制御配置モデルを用いたモデルベース開発に関する研究

M2015SE006 宮浦孝広
指導教員：阿草清滋

1 はじめに

組込みシステム開発においてモデルベース開発が注目を集めている．モデルベース開発は開発の上流工程で，仕様書として実行可能なモデルを作成する開発方法である．実行可能なモデルを作成することで，モデルを用いたシミュレーションを行い，単体テストからシステムレベルまでのテストが可能となる．また，シミュレーションによるテストを行うことで，システムの仕様を早期に決めることができ，システムを組み込む機械をシステムと同時に開発することが可能となり，開発期間を短縮できる．

モデルベース開発ではモデルの作成に，MATLAB/Simulink がよく用いられる．MATLAB/Simulink によるモデルの作成では，条件判断や繰返し処理の記述を行う場合，複数の処理から任意の処理を選択する必要がある．任意の処理の選択を行うために入力などの情報が必要となるが，これらの記述を行うと信号線数が多くなってしまいモデルが複雑化してしまう．UML モデルは条件判断や繰り返し処理のモデル化が得意なので，Simulink モデルと UML モデルによるモデル化を考える．しかし，Simulink モデルと UML モデルは異なったモデル化を行っており，2 つのモデルを用いた開発を行う場合，それぞれのモデル化をどのレベルまで行い，どのように結合するかが問題となる．

本研究では，制御配置モデルを用いたモデルの記述方式の提案を行う．制御配置モデルは UML モデルを Simulink モデルへの記述に対応させるモデルであり，このモデルによってモデルベース開発における組込みシステム開発の開発効率を向上させる．本研究におけるモデル化の手順は，UML モデルによるモデル化を行い，システムを詳細化していく．システムを詳細化していく中で，数学モデルで記述可能な処理や制御対象を制御している処理が出てきた場合には，Simulnk モデルを用いたモデル化を行う．詳細化していく中で出てきた処理を Simulink モデルで記述することで，Simulink モデルと UML モデルを関連付ける．詳細化が終わった後に，作成した Simulink モデルと UML モデルから制御配置モデルを作成し，システム全体をモデル化した Simulink モデルへ変換する．

2 既存技術

2.1 モデルベース開発

UML モデルによるモデルベース開発は，システム開発の上流工程において，様々な視点からのシステムのモデル化を行う開発である．様々な視点からのモデル化を行うことで，システムの理解を促すことができる．

Simulink モデルによるモデルベース開発は，システムの上流工程において，シミュレーションが可能なモデルを作成する開発である．シミュレーションによるテストを行うことでシステムへの修正を早期に行うことができる．

2.2 Simulink モデル

MATLAB/Simulink を用いて作成される Simulink モデルは制御対象を制御する数学モデルを表したコントローラ・モデル (図 1 の A) と制御対象を表したプラント・モデル (図 1 の B) から構成されている．

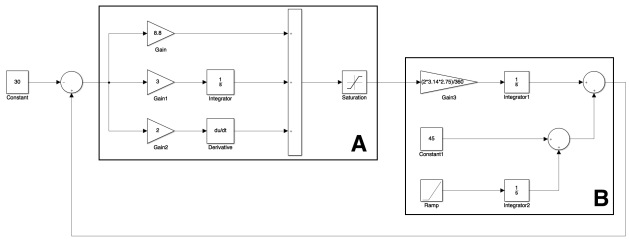


図 1 Simlulink モデル

条件によって制御を変更する場合には，コントローラ・モデル内に複数の制御を記述し，複数の制御中から条件に合う制御を判断する．この場合，判断を行うための入力が必要となり，この記述を行うと信号線数が多くなり，Simulink モデルが複雑化してしまう．

2.3 UML モデル

本研究ではユースケース図，状態遷移図，アクティビティ図の 3 つの UML モデルを用いる．ユースケース図はシステムに必要な機能や制御対象を明らかにすることができることからモデル化の最初に使用する．状態遷移図は状態からの遷移が可能な状態や遷移条件を明らかにすることができる．また，アクティビティ図は実行順序を知ることができ，本研究ではこれらの情報を制御配置モデルの作成に用いるので，モデル化の際に状態遷移図とアクティビティ図を用いる．

3 制御配置モデルを用いたモデル化

3.1 制御配置モデル

このモデルはシステムのモデル化の際に作成した UML モデルを Simulink モデルの記述に対応させる中間的な記述であり，作成した Simulink モデル間の制御の流れを表している．このモデルを制御配置モデルと呼ぶ．制御配置モデルの構成要素と使用方法を述べる．

- 制御
  角丸長方形で描かれる．Simulink モデルで記述されたシステムの処理を表す．
- 分岐点
  ひし形で描かれる．状態遷移図やアクティビティ図の分岐箇所を表す．
- 条件
  矢印で描かれる．矢印の接続先への遷移を表す．矢

印上部の遷移条件に対して真だった場合に遷移する．

- グループ
  太線四角形で描かれる．ある制御から遷移可能な制御をまとめたもの．区別をするので名前をつける．制御をまとめることをグループ化という．
- 分岐によるグループ
  四角形で描かれる．グループ内にある分岐点によって分岐している制御をまとめる．
- フロー
  横棒線で描かれる．制御後に遷移するグループ名を明らかにしたもの．

3.2 モデル化の手順

本研究におけるモデル化の手順を示す．

1. Simulnk と UML モデルによるモデル化
   (a) ユースケース図の作成
   (b) モデルの詳細化
2. 制御配置モデルの作成
   (a) 制御のグループ化
   (b) 条件分岐のまとめ
   (c) フローの追加
3. Simulink モデルへの変換

3.3 Simulink と UML モデルによるモデル化

3.3.1 ユースケース図の作成

ユースケース図の作成を行うことで，システムに必要な機能や制御対象を明らかにする．制御対象には機能によって実際に制御される制御対象と，機能中で用いる情報を取得するために制御される制御対象がある．機能によって実際に制御される制御対象をプラントアクタと呼び，プラントアクタには目印をつける．

3.3.2 モデルの詳細化

システムに必要な機能を状態遷移図とアクティビティ図を用いて詳細化を行っていく．そして，以下のような処理が記述された場合には Simulink モデルによるモデル化を行う．

- 数学モデルで記述が可能な処理
- プラントアクタを制御する処理

処理の詳細化が足りていなかった場合は，モデルの詳細化を続ける．Simulink モデルによるモデル化が行えた場合は，関連する UML モデルの部分に印をつける．印をつけた状態を数式状態，印をつけたアクションを数式アクションと呼ぶ．モデルの詳細化はプラントアクタを制御する処理が Simulink モデルで記述されるまで行う．

3.4 制御配置モデルの作成

3.4.1 制御のグループ化

1. システム開始時の処理の抜き出し
   状態遷移図とそれを詳細化したアクティビティ図から制御の流れをたどっていき，以下の条件に当てはまる処理をすべて抜き出す．抜き出した処理を制御配置モデルの「制御」として記述する．
   - システム開始後，最初に実行される可能性のある数式状態または数式アクション
2. 制御の判別
   抜き出して記述した「制御」にプラントアクタを制御していない処理があった場合には，その「制御」からの制御の流れを UML モデルを用いてたどっていき，以下の条件に当てはまる処理をすべて抜き出す．プラントアクタを制御していない処理が複数あった場合には 1 つずつの処理に対して作業を行う．
   - 「制御」後に実行される可能性のある数式状態または数式アクション

   抜き出した処理は，抜き出した処理の前に実行される「制御」の後ろに「制御」として記述し,「制御」と「制御」の間を制御の流れの順に「条件」で結ぶ．また，次に実行される可能性のある「制御」を実行するのに必要な条件があった場合には,「遷移条件」を「条件」上に記述する．
3. グループ化
   抜き出して記述した「制御」すべてが，プラントアクタを制御する処理となるまで「制御の判別」の作業を行い，作業が終了した場合には抜き出して記述した「制御」を制御配置モデルの「グループ」としてまとめる．
4. グループ内の制御からのグループ化
   「グループ化」した「グループ」内にある，プラントアクタを制御している「制御」からの遷移を確認し，以下の条件に当てはまる処理をすべて抜き出し,「制御」として記述を行う．プラントアクタを制御している「制御」が複数あった場合には 1 つずつの「制御」に対して作業を行っていく．
   - 「制御」後に実行される可能性のある数式状態または数式アクション

   抜き出して記述した「制御」に対して「制御の判別」と「グループ化」の作業を行うことで新しい「グループ」が作成されるが，この「グループ」がすでに作成された「グループ」と同じものであった場合にはこの「グループ」を破棄する．
5. 終了条件
   新しい「グループ」が作成されなくなるまで「グループ内の制御からのグループ化」を繰り返す．作成されなくなった場合には,「制御のグループ化」を終える．

3.4.2 条件分岐のまとめ

状態遷移図とアクティビティ図をたどっていき，制御配置モデルの「制御」が実行される条件を抜き出す．そして，それらの情報を「分岐点」,「条件」,「遷移条件」を用いて制御配置モデルへ記述していく．

「遷移条件」が同じものや途中まで「遷移条件」が同じものがないかを「グループ」ごとに確認していき，それらがあった場合には,「分岐によるグループ」を用いて「グループ化」を行う．または,「分岐点」や「条件」を用いて記述を追加する．まとめるものがなくなった場合には「条件分岐のまとめ」を終える．

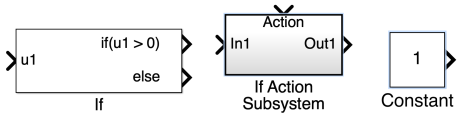
3.4.3 フローの追加

「制御」に制御配置モデルの「フロー」を記述し,「制御」の実行後に遷移する「グループ名」を記述する.「グループ」内の「制御」からの遷移先が同じ場合には,「グループ」に「フロー」を記述する．

3.5 Simulink モデルへの変換

作成した制御配置モデルを Simulink モデルへと変換する．システムを Simulink モデルと UML モデルを用いてモデル化を行い，作成したモデルを用いて制御配置モデルを作成することで，UML モデルを Simulink モデルでの記述に対応させた．この制御配置モデルを Simulink モデルへと変換することで，システム全体をモデル化した Simulink モデルに変換することができる．制御配置モデルの構成要素の扱いとそれに対応する Simulink モデルのブロックを示す.

- 制御は図 2 の (b) に変換し，ブロック中に制御の Simulink モデルを記述
- グループ・分岐によるグループは図 2 の (b) に変換
- 分岐点は図 2 の (a) に変換
- 条件は図 2 の (a) の設定に記述
- フローは図 2 の (c) を使って記述



(a) if Action Subsystem　(b) if Block　(c) Constant

図 2 Simulink モデルのブロック

4 評価・考察

4.1 先行車を追越すシステムの開発による検証

教育版 LEGO mindstorms EV3 による，先行車を追越すシステムの開発を行う．制御配置モデルを用いた記述方式によってシステム全体をモデル化した Simulink モデルが作成可能であるかを検証する．

4.1.1 概要

2 本の平行で黒色のラインがあり，左側のライン上に LEGO を 2 台縦に並べて走らせる．後続の LEGO と先行車との距離が一定の距離になると後続の LEGO が右側のラインへ移動する．そして，右に移動した LEGO が左側の LEGO を追い越したと判断すると，左側のラインへ戻って来るシステムを開発する．

4.1.2 実験に用いた LEGO

実験に用いた LEGO では，サーボモータ ×2，超音波センサ ×2，カラーセンサ ×1 以上のハードウェアを使用した．

4.1.3 システムの開発

はじめに，ユースケース図によるモデル化を行い，その後，状態遷移図，アクティビティ図，Simulink モデルを用いてモデルを詳細化した．次に，作成したモデルを用いて制御配置モデルを作成した．図 3 は制御配置モデルを一部抜粋したものである．

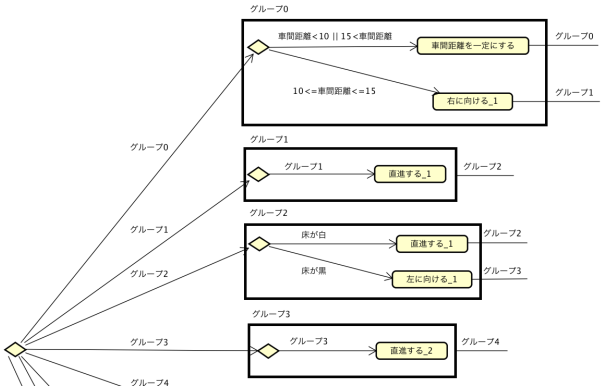


図 3 制御配置モデル

最後に，作成した制御配置モデルを Simulink モデルに変換する．変換されたシステム全体をモデル化した Simulink モデルが図 4 である．

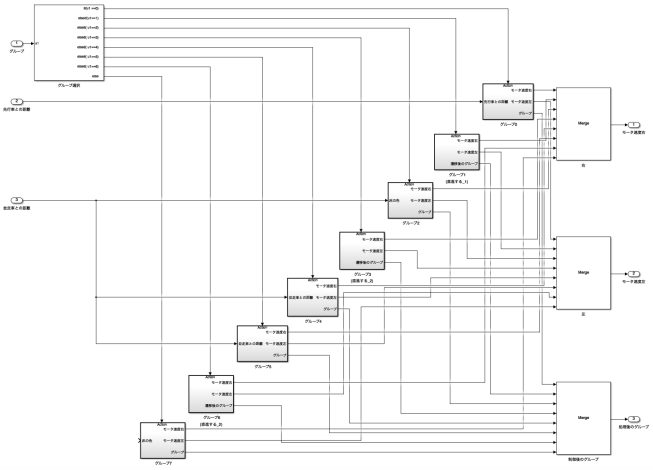


図 4 Simulink モデル

4.2 考察

4.2.1 アプリケーションへの適応

プラントアクタが複数存在し，そのプラントアクタを同時に操作する場合を考える．この場合，Simulink モデルの記述方法やアクティビティ図においてフォークノードによる記述が追加されるなど，記述が変化することが考えられるので，これらの記述に対応した方法を提案す

る必要があると考えられる．動作する手順や条件のあるシステムに対しては提案する手法によって実現が可能であるが，プラント・モデルの作成が行えないものや難しいものに対しては適応できないものが存在する．

4.2.2 処理の重複

今回作成した Simulink モデルは 4 個であったが，図 4 内には 15 個の処理が存在している．今回開発したシステムにおいて，同じモデルを用いた入出力の異なる処理が多く含まれていたことから，処理数が多くなった．このような処理が多いと，修正が必要な際に，関連する処理すべてに対して修正を行う必要がある．処理数の増加を防ぐ方法として，同じモデルを用いた入出力の異なる処理を 1 つの処理としてまとめる方法がある．しかし，条件による入力の判断や出力の判断を行う必要があり，この場合はモデルが複雑化してしまう．それぞれの問題における問題を検討し，それに応じて制御配置モデルの作成手順を検討する必要がある．

4.2.3 処理の発見

今回のシステムでは，手続き的な処理が多かったことで，グループ数が多くなってしまった．そして，グループ数が多くなってしまうとモデルが大規模化してしまうという問題がある．しかし，制御配置モデルは処理がどこのグループ内に記述されているかを確認することが可能であり，目的の処理を簡単に探すことができる．このことから，モデルが大規模化してしまっても問題がないと考えている．

4.3 自動化

現在はすべての作業を手作業で行っているので，エラーが発生してしまうことがある．制御配置モデルの作成には，制御後に遷移するグループ，制御が実行される条件，制御の遷移するグループの情報が必要であり，これらの情報を状態遷移図やアクティビティ図から抜き出し，記述を行ったテキストを用意することで，制御配置モデルの作成を自動化することができると考えられる．そして，自動化の結果として開発効率の向上やエラーの発生を下げることができる．

5 関連研究

Simulink モデルの複雑化にともない，Simulink モデルの保守性に関する研究がされている．小林等 [1] は Simulink モデルのブロックや配置が企業によって異なっていることが，モデルの保守性を低下させているとし，ブロックの使用方法やサブシステムの構造設計についての特性を数値化することで，設計者に依存しない統一的なモデルの作成を支援しようとした．

UML と Simulink モデルに関する研究としてモデルを変換する研究がなされている．田村等 [2] は Simulink モデルで記述できない処理を UML を用いて記述を行い，モデル設計の時に組み合わせることで，組込みシステム開発の効率化を目指している．しかし，Simulink モデルの作成時における，記述の問題について触れていない．本研究では Simulink モデルの作成時の問題を UML モデルを用いることで解決する．

小澤等 [3] は Simulink モデルが再利用性の検討・修正が困難であることから，再利用性の検討・修正が行いやすい UML モデルへ変換し，修正後 Simulink モデルへ戻すといったモデルの相互変換について研究していた．この研究では UML モデルを Simulink モデルの保守性を向上させる目的で使用している．本研究では，システム開発を効率化させる目的で UML を使用しており，UML の使用目的が異なる．

Peng 等 [4] は設計モデルをシミュレーションモデルへ変換することで，その設計モデルの正当性を示すことを目的とし，UML モデルから Simulink モデルへの変換を考察した．Simulink モデルを用いた正当性の確認や UML モデルを Simulink モデルへ変換する部分など，似ている部分が多い．しかし，Peng 等は状態遷移図のみの変換を行っている．また，アーキテクチャに注目をして変換を行うことを考えている部分が，本研究では制御間の流れに着目して変換を行っているので異なっている．

6 おわりに

本研究では，MATLAB/Simulink を用いた組込みシステム開発における Simulink モデル作成の際の問題を，制御配置モデルを用いることで解決する提案を行った．また，制御配置モデルを用いたモデルの記述方式を示した．

今後の課題として，制御配置モデルの作成や制御配置モデルから Simulink モデルを作成する工程における自動化の検討，重複した処理の扱いかたの検討，そして，プラントアクタが複数存在した場合の制御配置モデルの作成手順の見直しを行う必要がある．

参考文献

[1] 小林孝壽, 小林隆志, 久保孝行, “ブロック利用傾向に基づく Simulink モデルの設計スタイル抽出手法,” 電子情報通信学会技術研究報告. SS, ソフトウェアサイエンス, Vol. 114, No. 23, pp. 43–48, 2014.

[2] 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.

[3] 小澤貫之, 鷲崎弘宣, 深澤良彰,“Simulink モデルの保守性向上に向けたクラスタリングおよび UML モデルとの双方向変換に関する研究,” 電子情報通信学会技術研 究報告. SS, ソフトウェアサイエンス, Vol. 112, No. 373, pp. 49–53, 2012.

[4] Guo Peng, Li Yahui, Li Peng, Liu Shuai, Sun Dongya,”A UML Model to Simulink Model Transformation Method in the Design of Embedded Software,” Proceedings of the 2014 Tenth International Conference on Computational Intelligence and Security, no. 5, pp. 583-587, 2014.