MATLAB/Simulinkと計量機器コントローラで実現する

プロセス制御へのモデル・ベースド・デザインの適用と実装(前編)

小針 昌則/野口 芳和

21世紀の製造業は，国際的な競争にさらされているばかりでなく，製品ライフサイクルの短期化，ニーズの多様化，環境や安全に対する規制の強化など，ビジネス環境が大きくかつ急激に変化しています．企業がこのようなビジネス環境の変化に早急に対応するためには，新製品開発期間の短縮が課題となっており，設計期間や実装期間の短縮が望まれています．

本稿では，制御システムの設計および実装期間が従来の方法と比較して大幅に短縮可能な「モデル・ベースド・デザイン」(「モデル・ベース・デザイン」や「モデル・ベース設計」ともいう)を取り上げ，モデル・ベースド・デザインの導入例として，バッチ・プロセス[注1]の温度制御にモデル予測制御(Model Predictive Control ： MPC)[注2]を適用する事例を紹介します．

モデル・ベースド・デザインの導入

従来のコントローラ設計手法とモデル・ベースド・デザインによる設計手法の違いを説明し，モデル・ベースド・デザインの導入による効果，モデル・ベースド・デザインに必要となるソフトウェアやハードウェアについて紹介します．

● モデル・ベースド・デザインとは

▶ 従来は仕様書の解釈の違いによりトラブルが生じやすかった

従来，コントローラの設計は，客先の要求仕様に基づいて，基本設計仕様書を基本設計エンジニアが作成します．その仕様書に基づいて，詳細設計エンジニアが詳細設計仕様書を記述し，それらの仕様を実現するための実行用ハードウェアおよびソフトウェア環境を調達します．次に，プログラマがソース・コードを記述し，実行用ハードウェア上にコントローラを実装します(**図1**)[注3]．

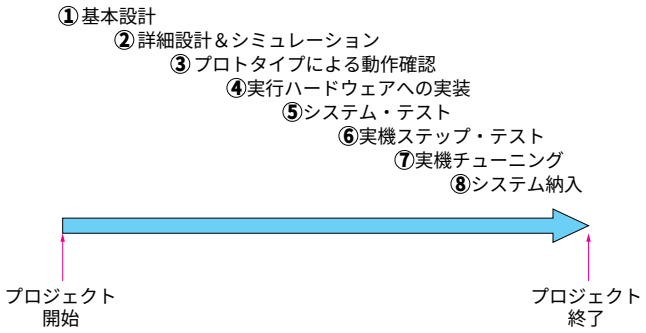

図1　プロセス制御のプロジェクトの流れ

完成したコントローラは実行用ハードウェア上での社内テスト通過後，客先を招いて検査を行います．ところが，この検査段階において，客先担当者，各設計者，プログラマがもつそれぞれの完成イメージの相違に気づき，完成イメージの再確認，設計修正，実装，検査のやり直しとなってしまうことがあります．

このようなトラブルのおもな原因は，設計やプログラミングの段階において担当者が異なるため，担当者間の仕様書の解釈の違い，すなわち，仕様書に記述されていないような詳細仕様部分の解釈の相違が生じるためと考えられます．

コントローラ実装の最終段階で，仕様を修正しようとすると，納期遅れ，費用増大という問題が生じてしまいます．かといって，各段階において，プロトタイプを作成し，完成イメージを合わせながら業務を進めようとすると，設計期間の長期化，費用増大といった問題が生じてしまいます．

このような問題を解決し，各担当者間での完成イメージを共有するために，近年，モデル・ベースド・デザインと呼ばれる設計手法が提唱され，自動車業界や通信業界で注目を浴びつつあります．

▶ 設計仕様をシミュレーション・モデルで表現し，完成イメージを共有化する

モデル・ベースド・デザインとは，基本設計段階から制御対象となるプロセスおよびコントローラの動特性や設計仕様を表現するシミュレーション・モデルを作成し，仕様書と併用することによって，各担当者間の仕様の解釈の相違を解消し，完成

注1：高付加価値製品の多品種少量生産に適した生産システム．一方，大量生産では連続プロセスが適用される．

注2：コントローラ内部に制御対象のプロセス・モデルを内包し，入出力変数の制約条件も考慮できる制御手法である．ただし，モデルの精度が制御性能の優劣を決定づけるので，制御対象プロセスを再現するレベルのモデル精度は必要となる．

注3：システムの規模が小さな場合には，基本設計エンジニア，詳細設計エンジニア，プログラマと1人で兼ねることも可能だが，これらの作業は，複数の担当者によって実現されているのが一般的である．ここでは，複数の担当者がコントローラを構築するものとして話を進める．

KEYWORD ── モデル・ベースド・デザイン，MPC，シミュレーション・モデル，CASEツール，MATLAB，Simulink，Stateflow，AD-4820，ジャケット付き反応槽，ステップ・テスト，モデル予測制御，PCRコントローラ

イメージを共有する設計手法です．初期段階で作られたシミュレーション・モデルは，基本設計，詳細設計，プログラミング，実装の各段階で同じモデルが使われ，段階が進むに従って完成形に近づくように詳細機能が追加されていきます．

設計段階で完成したシミュレーション・モデルは，Cコード自動生成機能を経由し，出力されたCコードは実行用ハードウェアへの実装にも使われます．このようにシミュレーション・モデルをベースとすることによって，仕様書の解釈の違いや設計環境とは異なった実装環境での手作業によるコントローラ実装作業に起因するヒューマン・エラーを削減できます．それによって，仕様変更による手待ち，手戻り時間の削減が可能となります．さらに，システム納入以降のオペレーション＆メンテナンス段階においても，納入したシミュレーション・モデルをベースに，プロセスの増設・更新の検討やプロセスの診断にも活用できます．

ただし，モデル・ベースド・デザインを導入したからといって，仕様書のもつ重要性に変わりはありません．シミュレーション・モデルだけで理解できないような複雑な制御仕様を第三者に伝えるには，シミュレーション・モデルだけでも不十分ですし，仕様書だけでも不十分です．

コントローラ設計から実装までの詳細ステップを**図2**に示します．それぞれのステップでは，以下に示す作業を行います．

① プロセス・モデル[注4]構築 ―― プロセスを一般性のある物理モデル[注5]もしくはブラックボックス・モデル[注6]として記述．精度のよいプロセス・モデルを構築することが肝要
② ステップ・テスト ―― 物理モデルもしくはブラックボックス・モデルのパラメータを決定するために，プロセスの入力変数を変化させたときの出力の時間変化（運転データ）を収集
③ プロセス・モデル評価 ―― 収集した運転データを基準とし，プロセス・モデルの評価を実施
④ コントローラ設計 ―― 構築したプロセス・モデルを制御対象とし，コントローラを設計
⑤ コントローラ評価 ―― プロセス・モデルと設計したコントローラを接続し，コントローラの制御性を評価
⑥ コントローラ実装 ―― 実行用ハードウェアにコントローラを実装

● モデル・ベースド・デザインの導入効果

従来の方法において，たとえば，現地調整で発見されたシステム仕様レベルの不具合の対応方法について考えてみます．その対応は，基本設計および詳細設計仕様書の修正を行い，その後シミュレーションによる動作確認，プログラム修正，現地と同一ハードウェアによる社内プログラム・テスト，現地でのプログラムの再インストールという複雑な過程を経由することになります．そのなかで，再度ヒューマン・エラーが混入されるかもしれません．

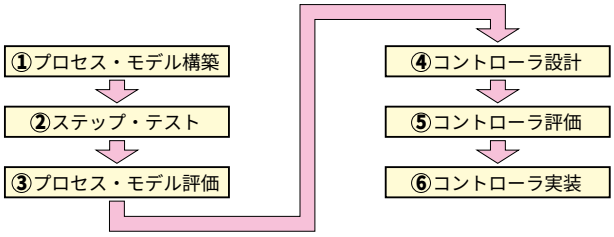

図2　コントローラ設計ステップ

ところが，モデル・ベースド・デザインを導入することにより，仕様変更に関連する動作確認がシミュレーション・モデル上で終了すれば，そのモデル・ファイルを現地にメールなどで転送し，現地ではシミュレーション・モデルからのCコード自動生成，コンパイル＆リンク，ダウンロードするだけで基本設計レベルの不具合に対応可能となります．

さて，モデル・ベースド・デザインをコントローラ設計とその実装に適用することによって，どのようなメリットがあるのでしょうか．

① Cコード自動生成機能によるプログラム・コーディング時間の削減，ヒューマン・エラーの混入防止ができる
② 仕様書とモデルを併用することによって，各担当者間での仕様の解釈に相違が生じない
③ プロジェクトに係る関係者が，プロジェクトを通じて同一モデルをレファレンスとすることができる
④ 設計の初期段階でモデルの評価ができるので，詳細設計以降の不具合が少なくなる
⑤ シミュレーション上でコントローラ実装前のエラーが検出できる
⑥ 追加仕様に対してシミュレーションによる検討をあらかじめできる

このように，モデル・ベースド・デザイン導入による効果は，プロジェクト進行に伴って生じるさまざまな問題を最短時間で解決でき，最終的にはコスト・ダウンに結びつくものとなります．

● モデル・ベースド・デザインに必要となるソフトウェアとハードウェア

さて，モデル・ベースド・デザインによるコントローラ設計とその実装を実現するためには，どのようなソフトウェアやハードウェアが必要となるでしょうか．

はじめに，ソフトウェアについて考えてみます．

▶ソフトウェアはMATLAB/Simulink，Stateflow，RTW

現在市販されているCASE（Computer Aided Software Engineering）ツールの中で，コントローラ設計に有効と考えられるのがMATLAB[注7]/Simulink[注8]（The MathWorks社）です．そ

注4：制御対象となるプロセスの挙動を表現できるモデルである．
注5：物理や化学の法則に基づいてプロセス動特性を微分方程式や代数方程式を表現するモデル．ブラックボックス・モデルとの対比で，ホワイトボックス・モデルと呼ぶことがある．
注6：プロセスの運転データから導出されるモデル．厳密な物理モデルの構築が困難であるプロセスに採用され，インパルス応答モデルやステップ応答モデルとして表現する．