情報処理学会論文誌 Vol.53 No.12 2660–2670 (Dec. 2012)

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

田村 雅成[1,a)] 神山 達哉[1,†1] 添田 隆弘[1] 兪 明連[1,b)] 横山 孝典[1,c)]

受付日 2012年3月5日, 採録日 2012年9月10日

概要：モデルベースによる組み込み制御ソフトウェア開発の効率向上のため，Simulink モデルを UML モデルへ自動変換するモデル変換環境を提案する．一般に，組み込み制御ソフトウェア開発は制御設計とソフトウェア設計の 2 段階で行う．制御設計では MATLAB/Simulink を用いて制御ロジックを Simulink モデルとして設計することが多い．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一般的である．本研究では，制御設計からソフトウェア設計への移行をスムーズに行うために，Simulink モデルから UML モデルへの変換ツールを開発した．本ツールが生成する UML モデルはデータとデータの算出処理をカプセル化したクラスからなる．また，変換後の UML モデルの再利用性向上のため，変換元の Simulink モデルに対して階層化を行うが，その階層化作業を効率化するため，階層化支援ツールを開発した．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．

キーワード：組み込みソフトウェア，モデルベース開発，MATLAB/Simulink，UML，モデル変換

A Model Trasformation Environment for Embedded Control Software Design with Simulink Models and UML Models

Masayoshi Tamura[1,a)] Tatsuya Kamiyama[1,†1] Takahiro Soeda[1]
Myungryun Yoo[1,b)] Takanori Yokoyama[1,c)]

Received: March 5, 2012, Accepted: September 10, 2012

Abstract: The paper presents a model transformation environment to transform a Simulink model to a UML model. The embedded control software development process consists of the control logic design phase and the software design phase. MATLAB/Simulink is widely used to build a controller model in the control logic design phase. On the other hand, UML is widely used in the software design phase. To shift from the control logic design phase to the software design phase smoothly, we have developed a model transformation tool to transform a Simulink model to a UML model. The UML model generated by the transformation tool consists of classes that encapsulate data and calculation methods of the data. To improve the reusability of the UML model, the Simulink model should be well-layered. We have also developed a layering support tool for efficient layering of the Simulink model. We have applied the model transformation environment to a number of Simulink models and found it useful for embedded control software design.

Keywords: embedded software, model-based design, MATLAB/Simulink, UML, model transformation

1. はじめに

自動車，家電製品，産業用機器等広い分野で組み込み制御ソフトウェアが用いられ，その開発量は増大している．一般に，組み込み制御ソフトウェアの開発は制御設計とソ

[1] 東京都市大学
Tokyo City University, Setagaya, Tokyo 158–8557, Japan
[†1] 現在，株式会社エー・アンド・デイ
Presently with A&D Company, Limited
[a)] g1181524@tcu.ac.jp
[b)] yoo@cs.tcu.ac.jp
[c)] yokoyama@cs.tcu.ac.jp

フトウェア設計の 2 段階で行う．制御設計では制御設計者が制御ロジックの設計を行う．ソフトウェア設計では制御ロジックに基づいて，ソフトウェア設計者がソフトウェアの構造や振舞いを設計する．

近年，制御設計は MATLAB/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になってきている．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 Simulink モデルを用いてシミュレーションを行うことで，制御ロジックの誤りを早期発見できる．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，MATLAB/Simulink の関連ツールである Real-Time Workshop Embedded Coder 等を用いることで，Simulink モデルからソースコードの自動生成が可能である．

しかし，MATLAB/Simulink のような制御系 CAE/CAD ツールはソフトウェア設計には適していない．Sangiovanni-Vincentelli らは制御系 CAE/CAD ツールをソフトウェア設計に用いることの問題点について述べている [2]．制御系 CAE/CAD ツールでソフトウェア設計を行う場合，構造と振舞いと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，スケジューリングに関連した遅延の分析やバック・アノテーションができない等といった問題がある．したがって，MATLAB/Simulink のような制御系 CAE/CAD ツールはソフトウェア設計には使用せず，制御設計のみに使用するべきである．

ソフトウェア設計には UML のようなソフトウェアモデリング言語が適している．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は適しているが，手続き的な処理の記述には不向きである．一方で，UML モデルは手続き的な処理が適したアプリケーションや通信処理等の設計に適している．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していない．実際の組み込み制御システムはプリエンプ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．しかし，Simulink モデルから自動生成されるソースコードは構造化されておらず，可読性も良くないため，機能の追加・修正は容易ではない．これらの機能を効率良く組み込むためには，ソースコードレベルではなくモデルレベルで設計できることが望ましい．したがって，制御設計で開発した Simulink モデルを 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を考える．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んだりすることが容易となる．そのためには，Simulink モデルをソフトウェア設計に適した形の UML モデルに変換できる開発環境が求められる．これまでに Simulink モデルから UML モデルに自動変換するツールがいくつか提案されているが [3], [4], [5], [6], [7]，それらはいずれも Simulink モデルの要素 1 つ 1 つを UML モデルのクラスに対応させているため，必ずしも再利用性の高い UML モデルにはならない．再利用性を向上させるには，変換後の UML モデルをオブジェクト指向の考え方に基づいた構成にすべきである．

本研究の目的は，Simulink モデルを再利用性の高い UML モデルに変換するモデル変換環境の開発である [8]．そのために，時間駆動オブジェクト指向ソフトウェア開発法 [9] に基づいて，Simulink モデルから UML モデルへの変換ルールを定義する．この手法では制御ロジック中のデータのうち，入力値，出力値，観測値，推定値，目標値，制御パラメータ等の制御上重要なデータ（物理量）をオブジェクトとする．我々はこの変換ルールに基づいた自動変換ツールを開発した．このツールは UML モデルのうち，構造図としてクラス図とオブジェクト図，振舞い図としてシーケンス図を出力する．UML モデル中の各オブジェクト（クラス）は Simulink モデル中の制御上重要なデータに対応付けている．そこで，変換元の Simulink モデルには階層化を行い，制御上重要なデータを算出する処理を 1 つの Subsystem ブロックの下位階層に収める．この Simulink モデルの階層化作業を効率化するために，自動変換ツールに加えて階層化支援ツールを開発した．

以下本論文では，2 章でモデル変換を用いた組み込み制御ソフトウェアの開発工程について述べる．続いて，3 章で開発したモデル変換環境について説明し，変換例を示す．さらに，4 章でモデル変換環境の適用実験について述べ，5 章で関連研究との比較を行い，6 章でまとめを述べる．

2. 組み込み制御ソフトウェアの開発工程

2.1 組み込み制御ソフトウェア開発全体の流れ

我々が提案している組み込み制御ソフトウェアの開発工程を**図 1** に示す．全体の開発工程は制御設計工程（Control Logic Design），ソフトウェア設計工程（Software Design），実装（プログラミング）工程（Programming）からなる．

2.1.1 制御設計工程

制御設計工程では，MATLAB/Simulink を用いて制御ロジック（Control Logic）を Simulink モデル（Simulink Model）として設計する．一般に，対象システムを表現する Simulink モデルは，制御対象を表す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，変換元となる Simulink モデルはコントローラモデルとなる．コントローラモデルはセンサ等からデー

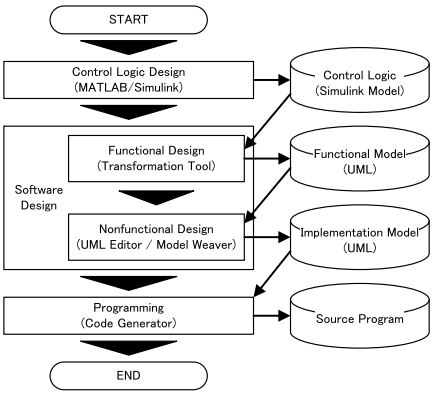

図 1 組み込み制御ソフトウェア開発の流れ
Fig. 1 Development flow of embedded control software.

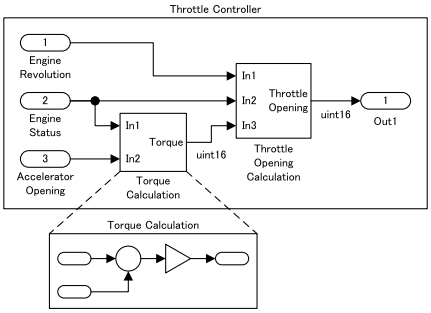

図 2 Simulink モデルの例
Fig. 2 Example Simulink model.

タを得るための入力と，プラントモデルへ制御データを送るための出力を持つ．

変換元となる Simulink モデルは上位階層に制御上重要なデータのみが現れるように階層化し，それらのデータを算出するための処理は Subsystem ブロックの下位階層に記述する．また，入力データと出力データは制御上重要なデータとして必ず上位階層に現れるようにする．したがって，階層化後の Simulink モデルは，上位階層が入力を表す Inport ブロック，出力を表す Outport ブロック，Subsystem ブロックの 3 種類のブロックと制御上重要なデータを表すラインで構成される形となる．なお，原則として Simulink モデルにループが含まれる場合は下位階層に隠蔽し，上位階層にループが現れないようにする．上記のような階層化を行うことで，変換元の Simulink モデルをデータごとに整理された再利用性の高い構造にすることができ，再利用性の高い形で UML モデルを生成することができる．

例として自動車車間距離システムのうち，スロットル制御（Throttle Controller）の Simulink モデルを**図 2** に示す．図 2 の Simulink モデルは入力されたエンジン回転数とエンジン状態，アクセル開度からスロットル開度を算出し出力する処理を表している．この Simulink 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）の入力を表す 3 つの Inport ブロック，トルク（Torque），スロットル開度（Throttle Opening）を算出する 2 つの Subsystem ブロック（Torque Calculation と Throttle Opening Calculation），算出したスロットル開度の出力を表す 1 つの Outport ブロックからなる．図 2 の上半分は階層化された Simulink モデルの上位階層であり，トルク，スロットル開度を算出する具体的な処理は Subsystem ブロックの下位階層に記述されている．この例では初めに，エンジン状態とアクセル開度のデータからトルクを算出する．次に，算出したトルクとエンジン回転数，エンジン状態のデータ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 16 bit 整数）としている．Simulink モデルは制御機能のみを表現し，ソフトウェアとしての実装については考慮されていない．

2.1.2 ソフトウェア設計工程

ソフトウェア設計工程では，制御設計で作成した Simulink モデルをもとに，ソフトウェアの構造と振舞いを UML で設計する．ソフトウェア設計工程は機能設計（Functional Design）と非機能設計（Nonfunctional Design）の 2 段階からなる．機能設計では，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．この段階の UML モデルは制御に関する機能のみを記述した機能モデル（Functional Model）である．機能モデルはソフトウェアとして実装するうえで必要な，タスク間の同期や通信の機構は考慮していない．非機能設計では，本来の機能ではないこのような非機能的な要求を満足するために，UML エディタ（UML Editor）等を用いて機能モデルに同期や通信の機構の追加・修正を行う．本論文では，非機能設計後の UML モデルを実装モデル（Implementation Model）と呼ぶ．機能設計については 2.2 節で，非機能設計については 2.3 節で詳しく述べる．

2.1.3 実装工程

実装工程では，Simulink モデルから Embedded Coder [1] を用いて生成した制御ソースコードと，実装モデルから UML モデリングツールを用いて生成したクラスのスケルトンのソースコードを合成し，最終成果物としてのソースコード（Source Program）を生成する．我々は，コード合成処理を自動化するコード合成ツール（Code Generator）も開発した [10]．Simulink モデルから生成されたデータを算出するためのコードを，それぞれ対応するクラスの update メソッド内に組み込む．組み込む際には，実装モデルのクラス構成に対応するように，他のオブジェクトの値へのアクセスは get メソッドの呼び出しに，また，算出結果を自分の属性値に代入するように書き換える．合成処理をすべてのクラスで行うことで，実装モデルに対応したソースコードを生成できる．

2.2 機能設計

機能設計では，開発したモデル変換ツール（Transformation Tool）を用いてSimulinkモデルをUMLの機能モデルに変換する．モデル変換ツールの詳細は3章で述べる．

我々のモデル変換方法は時間駆動オブジェクト指向ソフトウェア開発法[9]に基づいている．この開発法では，入力値，出力値，観測値，推定値，目標値，制御パラメータ等の制御ロジック上重要なデータ（意味を持つ物理量に対応するデータ）をオブジェクトに対応付ける．そして，それらのデータとその値を算出する処理とをまとめてカプセル化し，1つのクラスとする．これにより，算出方法の詳細を理解していなくてもクラスを再利用することができる．制御上重要な意味を持つデータは，制御ロジックを変更する場合にもその影響を受けにくい[11]．このため，制御ロジックの一部変更に対しては，UMLモデルのクラス構造を変えることなく，クラス内のメソッドの処理を変更することで対応できる．また，新たな物理量を追加する等，制御ロジック上大きな変更を行う場合も，既存の物理量を算出するクラスはそのまま再利用できることが多い．

UMLモデル上では，データを表すオブジェクトをデータ値オブジェクトとして記述する．データ値オブジェクトの基底クラス（ValueObject）を**図3**に示す．ValueObjectはデータ値を記憶する属性value，データ値を算出・更新するupdateメソッド，データ値を読み出すgetメソッドを持つ．データ算出時（updateメソッド実行時）に他のオブジェクトが持つデータが必要となる場合，そのオブジェクトのgetメソッドを呼び出してデータを参照する．また，制御ロジックのブロック線図が表すデータフローをオブジェクト間のデータの参照と見なす．関連consは始端側クラスが終端側クラスのデータを参照することを表す．

図2のSimulinkモデルから変換した機能モデルのクラス図を**図4**に示す．このクラス図はエンジン回転数（Engine Revolution），エンジン状態（Engine Status），アクセル開度（Accelerator Opening），トルク（Torque），スロットル開度（Throttle Opening），スロットル制御（Throttle Controller）の6つのクラスからなる．トルクはエンジン状態とアクセル開度，スロットル開度はトルクとエンジン回転数とエンジン状態のデータをそれぞれ参照する．Inportブロックのデータに対応するオブジェクトはupdateメソッドは持たない．スロットル制御は図2のSimulinkモデル全体を表すクラスである．スロットル制御はトルクとスロットル開度のupdateメソッドを呼び出すexecメソッドを持つ．execメソッドを制御周期ごとに呼び出すことで各データが周期的に更新される．

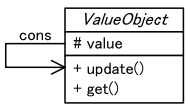

図3 データ値オブジェクトの基底クラス
Fig. 3 Base class of data object.

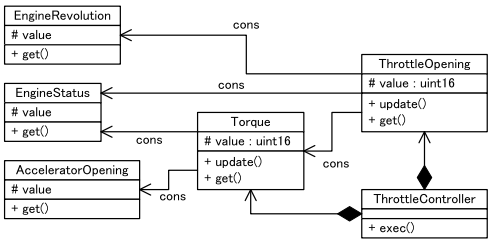

図4 機能モデルの例（クラス図）
Fig. 4 Example class diagram of functional model.

2.3 非機能設計

組み込み制御システムは時間制約を持つハードリアルタイムシステムである．非機能設計では，この時間制約を満たすために，タスク構成やスケジューリング方式，タスク優先度等の設計を行う．また，プリエンプティブなマルチタスク環境での動作を保証するために，タスク間の同期や通信，排他制御等の機構を追加する．これらの非機能的な要求の処理は，アスペクト指向プログラミング[12]によってアスペクトとして記述することで，機能的な要求の処理とは分離させることができる．非機能的な要求のアスペクト化をモデルレベルで行う方法も提案されている[13], [14]．

我々はすでに，アスペクト指向に基づいた非機能設計手法を提案している[15]．この手法では，組み込み制御ソフトウェアの非機能的要求をモデルレベルのアスペクトとしてパターン化し，このアスペクトパターンを我々が開発したモデルウィーバ（Model Weaver）によって機能モデルに織り込む．たとえば，システムの駆動方式（時間駆動かイベント駆動[16], [17]）やタスク間の同期，通信，排他制御の機構等がアスペクトパターンとして定義されている．設計者は必要な機能のアスペクトパターンを選択し，モデルウィーバを用いて機能モデルへ織り込むことで実装モデルを設計する．

例として，図4のシステムにおいてエンジン回転数，エンジン状態，アクセル開度の更新とトルク，スロットル開度の算出が異なる周期のタスクで実行される場合を考える．前者のタスクの方が後者のタスクよりも高優先度の場合，後者のタスクは前者のタスクによってプリエンプションされる可能性がある．このとき，後者のタスクのトルクの算出終了とスロットル開度の算出開始の間でプリエンプションが発生した場合，トルクの算出とスロットル開度の算出の間で使用するエンジン回転数の値が異なり，データの整合がとれなくなる．そのため，データの整合性を保つために排他制御やタスク間通信等の機構が必要となる．

この問題の解決法の1つにエンジン回転数の値をバッファリングする手法がある．後者のタスクの実行開始時にエンジン回転数の値をバッファに記憶しておき，トルクの

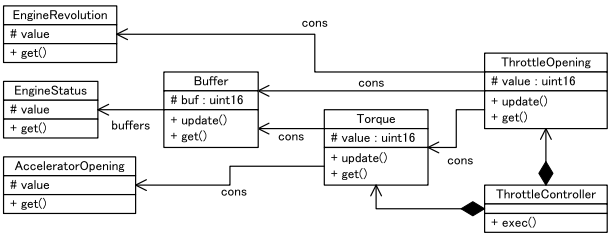

図 5 実装モデルの例（クラス図）
Fig. 5 Example class diagram of implementation model.

算出とスロットル開度の算出で使用するエンジン回転数の値をバッファから読み出すことでデータの整合性を保持できる．図 4 のクラス図にバッファリング機能のアスペクトパターンを織り込むことで，**図 5** のような，バッファ（Buffer）クラスが追加された実装モデルが得られる．

3. モデル変換環境

3.1 階層化支援ツール

UML モデルへの変換は階層化された Simulink モデルの上位階層を利用して行う．2.1.1 項で述べたように，変換元の Simulink モデルは，最上位階層が Inport ブロック，Outport ブロック，Subsystem ブロックの 3 種類のブロックと制御上重要なデータを表すラインのみで構成されるように階層化する．モデルを Subsystem ブロック単位で再利用可能とするため，制御ロジック上重要なデータのみが上位階層に現れるように階層化を行い，それらを計算する処理の詳細は Subsystem ブロックの下位階層に収める．2.2 節で示したように，データ値オブジェクトは Simulink モデル内の制御上重要なデータに，データ値オブジェクトが持つ update メソッドはデータ値を算出する Subsystem ブロックにそれぞれ対応するため，階層化により，再利用性の高いクラスを生成することが可能になる．

なお，Simulink モデルがループを含む場合には下位階層に隠蔽し，上位階層にループが現れないようにする．これは，実装工程では下位階層のモデル単位，すなわちクラス単位でコード生成を行うため，ループを下位階層に隠蔽した方が，クラス単位での部品化再利用に適しているという考えによる．本研究でこれまで対象とした Simulink モデルに存在したループは，下位階層に隠蔽して問題ないローカルなループと，プラントモデルを含んだオーバオールのフィードバックのみであった．後者については，モデル変換ツールが変換対象としているのはコントローラモデルのみであり，プラントモデルを介したループは変換対象には含まれないため，問題はない．

我々は制御ロジック上重要なデータを選択するのみで Simulink モデルの階層化が行える，階層化支援ツールを開発した．本ツールは，Simulink モデルの上位階層に表示する制御上重要なデータを選択することで Simulink モデルの階層化を行う．

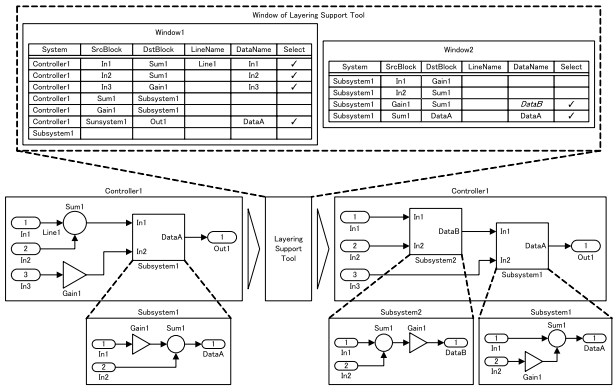

図 6 Simulink モデルの階層化
Fig. 6 Layering of Simulink model.

図 6 に，階層化支援ツール（Layering Support Tool）の入出力およびウィンドウ表示を示す．本ツールは階層化前の Simulink モデルを記述した mdl ファイルを入力し，階層化後の Simulink モデルの mdl ファイルを出力する．

ツールは入力された Simulink モデルを解析し，ツールウィンドウの表の各行に Simulink モデル中のデータを表示する．ツールウィンドウは Simulink モデルの階層別に表示される．また，データは Simulink モデル中の各ラインに対応して表示する．図 6 の例では，左側の階層化された Simulink モデル（上位階層が Controller1，下位階層が Subsystem1）を入力し，図の上部に示した 2 つのウィンドウを表示している（Window1 が Controller1，Window2 が Subsystem1 のデータを表示している）．列 System は，そのデータが Simulink モデル内のどの階層にあるかを表しており，最上位階層のデータであれば Simulink モデル全体が表すシステム名，Subsystem ブロックの下位階層のデータであればその Subsystem ブロック名を表示する．列 SrcBlock はデータの始端側のブロック，列 DstBlock は終端側のブロックを表し，ライン，データが名前を持っている場合は列 LineName にライン名，列 DataName にデータ名を表示する．列 DataName に表示されるデータ名は，Inport ブロックから出ているデータの場合は Inport ブロック名に，Subsystem ブロックから出ているデータの場合はその下位階層の Outport ブロック名に対応する．このような名前の対応付けは Simulink で一般的に用いられている方法である．たとえば図 6 の Window1 の表の 1 行目は，ブロック In1 から出力されてブロック Subsystem1 に入力されるライン Line1 が表す，Controller1 内のデータ In1 を表している．

列 Select はそのデータを重要なデータとするかどうかを選択するためのチェックボックスである．ここにチェックを入れることで上位階層に現れるデータを選択させる．この例ではブロック Subsystem1 から出力されてブロック Out1 に入力される最上位階層のデータ DataA と，ブロック Gain1 から出力されて Sum1 に入力されるブロッ

ク Subsystem1 の下位階層のデータ（名前なし）が制御上重要なデータとして選択されている．なお，最上位階層の Inport ブロックから出力されるデータおよび Outport ブロックに入力されるデータはあらかじめチェックを入れた状態で表示する．これは 2.1.1 項で述べたように，入力データと出力データは制御上重要なデータとして必ず上位階層に現れるようにするためである．また，原則として制御上重要なデータには名前を付けるものとし，選択されたデータが名前を持たない場合は列 DataName に名前を記述させる．この例では後者のデータに DataB という名前を付けている．

ユーザがデータを選択後，階層化支援ツールは階層化処理を行う．まず，入力した Simulink モデルの mdl ファイルから，階層ごとにブロックとデータフローをノードとする木構造の中間データを生成する．次に，木構造の下位階層から順にチェックしていき，ユーザ指定のデータがある場合は，Subsystem ブロックとユーザ指定のデータのみが現れるようなノード構成とし，1 つ上の階層に加える．これを最上位階層まで繰り返していくことで，階層化後の中間データを生成する．最後に，生成した中間データから Simulink モデルの mdl ファイルを生成する．以上により，階層化後の Simulink モデルを得ることができる．最上位階層には選択されたデータのみが現れるようになり，選択されていないデータ（ライン）とブロック群は Subsystem ブロックの下位階層に収められる．

図 6 の例では，右側に示す階層化された Simulink モデル（上位階層が Controller1，下位階層が Subsystem1 と Subsystem2）を出力している．この場合，選択した 2 つのデータ（DataA と DataB）を算出する Subsystem ブロックを最上位階層に持ち，それ以外のデータとブロックを各 Subsystem ブロックの下位階層に収めた Simulink モデルが生成される．

3.2 モデル変換ツール

我々は階層化された Simulink モデルを UML モデルに変換するモデル変換ツールを開発した．このモデル変換ツールは，UML モデルのうち構造図としてクラス図とオブジェクト図，振舞い図としてシーケンス図を生成する．オブジェクト指向に基づくソフトウェア設計において，クラス図は最も重要な静的構造図である．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図である．組み込み制御システムではオブジェクトを動的には生成せず，固定的なオブジェクト構成とすることが多い．そのため，オブジェクト図によって定常的なオブジェクト構成を表すことができる．シーケンス図はオブジェクト間のメソッド呼び出しを時系列的に表現する図である．

モデル変換ツールの内部処理を**図 7** に示す．本ツールは

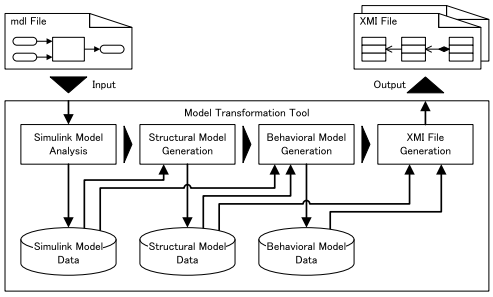

図 7 モデル変換ツール
Fig. 7 Model transformation tool.

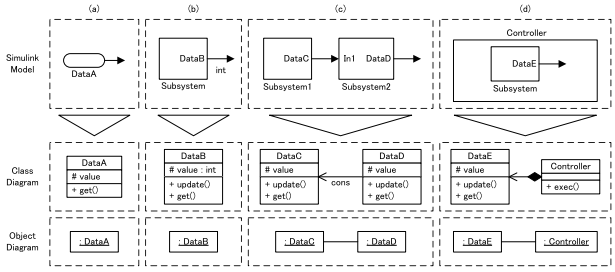

図 8 構造図への変換ルール
Fig. 8 Transformation rules for structural model.

Simulink モデルのファイル保存形式である mdl ファイルを入力とする．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 モデルデータをもとに，構造図（クラス図とオブジェクト図）のデータを生成する．さらに生成した構造図データと Simulink モデルデータをもとに，振舞い図（シーケンス図）データを生成する．最後に，構造モデルデータと振舞いモデルデータを XMI ファイルに変換して出力する．XMI ファイルは UML モデルの標準的なファイル保存形式である [18]．構造図の生成法は 3.2.1 項で，振舞い図の生成法は 3.2.2 項で詳しく述べる．

3.2.1 構造図の生成

階層化された Simulink モデルの各要素から構造図（クラス図とオブジェクト図）の要素への変換ルールを**図 8** に示す．

図 8 の (a) は Inport ブロックからクラス図のクラスおよびオブジェクト図のオブジェクトへの変換ルールである．クラスは Inport ブロックが表すデータに対応付けて生成する．変換後のクラスはクラス名をデータ名（この例では DataA）とし，属性 value とメソッド get を持つ．オブジェクトも同様にデータに対応付けて生成する．オブジェクトは対応するクラスのインスタンスとなる．

図 8 の (b) は Subsystem ブロックからクラス図のクラスおよびオブジェクト図のオブジェクトへの変換ルールである．クラスは Subsystem ブロックが出力するデータに対応付けて生成する．変換後のクラスはクラス名をデータ名（この例では DataB）とし，属性 value，メソッド update，

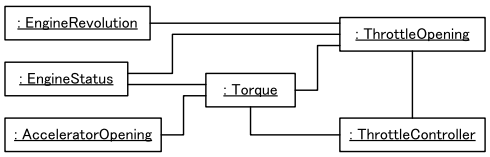

図 9 機能モデルの例（オブジェクト図）
Fig. 9 Example object diagram of functional model.

メソッド get を持つ．また，Simulink モデル内でデータの型が定義されている場合（この例では int）には，属性 value の型を同様に定義する．オブジェクトも同様にデータに対応付けて生成する．

図 8 の (c) は Subsystem ブロック間（この例では Subsystem1 と Subsystem2 間）をつなぐラインから，クラス図のクラス間（この例では DataC と DataD 間）の関連およびオブジェクト図のオブジェクト間のリンクに変換するルールである．この変換ルールはブロックが Inport ブロックの場合も同様である．クラス図には，関連 cons がラインに対応付けて生成される．同様にオブジェクト図にも，リンクがラインに対応付けて生成される．リンクは対応する関連のインスタンスとなる．

図 8 の (d) は Simulink モデルから Simulink モデル全体を表すオブジェクト（全体オブジェクト）とコンポジションへの変換ルールである．全体オブジェクトは Simulink モデル全体が表すシステム名（この例では Controller）をクラス名に持ち，exec メソッドを持つ．コンポジションとは全体部分関係を意味する関連である．コンポジションは全体オブジェクトと Subsystem ブロックに対応するデータ値オブジェクト（この例では DataE）間をつなぐ形で生成される．このコンポジションはデータ値オブジェクトが全体オブジェクトの一部分であることを表している．オブジェクト図についても同様に，全体オブジェクトのインスタンスとリンクを生成する．

以上のルールに基づいて図 2 の Simulink モデルから変換したクラス図は図 4 となる．同じ Simulink モデルから変換したオブジェクト図を**図 9** に示す．このオブジェクト図はエンジン回転数（Engine Revolution），エンジン状態（Engine Status），アクセル開度（Accelerator Opening），トルク（Torque），スロットル開度（ThrottleOpening），スロットル制御（Throttle Controller）の 6 つのオブジェクトからなる．各オブジェクトは図 9 のクラス図の各クラスに 1 対 1 に対応している．また，オブジェクト間をつなぐリンクは図 9 のクラス間の関連のインスタンスであり，各関連と 1 対 1 に対応している．

3.2.2 振舞い図の生成

階層化された Simulink モデルの各要素から振舞い図（シーケンス図）の要素への変換ルールを**図 10** に示す．シーケンス図の各要素は，Simulink モデルの情報だけでなく，クラス図やオブジェクト図の情報もあわせて参照して

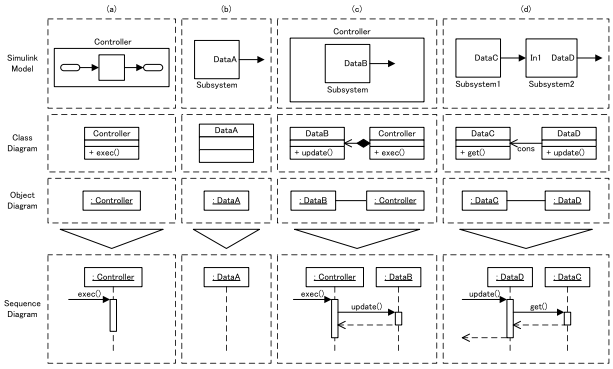

図 10 振舞い図への変換ルール
Fig. 10 Transformation rules for behavioral model.

生成される．変換後のシーケンス図は，全体オブジェクトの exec メソッド実行時の処理を表している．

図 10 の (a) は全体オブジェクト（この例では Controller）の生存線と exec メソッドの呼び出しメッセージへの変換ルールである．変換後のシーケンス図が表す処理は，全体オブジェクトの exec メソッドの呼び出しから開始される．

図 10 の (b) はデータ値オブジェクト（この例では DataA）の生存線への変換ルールである．生存線は各オブジェクトに 1 対 1 で対応するように生成する．

図 10 の (c) は update メソッドの呼び出しメッセージとリターンメッセージの生成ルールである．update メソッドの呼び出しメッセージは Simulink モデルの Subsystem ブロックに対応している．全体オブジェクト（この例では Controller）の exec メソッドがデータ値オブジェクト（この例では DataB）の update メソッドを呼び出すことで，データの算出・更新が行われる．また，update メソッドの呼び出しメッセージと対になる形でリターンメッセージを生成する．

図 10 の (d) は get メソッドの呼び出しメッセージとリターンメッセージの生成ルールである．このルールは，ラインの始端側クラス（この例では DataC）が Inport ブロックの場合も同様である．Simulink モデルのラインが表すデータフローおよびクラス図上の関連 cons に対応して，get メソッドの呼び出しを行う．オブジェクトが他のオブジェクトのデータを参照（この例では DataD が DataC のデータを参照）する場合，前者の update メソッドが後者の get メソッドを呼び出してデータを参照する．前者は参照したデータを用いて update メソッドの処理を行い，自身のデータを算出し，更新する．また，get メソッドの呼び出しメッセージと対になる形でリターンメッセージを生成する．

Subsystem ブロックに対応するすべてのオブジェクトの update メソッドを，Simulink モデルのデータフローに従って呼び出すことで，制御ロジック上のすべてのデータの値が更新される．全体オブジェクトの exec メソッドがデー

タ値オブジェクトの update メソッドを呼び出す順番の決定ルールを**図 11** に示す．

図 11 の (a), (b) は Simulink モデルのデータフローから，update メソッドの呼び出し順を一意に特定できる場合の変換ルールである．Simulink モデルのデータフロー上でより上流にあるオブジェクトから順に逐次的に update メソッドを呼び出す．(a) の場合は，DataF の算出に DataE の値を使用するので DataE を先に算出しなければならない．そのため，DataF の update メソッドよりも DataE の update メソッドを先に呼び出す．(b) の場合は，DataI の算出に DataH，DataH の算出に DataG の値を使用するので，DataI よりも DataH，DataH よりも DataG を先に算出しなければならない．そのため，DataG，DataH，DataI の順に update メソッドを呼び出す．

図 11 の (c)，(d) は Simulink モデルのデータフローからは，呼び出し順を一意に特定できない場合の変換ルールである．ある複数のブロックの呼び出し順がデータフローから特定できない場合，並行に呼び出すものとしてシーケンス図を生成する．具体的には，該当する update メソッドの処理を並行処理を表す par のフラグメントで囲う．(c) の場合は，DataJ と DataK のどちらを先に算出しても DataL の算出には影響しないため，DataJ と DataK の update メソッドの呼び出しを並行処理するものとしてシーケンス図を生成する．(d) の場合は DataM と DataN のどちらを先に算出しても DataO の算出には影響しないため，DataM と DataN の update の呼び出しを並行処理する．さらに，DataM，DataN の算出から DataO の算出までの処理と DataP の算出の処理はどちらを先に行っても DataQ の算出には影響しないため，DataM，DataN の算出から DataO の算出までの処理と DataP の算出の処理を並行処理する．

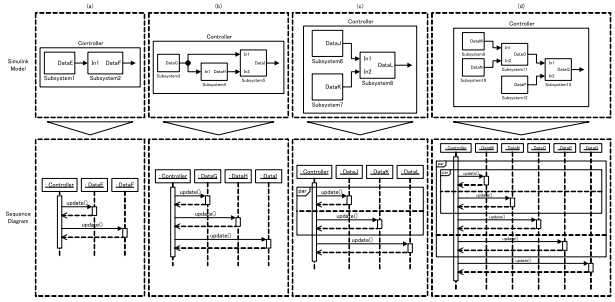

図 11 update メソッドの呼び出し順の決定ルール
Fig. 11 Transformation rules for update method calling sequence.

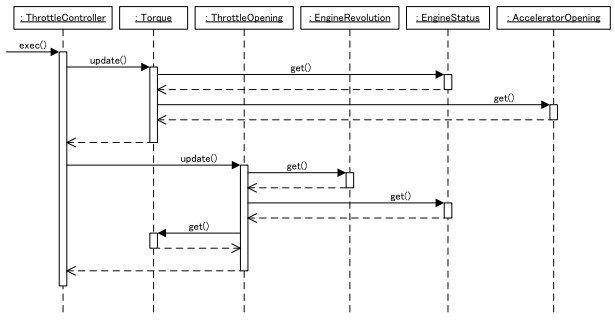

図 12 機能モデルの例（シーケンス図）
Fig. 12 Example sequence diagram of functional model.

図 2 の Simulink モデルから変換したシーケンス図を**図 12** に示す．このシーケンス図はスロットル制御（ThrottleController）の exec メソッドの処理を表している．スロットル制御の exec メソッドは，図 2 の Simulink モデルのデータフローに従い，トルク（Torque），スロットル開度（ThrottleOpening）の順に，update メソッドを呼び出している．また，トルクの update メソッドはエンジン状態（EngineStatus）とアクセル開度（AcceleratorOpening）の get メソッドを，スロットル開度の update メソッドはエンジン回転数（EngineRevolution），エンジン状態，トルクの get メソッドを呼び出してデータを参照している．この exec メソッドを制御周期に基づいて周期的に実行することで，制御処理が実現される．

4. 評価

開発したモデル変換環境の適用実験を行って，その有用性を評価する．適用実験に用いる Simulink モデルは，制御設計者が実装を考慮せずに作成したものである必要がある．そこで，MathWorks 社 [1] が提供している燃料噴射システム，ハイブリッド駆動システム，ステッピングモータ制御システムの Simulink モデルに対して適用実験を行った．最初に，それらの Simulink モデルを階層化支援ツールを用いて階層化した．適用実験に用いた各 Simulink モデルの階層化後のブロック数を**表 1** に示す．続いてモデル変換ツールを用いて，階層化した Simulink モデルをクラス図，オブジェクト図，シーケンス図に変換した．

実験を行ったモデルのうち，ハイブリッド駆動システムの例を紹介する．本ハイブリッド駆動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ドシステムである．階層化したハイブリッド駆動システムの Simulink モデルの最上位階層を**図 13** に示す．また，変換後のクラス図を**図 14** に，オブジェクト

表 1 適用実験に使用した Simulink モデル
Table 1 Models Used in Experiments

対象システム	ブロック数		
	Subsystem ブロック	Inport ブロック	Outport ブロック
燃料噴射システム	15	4	1
ハイブリッド駆動システム	30	6	5
ステッピングモータ制御システム	8	1	4

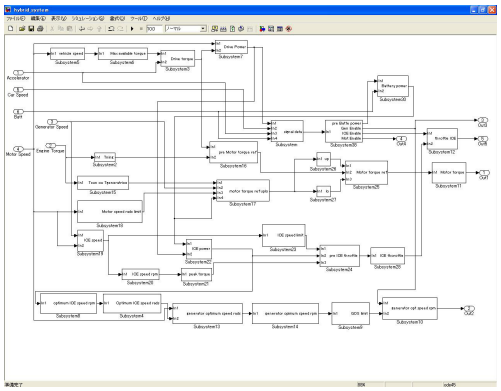

図 13 ハイブリッド駆動システムの Simulink モデル
Fig. 13 Simlink model of hybrid electric vehicle system.

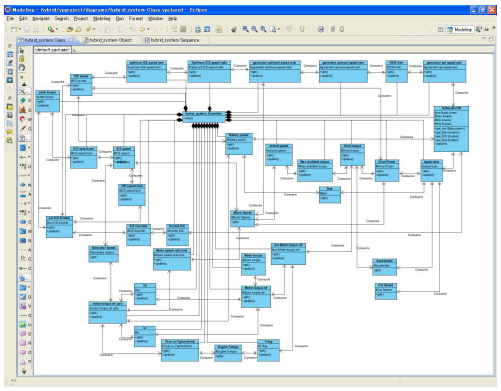

図 14 ハイブリッド駆動システムのクラス図
Fig. 14 Class diagram of hybrid electric vehicle system.

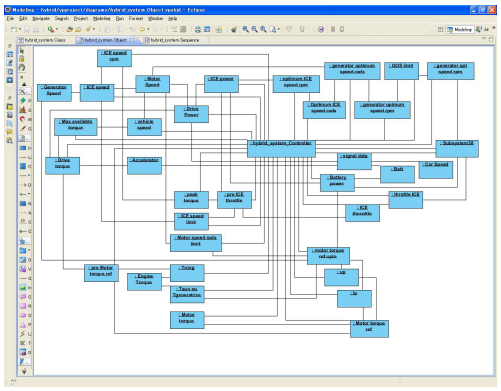

図 15 ハイブリッド駆動システムのオブジェクト図
Fig. 15 Object diagram of hybrid electric vehicle system.

図を図 15 に，シーケンス図を図 16 に示す．

以上のように，ソフトウェアとしての実装を考慮していない純粋な制御ロジックを記述した Simulink モデルを階層化し，階層化した Simulink モデルからソフトウェアの構造と振舞いを表す UML モデルに変換することができた．これらの結果から，本モデル変換環境は実際の組み込み制御ソフトウェア開発に適用可能と考える．

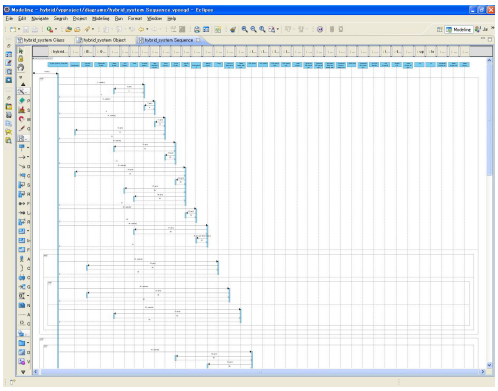

図 16 ハイブリッド駆動システムのシーケンス図
Fig. 16 Sequence diagram of hybrid electric vehicle system.

5. 関連研究との比較

これまでに Simulink モデルから UML モデルに自動変換するツールがいくつか提案されている．Ramos-Hernandez らは Simulink モデルを UML モデルに自動変換するツールを提案している [3], [4]．このツールは，Simulink モデルのブロック 1 つ 1 つをインタフェースクラスに変換する．Müller-Glaser らは，Simulink モデルの各要素をオブジェクトとして UML モデルに変換する自動変換ツールを提案している [5], [6]．この手法では，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．Sjöstedt らもまた，Simulink モデルを UML モデルに自動変換するツールを提案している [7]．このツールは，構造を表す UML モデル（以下，構造図）の 1 つである複合構造図と振舞いを表す UML モデル（振舞い図）の 1 つであるアクティビティ図を生成する．

これらの手法では，Simulink モデルのブロック 1 つ 1 つを UML モデルのクラスとしている．そのため，いくつかのクラスを組み合わせなければ制御に必要なデータを算出できない．よって，クラスを再利用する場合でも，Simulink モデルを用いて制御設計を行うのと同様の設計作業が必要となり，それをソフトウェア設計者が行うのは難しい．したがって，これらの手法では UML モデルの部品化再利用に適しているとはいえない．

これに対して本手法では，2.2 節で述べているように，制御的に重要な意味を持つデータ（物理量）をオブジェクトととらえ，データとその値を算出する処理とをまとめてカプセル化し，1 つのクラスとしている．そのため，制御ロジックの詳細を理解していなくてもクラスを再利用することができる．また，制御ロジックの変更やデータの追加を行う場合も，UML モデルのクラス構造や他のクラスへの影響が少なくて済む．以上のように，本研究の変換方法は他の関連研究に比べ再利用性に優れていると考えられる．

6. まとめ

組み込み制御ソフトウェア開発向けのモデル変換環境として，Simulink モデルの階層化支援ツールおよび Simulink モデルから UML モデルへのモデル変換ツールを開発した．モデル変換ツールは階層化された Simulink モデルからクラス図，オブジェクト図，シーケンス図の 3 種類の UML モデルを生成する．そして，いくつかの Simulink モデルに対して適用実験を行い，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．

今後は，より広範囲の制御モデルに対応できるようにモデル変換環境を拡張し，組み込み制御ソフトウェア開発のさらなる効率化を実現したいと考えている．具体的には，上位階層にループが現れる Simulink モデルや，状態遷移を扱う Stateflow ブロックを含む Simulink モデルへの対応を検討している．後者については，振舞いモデルとしてステートマシン図の生成を計画している．

参考文献

[1] The Math Works Inc., available from ⟨http://www.mathworks.com/⟩.

[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. and Müller-Glaser, K.D.: 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] 神山達哉，添田隆弘，兪　明連，横山孝典：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，情報処理学会研究報告，Vol.2010-EMB-18, No.7, pp.1–7 (2010).

[9] 横山孝典，納谷英光，成沢文雄，倉垣　智，永浦　歩，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.34, No.2, pp.338–349 (2003).

[10] 神山達哉，兪　明連，横山孝典：モデル変換とコード生成機能を有する組み込み制御ソフトウェア開発支援ツール，情報処理学会第 74 回全国大会講演論文集，1L-5 (2012).

[11] 吉村健太郎，宮崎泰三，横山孝典：オブジェクト指向組み込み制御システムのモデルベース開発法，情報処理学会論文誌，Vol.46, No.6, pp.1436–1446 (2005).

[12] Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C. Loingtier, J.M. and Irwin, J.: Aspect-Oriented Programming, *Proc. 11th European Conference on Object-Oriented Programming*, pp.220–242 (1997).

[13] Wehrmeister, M.A., Freitas, E., Pereira, C.E. and Wagner, F.R.: An Aspect-Oriented Approach for Dealing with Non-Functional Requirements in a Model-Driven Development of Distributed Embedded Real-Time Systems, *Proc. 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing*, pp.428–432 (2007).

[14] Driver, C., Reilly, S., Linehan, E., Cahill, V. and Clarke, S.: Managing Embedded Systems with Aspect-Oriented Model-Driven Engineering, *ACM Trans. Embedded Computing Systems*, Vol.10, No.2, pp.21:1–26 (2010).

[15] Soeda, T., Yanagidate, Y. and Yokoyama T.: Embedded Control Software Design with Aspect Patterns, *Journal of the Chinese Institute of Engineers*, Vol.34, No.2, pp.213–225 (2011).

[16] Kopetz, H.: Should Responsive Systems be Event-Triggered or Time-Triggered?, *IEICE Trans. Information & Systems*, Vol.E76-D, No.11, pp.1325–1332 (1993).

[17] 横山孝典：組み込み制御ソフトウェアのアスペクト指向に基づく設計法，情報処理学会論文誌，Vol.47, No.4, pp.1185–1194 (2006).

[18] Object Management Group: XML Metadata Interchange Specification, Version 2.0.1 (2005).

田村 雅成 （学生会員）

1988 年生．2011 年武蔵工業大学知識工学部情報科学科卒業．同年東京都市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．

神山 達哉 （正会員）

1987 年生．2010 年武蔵工業大学工学部コンピュータ・メディア工学科卒業．2012 年東京都市大学大学院工学研究科情報工学専攻修士課程修了．同年株式会社エー・アンド・デイ入社．

添田 隆弘

1986 年生．2009 年武蔵工業大学工学部コンピュータ・メディア工学科卒業．2012 年東京都市大学大学院工学研究科情報工学専攻修士課程修了．

兪 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．