink モデルをアクティビティ図に変換するルールは，全体を表すクラス（ControllerA）の exec メソッドと Switch ブロックに対応するクラス（DataC）の update メソッドのアクティビティを表すアクティビティ図を生成する．まず全体を表すクラス（ControllerA）の exec メソッドは，Stateflow ブロックに対応するクラス（Mode）の get メソッドを呼び出す．次に，条件ごとに，Switch ブロックにつながる Subsystem ブロック（Mode > 0.5 の場合は DataX〜DataA，そうでない場合は DataY〜DataB）をすべて見つけ出す．そして，条件分岐を表すデシジョンノードを生成し，条件ごとに Inport ブロックに最も近い Subsystem ブロックから順に Subsystem ブロックに対応するクラスの update メソッドを呼び出す．最後に，Switch ブロックに対応するクラス（DataC）の update メソッドを呼び出す．Switch ブロックに対応するクラス（DataC）の update メソッドは，Stateflow ブロックに対応するクラス（Mode）の get メソッドを呼び出す．次にデシジョンノードを生成し，Switch ブロックが持つ条件ごとに Subsystem ブロックに対応するクラス（DataA または DataB）の get メソッドを呼び出す．条件式は全体を表すクラス（ControllerA）の exec メソッドにおける条件式と同じものを記述する．

図 25 (b) の Simulink モデルにおいて，Stateflow ブロックが出力する選択信号がしきい値 0.5 よりも大きい場合は，Subsystem3 ブロック〜Subsystem1 ブロックのみ，しきい値 0.5 よりも小さい場合は，Subsystem3 ブロックと Subsystem2 ブロック〜Subsystem4 ブロックのみの計算を

実行すればよい．(a) と異なり Subsystem3 ブロックはいずれの場合も実行する．

図 25 (b) の Simulink モデルをシーケンス図に変換するルールは以下のとおりである．まず全体オブジェクト（ControllerA）が Stateflow ブロックに対応したオブジェクト（Mode）の get メソッドを呼び出すメッセージを生成する．次に，条件ごとに，Switch ブロックにつながる Subsystem ブロック（Mode > 0.5 の場合は，DataX～DataA，そうでない場合は，DataX，DataZ～DataB）をすべて探索する．そして，条件分岐を表す Alternative フラグメントを生成し，条件ごとに，入力である Inport ブロックに最も近い Subsystem ブロックから順に Subsystem ブロックに対応したオブジェクトの update メソッドを呼び出すメッセージを生成する．この際，同じ Subsystem ブロックに出力される複数の Subsystem ブロック（DataX，DataZ）は，並行実行が可能である．よって，並行実行を表す Paralell フラグメント（図中の par）を生成し，並行実行可能な Subsystem ブロックに対応したオブジェクト（DataX，DataZ）の update メソッドを呼び出すメッセージを生成する．最後に，(a) の Simulink モデルをシーケンス図に変換する場合と同様，Switch ブロックに対応したオブジェクト（DataC）の update メソッドを呼び出すメッセージを生成する．

図 25 (b) の Simulink モデルをアクティビティ図に変換するルールは，全体を表すクラス（ControllerA）の exec メソッドと Switch ブロックに対応するクラス（DataC）の update メソッドのアクティビティを表すアクティビティ図を生成する．まず全体を表すクラス（ControllerA）の exec メソッドは，Stateflow ブロックに対応するクラス（Mode）の get メソッドを呼び出す．次に，条件ごとに，Switch ブロックにつながる Subsystem ブロック（Mode > 0.5 の場合は DataX～DataA，そうでない場合は，DataX，DataZ～DataB）をすべて探索する．そして，条件分岐を表すデシジョンノードを生成し，条件ごとに Inport ブロックに最も近い Subsystem ブロックから順に Subsystem ブロックに対応するクラスの update メソッドを呼び出す．この際，同じ Subsystem ブロックに出力される複数の Subsystem ブロック（DataX，DataZ）は，並行実行が可能である．よって，並行実行を表すフォークノードを生成し，並行実行可能な Subsystem ブロックに対応したクラス（DataX，DataZ）の update メソッドを呼び出す．最後に，Switch ブロックに対応するクラス（DataC）の update メソッドを呼び出す．Switch ブロックに対応するクラス（DataC）の update メソッドは，(a) の Simulink モデルをアクティビティ図に変換する場合と同様の変換を行う．

コンポジションを考慮した変換を行った場合においても同様の考え方で変換できる．そして，3 章および 4 章で述べた変換規則と本章で述べた変換規則を組み合わせることで複雑な Simulink モデルを変換することができる．

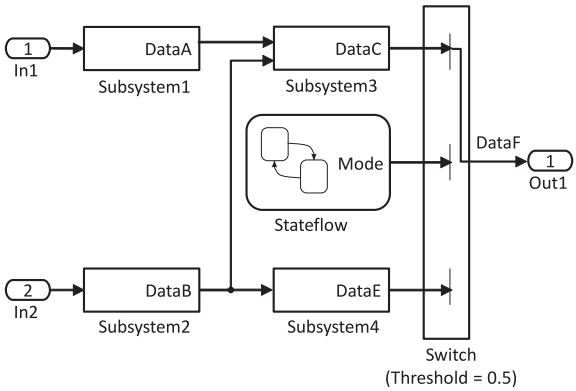

図 26 データフローの分岐や合流を含む Simulink モデル
Fig. 26 Simulink Model with condition and joint of data flows.

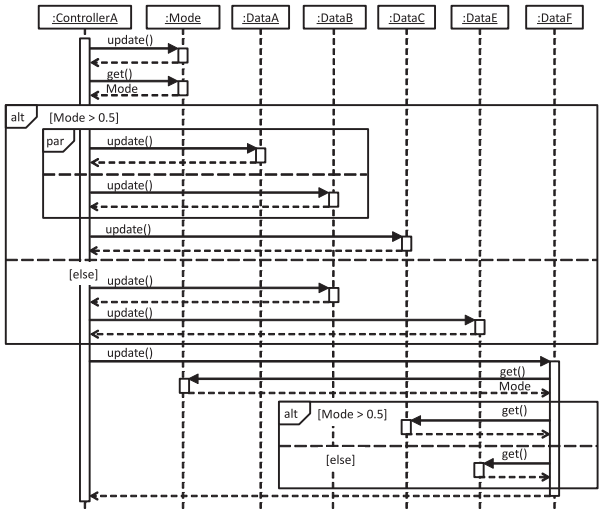

図 27 図 26 の Simulink モデルに対応するシーケンス図
Fig. 27 Sequence diagram corresponding to the Simulink Model shown by Fig. 26.

5.2 変換例

図 26 の Simulink モデルをシーケンス図に変換すると，**図 27** に示すシーケンス図が得られる．ControllerA オブジェクトの exec メソッドは状態値の更新および状態値の取得をした後，実行する処理の切替えを行う．状態値がしきい値 0.5 よりも大きい場合は，DataA オブジェクトと DataB オブジェクトの update メソッドを呼び出した後，DataC オブジェクトの update メソッドを呼び出す．状態値がしきい値 0.5 よりも小さい場合は，DataB オブジェクトの update メソッドを呼び出した後，DataE オブジェクトの update メソッドを呼び出す．最後に Switch ブロックに対応した DataF オブジェクトの update メソッドを呼び出し，実行した処理の結果を取得する．

図 26 の Simulink モデルをアクティビティ図に変換すると，**図 28** に示すアクティビティ図が得られる．ControllerA クラスの exec メソッドは状態値の更新および状態

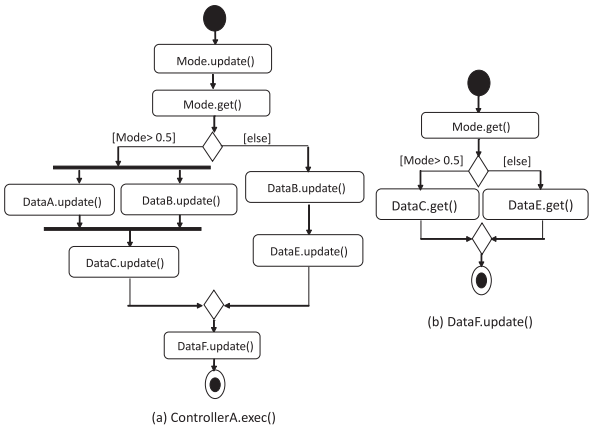

図 28 図 26 の Simulink モデルに対応するアクティビティ図
Fig. 28 Activity diagram corresponding to the Simulink Model shown by Fig. 26.

値の取得をした後，実行する処理の切替えを行う．状態値がしきい値 0.5 よりも大きい場合は，DataA クラスと DataB クラスの update メソッドを呼び出し，その後 DataC クラスの update メソッドを呼び出す．状態値がしきい値 0.5 よりも小さい場合は DataB クラスの update メソッドを並行に呼び出し，その後 DataE クラスの update メソッドを呼び出す．次に Switch ブロックに対応した DataF クラスの update メソッドを呼び出す．DataF クラスの update メソッドは状態値の取得を行い，状態値がしきい値 0.5 より大きい場合は DataC クラスの get メソッドを呼び出す．状態値がしきい値 0.5 より小さい場合は DataE クラスの get メソッドを呼び出す．

6. 変換ツールの実装

本研究で開発したモデル変換ツールの構成を**図 29** に示す．本ツールは Java 言語で開発しており，Simulink モデルのファイル保存形式である mdl ファイルを入力し，UML モデルの標準的なファイル保存形式である XMI ファイルを出力する．

まず，ツールは入力された mdl ファイルを解析し変換に必要なデータを抽出した Simulink モデルデータを生成する．続いて生成した Simulink データをもとに，クラス図変換を行う．クラス図変換では，Simulink モデルデータからクラス図データであるクラスや関連等を生成する．そして，Simulink モデルデータと生成したクラス図データをもとにオブジェクト図変換を行う．オブジェクト図変換では，Simulink モデルのラインからリンクをクラスからオブジェクトを生成する．その後，Simulink モデルデータとクラス図データをもとにシーケンス図変換を行う．シーケンス図変換では，Simulink モデルデータのラインデータとクラス図データのクラスからシーケンス図を生成する．アクティビティ図変換では，Simulink モデルデータのラインデータとクラス図データのクラスからアクティビティ図を

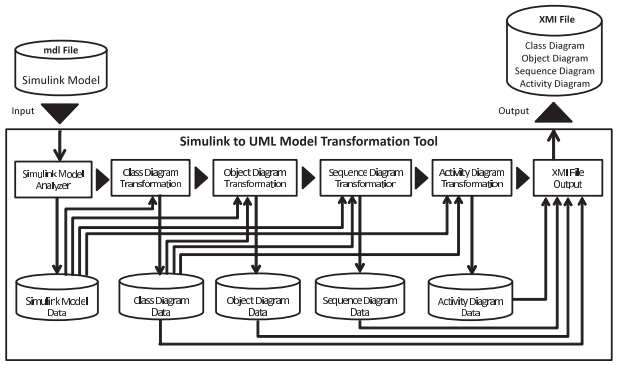

図 29 モデル変換ツール
Fig. 29 Model Transformation Tool.

表 1 ブロックの内訳
Table 1 Number of blocks of Simulink Models.

ブロックの種類	クルーズコントロール制御システム	フォールトトレラント燃料噴射システム	
		解説書	R2013
階層化前総ブロック	21	53	31
階層化後総ブロック	17	30	21
Inport	8	4	4
Subsystem	4	20	13
Stateflow	1	1	1
Switch	1	3	0
Outport	3	1	1
MultiportSwitch	0	1	0
SwitchCase	0	0	1
Merge	0	0	1

生成する．最後に，クラス図データとオブジェクト図データ，シーケンス図データ，アクティビティ図データを XMI ファイルとして生成する．

7. 適用実験

開発したモデル変換ツールの有用性を評価するため，状態遷移や条件分岐を含むクルーズコントロール制御システム [12] と 2 つのフォールトトレラント燃料噴射システム [13], [14] の 3 つの Simulink モデルに対して適用実験を行った．

3 つの Simulink モデルの階層化前と階層化後のブロックの内訳を**表 1** に示す．3 つの Simulink モデルをそれぞれ入力し本モデル変換ツールから出力された UML モデルの各要素数を**表 2** に示す．

また，例としてフォールトトレラント燃料噴射システム R2013 版の Simulink モデルを**図 30** に示す．また，変換後の UML モデルのクラス図を**図 31** に，オブジェクト図を**図 32** に，シーケンス図を**図 33** に，アクティビティ図の一部を**図 34** に示す．

クルーズコントロール制御システムは Switch ブロック

表 2 UML モデルの要素数
Table 2 Number of elements of UML Models.

要素	クルーズコントロール制御システム	フォールトトレラント燃料噴射システム	
		解説書	R2013
クラス数	17	32	22
オブジェクト数	15	30	20
メッセージ数	44	288	154
アクティビティ数	6	28	15
ノード数	35	203	108

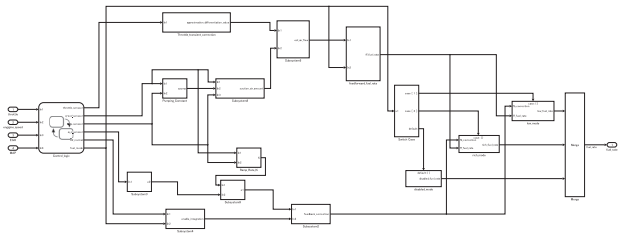

図 30 フォールトトレラント燃料噴射システムの Simulink モデル
Fig. 30 Simulink Model for Fault Tolerant Fuel System.

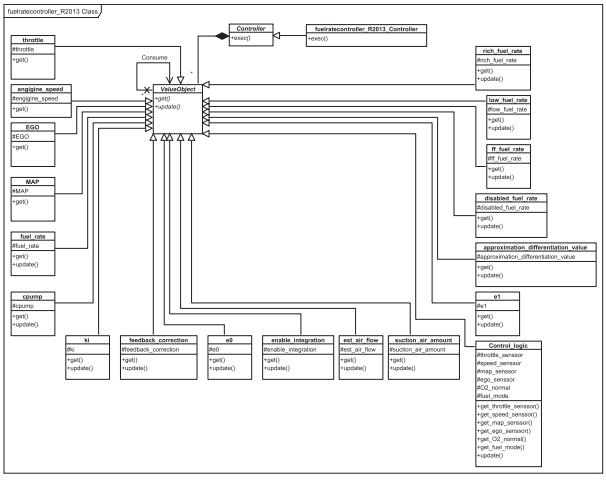

図 31 フォールトトレラント燃料噴射システムのクラス図
Fig. 31 Class diagram for Fault Tolerant Fuel System.

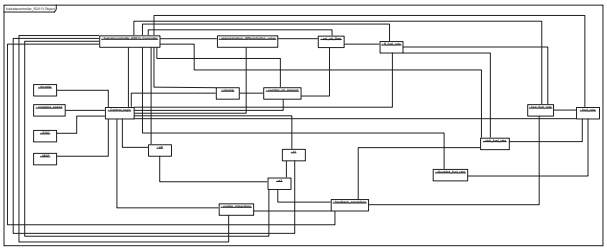

図 32 フォールトトレラント燃料噴射システムのオブジェクト図
Fig. 32 Object diagram for Fault Tolerant Fuel System.

を用いて記述されており，コンポジションを考慮した変換方法を適用している．また，サイバネットシステム株式会社のフォールトトレラント燃料噴射システムは Switch ブロックと MultiportSwitch ブロックを用いて記述されており，Switch ブロックにはコンポジションを考慮した変換方

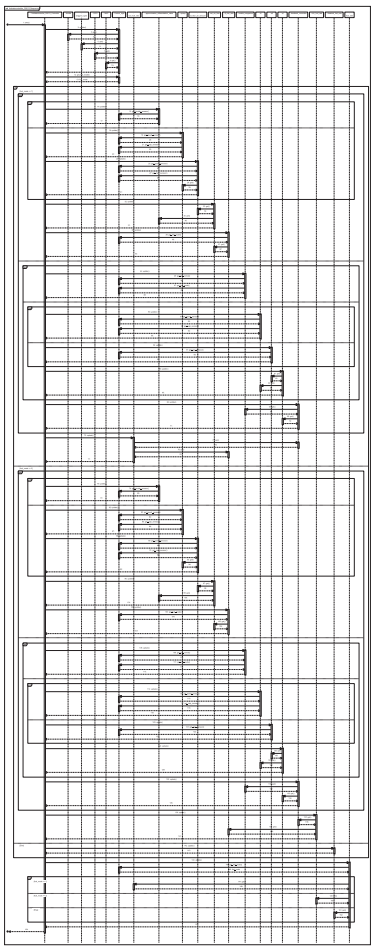

図 33 フォールトトレラント燃料噴射システムのシーケンス図
Fig. 33 Sequence diagram for Fault Tolerant Fuel System.

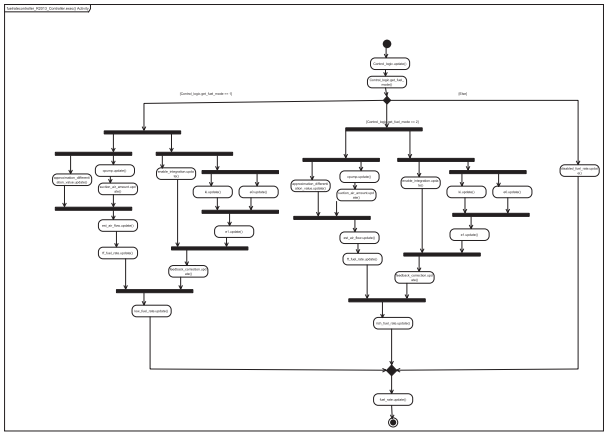

図 34 フォールトトレラント燃料噴射システムのアクティビティ図
Fig. 34 Activity diagram for Fault Tolerant Fuel System.

法を，MultiportSwitch ブロックには選択のあるデータフローに関する変換方法を適用している．

MATLAB R2013 のサンプルモデルであるフォールトトレラント燃料噴射システムは，MultiportSwitch ブロックを SwitchCase ブロックのパターンを用いて記述されており，制御フローが条件分岐を含む場合の変換方法を適用している．

以上により，状態遷移や条件分岐を含む Simulink モデルを本モデル変換ツールに入力することで，状態遷移や条件分岐に応じて実行する処理を切り替えることで不要な計算を排除した効率の良い UML モデルに変換できることを確認した．

8. 関連研究との比較

変換方法について，1 章で取りあげた 3 つの従来研究と本研究とを比較し，違いを明らかにする．

Ramos-Hernandez らは，Simulink モデルの各ブロックをインタフェースクラスに，各ラインを依存関係に変換している．これらの手法は，図 3 のブロックにはすべて対応しているが 1 対 1 対応変換であるため，本研究のようなコンポジションを考慮した変換は行っていない．

また，Müller-Glaser らは Simulink モデルの各ブロック，ライン，分岐をクラスとオブジェクトに変換している．こちらも，図 3 のブロックにはすべて対応しているが 1 対 1 対応変換であるため，本研究のようなコンポジションを考慮した変換は行っていない．

そして，Sjöstedt らは Simulink モデルを UML モデルの複合構造図とアクティビティ図に変換する方法を提案している．Simulink モデルを UML モデルの複合構造図に変換する際は，各ブロックをクラスに，ラインをコネクタに変換している．こちらも，図 3 のブロックにはすべて対応しているが 1 対 1 対応変換であるため，本研究のようなコンポジションを考慮した変換は行っていない．また，Simulink モデルを UML モデルのアクティビティ図に変換する際は，Simulink モデルのシミュレーション実行順序をアクティビティ図に変換しており，データフローと制御フローを考慮した UML モデルとはなっていない．

これらに対し我々のモデル変換ツールは，データフローだけでなく制御フローを考慮した効率の良い UML モデルを生成できるという点で利点があると考える．

9. おわりに

状態遷移や条件分岐を含む Simulink モデルを入力し，状態遷移や条件分岐に応じて実行する処理を切り替える UML モデルを出力するモデル変換ツールを開発した．そして，複数の Simulink モデルに対して適用実験を行い，その有用性を確認した．

謝辞 本研究は JSPS 科研費 24500046 および 15K00084 の助成を受けたものである．

参考文献

[1] The MathWorks Inc.: Simulink (online), available from ⟨http://www.mathworks.com/products/simulink/⟩.
[2] Sangiovanni-Vincentelli, A. and Di Natale, M.: Embedded System Design for Automotive Applications, *IEEE Computer*, Vol.40, No.10, pp.42–51 (2007).
[3] Ramos-Hernandez, D.N., Fleming, P.J., Bennett, S., Hope, S., Bass, J.M. and Baxter, M.J.: Process control systems integration using object oriented technology, *Proc. Technology of Object-Oriented Languages and Systems TOOLS 38*, pp.148–158 (2001).
[4] Ramos-Hernandez, D.N., Fleming, P.J. and Bass, J.M.: A novel object-oriented environment for distributed process control systems, *Control Engineering Practice*, Vol.13, No.2, pp.213–230 (2005).
[5] Kühl, M., Spitzer, B., Müller-Glaser, K.D. and Dambacher, U.: Universal object-oriented modeling for rapid prototyping of embedded electronic systems, *Proc. 12th IEEE International Workshop on Rapid System Prototyping*, pp.149–154 (2001).
[6] Müller-Glaser, K.D., Frick, G., Sax, E. and Kühl, M.: Multiparadigm modeling in embedded systems design, *IEEE Trans. Control Systems Technology*, Vol.12, No.2, pp.279–292 (2004).
[7] Sjöstedt, C.-J., Shi, J., Törngren, M., Servat, D., Chen, D., Ahlsten, V. and Lönn, H.: Mapping Simulink to UML in the design of embedded systems: Investigating scenarios and structural and behavioral mapping, *Proc. OMER 4 Post Workshop* (2008).
[8] 田村雅成，神山達哉，添田隆弘，兪　明連，横山孝典：Simulink モデルと UML モデルを用いた組み込み制御ソフトウェア開発のためのモデル変換環境，情報処理学会論文誌，Vol.53, No.12, pp.2660–2670 (2012).
[9] MathWorks Automotive Advisory Board (MAAB): Control Algorithm Modeling Guidelines Using MATLAB(R), Simulink(R), and Stateflow(R), version 3.0 edition (2012).
[10] Japan MBD Automotive Advisory Board (JMAAB): Control Algorithm Modeling Guidelines Using MATLAB(R), Simulink(R), and Stateflow(R), version 4.0 edition (2015).
[11] 横山孝典，納谷英光，成沢文雄，倉垣　智，永浦　渉，今井崇明，鈴木昭二：組込み制御システムのための時間駆動オブジェクト指向ソフトウェア開発法，電子情報通信学会論文誌 D-I，Vol.J84-D1, No.4, pp.338–349 (2001).
[12] サイバネットシステム株式会社：Simulink/Stateflow サンプルモデル解説書—クルーズコントロール制御編 (2004).
[13] サイバネットシステム株式会社：Simulink/Stateflow サンプルモデル解説書—フォールトトレラント燃料噴射システム編 (2003).
[14] The MathWorks Inc.：フォールトトレラント燃料制御システムのモデル化（オンライン），入手先 ⟨http://jp.mathworks.com/help/simulink/examples/modeling-a-fault-tolerant-fuel-control-system.html⟩.

黒木 裕太 （正会員）

1990 年生．2014 年東京都市大学知識工学部情報科学科卒業．2016 年同大学大学院工学研究科情報工学専攻修士課程修了．同年株式会社シンカーミクセル入社．業務システムのソフトウェア開発に従事．

田中 亨祐 （学生会員）

1993 年生．2015 年東京都市大学知識工学部情報科学科卒業．同年同大学大学院工学研究科情報工学専攻修士課程入学．現在，同課程在学中．ソフトウェア工学の研究に従事．

俞 明連 （正会員）

1994 年安東国立大学校工学部コンピュータ工学科卒業．1996 年浦項工科大学情報通信専攻修士課程修了．同年安東情報大学講師．2002 年嶺南大学コンピュータ工学専攻博士課程修了．2006 年早稲田大学大学院情報生産システム研究科博士後期課程修了．2007 年武蔵工業大学．現在，東京都市大学准教授．スケジューリング理論の研究に従事．博士（工学）．電子情報通信学会，IEEE 各会員．

横山 孝典 （正会員）

1981 年東北大学工学部通信工学科卒業．1983 年同大学大学院工学研究科電気及通信工学専攻修士課程修了．同年（株）日立製作所入社．1987 年から 1990 年まで（財）新世代コンピュータ技術開発機構出向．2004 年武蔵工業大学．現在，東京都市大学教授．組み込みシステム，分散システム，ソフトウェア工学等の研究に従事．博士（情報科学）．電子情報通信学会，IEEE，ACM 各会員．