技術紹介

モデルベースデザインによる制御設計

石田　修一 *1
Ishida Shuuichi

近年、制御システムは高機能化や大規模化が急速に進み、従来の開発手法では開発費用の増加、開発期間の拡大、および品質の確保が困難となった。対して顧客からは、開発期間の短縮やコストダウンへの強い要求がある。この現状を改善する方法として、モデルベースデザイン（MBD：Model-Based Design）が自動車分野、航空防衛分野など広い範囲の分野で製品開発に生かされてきている。本稿においてモデルベースデザインによる開発プロセスを紹介する。

キーワード：制御設計ツール、MATLAB/Simulink、LabVIEW

1. はじめに

近年、制御装置開発は高機能化、大規模化に加え、顧客からは開発期間の短縮、コストの削減への対応が求められている。これらの要求に対応する開発手法として近年、自動車分野、航空防衛分野を中心に製品開発プロセスにモデルベースデザインを適用する企業が増えている。

本稿ではモデルベースデザインを紹介するが、まず従来の開発プロセスについて説明する。**図 1** に制御システムにおける一般的な V 字開発プロセスを示す。

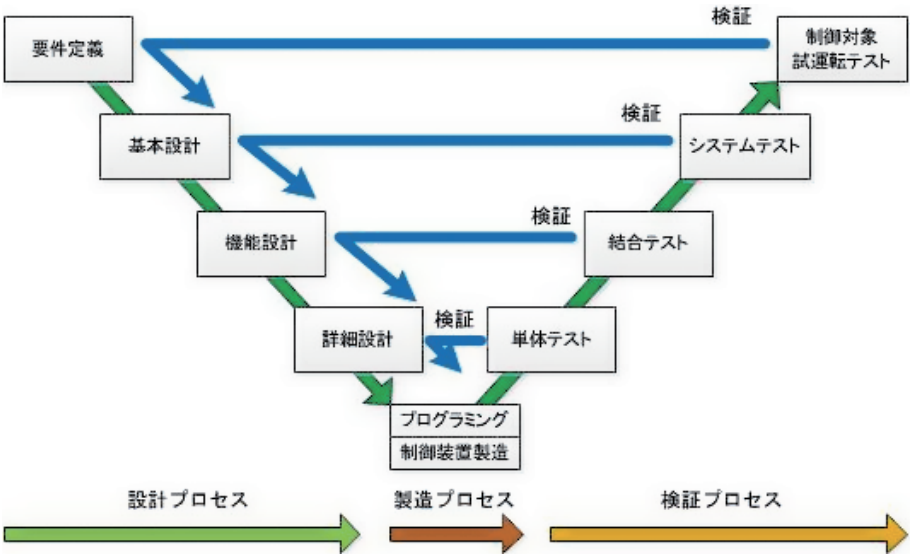

図 1　V 字プロセス

*1：制御システム事業部　技師長

従来の開発プロセスは、規定した要件定義を基に設計プロセスで制御システムを機能単位に設計し制御装置を製作する。

次に検証プロセスで制御装置を用いて機能単位の検証作業を行い、設計した機能が要件定義に合致していることを確認する。この検証作業において確認された要件定義と機能の不整合は、**図 1** に示す通り検証した機能の設計プロセスまで遡り検討・修正を行う。以上のように従来の開発プロセスは、検証作業と検討・修正作業を繰り返し制御システムの品質を作り込む作業となる。この開発プロセスを近年の高機能化、大規模化した制御システムの開発に適用すると検証、検討・修正作業の繰り返しが肥大し、開発期間と開発コストの増大や検証不足による品質低下が生じる結果となる。

この問題を解決する手法として、モデルベースデザインによる製品開発プロセスが進んでいる。

図 2 にモデルベースデザイン（MBD：Model-Based Design）による V 字プロセスを示す。

MBD ベース V 字プロセスと従来の V 字プロセスで大きく異なる点は、設計プロセスを構成する基本設計、機能設計、詳細設計、各作業においてモデルシミュレーションを行い、設計結果の検証を行うことにある。

よって、モデルベースデザインでは制御設計ツールを用い、設計プロセスで制御対象となるプラントと開発対象である制御装置をモデル化し、プラントモデルと制御装置モデルを接続して制御システムとしてのモデルを構築する。この制御システムモデルで運用状態を想定した閉ループシミュレーションを繰り返し行うことで、検証プロセスの検証作業を設計プロセスの各作業と並行して行う。この結果、開発プロセスを通して適正な検証時間を確保できるようになり、「品質の向上」、「後戻り作業の削減」により開発期間とコストの削減を実現できる。

モデルベースデザインを支援する制御設計ツールと主な提供メーカ 2 社を次に示す。

- Math Works 社の MATLAB/Simulink
- National Instruments 社の LabVIEW

制御設計ツールは、機械、電力、電気、ロジックが混在したマルチドメインモデルの作成とシミュレーションを支援するブロック線図作成環境

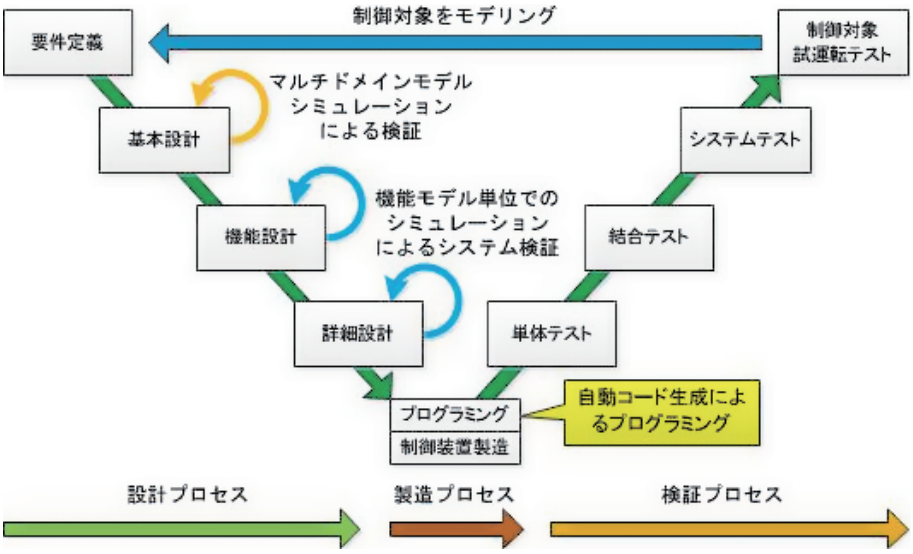

図 2　MBD ベース V 字プロセス

とからなる。このツールには制御設計／解析機能とともに、自動コード生成機能（ANSI/ISO 準拠の C および C++ コードに対応）やデバッグ機能が提供されている。

2. マルチドメインモデルの作成

我々の制御システム開発における制御対象は、モータやエンジンなどを動力源とした機械的な要素を制御する装置が多い。よって、モデルベースデザインで作成するモデルは、制御対象の機械的要素と、開発対象である制御装置の電気的要素が混在するマルチドメインモデルとなる。

次に制御設計ツールに Math Works 社の MATLAB/Simulink を使用したプロペラ機の推進システムモデル（**図 3** 参照）を例にとり、モデルベースデザインの進め方を説明する。

この推進システムではガスタービンの出力トルクを、減速装置を介してプロペラ軸に入力しプロペラ回転数を得る。また、プロペラ回転数とプロペラ角度により推力を発生させ、速力を得るプラントである。このプラントを制御する制御装置の要件定義を次のように設定する。

・ スロットルの操作量に比例した速力を得る。
・ プロペラ回転速度の応答時間を短縮する。

この要件定義から制御する信号は、プロペラ回転数とプロペラ角度となる。

図 3 の推進システムモデルは、①入力トルクと負荷トルクからプロペラ回転数を演算する軸回転数演算ブロック、②プロペラ回転数とプロペラ角度から推力と負荷トルクを求め、推力から空気抗力を差し引いて速力を演算する推力および速度演算ブロック、③入力トルクを発生させるガスタービンエンジンブロック、以上の 3 ブロックが制御対象モデルとなる。この 3 ブロックに開発対象となる制御装置のモデルを接続して、推進システムモデルとなる。

図 4 に推進システムモデルを示す。

図 4 を構成する各ブロックを **2.1 項**以降に示す。

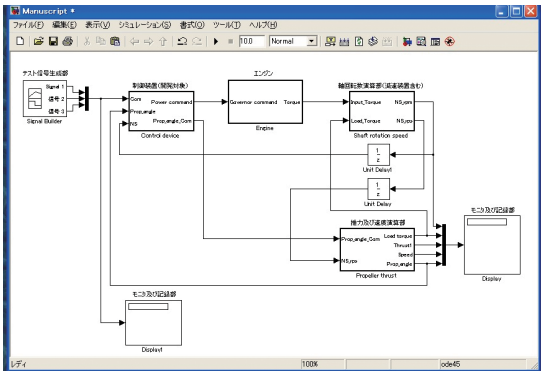

図 4　推進システムモデル図

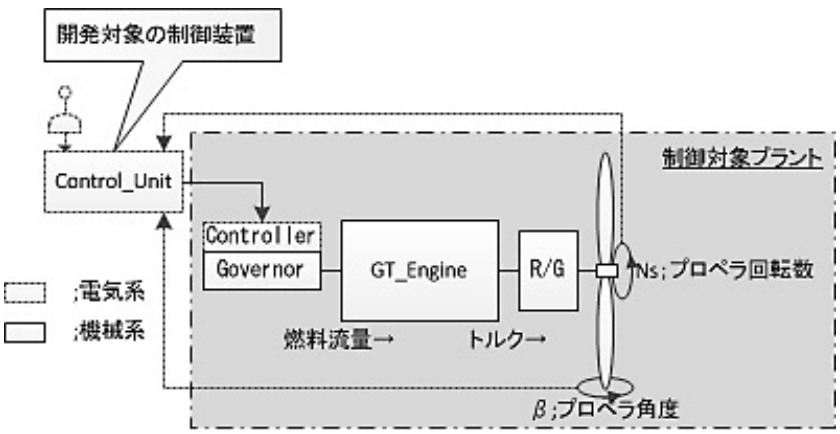

図 3　推進システム

2.1 軸回転数演算ブロック

エンジンからの入力トルクを減速機により増幅し、負荷トルクを差し引いてプロペラ回転数を演算する。

この運動方程式を式（1）に示す[(1)]。

$$\left(\eta n^2 I_1 + I_2\right)\frac{d\omega_2}{dt} = \eta n M_d - M_r \qquad (1)$$

η ：減速機動力伝達効率

1:n ：変速比

I_1 ：減速装置入力端慣性モーメント

I_2 ：減速装置出力端慣性モーメント

ω_2 ：出力軸角速度

M_d ：入力トルク

M_r ：負荷トルク

運動方程式をモデル化する場合、運動方程式を高次の微分係数について整理して行う。上記運動方程式を整理した式を式（2）に、本項モデル図を**図 5** に示す。

$$\frac{d\omega_2}{dt} = \frac{\eta n M_d - M_r}{\eta n^2 I_1 + I_2} \qquad (2)$$

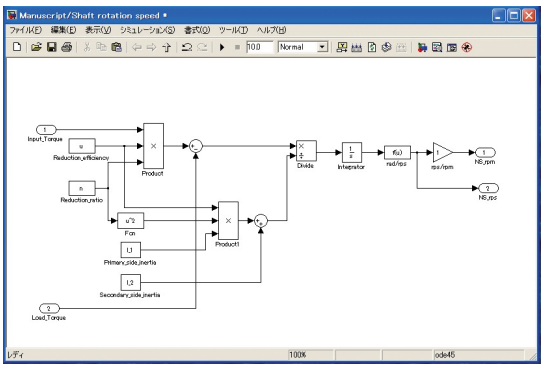

図 5　回転数演算ブロック図

2.2 推力および速度演算ブロック

空気抗力 D は、空気抗力係数 C_d、空気密度 ρ、主翼の面積を S とすると式（3）となる。

$$D = 1/2 C_d \rho V^2 S \qquad (3)$$

大気速度 V は、対地速度 V_E、追い風速度 U とすると式（4）となる。

$$V = V_E - U \qquad (4)$$

プロペラ前進係数 J は、プロペラ回転数 N_s、プロペラ直径 D_p、大気速度 V により式（5）となる。

$$J = \frac{V}{N_s D_p} \qquad (5)$$

推力 T とトルク Q は、それぞれ空気密度 ρ、プロペラ回転数 N_s、プロペラ直径 D_p とプロペラ性能から求まるスラスト特性 K_T、トルク特性 K_Q により式（6）、式（7）となる。なお、スラスト特性 K_T、トルク特性 K_Q は、プロペラ前進係数 J とプロペラ角度 β による非線形のプロペラ特性である。

$$T = K_T(J,\beta)\rho N_s^2 D_p^4 \qquad (6)$$

$$Q = K_Q(J,\beta)\rho N_s^2 D_p^5 \qquad (7)$$

上記の式（3）～式（7）から推力および速度演算ブロックモデルを作成する。本項モデル図を**図 6** に示す。

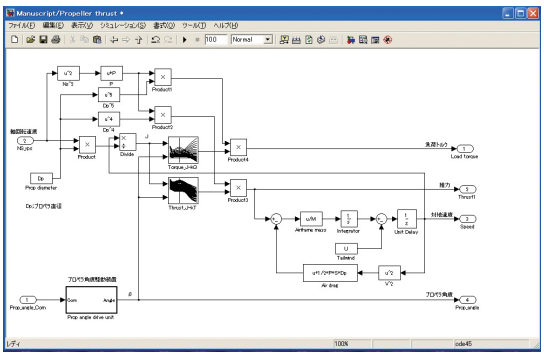

図 6　速度演算ブロック図

2.3 エンジンブロック

エンジンブロックは、ガスタービン本体と燃料流量を制御する燃料計量弁からなる。燃料計量弁は、制御装置から入力された出力指令に比例するガスゼネレータ回転数に合わせて燃料流量をガスタービンに出力する。ガスタービンは、入力された燃料流量からトルクとエンジン諸元を演算し出力する。**図 7** に本項モデル図を示す。

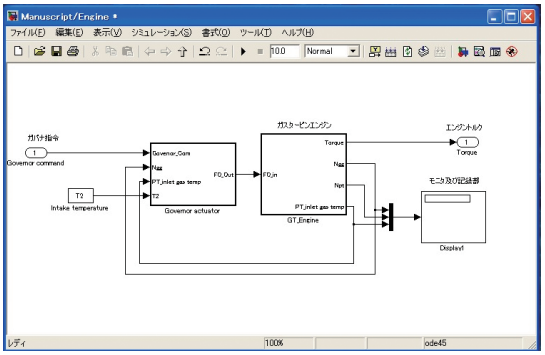

図 7　エンジンブロック

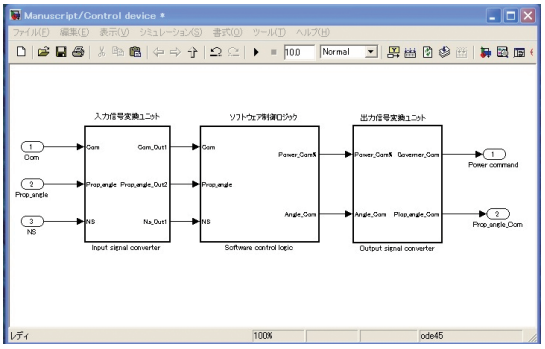

図 8　制御装置ブロック

2.4 制御装置ブロック

制御装置ブロックは、ハードウェアである入出力信号変換器とソフトウェアである制御ロジックで構成される。ハードウェアである入出力信号変換器は、既存基板を使用することが多く、一度モデル化すると、同様の制御装置を開発する場合にプラットフォームモデルとして流用できる。

制御ロジックのモデル設計においても流用を考慮した機能分析により共通部を抽出し、モデルのプラットフォーム化を図ることが重要である。

この推進システムの制御ロジックを構成するブロックは、プロペラ回転数を制御する PID（Proportional-Integral-Derivative）制御ブロック、プロペラ角度と角速度を制御するプロペラ角度制御ブロック、操縦モードを選択するシーケンス制御ブロックの 3 つに機能分解すると良い。

なお､プロペラ回転数を制御する PID 制御部は、回転数のオーバーシュートを防止するアンチワインドアップ制御を適用する必要がある。

図 8 に本項ブロック図を示す。

ここでモデルのプラットフォーム化を考えると、プロペラ回転数のアンチワインドアップ PID 制御部、およびプロペラ制御部をプラットフォームとして、流用性を上げた汎用モデルとして設計する。エンジン馬力／回転速度、プロペラ角度／速力指令スケジュールと PID 制御定数などは、機種ごとに異なるアプリケーション部として可変性を高め設計する。

3. シミュレーション作業

次に、開発プロセスにおいて実施する主なシミュレーション作業を示す。

3.1 ソフトウェア・イン・ザ・ループ（SILS：Software In the Loop Simulation）

この作業は、設計プロセスで行うシミュレーションで、作成した推進システムモデル（**図 4** 参照）を使用してコンピュータ上で実運用を想定した閉ループシミュレーションを繰り返すことになる。この作業は、コンピュータの仮想空間を使って実施することから、ソフトウェア・イン・ザ・ループシミュレーション、通称シルズ（SILS：Software In the Loop Simulation）と呼ばれる。

シミュレーションでは、主要な制御機能を設計するたびに行い、機能が要件定義に合致していることを確認し、不適合点を修正しロジックに反映してゆく。

図 9 に SILS によりアンチワインドアップ PID 制御部の構造を調整し、プロペラ回転数の応答性を改善させた結果をグラフで示す。

上記のとおり、設計プロセスにおいて SILS を繰り返すことにより設計と平行して検証が行える

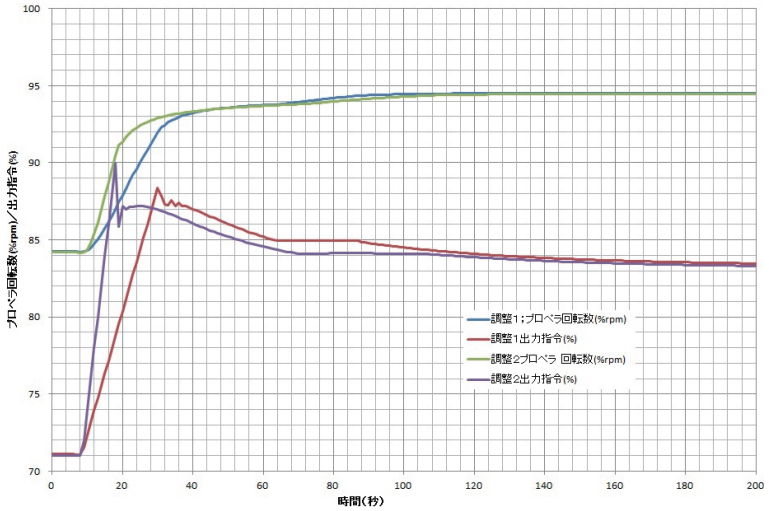

図 9　SILS による調整データ

ため、検証プロセスの期間短縮と検証時間の最適化が可能となり品質の向上が図れる。

3.2 ハードウェア・イン・ザ・ループ（HILS：Hardware In the Loop Simulation）

この作業は、検証プロセスで行うシミュレーションで実機の制御装置を使用して行う。実際の制御装置とシミュレータを組み合わせてシミュレーションすることから、ハードウェア・イン・ザ･ループシミュレーション、通称ヒルズ（HILS：Hardware In the Loop Simulation）と呼ぶ。

SILS で使用したプラントモデルをシミュレータに搭載し、制御ロジックを搭載した実際の制御装置と組み合わせて実際の運用に即した操作を行うことにより、開発対象である制御装置が要件定義を満たしていることを確認する作業である。モデルベースデザインにおいて、HILS の段階で設計プロセスまで遡って修正される機能は最小に管理される。

従来の開発においても、HILS は検証プロセスにおいて実施されていた。IHI は 1990 年頃の舶用ガスタービンの操縦装置開発において HILS を利用した検証作業を行い、当事業部もこの作業に深く関与していた。

当時、SILS 環境はなく、設計した制御ロジックとプラントモデルは HILS の作業に入って初めて組み合わせるために、プログラムチェック、および制御ロジックの検証作業に膨大な時間を必要としたことを覚えている。

当事業部では平成 11 年頃からモデルベースデザインに取り組み、制御アルゴリズムの設計に加え、主制御装置や HILS シミュレータなどを自社開発し IHI の製品開発に供している。

図 10 に HILS シミュレータ用の基板を示す。

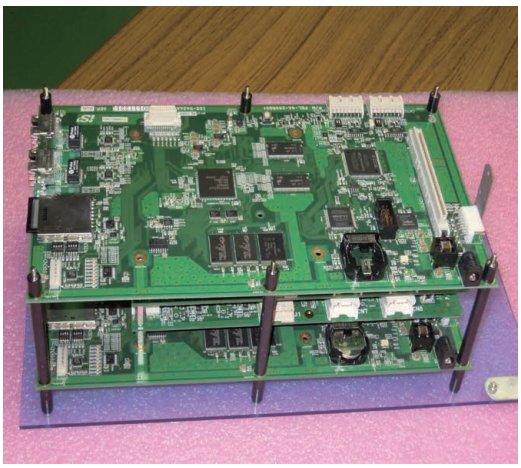

図 10　シミュレータ用基板

ハードウェア仕様

・CPU　SH4
・ネットワーク　100BASE-TX × 2 ポート
・シリアルポート　RS422 × 1CH
　RS485 × 2CH

4. まとめ

以上のようにモデルベースデザインでは、プラントモデルと制御装置モデルを設計プロセスで作成し、SILS を通じて制御システムとしての検証を十分に行うことによって検証プロセスの繰り返し作業を削減できる。また、制御設計ツールのオートコード機能を活用することによってヒューマンエラーの排除､プログラミング工数の削減も合わせて行える。

設計プロセスの SILS から検証プロセスの HILS までの開発プロセス全体を通して開発チームが同じモデルを使用することで、開発チームのコミュニケーションが向上し、作業の効率化が図れることから製品品質の向上も図れる。

ただし、モデルベースデザインにおいても次の問題点がある。

“モデルシミュレーションを重視するあまり、モデルのみを作成し開発の経緯を残していない。”

“設計プロセスの機能分析が不十分なままプラットフォーム化が行えない複雑なモデルを構築する。”

その結果、保守性が悪く機能追加、変更および他機種への機能流用などが困難となる問題が発生している。

これを解決するには､要件定義の“目的”“要求”“手段”に分けた文書化と制御装置の十分な機能分析によるプラットフォーム化が重要であるが、未だ十分とは言えない。今後、要件定義のフォーマット化を含め取り組む課題は多い。

また、最近、制御設計ツールは、FPGA（Field Programmable Gate Array）や SoC（System-on- a-chip）といった新たなロジックデバイスの開発環境、画像処理アルゴリズム開発環境などを提供してきている。

今後は新たな事業分野を考えるためにも、新たな開発環境の情報収集や分析にも取り組んで行く必要がある。

参考文献

(1) 日本機械学会編 機械工学便覧

制御システム事業部
技師長
石田 修一
TEL. 042-523-8313
FAX. 042-523-8320