CATS CESL

制御システムモデルと組み込みソフトモデル (1) モデルベース開発

今井 良和†

制御システムのコントローラのソフトウェア開発に，モデルベース開発が有効であると提唱されている．モデルは実行可能な仕様書であり，シミュレーションによって妥当性を確認しながら開発できる．しかしながら，物理現象であるプラントの連続系ダイナミクスと，機能を実現する離散事象が混在するシステムを，単一のモデリング手法で表現するのは難しい．本稿では，HEV（Hybrid Electric Vehicle; ハイブリッド自動車）を例題にして，複数のモデリング手法の組み合わせ方とその効果を解説する．

Control Design Model and Embedded-Software Design Model (1) model-based development

YOSHIKAZU IMAI†

Model-based development is an effective method for software of control systems. Model is an executable specification, and simulation provides that the model has validity.

Though, using single modeling technique is difficult to model systems including time-continuous dynamics and discrete event dynamics. This paper uses the example of HEV (hybrid electric vehicle) model to explain how to use multi modeling techniques and the effect.

1. 制御システムのモデリング

まず，対象とする制御システムとモデリング手法について簡単に解説する．

本稿で対象とする「制御」とは，形のある「対象」に働きかけ，「対象」をある目的を達成するように機能させることである．例えば，ロボット・アームのアクチュエータ，エアコンのコンプレッサ，自動車のエンジンなどである．多くの工業製品は「制御」によってその機能を果たしている．

モデルベース開発は，制御対象（プラント）をモデル化することから始まり，それを求められる状態へ導くための制御器（コントローラ）をモデル化する．モデルはシミュレーションによって妥当性を検証し，モデルを改善していく．そして十分に検証されたモデルを元に製品を製作する．これにはソフトウェアの自動生成も含まれる．

コンピュータによるツールが実用化される以前は，モデルとは物理現象に基づいた数式であり，コントローラの設計は，数学的な解析手法によっていた．現在，モデリングには，ブロックダイアグラムによるツールがあり，広く使われている．代表的なものとして Simulink[1] , Scicos[2] などがある．さらに物理モデリングに適したツールとして Simscape[1], Dymola[3], MapleSim[4], OpenModelica[5] などがある．物理モデリングツールは，プラントをその物理的な構成から直感的に記述することができ，モデリングが飛躍的に容易になる．

ここまでに述べたブロックダイアグラムで連続時間ダイナミクスをモデリングするツールを，便宜上「制御設計ツール」と呼ぶ．

次にコントローラに着目したモデリングを考える．一般的にコントローラは，離散事象を扱う．離散事象とは，例えば，ユーザーのスイッチ操作，室内気温が目標温度に達した，センサ/アクチュエータなどの機器の故障などである．離散事象を扱うために，コントローラは，マイクロプロセッサ上のソフトウェアーいわゆる"組み込みソフトウェア"で実現する．

離散事象システムは，有限状態機械(FSM)でモデルにできる．組み込みソフトウェアでは，FSM をベースとした「状態遷移モデル」が使われる[6].

†キャッツ組込みソフトウェア研究所, CATS Embedded Software Laboratory

以下，HEV を例として，制御設計ツールによるモデリングと，それに状態遷移モデルを組み合わせて，コントローラのソフトウェアを開発する過程を説明する．そして，2つのモデリング技法を組み合わせることの効果について述べる．

2. 制御設計ツールによるモデリング

まず，制御システムのプラントとコントローラを，制御設計ツールでモデリングし，シミュレーション結果からコントローラのモデルを改善する手順について述べる．

2.1. HEV システムのモデリング

図 1は，HEV のパワープラントのモデルである[7]．車輪はモータだけで駆動する(エンジンの動力を発電だけに使う)「シリーズハイブリッド」と呼ばれるシステムである．システムは，自動車の車体と車輪に相当する負荷，車輪を駆動するモータ，モータを駆動するためバッテリー，バッテリーを充電するためのエンジン，そしてエンジンの運転を制御するコントローラなどで構成される．このモデルでは，エンジン制御装置が「コントローラ」，のこりの要素が「プラント」である．

プラントを含む制御システムを物理モデリングツール MapleSim で作成した(図 1が)．MapleSim では作成したモデルの一部をコンポーネントとして再利用することができる．ここでは，別に設計されたエンジンやモータなどのコンポーネントを利用した．

物理システムを組み立てるように，コンポーネントを配置することで，プラントのモデリングができる．この機能によって，システムの変更にともなうモデルの修正が簡単になる．例えば，他のエンジンへの変更(4気筒から6気筒のエンジンに載せかえる)や別の車体に適用といった修正は，コンポーネントの置き換えるだけである．また，他のシステム開発(これもモデルベース開発)と共通するコンポーネントを一元化できる．

コントローラは，バッテリーの充電量を適切な範囲に収めることを目的とし，エンジンの回転数を制御する．ここでは充電量を入力とする PID 制御で実現している．

2.2. シミュレーションと改善

図 2は，前述のモデルをシミュレーションした結果である．シミュレーション結果における ○ は，エンジンが短時間の間に停止-運転していることを表している．燃費の向上を目的とする場合，この短い間欠は好ましくない．

それを改善するために，PID 制御の目標値を切り替えるようにしたのが，図 3である．ブロックダイアグラムでは，コントローラを多機能にするとモデルは複雑になる．

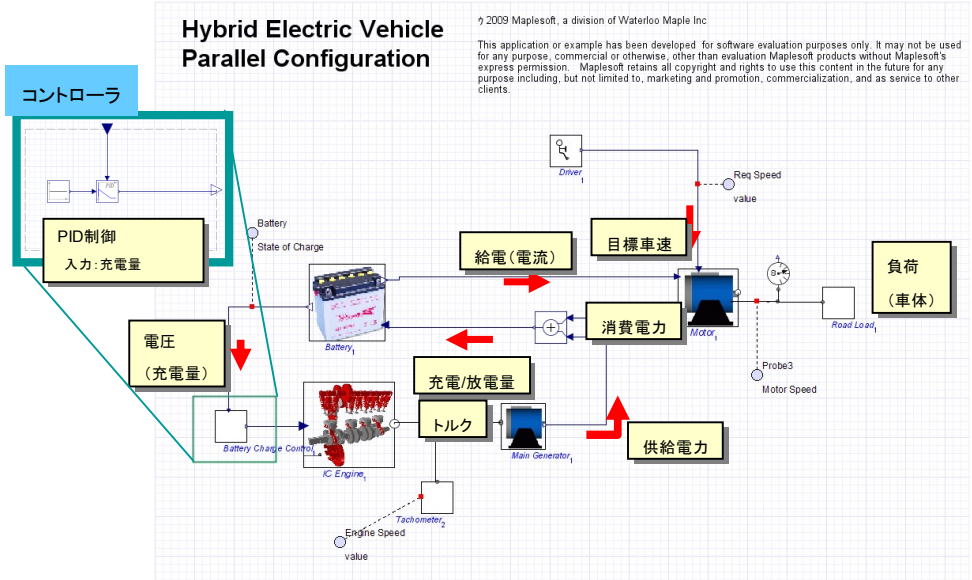


図1 HEV システムの制御設計モデル(1)

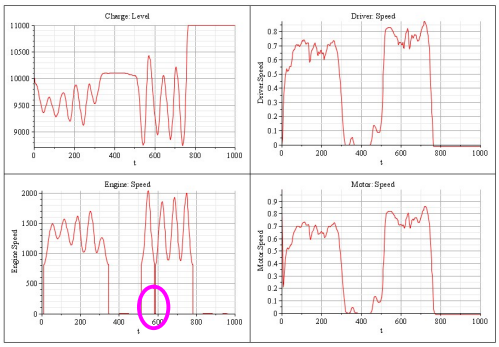


図2　シミュレーション結果(1)

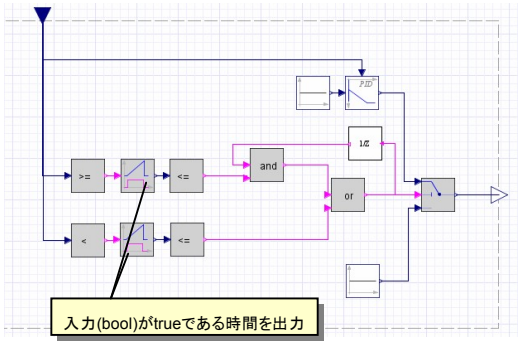


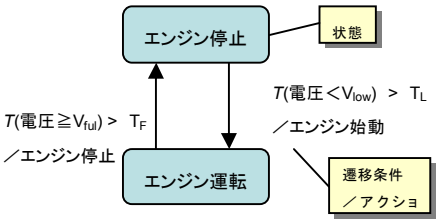
図 3　改良したコントローラ

3. 組み込みソフトウェアツール

3.1. HEV コントローラの状態遷移モデル

2.2のコントローラを状態遷移図で実現したのが図 4 である. エンジンは"運転", "停止"で特性が異なるので, コントローラは2つの状態を設ける. 状態に応じてエンジンの目標回転数を変えている. そのシミュレーション結果が図 5 である.

ここで, ブロックダイアグラム(図 3)よりも, 状態遷移モデル(図 4)の方が,「エンジンの短い間欠を抑止する」という要求を直接的に表現している.



T(p) ：条件 p が成立している時間

図 4　コントローラの状態遷移モデル

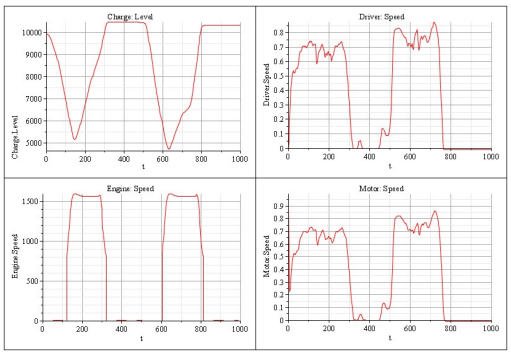


図 5　シミュレーション結果(2)

3.2. 状態遷移表と異常系への対応

次に, コントローラのソフトウェアの実装に向けて, 状態遷移モデルを詳細化, そして異常系に対応した設計にする手順について説明する.

制御システムの特性にあわせ, "運転", "停止"の2つの状態を設けた. 実際には, エンジンの運転, 停止はある程度の時間がかかる. そこでエンジンの運転, 停止の間の過渡状態を追加する(図 6). 過渡状態からの遷移をエンジンの回転数で判断している.

この状態遷移モデルを, 異常系に対応するモデルへ拡張する. どれだけ多くの異常系を想定できるかが, 製品の品質に影響する. そのために状態遷移図で表してきたモデルを表形式の「状態遷移表」にする(図 7). 状態遷移図と状態遷移表は等価であり, 相互に変換できる.

事象と状態の組み合わせを確認することで,「エンジンを始動しても運転できない(スタータの故障)」,「エンジンを運転しているのに充電できない(ジェネレータの故障)」という異常系を発見できた(図 7 中, 赤枠内).

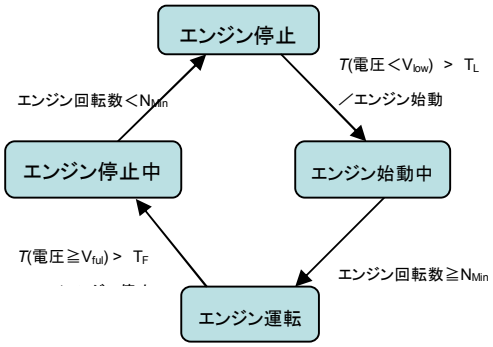


図 6　過渡状態を考慮した状態遷移モデル

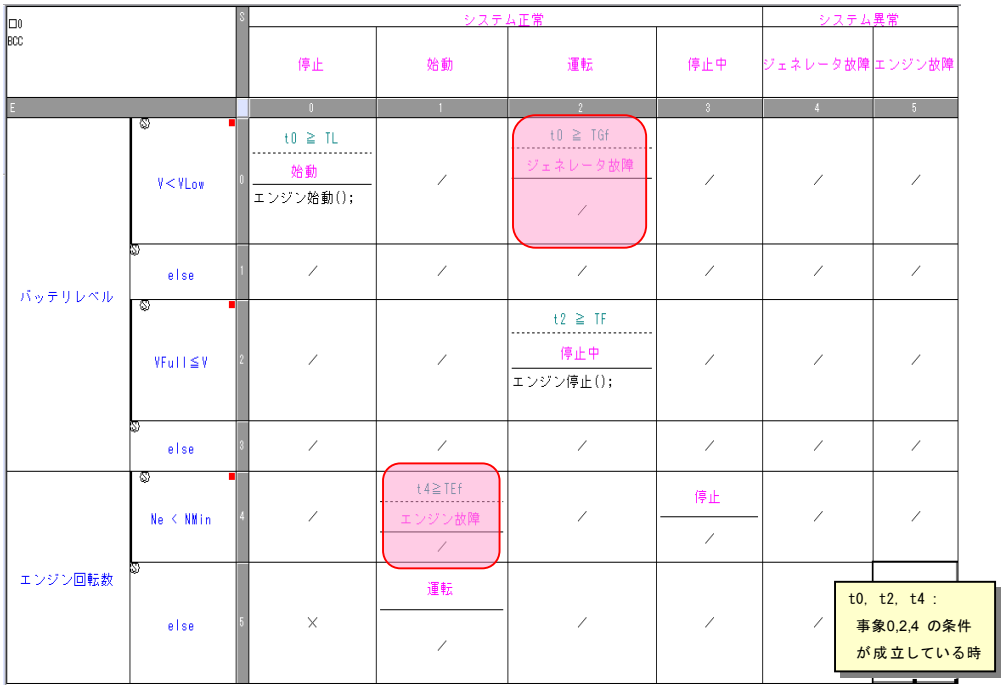


図 7 異常系に対応した状態遷移表

4. 効果

コントローラのソフトウェアを制御設計ツールだけでモデリングする場合と，状態遷移モデルを組み合わせた場合の違いをまとめる．

応答時間，安定性やコストなどの「非機能要件」は，主に連続時間ダイナミクスに依存する．例えば PID 制御の時定数で応答時間を調整したり，バッテリーの寿命を延ばすために充電量の目標値を変えたりする．一方，"～のとき，・・・する"のような論理で表現できる「機能要件」は，主に離散事象として現れる．例えば，バッテリーの充電量が低下したときエンジンを運転する、という要求は，「バッテリーの充電量が低下」というのが離散事象である．

コントローラのソフトウェア開発を，要求分析，制御設計，ソフト設計の3つのフェーズに分けたときの成果物の流れを，図 8，図 9に示す．図中点線は設計者の作業を，実線はツールによる変換をあらわす．なお，ここでいうソフト設計フェーズは，コーディングだけでなく，CASE ツールなどを使う場合も含む．

制御設計ツールだけの場合（図 8），また3.2であげたような異常系への対応といった機能要求は，ソフト設計フェーズで扱う（あるいは複雑なモデルを作成する）．また，離散事象のモデルは，次フェーズではリファレンスにしかならず，同じ要求に対する設計を二度行うことになる．検証-改善を繰り返すモデルベース開発では，甚だ効率が悪い．

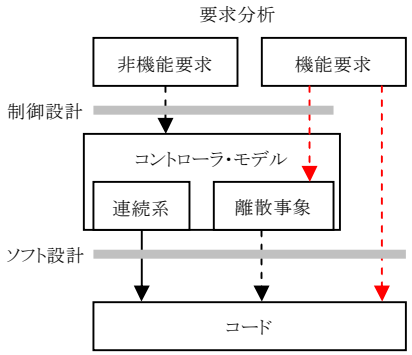


図 8 制御設計ツールだけ場合

これに対して状態遷移モデルを組み合わせた場合（図 9）では，制御設計フェーズの成果のすべてを次のフェーズに反映できることを示している．また3.2に示したように，状態遷移表によるモデリングは，要求分析にも有効である．

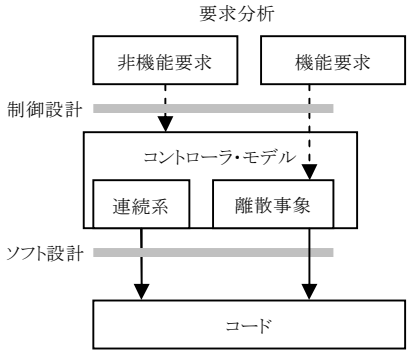


図 9 状態遷移モデルを組み合わせた場合

5. 今後の課題

先述したように，モデルベース開発は，モデリングとシミュレーションを繰り返してモデルを改善していく．また要求分析からコードにいたるまでの成果物（文書，モデル，コード）は関連を持っている．制御システムのソフトウェア開発は，ドメイン，ソフトウェアの複数を必要としている．このことから，成果物間の関連を管理する仕掛けが必要である[8]．この課題については別稿で解説する．

6. まとめ

制御システムのソフトウェア開発において，制御設計ツールに状態遷移モデルを組み合わせてコントローラをモデル化する利点について説明した．

状態遷移モデル/表の効果を強調したが，連続時間ダイナミクスのモデリングには，ブロックダイアグラムが有効である．

制御設計において，ブロックダイアグラムと状態遷移モデルという複数のセマンティクスが混在することになるが，開発フェーズ全体で得られる効果は高い．

本稿が，複雑化する制御システムに対するモデリングの指針になることを期待する．

謝辞

今回，プラントモデルの作成に協力をいただいたサイバネットシステムズの石塚真一氏に感謝する．

商標

参考文献

[1] The MathWorks, Inc., http://www.mathworks.com
[2] INRIA, http://www.scicos.org
[3] Dassault Systèmes K.K., http://www.3ds.com
[4] サイバネットシステムズ株式会社, http://www.cybernet.co.jp
[5] The Open Source Modelica Consortium, http://www.ida.liu.se/labs/pelab/modelica/OpenSourceModelicaConsortium.html
[6] 2008 年版 組込みソフトウェア産業実態調査報告書－プロジェクト責任者向け調査－（経済産業省）, http://www.meti.go.jp/policy/mono_info_service/joho/downloadfiles/2008software_resarch/project_houkokusho.pdf
[7] NEDO 技術開発機構，技術解説：省エネルギー：ハイブリッド車, http://app2.infoc.nedo.go.jp/kaisetsu/seg/seg04/index.html#elmtop
[8] 渡辺．組込みソフトウェア開発課題への挑戦 ～統合化～, http://www.zipc.com/cesl/info/07.pdf

今井 良和

1996 年豊橋技術科学大学 知識情報工学専攻 修士課程修了．

現在，キャッツ株式会社 ．