2 サイバーフィジカルシステムを支える技術 —フィジカルシステムを理解する

基般

安積　卓也（大阪大学大学院基礎工学研究科）

フィジカルシステム

「サイバーフィジカルシステム（CPS）」の主要な分野として，制御工学，リアルタイムシステム・組込みシステム，検証・分析が CPS 向けの教科書図 -1[1] で説明されている．さらに，本特集 1. の概要と動向でも述べられていた通りセンサネットワークが主な分野として挙げられることが多い．上記分野のトップレベルの国際会議や CPS の関連のワークショップを集めて，CPS Week が開催されている．その中でも ICCPS（International Conference on Cyber-Physical Systems）や HSCC（Hybrid Systems: Computation and Control）の発表は，制御工学の基本的な知識があると，研究の内容を理解しやすくなる．本稿では，CPS の基本となる制御（連続）とコンピュータの世界（離散）のかかわりを中心に，フィジカルシステムを理解するために必要な制御工学の基本とモデルベース開発について解説する．

▶連続時間システムと離散時間システム

フィジカルシステムは，時間の切れ目のないシステムである連続時間システム図 -2 上で表現される．連続時間システムは，微分方程式などを用いて実世界の物理現象を扱う．一方，コンピュータが扱うディジタルデータのように飛び飛びの値を処理するシステムは離散時間システム図 -2 下と呼ばれる．

▶フィードバック制御

CPS は，制御系の組込みシステム開発で利用されている制御方法としてフィードバック制御を基にしている．図 -3 に基本的なフィードバック制御の構造を示す．制御対象は，自

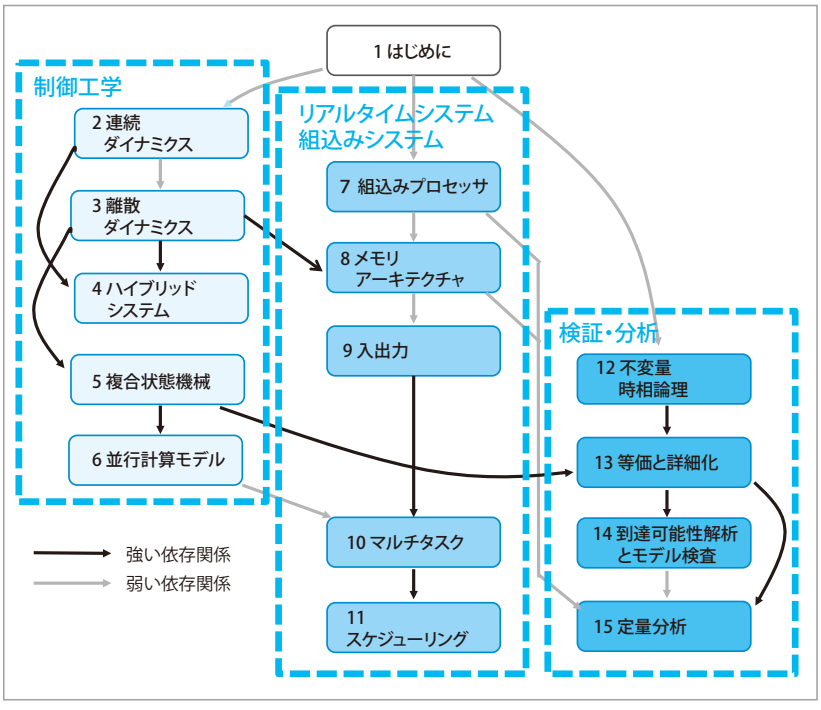

図 -1　CPS 教科書の章構成

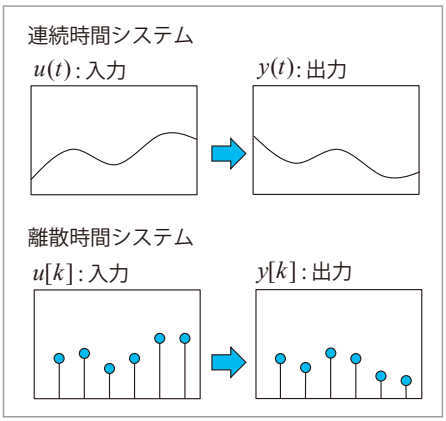

図 -2　連続時間・離散時間システム

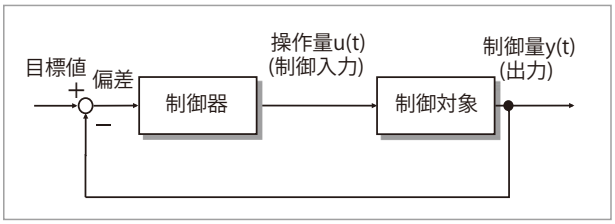

図 -3　フィードバック制御

動車，エアコン，ロボットアーム，電力などさまざまなものが挙げられる．フィードバック制御は，ある目標値と現在の値（制御量）を比較して，適切な操作量を制御器（コントローラ）で計算して制御対象をコントロールする．たとえば，制御目標値が 60km/h で現在の速度が 20km/h と 40km/h の場合では，操作量が異なる．この例では，20km/h のときの方が，目標値との差（偏差）が大きいため操作量も大きくなる．

制御対象への入力はモータなどのアクチュエータに対して制御ソフトウェアでコントロールする．モータの回転数など制御対象の状態（制御対象の出力）をさまざまなセンサを利用して取得する．CPS の場合，図 -4 に示す通り制御対象・制御器がネットワークで接続されており，複数の制御器・制御対象が共存しているところが複雑さを増している．さらに，制御対象が，機器だけではなく人間や社会であることにより，安定状態に持っていくことが難しくなっている．しかしながら，次章で述べる制御工学の基本的な知識は，CPS に活用できる．

元々組込みシステムは閉じた世界で動作していたため，悪意のあるプログラムが混在しないことを前提に開発できていた．しかし，CPS では，組込み機器がオープンネットワークに接続されるようになり，パソコンなど汎用システムと同様にセキュリティに関する問題にも対処する必要性がある．ICCPS 2014 の発表でも，サイバーセキュリティに関する論文がベストペーパーを受賞している．

▶モデリング

モデリングとは，着目しているある側面だけを抽出し，抽象化した図形モデルあるいは数理モデルを作成することである．モデリングの方法は，着目す

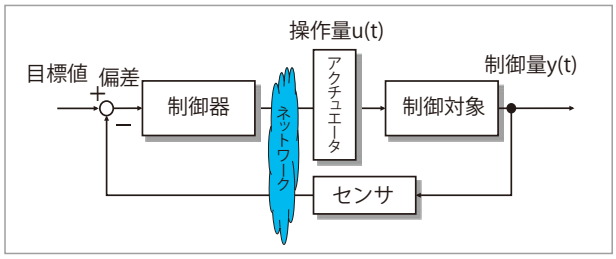

図 -4　CPS のフィードバック制御

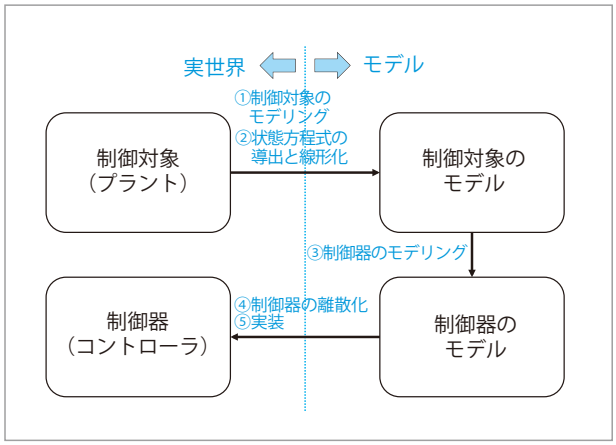

図 -5　制御器の設計・実装の流れ

る種類によってさまざまである．モデリングを間違えると，現実とかけ離れたモデルができることがあるため，注意が必要である．CPS では，1 つのモデルだけでなく，粒度や種類の違う複数のモデルを一度に扱う．たとえば，交通シミュレーション用のモデル，ネットワークモデル，自動車のモデル，エンジンのモデルなどを同時に扱う．次章で説明する微分方程式に代表される連続時間を扱う数理モデルや，状態遷移図など離散事象のモデルをどう統合的に扱うかも CPS の課題の 1 つである．

知っておくべき制御工学の知識

情報系の研究者が，CPS の研究を理解する上で必要な最低限知っておくべき制御工学の知識を情報系の言葉にかみ砕いて解説する．

制御器の設計・実装は，図 -5 の①制御対象（プラント）のモデリング，②状態方程式の導出，③制御器のモデリング，④制御器の離散化，⑤実装の順で行う．基本的な制御器の設計・実装の流れを図 -6 の倒立振子（とうりつしんし）の制御設計を例にして解説する．

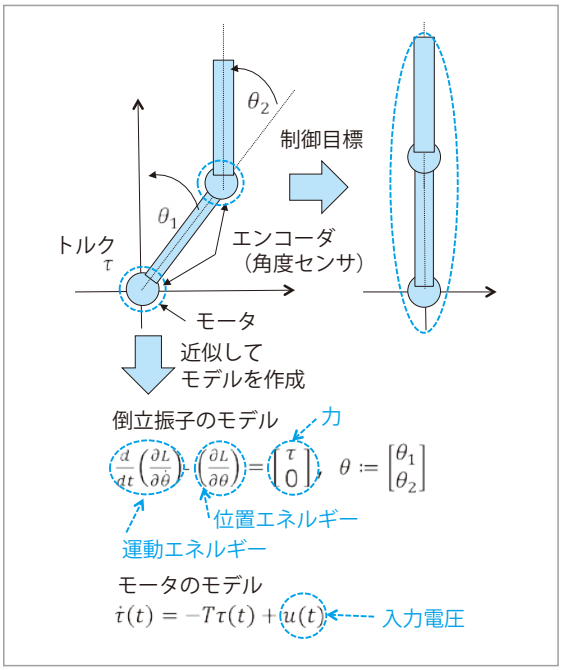

図 -6　①制御対象のモデリング

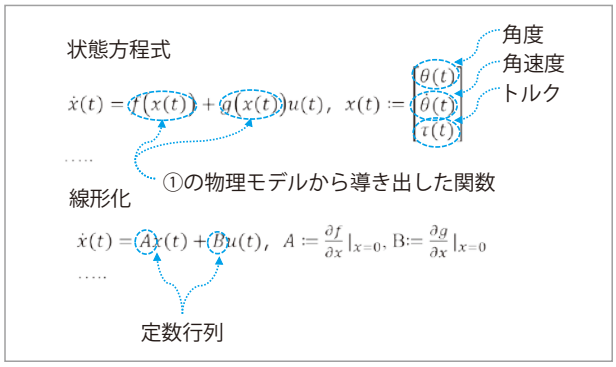

図 -7　②状態方程式の導出と線形化

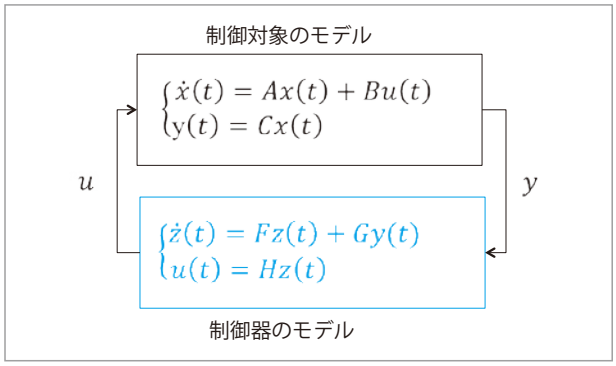

図 -8　③制御器のモデリング

倒立振子は，大学の制御の実験でよく用いられる教材である．

• 倒立振子の内部状態 *x(t)*

角度 θ，角速度 $\dot{\theta}$，トルク τ で表現された内部状態

• 倒立振子のセンサ出力 *y(t)*

センサで取得できる角度 θ

• 制御器の操作量 *u(t)*

モータの入力電圧

倒立振子の例では，内部状態 $x(t)$ を $\theta=0$ 付近で安定化させるために，センサ出力 $y(t)$ の情報を利用して制御器でいかに操作量 $u(t)$ を決めるかを目的として制御器の設計を行う．具体的には，トルク（力）を加え，モータを回転させ振子を鉛直上向きに安定して倒立させる（θ 1 と θ 2 の角度両方を 0 にする）ための入力電圧を決める制御器を設計する．

• 制御器の設計・実装の流れ[☆1]

① 制御対象のモデリング（図 -6）

まずは，倒立振子の運動モデルとモータの電気系のモデルを作成する．

・ラグランジュの運動方程式

（運動エネルギー）－ 位置エネルギー ＝ 力

・モータの電気系のモデル

制御器の操作量 $u(t)$ とトルクの関係式

$\dot{\tau}(t) = -T\tau(t) + u(t)$

その他のモデリング方法としては，ニュートンの運動方程式，回転運動方程式などが利用される．

② 状態方程式の導出と線形化（図 -7）

①のモデルから操作量 $u(t)$ と倒立振子の内部状態 $x(t)$ の状態方程式（制御対象の入力に対してどのような応答をするかを決定する方程式）を求める．ただし，導き出した状態方程式は，非線形関数であるため，計算が複雑である．そこで，この状態方程式を動作平衡点（$\theta=0$）付近で線形化することにより，計算をしやすくする．たとえば，動作平衡点でのテーラー展開を行うことによって線形化できる．ここでは詳細は省略する．線形化することで，状態方程式を定数行列 A，B，倒立振子の内部状態 $x(t)$，制御器の操作量 $u(t)$ で表現できる．

③ 制御器のモデリング（図 -8）

倒立振子のセンサ出力 $y(t)$ を利用して，フィー

☆1 今回紹介する制御設計の方法以外に，制御対象のモデルを離散化してから離散時間の制御器を作成する方法や，連続時間の制御対象から離散時間の制御器を直接作成する方法がある．

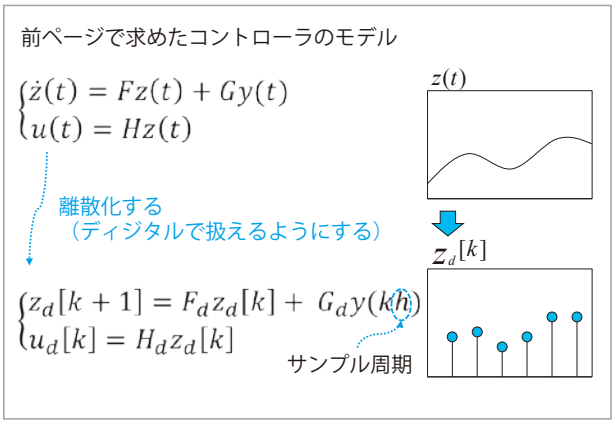

図 -9　④制御器の離散化

ドバック制御（図 -3）を行う．すなわち，直近の制御結果から操作量 $u(t)$ を導く制御器の数学モデルを作成する．制御器のモデリングには，②で求めた線形システムを安定化させる極配置法や，制御の善し悪しを決める評価関数を最大または最小にする最適制御などが用いられる．$z(t)$ は，倒立振子の状態を推定した数学モデルである．制御器内の推定した状態 $z(t)$ が実際の倒立振子の内部状態 $x(t)$ と大きく違っていると，操作量 $u(t)$ が大きくずれる．

④,⑤制御器の離散化と実装（図 -9）

③で求めた制御器のモデルをコンピュータ（ディジタル）で扱うため，制御器のモデルの離散化を行う．時間変数は，連続時間の場合は「t」，離散時間の場合は「k」で表現されることが多い．離散時間は制御器の実装以外にも，システム同定（慣性モーメントなど測定が難しいパラメータを推測する方法）などにも利用する．

次章で解説するモデルベース開発で用いられる Matlab/Simulink などのツールを利用することで，離散化や制御器のコード生成できる．

本来は，ラプラス変換（微分方程式の問題を代数の問題に変換する一手法）や制御システムの特性を表す伝達関数など抑えるべき項目がある．しかし，本稿の内容を抑えれば，ICCPS や本特集の筆者らが運営にかかわっている国際会議である CPSNA（Cyber-Physical Systems, Networks, and Applications）の論文の内容が大分読みやすくなる．たとえば，式の内部状態 $x(t)$，センサ出力 $y(t)$，操作量 $u(t)$ に着目すれば，制御対象と制御器のどちらを議論しているかが一目で分かる．

本稿では，詳細設計については，省略しているので，もう少し詳しく知りたい方は文献[2] などを参考にするとよい．

CPS での制御システムでは，制御器とセンサがネットワークで接続されていることもあり，センサ出力 $y(t)$ に遅延が発生することを考慮する必要がある．CPS では，複数の制御器や制御対象が複雑に絡み合ってはいるが，1 つのフィードバック制御に着目した場合には，ほかの制御対象や制御器などの外部要因は外乱として扱うことによって，制御工学の知識を活用できる．

人間をフィードバック制御のループの中に取り入れるという Humans in the loop が CPS でも考えられ始めている．ドイツを中心にまとめられている agendaCPS[3] では，CPS のアプリケーションと人間とのインタラクションをどう行えばよいかまとめられているので，参考にするとよい．

モデルベース開発

複数のモデルのシミュレーションやコード生成をすることができるため，モデルを利用したソフトウェア開発（モデルベース開発）が，CPS でも重要な要素の 1 つになってきている．CPSWeek では，モデルベース開発のさまざまな側面（シミュレーション，モデリング，コード生成など）について議論するために，複数のワークショップが開催されている．

広義のモデルベース開発は，狭義のモデルベース（MBD : Model Based Development：一般的に単にモデルベース開発と呼ぶ），モデル駆動開発（MDD : Model Driven Development）に 2 つに分類される．

モデルベース開発は，連続時間システムのモデルをベースとしたシミュレーション可能なモデルを用いるソフトウェア開発手法であり，制御器および制

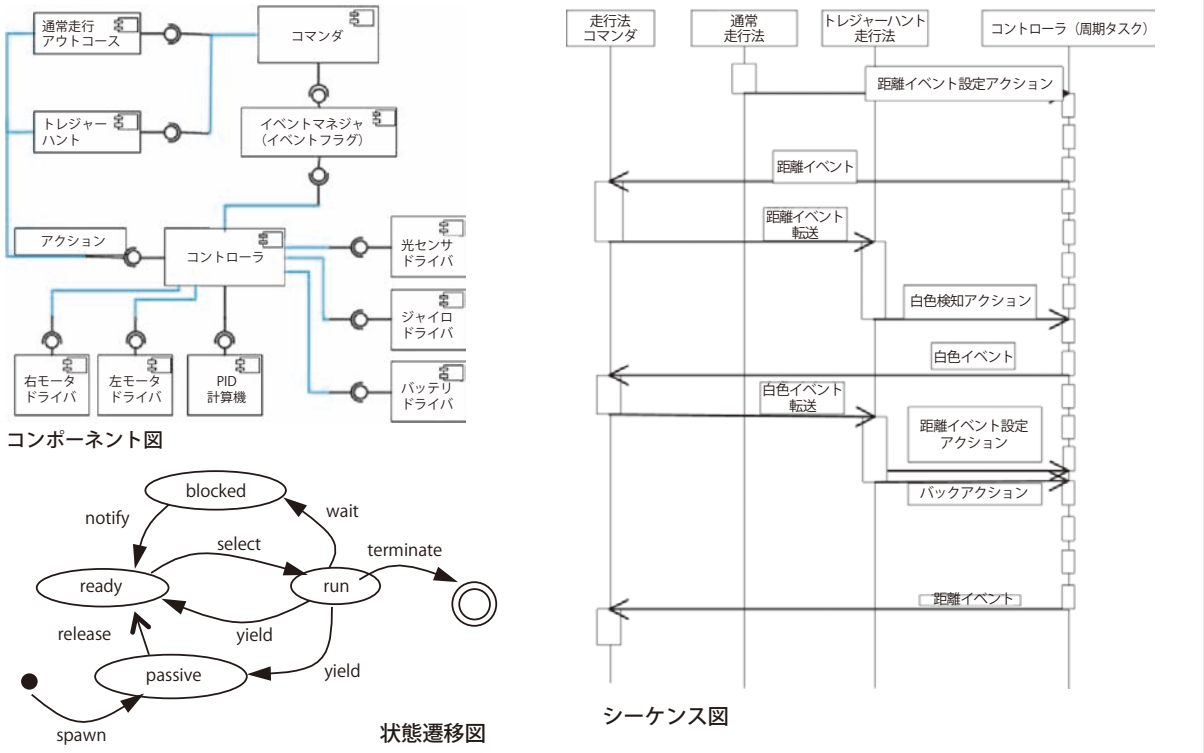

図 -10　モデル駆動開発のモデルの例

御対象の一部をモデルで表現し，シミュレーションにより制御アルゴリズムの開発・検証を行う．

一方，モデル駆動開発は，UML（Unified Modeling Language）に代表されるアーキテクチャ記述言語によるモデル化しソフトウェア開発を行う．図 -10 のコンポーネント図，シーケンス図，状態遷移図のように図で表現できるモデルを利用してソフトウェアの一側面を表現する．モデル駆動開発では，シーケンス図や，状態遷移図の情報からイベント管理などのプログラムを自動生成できる．SysML（Systems Modeling Language）や AADL（Architecture Analysis &Design Language）では，UML と違い，メモリ構成などの装置を含むコンピュータアーキテクチャもモデリングが可能である．

図 -11 にモデルベース開発と組込みソフトウェア・ハードウェアの関係を示す．制御対象のモデルには，Modelica という制御対象のモデル作成に適した言語を用いて物理シミュレーションを行う．Modelica は，Matlab/Simulink などで作成された制御器のモデルと連携してシミュレーションできる．モデルベース開発の利点は，実機を使わず検証できることであり，手戻りを減らすことができる．

Matlab/Simulink は，自動車業界を中心に，制御器，制御対象のモデルを作成，シミュレーションすることを目的に利用されている．これまでは，制御器のモデルを基に人手によって制御ソフトウェア（制御器のプログラム）を実装していたが，最近では，Embedded Coder や TargetLink などのツールを利用することにより，組込み機器用の実用的なソースコードを生成できるようになってきた．図 -11 に示す通り，制御ソフトウェアは，組込みシステムのソフトウェアの一部であり，その他，リアルタイム OS，機器全体の制御，例外処理，割込み処理，通信処理などは別途開発が必要になる．

CPS では，これまでは個別で扱ってきたモデルを，複合的に扱うことも増えてきている．たとえば，ハイブリッドシステムは，連続時間システムと，離散事象システムを同時に扱うシステムである．離散事象システムは，情報系でもなじみの深い状態遷移図を扱う．ハイブリッドシステムでは，状態遷移

図の各状態で制御器のモデル保持しており，ある事象（イベント）で，状態を切り替え，制御方式を切り替える．CPS Week の国際会議の 1 つである HSCC は，ハイブリッドシステムに焦点を当てた学会である．

スマートグリッドなど CPS の応用分野は，初期段階で実機を用いた動作確認をすることは難しい．モデルベース開発では，複数のモデル（ネットワークモデル，交通シミュレーション用のモデル，自動車など）を利用したシミュレーションを行うことで，実機なしで検証することができる．初期段階では，複数の制御対象は，Modelica や Matlab/Simulink などを利用しシミュレーションで動作確認を行う．シミュレーションで検証したモデルを基に，複数の制御器のコードを自動生成できるため，CPS のアプリケーションの開発に，モデルベース開発は，大きな役割を果たすと期待されている．

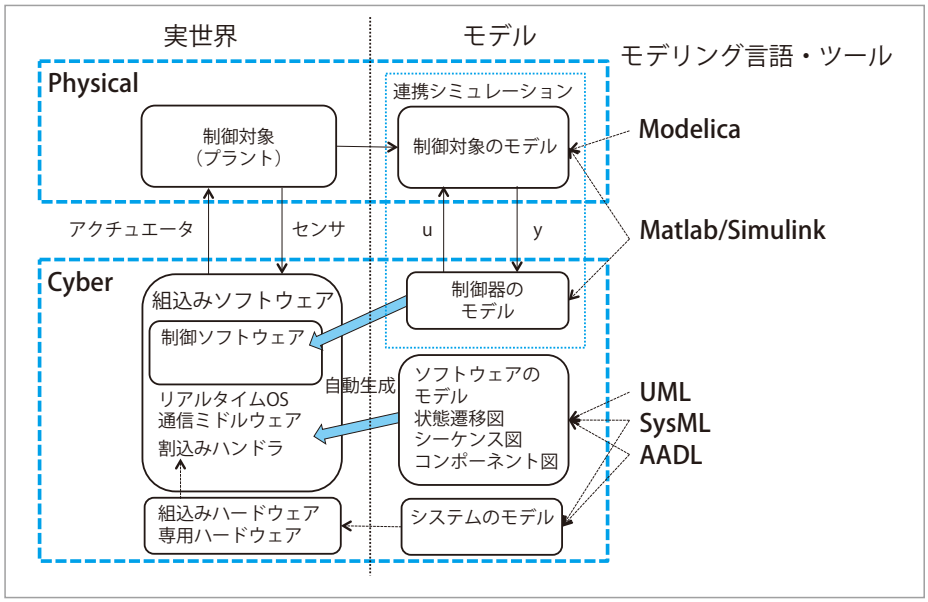

図 -11　モデルベース開発

さらに理解を深めるために

筆者も含め多くの情報系の教育を受けた人は，制御理論の授業を受講していないことが多い．本稿では，CPS を理解する上で必要なフィジカルシステム（連続時間）の基本をはじめの一歩として解説した．しかし，CPS をより理解するには，基本的な制御理論（古典制御，現代制御，モデリング）を理解しておくことが必要である．制御工学の講義の動画 [4] や資料が大学 OCW（Open Course Ware）などで公開されているので，活用されるとよい．冒頭で紹介した教科書（pdf 版は無料で公開いている [1]）は，CPS という視点から最低限必要な知識を網羅しているので，研究室単位での英語輪講で活用するには，ちょうどいい教材である．

参考文献

1) Lee, E. A. and Seshia, S. A. : Introduction to Embedded Systems-A Cyber-Physical Systems Approach (2014), http://leeseshia.org/
2) Mathworks : MATLAB/Simulink サンプルモデル 解説書 - 倒立振子の安定化制御のラピッドプロトタイピング編 - , http://www.mathworks.com/tagteam/56581_TA014_Inverted_Pendulum_Stabilization_Control_Rapid_Prototyping.pdf
3) agendaCPS, http://www.fortiss.org/en/research/projects/agendacps/
4) http://www.appi.keio.ac.jp/?page_id=437

（2014 年 4 月 30 日受付）

安積卓也（正会員）　takuya@sys.es.osaka-u.ac.jp

2009 年名古屋大学情報科学研究科博士課程修了．博士（情報科学）．日本学術振興会特別研究員 PD，立命館大学，カリフォルニア大学アーバイン校を経て，2014 年より現職．組込みシステム，リアルタイムシステムの研究に従事．