Vol.2010-EMB-18 No.7
2010/8/9

組み込み制御ソフトウェア開発のための Simulink・UML モデル変換ツール

神山達哉[†1]　添田隆弘[†1]
兪明連[†1]　横山孝典[†1]

モデルベース開発による組み込み制御ソフトウェアの開発効率向上のため，Simulink モデルから UML モデルへの変換方法および自動変換ツールを提案する．一般に組み込み制御ソフトウェアの開発は制御設計とソフトウェア設計の２段階で行われる．制御設計では MATLAB/Simulink を用いて制御ロジックを Simulink モデルで表すことが多い．一方ソフトウェア設計では UML モデルを用いて設計を行うのが普通である．しかし，Simulink モデルと UML モデルとの間で明確な対応付けがないため，それらモデルの統合は容易ではない．そこで本研究では，Simulink モデルからソフトウェア設計に適した形の UML モデルへ変換する方法を提案するとともに，その方法に基づき自動変換するツールを開発した．そして，本変換ツールを用いて，いくつかの組み込み制御システムの Simulink モデルに対して変換実験を行い，実際の組み込み制御ソフトウェア開発に適用できる見通しを得た．

A Simulink to UML Model Translator for Embedded Control Software Development

Tatsuya Kamiyama,[†1] Takahiro Soeda,[†1]
Myungryun Yoo[†1] and Takanori Yokoyama[†1]

The paper presents a method and a tool to transrate a Simulink model into a UML model. The embedded control software development process can be divided into the control logic design phase and the software design phase. In the control logic design phase, a Simulink model is built with MATLAB/Simulink. In the software design phase, UML is widely used to build a software model. There is no standard method to translate a Simulink model into a UML model, so it is difficult to integrate those models. We have developed a Simulink to UML model translator to generate a UML model appropriate for software design. We have applied the translator to a number of Simulink models and have found it useful for embedded control software dvelopment.

1. はじめに

自動車，FA，ロボットなど広い分野で組み込み制御システムが用いられ，そのソフトウェア開発量が増大している．一般に組み込み制御ソフトウェアの開発は，制御設計者が制御ロジックを設計する制御設計と，制御ロジックに基づいてソフトウェア設計者が制御プログラムを作成するソフトウェア設計の２段階で行われる．

近年，制御設計は MATALAB/Simulink[1)] 等の制御系 CAE/CAD ツールを用いたモデルベース開発が主流となってきている．MATLAB/Simulink では，ブロック線図形式のモデル（以下 Simulink モデル）で制御ロジックを記述する．そして，作成したモデルのシミュレーション検証を行うことで，制御ロジック誤りの早期発見が可能となる．さらに，制御ロジックをモデルで表すことで，自然言語による記述で発生しがちであった，制御ロジックの誤認識によるヒューマンエラーやコミュニケーションミスを削減できる．また，Simulink モデルからプログラムを自動生成するツールである Real-Time Workshop Embedded Coder[1)] を活用することで，プログラム自動生成による開発効率向上も期待されている．

しかし，組み込み制御システムを構成するすべてのソフトウェアを Simulink モデルで記述できるわけではない．Simulink モデルはフィードバック制御やフィードフォワード制御等の制御ロジックを表すのに適しているが，手続き的な処理の記述には向いていない．そのため，手続き的な記述が適したアプリケーションや通信処理等は UML 等を用いて別に設計し，実装時に制御プログラムと組み合わせなければならない．

また，MATLAB/Simulink のシミュレーションは，制御ロジックそのものの検証が目的のため処理時間をゼロとして行っており，タスク間のプリエンプション等は考慮されていない．ところが現実の組み込み制御システムは，プリエンティブなマルチタスク環境で動作させるため，タスク間の同期や通信の機構を組み込む必要がある．組み込み制御システムの実装方法には時間駆動に基づく構成法とイベント駆動に基づく構成法があるが[2)]，両者を混在させることも多く，タスク間の同期や通信の設計，実装は重要である．しかし，Simulink モデルから自動生成されるプログラムは構造化されておらず，可読性もよくないため，同期や通信処理の追加や修正は容易ではない．Simulink モデルから，タスク間通信機構を含ん

†1 東京都市大学
Tokyo City University

だソースコードを生成する研究もなされているが[3]，必ずしも最適なコードを生成するわけではない．また，コンポーネント単位の Simulink モデルからソースコードを自動生成する場合には，コンポーネント間を接続する機構の自動生成はできない．ソースコードレベルではなく，モデルレベルでコンポーネントを組み合わせたり，タスク間の同期や通信機構を設計できることが望ましい．

そこで我々は，制御設計で開発した Simulink モデルを一旦 UML モデルに変換し，その UML モデルをもとにソフトウェア設計を行うことが有効と考える．制御ロジックを UML モデルに変換することで，制御ロジック以外の UML モデルと組み合わせたり，タスク間の同期や通信の機構を組み込むことが容易になる．そのためには，ソフトウェア設計に適した形の UML モデルに効率よく変換することが求められる．

ところが，制御設計で用いられる Simulink モデルとソフトウェア設計で使用される UML モデルでは，その狙いや対象，モジュール化のしかた，表記法などが異なり，それらの間に明確な対応付けはなされていない．そのため，Simulink モデルをソフトウェア設計に適した形の UML モデルに変換する標準的な手法は存在しない．

吉田らは，Simulink モデルから UML モデルへの変換方法を提案している[4]．この手法では，Simulink モデルはタスクを呼び出すスケジューラ，インタフェースであるデバイス，そして個々のタスクのみからなると規定している．そして，Simulink モデル内のスケジューラ，デバイス，タスクを構成している部分をクラスとして抽出し，UML モデルへ変換する．しかし，この方法は Simulink モデルの構成を限定しているため汎用性に欠ける．また，変換のためには Simulink モデルの解析及び UML モデルの作成を人手で行わなければならないため，変換に多くの工数がかかる．

Ramos-Hernandez らは，ツールにより自動変換を行う方法を提案している[5][6]．このツールは，ブロックひとつひとつをインタフェースクラスに変換する．ブロック間を接続するラインが分岐していた場合には，その部分に Demux というインタフェースクラスを作成する．また，ブロック間の繋がりを依存関係を用いて表す．この手法では，Simulink モデルの構成がそのまま UML モデルに反映されるが，ブロックひとつひとつをクラスとしているため，必ずしも再利用性の高い UML モデルにはならない．

Müller-Glaser らは，Simulink モデルの要素をそれぞれオブジェクトとして UML モデルに自動変換する方法を提案している[7]–[9]．この手法は，Simulink モデルのブロック，ライン，分岐をそれぞれオブジェクトに対応させたオブジェクト図を生成する．ラインが分岐している場合は，分岐を表すオブジェクトを生成する．この手法では，Simulink モデルの構造がそのまま UML モデルに反映されるため，必ずしもソフトウェアとして好適なオブジェクト構造にはならず，新たな機能追加や部品化再利用は容易ではない．

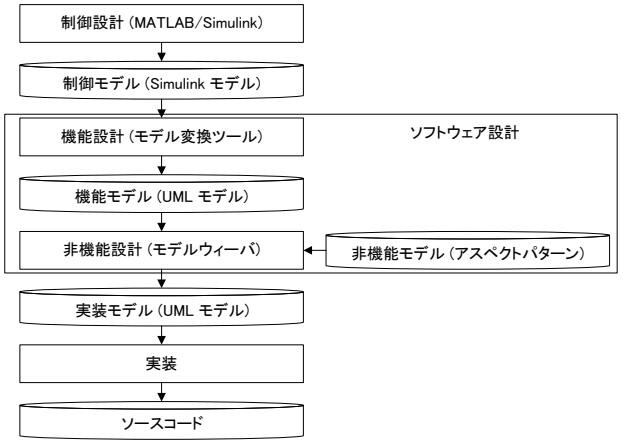

図 1 組み込み制御ソフトウェア開発の流れ
Fig. 1 Development flow of embeded control software

そこで本研究の目的は，Simulink モデルを用いた制御設計から UML を用いたソフトウェア設計への移行手段として，Simulink モデルをソフトウェア設計に適した形の UML モデルに変換する方法を提案するとともに，その方法に基づいて自動変換するツールを開発することである．これにより，制御設計からソフトウェア設計への自然な移行が可能となる．また，変換作業を自動化することで，開発効率を向上できるとともに人手による変換ミスを無すこともできる．

以下本論文では，第 2 章でモデル変換を用いた組み込み制御ソフトウェア開発全体の流れについて述べる．第 3 章では Simulink モデルから UML モデルへの変換の方法について説明し，第 4 章で開発した自動変換ツールについて述べる．そして第 5 章で自動変換ツールを用いたモデルの変換例を紹介し，第 6 章でまとめを述べる．

2. 組み込み制御ソフトウェア開発の流れ

我々の提案している組み込み制御ソフトウェア開発の流れを**図 1** に示す．開発全体は，制御設計工程，ソフトウェア設計工程，実装工程からなる．制御設計工程では，MATLAB/Simulink

を用いて制御モデル（Simulink モデル）を作成する．ソフトウェア設計工程では，制御機能のみを表現し，実装については考慮していない制御モデルをもとに，実装を考慮した実装モデル（UML モデル）を開発する．我々は図 1 に示したように，ソフトウェア設計工程をさらに機能設計フェーズと非機能設計フェーズの 2 段階に分けている．

機能設計では，制御ロジックを表現した Simulink モデルを，ソフトウェア設計に適した形態の UML モデルに変換する．変換により得られた UML モデルは，制御に関する機能モデルである．従来は手作業により Simulink モデルを UML モデルに変換していたが[10]，本論文ではこのモデル変換作業を自動化するモデル変換ツールを提案する．

次に非機能設計において，機能モデルである UML モデルをもとに，リアルタイム性や信頼性を考慮した実装モデルである UML モデルを作成する．我々は効率的な非機能設計を実現するため，アスペクト指向モデリングに基づいて非機能的処理をアスペクトパターン化し，それを機能モデルに織り込む手法を提案している[11]．これまでに，リアルタイム性を実現するタイミング設計のためのアスペクトパターンを提案し，アスペクトパターンをモデルに織り込むモデルウィーバを開発している．機能モデルである UML モデルに対して，モデルウィーバを用いてアスペクトパターンを織り込むことで，実装モデルである UML モデルを得ることができる．

最後の実装工程では，実装モデルに基づいて制御ソフトウェアのプログラムを作成する．プログラムは，実装モデルである UML モデルから，あるいは制御ロジックの詳細を記述した Simulink モデルから自動生成することが可能である．

3. Simulink モデルと UML モデル

3.1 Simulink モデル

Simulink モデルの記述は自由度が高く，用途や設計者により様々な記述方法を用いることが可能である．しかし，製品開発における制御設計では制御ロジックの理解しやすさや制御モデルの再利用性のため，一定の基準に基づいて Simulink モデルを記述すべきである．またモデル変換を自動化するためにも，変換対象とする Simulink モデルの記述法の基準を明確にする必要がある．以下，本研究で変換対象とする Simulink モデルについて説明する．

一般に，対象システムを表現する Simulink モデルは，制御対象であるプラントモデルと，対象を制御するコントローラモデルの 2 つからなる．制御ソフトウェアとして実装すべき部分はコントローラモデルであるため，変換対象とするのはコントローラモデルである．コントローラモデルは，センサ等からのデータを得るための入力と，プラントモデルへ制御

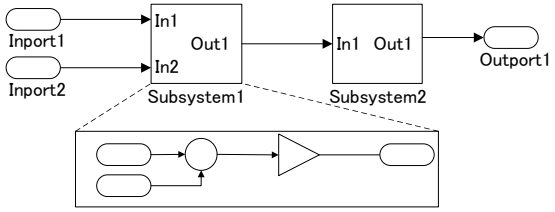

図 2 階層化された Simulink モデルの例
Fig. 2 Layered Simulink model

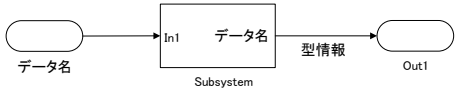

図 3 データ名の記述
Fig. 3 Discription of data names

データを送るための出力を持つ．Simulink モデルでは，入力には Inport ブロック，出力には Outport ブロックを用いる．

ごく単純な場合を除いて，制御ロジックを理解しやすくするため Simulink モデルは階層化して記述すべきである．したがって変換対象とする Simulink モデルも階層化を行ったものとする．Simulink では，Subsystem ブロックを用いてモデルを階層化する．データを算出するための具体的な処理は Subsystem ブロックの下位階層に記述する．**図 2** に階層化した Simulink モデルの例を示す．

以上により，変換対象とする Simulink モデルの最上位階層は Inport ブロック，Outport ブロック，Subsystem ブロックの 3 種類のブロックで構成する．ここで，最上位階層に現れる Subsystem ブロックは，制御ロジックにおいて重要な意味を持つデータ（物理量）を算出するブロックとなるように構成する．これにより，Simulink モデルを Subsystem 単位で再利用可能な構造にすることができる．

最上位階層に現れるデータには名前をつけることとする．前述のように，通常は制御ロジック上意味のある物理量がデータ名となる．データ名の記述法を**図 3** に示す．Inport ブロックから出ているデータの名前は Inport ブロック名に記述する．次に，Subsystem ブロックから出ているデータの名前は，Subsystem ブロック内の Outport ブロック名に記述する．そうすることで，Subsystem ブロックにデータ名が表示される．また，名前とは別に，データの型情報を付けたい場合は，そのデータを表しているラインのライン名にデータ

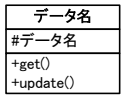

図 4 データを表す基底クラス
Fig. 4 Base class of data object

の型を記述する．これらは Simulink で一般的な名前の付け方である．

以上をまとめ，変換対象とする Simulink モデルが従うべき基準を以下に示す．

- 上位階層では，Inport ブロック，Outport ブロック，Subsystem ブロックの 3 種類のブロックで構成されるように階層化する．
- 上位階層に現れるデータには名前を付ける．

3.2 UML モデル

本論文で提案するモデル変換ツールが生成する UML モデルはクラス図とオブジェクト図とする．クラス図を生成対象とするのは，それがオブジェクト指向に基づくソフトウェア設計において最も重要な静的構造図であるためである．

一方オブジェクト図は，システムが動作している特定の時点での，オブジェクトとオブジェクト間の関連を表現するための図である．多くの組み込み制御システムでは，オブジェクトを動的に生成する必要はなく，固定的なオブジェクト構成となることが多い[12]．このため，組み込み制御システムの定常的なオブジェクト構成をオブジェクト図を用いて表すことが可能である．そこで，オブジェクト図も生成対象とする．

本論文で提案するモデル変換は，制御ブロック図を用いた時間駆動オブジェクト指向ソフトウェア開発法[10] をベースとしており，制御ロジック上に現れるデータ（物理量）をオブジェクト（クラス）と見なす．データを表すオブジェクトの基底クラスを**図 4** に示す．クラス名はデータ名とする．データの値を記憶する属性と，そのデータの値を算出するメソッド update，値を読み出すメソッド get を持つ．

3.3 Simulink モデルと UML モデルの対応

Simulink モデルと UML モデルの対応付けについて説明する．例として，自動車車間システムのスロットル開度を算出する部分の Simulink モデルを**図 5** に示す．図 5 のモデルは，エンジン回転数（engine revolution），エンジン状態（engine status），アクセル開度（accelerator opening）を入力し，スロットル開度（throttle opening）を出力する．エンジン回転数，エンジン状態，アクセル開度，トルク（torque），スロットル開度の５つのみが，上位階層に現れるように階層化している．トルクとスロットル開度を求める具体的な処理は，それぞれの Subsystem ブロックであるトルク算出（torque calculation）とスロットル開度算出（throttle opening calculation）の下位階層で行っている．本モデルは，まず，エンジン状態とアクセル開度を用いてトルクを算出する．次に，算出したトルクとエンジン回転数とエンジン状態を用いてスロットル開度を算出し，出力する．なお，トルクとスロットル開度のデータ型は uint16（符号なし 16bit 整数）としている．

図 5 の Simulink モデルに対応するクラス図を**図 6** に示す．図 6 のクラス図は，エンジン回転数，エンジン状態，アクセル開度，トルク，スロットル開度の 5 つのクラスから構成されている．クラス間の関連 cons は，参照先のクラスの属性値を参照することを表している．例えば，トルククラスはエンジン状態クラスの属性値とアクセル開度の属性値を参照する．具体的にはトルククラスのメソッド update 中で，エンジン状態クラスのメソッド get，アクセル開度クラスのメソッド get を呼び出して，それらの属性値を得ている．そして，それらの値を用いてトルクの値を算出し，属性値に記憶する．同様にスロットル開度はエンジン回転数とエンジン状態とトルクを参照し，トルク値を算出して記憶する．以上により，Simulink モデルに対応したクラス図を実現できる．クラスは，上位階層の Subsystem ブロックに対応するため，クラス図は制御上重要な意味を持つデータに対応するクラスから構成される．このためクラス単位での再利用に適した構成となっている．

図 5 の Simulink モデルから変換したオブジェクト図を**図 7** に示す．図 7 のオブジェクト図では，Simulink モデルの５つのデータに対応したクラスのオブジェクトが生成され、それらオブジェクト間はリンクでつながっている．リンクのつながりは，図 5 の Simulink モデルに，また，図 6 のクラス図とも対応している．このオブジェクト図により，クラス図では表せない実際のオブジェクト間のつながりを表現できる．なお，通常 Simulink モデルは異なる種類のデータ及び，異なる計算をするブロックから構成されるため，各クラスにひとつずつオブジェクトが生成されることとなる．

4. モデル変換ツール

4.1 変 換 方 法

Simulink モデルからクラス図とオブジェクト図への変換方法を述べる．変換は Simulink モデルの上位階層に現れるデータに着目して，**図 8** に示す変換規則に従い，以下の手順で行う．

(1) 上位階層のデータからクラスとオブジェクトを生成．
(2) 上位階層のデータのつながりから，関連やリンクを生成．

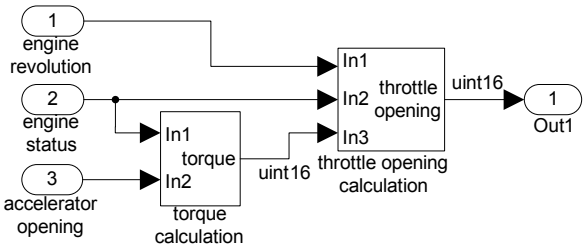

図 5 Simulink モデルの例
Fig. 5 An example Simulink model

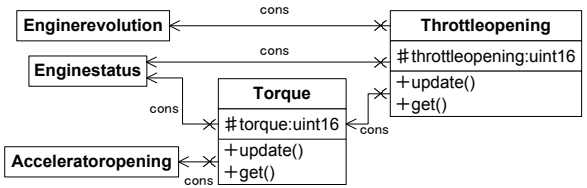

図 6 クラス図の例
Fig. 6 An example class diagram

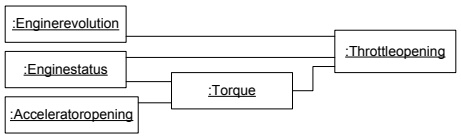

図 7 オブジェクト図の例
Fig. 7 An example object diagram

まず，手順（1）でクラスとオブジェクトを生成する．上位階層のデータは Inport ブロックから出ているものと，Subsystem ブロックから出ているものの 2 通りある．図 8（a）に示すように，Inport ブロックから出ているデータからは，データ名をクラス名に持つクラスと，データ名を分類子に持つオブジェクトを生成する．図 8（b）に示すように，Subsystem ブロックから出ているデータからは，属性値とメソッドを持つクラスとデータ名を分類子に持つオブジェクトを生成する．クラスのクラス名と属性値名にはデータ名を記述し，型情報があるものには，属性値に型情報を付加する．メソッドは，属性値を渡すための get メソッドと，属性値を更新する update メソッドを記述する．

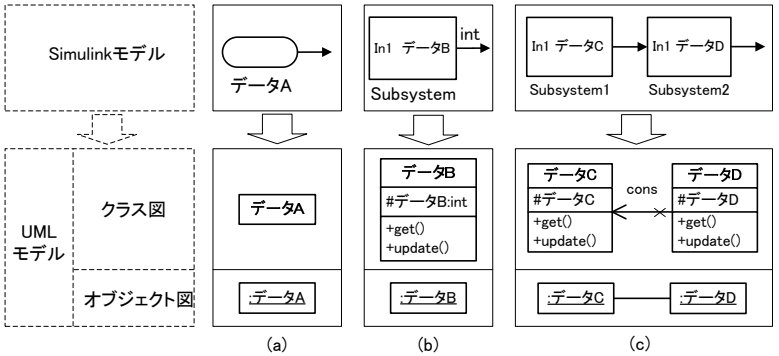

図 8 Simulink モデルと UML モデルの変換規則
Fig. 8 Translation rules from Simulink model to UML model

次に，手順（2）で Simulink モデルでのデータのつながりから，クラス間の関連とオブジェクト間のリンクを生成する．それらの対応を図 8（c）に示す．図 8（c）の Simulink モデルは，データ C を用いてデータ D を算出しているモデルである．ここから，クラス図では誘導可能性を用いて関連を生成する．オブジェクト図ではリンクを生成する．

以上の変換手順を Simulink モデル全体に行うことで，クラス図とオブジェクト図に変換することができる．

4.2 変換ツールの動作

開発した変換ツールの入出力と処理の流れを**図 9** に示す．変換ツールは，Simulink モデル情報が記述された mdl ファイルを入力し，生成したの UML モデルの XMI(XML Metadata Interchange) ファイルを出力する．変換ツールの内部処理は，まず入力した mdl ファイルを解析し，変換に必要なデータを抽出する．次に，変換処理を行い，抽出されたデータから UML モデルのデータを作成する．作成された UML モデルデータを用いて XMI ファイルを生成し出力する．

5. 適 用 実 験

これまでに燃料噴射システム[13]，自動車ハイブリッド駆動システム[14]，ステッピングモータ制御[15] 等の Simulink モデルに対して, 開発した変換ツールを用いてモデル変換実験を行った．そして，それら組み込みシステムの Simulink モデルについて，UML モデルに変換できることを確認した．変換後の XMI フィルを表示させる UML エディタは，SDE for

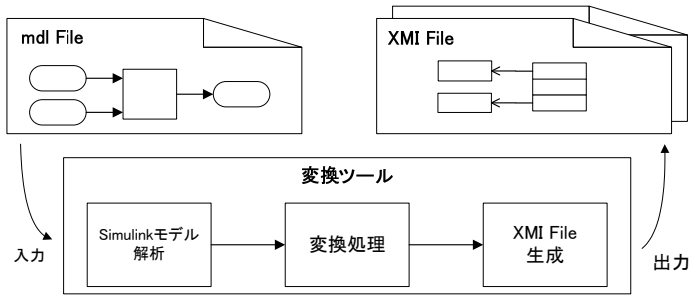

図 9 変換ツール
Fig. 9 Model translator

Eclipse(Smart Dvelopment Environment for Eclipse)[16] を用いた．

実験を行ったモデルのうち自動車ハイブリッド駆動システムの例を紹介する．本ハイブリットシステムは，エンジンとモータの２種類の動力源から構成されるシリーズ・パラレルハイブリットシステムである，エンジンの動力を動力分割機構により２つの動力に分割し，一方の動力で直接車輪を駆動し，他方の動力でジェネレータを駆動させて発電する．分割する動力の割合を自在に制御することで，エンジンとモータのそれぞれが持つ長所を活かすとともにお互いの不得意な点を補い，高効率な走行が可能となる．このシステムの車両走行を高効率化を目的に，エンジン，モータ，ジェネレータ，バッテリーのエネルギー収支を最適制御する制御コントローラの Simulink モデルが変換対象である．ただし，実際に変換対象とした Simulink モデルは，制御設計者が作成したものを 3.1 で示した基準に合致するように若干の修正を加えたものである．変換前の Simulink モデル，変換後のクラス図とオブジェクト図を**図 10** に示す．

以上のようにソフトウェアでの実装を考えていない純粋な制御ロジックを記述したモデルから，ソフトウェア構造を表す UML モデルに変換することができた．これにより，本ツールは実際の組み込み制御システムに適用可能と考えている．

6. おわりに

モデルベース開発による組み込み制御ソフトウェア開発における機能設計の効率向上を目的に，Simulink モデルから UML モデルへの変換方法を提案し，モデル変換ツールを開発した．本ツールにより制御ロジックを表す Simulink モデルからソフトウェア設計に適した

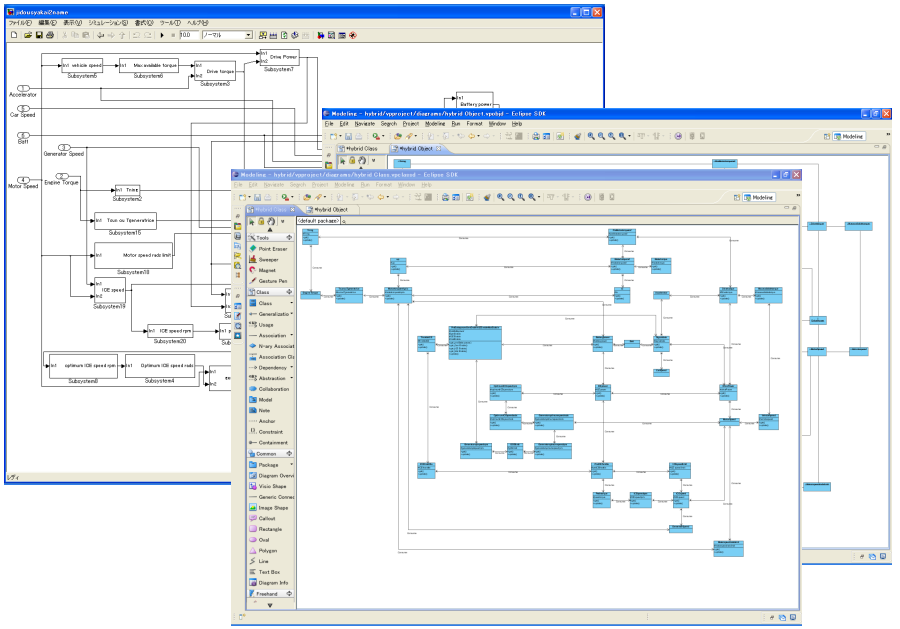

図 10 自動車ハイブリット駆動システムのモデル
Fig. 10 Simulink and UML models of a hybrid electric vehicle

UML モデルへの自動変換が可能になる．また，自動車ハイブリット駆動システムほか複数の Simulink モデルを対象に，開発した変換ツールの適用実験を行い，組み込み制御ソフトウェア開発に適用できる見通しを得た．

今後の課題として，まず，現在手作業で行っている Simulink モデルの階層化を自動化することが挙げられる．最上位階層に現れるデータを指定することで，階層化後の Simulink モデルを自動生成する機能を追加する予定である．また，クラス図やオブジェクト図以外の UML モデルも生成対象とすることを計画している．これにより，多岐にわたったソフトウェア設計に対応させることが可能となる．

謝辞 本研究は科研費（20500037）の助成を受けたものである．

Vol.2010-EMB-18 No.7
2010/8/9

参 考 文 献

1) The MathWorks, Inc., http://www.mathworks.com/
2) Kopetz, H.：Should Responsive Systems be Event-Triggered or Time-Triggered?, IEICE Transaction on Information & Systems, Vol.E76-D, No.11, pp.1325–1332 (1993).
3) Scaife, N. and Caspi, P.：Integrating Model-Based Design and Preemtive Scheduling in Mixed Time- and Event-Triggered Systems, Proceedings of 16th Eutomicro Conference on Real-Time Systems, pp.119–126 (2004).
4) 吉田聡，上田賀一，中島震：UML と Simulink のモデル変換手法の検討，電子情報通信学会信学技法，vol.109，no.231，pp.25–30 (2009).
5) 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, Issue2, pp.213–230 (2005).
6) Ramos-Hernandez, D.N., Zubuzarreta, I., Freming, P.J., Bennett, S. and Bass J.M.：Towards a Control Software Design Enviroment Using a Meta-Modelling Technique, Proceeding of 15th IFAC World Congress (2002).
7) Müller-Glaser, K.D., Frick, G., Sax E. and Kühl, M.：Multiparadigm Modeling in Embedded Systems Design, IEEE Transactions on Control Systems Technology, Vol.12, No.2, pp.279–292 (2004).
8) Kühl, M., Reichmann, C., Prötel, I. and Müller-Glaser K.D.:From Object-Oriented Modeling to Code Generation for Rapid Prototyping of Electronic Systems, Proceedings of the 13th IEEE International Workshop on Rapid System Prototyping, pp.108–114 (2002).
9) Kühl, M., Spitzer, B. and Müller-Glaser, K.D.：Universal Object-Oriented Modeling for Rapid Prototyping of Embedded Electronic Systems, Proceedings of the 12th IEEE International Workshop on Rapid System Prototyping, pp.149–154 (2001).
10) 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，Vol.J84-D-I，No.4，pp.338–349 (2001).
11) Soeda, T., Yanagidate, Y. and Yokoyama T.：Embedded Control Software Design with Aspect Patterns, Proceedings of International Conference on Advanced Software Engineering and Its Applications 2009, pp.34–41 (2009).
12) 成沢文雄，納谷英光，横山孝典：組み込み制御システムのためのオブジェクト指向コード生成ツール，情報処理学会論文誌，Vol.46，No.5，pp.1306–1317 (2005).
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.：SimPowerSystems サムプルモデル解説書-自動車用ハイブリッド駆動システムのモデリングとシミュレーション- (2009).
15) The MathWorks, Inc.：ステッピングモータ制御のモデルベース開発 (2009).
16) SDE for Eclipse, http://www.visual-paradigm.com/