PCA と MVC の複合アーキテクチャスタイルを用いた組込みモデリング技法の提案

松澤芳昭[†] 児玉公信[††]
塩見彰睦[†] 酒井三四郎[†]

本研究では，PCA(Process Control Architecture) と MVC(Model-View-Controller) の複合アーキテクチャスタイルを用いたモデリング技法を提案する．制御系組込みシステムのアーキテクチャについては，オブジェクト指向システムが必ずしも最適ではなく，PCA を骨格とするアーキテクチャが適すことが Shaw(1995) によって提案されている．この議論を基礎として，本研究では PCA を骨格に MVC システムを埋込む形で複合アーキテクチャスタイルを形成した．追加された MVC システムは制御システムに渡す目標値の算出システムとしての役割を担う．目標値の保持を Model に，センサからのフィードバックの受信を View，並行状態機械を Controller として，責務を整理してモデリングすることができる．オブジェクト指向，制御，状態機械の 3 種のモデルが扱え，かつシンプルなアーキテクチャスタイルであることが特徴である．社会人を対象とした教育カリキュラムにおける 2 つの課題に対して適用し，組込みモデリング初学者でも効果的なモデルが作成できることを具体例で示した．

A Proposal of Object-Oriented Modeling Method for Embedded Software using Combined Architectural Style with PCA and MVC

Yoshiaki Matsuzawa,[†] Kiminobu Kodama,[††] Akichika Shiomi[†] and Sanshiro Sakai[†]

In this paper, we propose a combined software architectural style for embedded software with PCA(Process Control Architecture) and MVC(Model-View-Controller) architecture. Shaw(1995) proposed the software organization paradigm motivated by process control loops for the process control software rather than in an object-oriented design. Based on the Shaw's proposal, we developed an architectural style in which MVC unit is embedded in PCA. The responsibility of the embedded MVC unit is defined as to produce desired values for a PC unit. We can manage the model disciplines the internal MVC unit by Model is modeled as a holder of desired values, View is modeled as receivers by sensors, and Controller is modeled as the concurrent finite state machines. Object-Oriented model, Process Control model, and Finite State Machine Model are simply managed by applying the proposed architectural style. The proposed architectural style has been tried by modeling beginners but working in embedded software industry and succeeded.

1. はじめに

近年，ソフトウェア開発一般に，モデルベース開発が普及段階にある．

企業情報システムの分野では，UML(Unified Modeling Language) を用いたオブジェクト指向に基づく概念データモデリング，およびフレームワークを利用したモデルベース開発 (MDD: Model Driven Development) が広範に普及している．組込みシステムの分野でも，多機能化・複雑化に伴う開発規模の増大とプロセッサ性能の向上により，企業情報システムで培われたオブジェクト指向技術を適用しようという動き（例えば,[1)]）が始まっている．

その一方で，組込み開発現場では，厳しい時間制約や信頼性向上に培われた別のモデルベース開発が進化している．その一つは制御系組み込みシステムにみられる，制御システムのモデルベース開発である．これは，プラント（制御対象）およびコントローラ（制御方式）を数理モデル（微分方程式）でモデル化する方法である．モデル化された数理モデルが，CAD ツー

† 静岡大学情報学部
Faculty of Informatics, Shizuoka University
†† 株式会社情報システム総研
Information Systems Research Institute

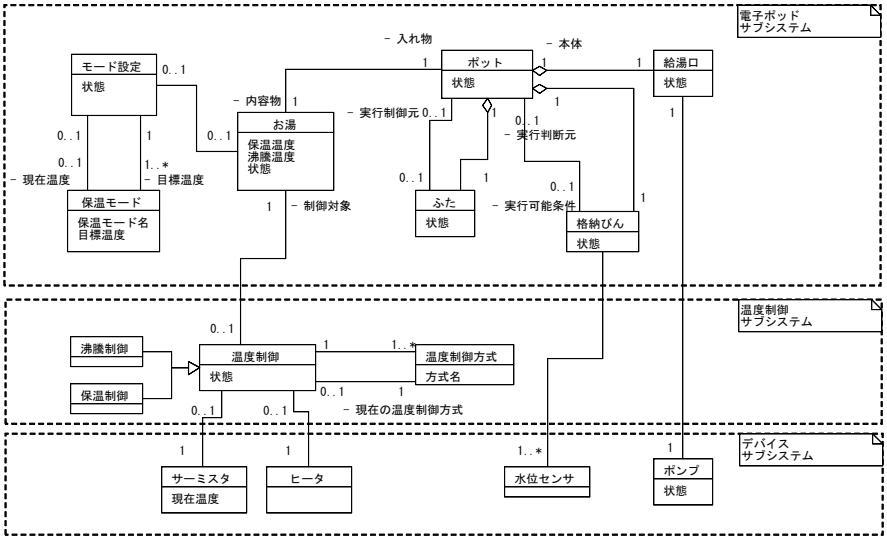

図 1 SESSAME による電気ポットのモデル（文献[2] p.149 より）
Fig. 1 The model of an electric kettle control software by SESSAME

ル上のブロック図とよばれる図に展開されると，CAD ツールは数理モデルに基づいたシミュレーションと，コードの自動生成を行う．数理モデルに基づいた検証は，一定以上の信頼性を要求される制御系組込みソフトウェアではすでに必須の開発プロセスとなっている．

もう一つは，状態機械，論理式によるモデル化とその検証技術の実用化である．これらの方法論自体は古典的な技法であるが，近年の計算機の高速化によって，実用的な速度でモデルの動作検証ができるようになってきた．この技術もまた，一定以上の信頼性を要求される組込みシステムには必須の開発プロセスになっている．

こうした背景の中で，筆者は地域の組込み技術者を対象としたモデルベース開発講座を実施してきた．その過程で，オブジェクト指向モデリングの方法論の組込みソフトウェア分野への適用技術の整理が不十分であることが分かってきた．この要因として，ハードウェアを扱う組込み特有の環境に企業情報システムの方法論を無理矢理適用することによって生じた問題（2 章で詳説）や，組込み分野特有の学際的な問題，特に，先に述べた多様なモデルベース開発の混在による問題がある（3 章で詳説）．

本研究の目的は，こうした組込み向けオブジェクト指向モデリング技法を整理し，組込みモデリングのガイドラインを示すことである．我々が開発講座で現場の技術者と議論しながら構成してきたガイドラインについて，全体を鳥瞰するアーキテクチャスタイルという形で提案する．社会人を対象とした教育カリキュラムにおける 2 つの課題に対して適用し，組込みモデリング初学者でも効果的なモデルが作成できることを具体例で示す．

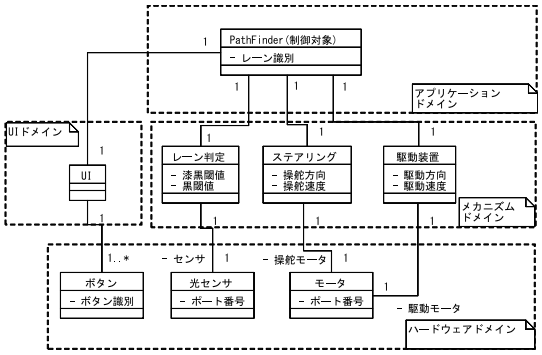

図 2 ET ロボコン実行委員会によるロボットカー制御のモデル（文献[3] p.145 より）
Fig. 2 The model of a robot car control software by the committee of ETRobot contest

2. 組込モデリング事例と議論

本章では，市販されている書籍の例を引用しながら，組込モデリングの方法論について検討し，組込モデリングの特徴と問題について整理したい．ここで利用するのは，SESSAME によるモデル[2] と ET ロボコン実行委員会[3] によるモデルである．それぞれを図 1，図 2 に示し，以後これらを参照し議論を進める．

2.1 真性実世界モデル化の是非

第一の論点は，ハードウェア，プラント（制御対象）がモデル化の対象にされていることに関するものであ

る．図 1 でいえば，デバイスサブシステムの全クラス，電子ポットサブシステム右部のポット，ふたなどの部品モデルである．実世界があるがままにモデル化されており，これは磯田のいう，「真性実世界モデル化」[4] と呼ぶことができる．こうして作られたモデルについて磯田は，実世界をシミュレートするソフトウェアに有効であるが，業務支援ソフトウェアでは役に立たないと述べている．しかし，実世界の業務を自動化するソフトウェアを開発するときには有効とも述べており，この用途について検討が必要である．

制御理論では，制御対象（プラント）はシミュレーションのためにモデル化されるが，制御システムの実装には不要である．ただし，センサの値を基にシミュレーション（積分）を行って，プラント状態の予測を行う制御システムの実装には必要である．実世界のハードウェアを状態機械としてモデル化する場合についても同様に考えることができる．ソフトウェアがオブジェクトの状態をシミュレートすることによって，なんらかの制御をする目的であれば，そのオブジェクトは必要である．ただし，その場合にも，たとえば図 1 の「ポット」クラスの状態の例が何を指すかが明らかでないように，ソフトウェアが制御の対象としない，もしくはデータとして利用しないクラスは不要である．それ以外には疑似実世界モデル化によって，たとえば図 1 に左上部の「保温モード」のように，必要なデータ構造のみクラス抽出すればよい．

2.2 機能分割の是非

第二の論点は，機能分割に関するものである．図 2 の「メカニズムドメイン」では，上位クラス（アプリケーション）の機能が分割されたクラスが確認できる．ここで言うまでもなく，オブジェクト指向設計は，抽象データ構造のモデル化が目的である．Strategy パターンなど，機能がオブジェクトとされることはあるものの，それは機能をデータとして扱うことで動的な変更が可能になるという利点を得るためであり，機能の分割統治の目的ではない．特に，制御系組込みソフトウェアでは，ストアデータは少なく，フローデータが多くなる．したがって，データを中心にクラス分析を行った場合，データは見つからず，クラス分割はできなくなる．なお，そのようなシステムを無理矢理クラス分割する必要はない．

2.3 レイヤーアーキテクチャ適用の是非

第三の論点は，レイヤーアーキテクチャの適用に関するものである．ドライバも含めた全体の構造として，レイヤアーキテクチャになることには異論はない．しかしながら，図 1 の例では，真ん中の制御層はレイヤになりきれておらず，一部飛び越して下位にアクセスしている箇所がある．図 2 の方は，機能分割をしたために，中央レイヤの凝集度が低くなっている．UI レイヤがアプリケーションレイヤの下位にあるのも気になるし，アプリケーションレイヤにクラスが一つしかないのも気になる．この構造は，機能の視点から捉えた構造化設計と考えれば理解できるが，オブジェクト指向の観点からは，その利点が活かされた設計とはいえない．

例えば，ハードウェアを通した対話型システムなのであれば，MVC(Model-View-Controller) を利用した整理が行えるはずである．制御システムであれば，パイプ&フィルタアーキテクチャを採用して，例えばローパスフィルタのようなフィルタオブジェクトは使い回すことが可能であり，そのような部品を開発し再利用性を高めることもできるはずである．

以上のように，制御，ハード，フローデータが錯綜する組込みシステムにおいて，企業情報システム系のストアデータに注目するモデリング技法をそのまま適用しようとして，混乱している例を示した．特に，「真性実世界モデル化」を問題は，海外の書籍（例えば[5]）にもみられる問題である．

3. 組込アーキテクチャの先行研究

本章では，組込モデリング技法とアーキテクチャの視点から先行研究レビューを行う．

3.1 オブジェクト指向導入以前の時代 (〜1990 年)

Gomma の 1993 年の著書[6] では，並行リアルタイムシステムの設計手法として，DFD(Data Flow Diagram)，ブロック図，モジュール構造図が主として用いられている．オブジェクト指向設計はその手法の紹介にとどまっている．組込みシステムでは，フローデータが複雑になるため，この手法は自然に見える．

3.2 Shaw と Garlan によるアーキテクチャスタイル (1994 年〜)

その後，Shaw と Garlan は，制御システムを扱うソフトウェアアーキテクチャスタイルについて検討を行っている．1995 年発表の論文[7] では，制御ソフトウェアでは，オブジェクト指向設計がベストとは限らず，制御システムフローを用いたアーキテクチャに基づくべきという結論を述べている．ここで制御システムフローとは，一般的に用いられる閉ループモデル（図 3）を指している．図 3 中において，制御プログラム (Controller) にとっての関心事は，設定された目標値 (Desired Value)，センサから得られる観測値（De-

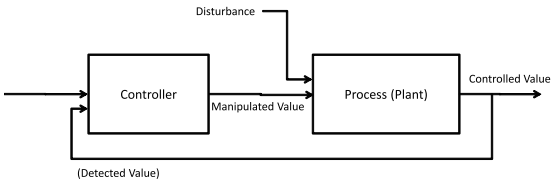

図 3 フィードバック制御の一般モデル
Fig. 3 The feedback process control loop model

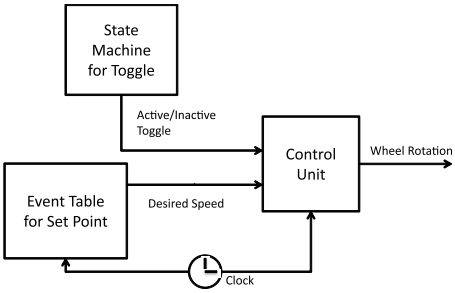

図 4 Shaw(1995) でのクルーズコントロール制御システムアーキテクチャ
Fig. 4 The architecture of a cruise control software by Shaw(1995)

tected Value），および操作量 (Maniputated Value) の算出である．操作量を制御対象（Plant）に送信し，実際の制御量 (Controlled Value) が得られる．

この結論は，アーキテクチャスタイルを提案した書籍[8] において，プロセス制御を扱うためのソフトウェアアーキテクチャとしてまとめられている．自動車のクルーズコントロールシステムの制御ソフトウェアが課題として取り上げられており，様々なアーキテクチャスタイル適用を試みた結果，プロセス制御コントロールを中心とした，図 4 に示すアーキテクチャが，データフローアーキテクチャの特別な形態として提案されている．

このモデルは，制御ユニット（ControlUnit）を中心として，そのトグルスイッチとしての状態機械 (State Machine for Toggle) と，各種のイベントに応じた目標値の設定テーブル (Event Table）をもち，これが制御ユニットに入力を行う基本モデルである．彼らはこのモデルを明示的にプロセス制御アーキテクチャとは呼んでいないが，本論文では，これを PCA（Process Control Architecture）と呼ぶことにする．

3.3 モデルベース開発・アスペクト指向の時代 (2000 年〜)

その後，2001 年に横山ら[9] は，組込みオブジェクト指向開発の方法論を提案している．この方法論ではすでに，制御システムとの連携が意識されており，制御値，目標値等を「データ値オブジェクト」という概念を用いてモデル化する．制御システムをすべてオブジェクトと見立ててモデルを作成することで，制御システムをオブジェクト指向システムを連携させて，データフローを表現させようという試みである．

続く 2005 年の論文[10] では，「ブロック図を用いたモデルベース制御開発と，オブジェクト指向分析・設計とのギャップが大きく，従来のオブジェクト指向指向開発をそのまま適用するのは困難である.」と指摘しており，制御系のモデルベース制御開発とオブジェクト指向開発を接続するフレームワークを提案している．ここで，システム全体を UML を用いたオブジェクト指向モデルで記述し，ブロック図から生成されたプログラムをカプセル化するためのクラスを生成する方法よりも，データ値オブジェクト方式のほうが再利用性が高まると主張している．

野呂ら[11] は，20 年余りの産学連携研究に基づいて，E-AoSAS++と銘打つアスペクト指向のモデリング方法論とその処理系を独自提案している．並行性, 状態遷移, 耐故障性, 実時間性, エラー処理等の組込システムにおける横断的関心事の解決についての一般解を与えるアーキテクチャの構築を提案している．

モデル化の基本的方略として，

- 並行に稼働するハードウェアを状態遷移機械でモデル化
- 主要データ構造をオブジェクト指向でモデル化
- 実時間，耐故障性等のアスペクトとの共存をモデル化

が挙げられており，組込みシステムの特徴と方法論の長所が掛け合わされている点もこの提案の特徴であり，現在は並行状態機械のモデル検証ができるところまで処理系が実用になっている．

3.4 Research Question

ここまでに，組込みシステムとオブジェクト指向の横断的関心事の問題は Shaw(1995) から引き続く問題であり，横山らは制御システムを中心としたオブジェクト指向の連携について，野呂らは並行状態機械を基礎としたオブジェクト指向の連携について議論してきた．野呂らの提案は横断的関心事のアスペクト指向による解決であり，本質的には万能の解決策である．しかしながら，万能であるが故に単一の制御システムを扱うには重厚感があることや，制約が弱いことからくる適用の難しさが特に教育現場での利用の際には問題となる．そこで，本研究では少し制約を強くして，適用しやすく，かつ近年のモデルベース開発に対応したアーキテクチャスタイルの提案を目的とする．

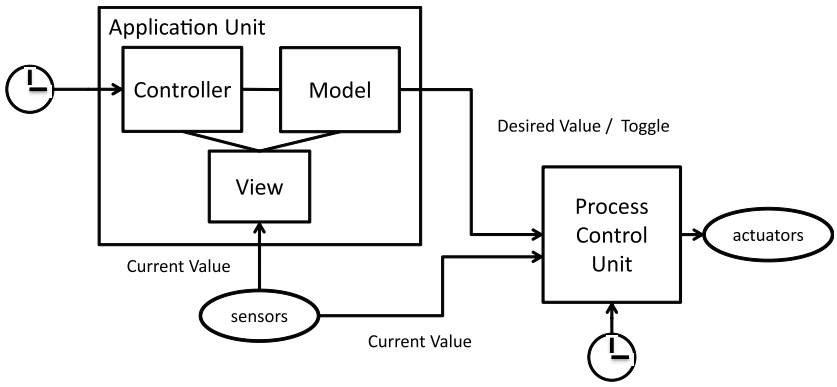

図 5 提案する MVC-PCA アーキテクチャスタイル
Fig. 5 The proposed MVC-PCA architectural style

4. アーキテクチャスタイルの提案

本章では，提案する MVC-PCA アーキテクチャスタイルについて説明を行う☆.

提案するアーキテクチャスタイルを図 5 に示す．全体構造としては Shaw(1995) の PCA モデルを採用する．制御システムは Process Control Unit(以下 PCU) を中心として sensor, actuator を通したフィードバックシステムループとしてモデル化できる．我々のアーキテクチャスタイルでは制御システムのトグルと目標値を算出するシステムとして，Application Unit(以下 AU) を PCU の入力位置に設置する．即ち，AU は制御システムに渡す目標値の算出システムとしての役割を担う.

AU の内部については MVC でモデル化する．近年のプロセッサの性能向上によって，制御系以外の組込みシステムについては，企業系の情報システムのモデリング技法の多くが再利用できると考えたためである．AU 内の MVC 各オブジェクトの責務の配布は次のようになる.

Model データ構造，目標値の保持

View センサからの情報受信，アクチュエータなど外界との boundary，センサからのフィードバックの受信

Controller アプリケーションを駆動する並行状態機械

Model はデータ構造の保持の他に状況によって変化する目標値の保持という責務を負う．これは横山モデル[10] の流用である．ハードウェアのモデル化が必要ならば View で行う．ここでのハードウェアのモデル化を許すことで，真性実世界モデル化問題への解決も効果が期待できる．Controller にアプリケーションを駆動する並行状態機械をモデル化する，ここに必要ならば Clock を入れる.

提案する PCA-MVC アーキテクチャスタイルは，オブジェクト指向，制御，状態機械の 3 種のモデルが扱え，かつシンプルなアーキテクチャスタイルであることが特徴である．制御がない情報システムでは，PCU を省略し，AU のみに通常の企業系アプリケーションと同様に MVC でモデル化すればよいし，制御系のシステムでは，AU を単一の Controller からなる状態機械として考えてもよい．制御システムや状態機械系の検査ツールとの相性もよいはずである.

5. 適 用 例

本章では，我々が実施している社会人のための組込みシステム講座[12] で作られたモデルの例を用いて，提案するアーキテクチャスタイルがどのように適用されるかを示す.

5.1 レゴ搬送車への適用

一つ目の適用例として，Lego Mindstorm NXT を利用した搬送車システムを示す．この課題は，荷物の状態などを管理する状態機械と，ライントレースの制御システムが必要であるため，提案アーキテクチャスタイルを利用することによって，前者を AU に，後者を PCU として，素直にモデル化できる．その一例として，ある受講者グループによって作られたシステムのクラス図を図 6 に示す.

View クラス群は，利用するセンサをモデル化し，ハードウェアとのインタフェイスとしている．Button は Lego Mindstorm NXT に本体についているボタンを表現するクラスであり，TouchSensor, LightSensor はクラス名と同様の Lego MindStorm 部品を表現す

☆ 実際には教育現場で議論しながらモデルを開発し，アーキテクチャスタイルとしてまとめたものであるが，論文の説明の順序の関係で，先にアーキテクチャスタイルの解説を行う.

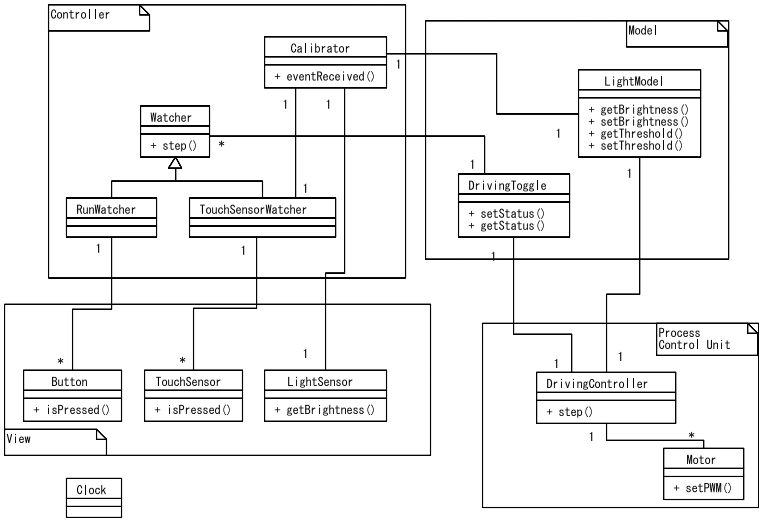

図 6 レゴ搬送車モデルへの適用例 (クラス図)

Fig. 6 An example the proposed architecture is applied for delivery system made of LEGO Mindstorms (class diagram)

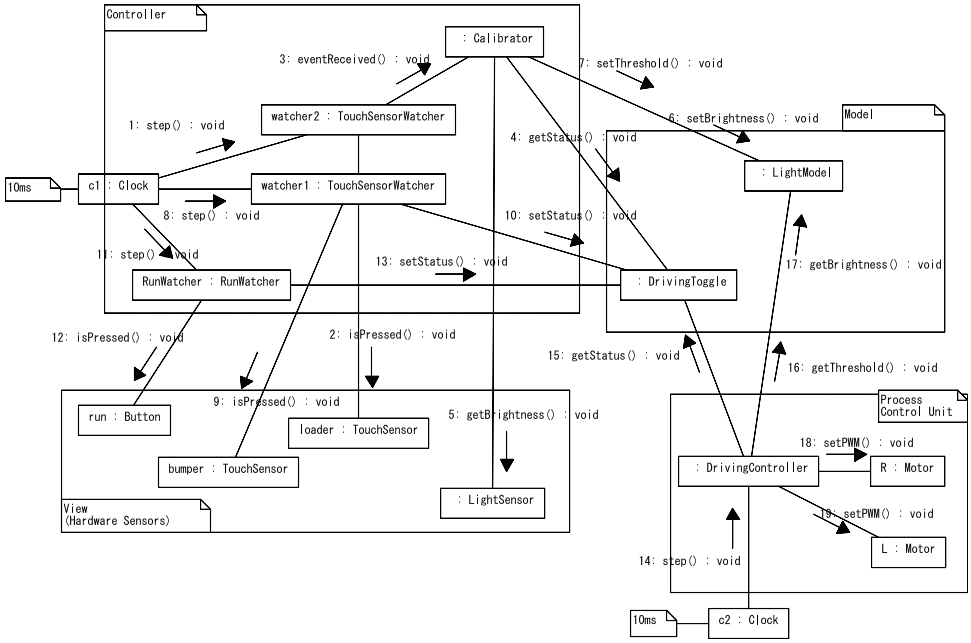

図 7 レゴ搬送車モデルへの適用例 (コミュニケーション図)

Fig. 7 An example the proposed architecture is applied for delivery system made of LEGO Mindstorms (communication diagram)

るクラスで，値を取得する操作を持っている．Controller クラス群は主にセンサを見張る Watcher 群の状態機械と光センサを見張る状態機械から構成される．Watcher 群には一定周期毎に Clock が入れられる（これは，step() 操作が呼ばれることによって実現する）．Calibrator は Watcher からのイベントを受け取って動作する状態機械である．Model クラス群では，目標値 (LightModel クラス) とトグル (DrivingToggle) をそのままモデル化している．それらには，データの設定，取得操作が配布されている．PCU においては，DrivingController クラスが AU の Model と接続し，目標値とトグルを受け取って，独自に Clock 信号を受理して Motor を駆動する．

このシステムにおけるモデリングで注目すべき点は，

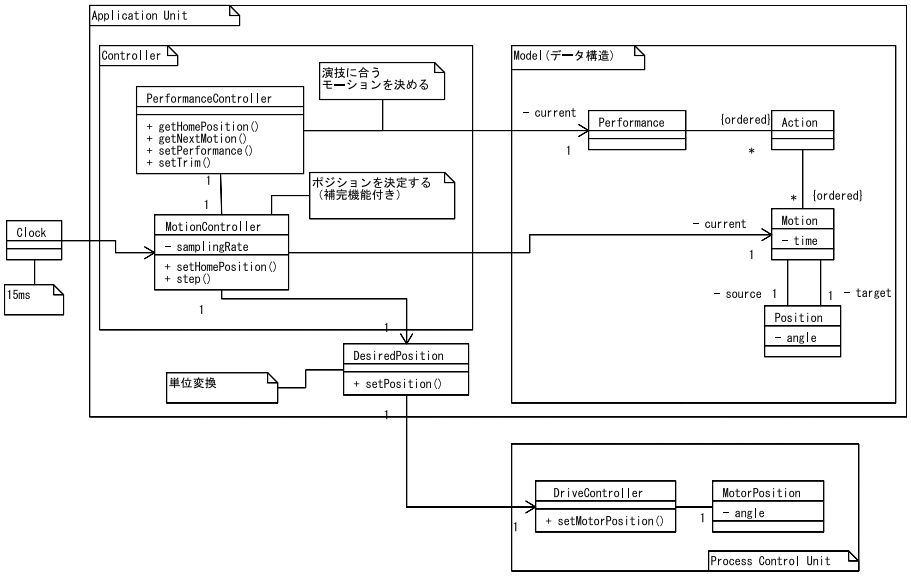

図 8　二足歩行ロボットへの適用（クラス図）
Fig. 8　An example the proposed architecture is applied for a biped walking robot system (class diagram)

保持すべきデータ構造がない点である．したがって，企業情報システム流のモデリングを試みると，「クラスが定義できない」という結果になることが予想される．その一方で，この例においては，制御システムにおいて重要なトグルと目標値が Model として抽出できており，これが制御システムから見て，目標値算出システムとして抽象化されている．AU 内は MVC であるため View に表示機器（ディスプレイなど）が追加された場合にも，MVC の利点がそのまま活かせる形になっている．

さらに，情報の流れと依存関係も整理されていることを，図 6 と同じシステムについて描かれたコミュニケーション図（図 7）を例として示す．☆．全体として，Controller と PCU が Model の情報を管理，取得する（＝ Model に依存する）形となっており，これを通じて Model, Controller, PCU の独立性が確保されていることが読み取れる．

5.2　二足歩行ロボットへの適用

二つ目の適用例として，講座の最終課題を示す．課題は，16 個のモータで構成される二足歩行ロボットに，魅力的な演技をさせるシステムの開発である．ある受講者グループによって作られたクラス図を図 8 に示す．

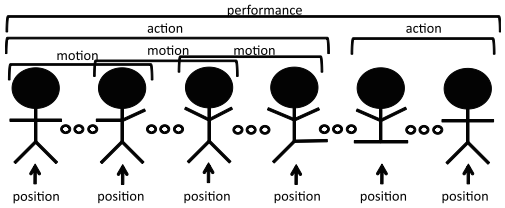

図 9　モーション概念のモデル化
Fig. 9　Modeling for Motions

この課題においては，演技データの概念モデリングをする必要がある．これは一般的な情報システムと同様の方法でできる．この例では，図 9 に示すようにモーションとポジションのデータを定義し，図 8 における Model のデータ構造として表現されている．このモーションという概念から補完計算をするのが Controller の責務になる．このシステムのポイントはフィードバックシステムではなく，時間駆動のみのシステムであることで，View としてモデル化されるセンサがない．

なお，この例においては PCU と AU でポジションの単位系が異なるように設計してしまったため，DesiredPosition クラスで単位変換する必要性が生じている．PCU と AU で独立にデータ構造を持てることの欠点ではあるが，制御システムを独立させる利点のほうが大きく，例で示したように，目標値クラスへの変換規則のカプセル化で局所的に解決できる．

☆ なお，データフローが中心の組込モデルに関しては，クラス図よりもコミュニケーション図が読みやすい（データフローが表現できるため）したがって，講座ではコミュニケーション図とクラス図の併用を推奨している．

6. 考　　察

オブジェクト指向，制御，状態機械の 3 種のモデルが容易に扱えることが提案するアーキテクチャスタイルの目的であった．この目的に関しては，5 章で提示した 2 つの例によって，少なくとも教育現場で扱う課題のレベルでは，3 種の側面を持つモデルが整理されることが実証された．アーキテクチャを構築した学習者はモデリング初学者であったことから，理解容易性が高いことも示唆された．

2 章で提示したロボット車の課題（図 2）は，5 章適用例の一つ目で取り上げた課題（図 6）と類似したロボット車の問題であり，双方共設計レベルの記述であるので比較が可能である．提案アーキテクチャスタイル適用モデル（図 6）においては，2.1 節で述べた「真性実世界モデル化」の問題について，ハードウェアのモデリング対象と目的を明確にすることで解決している．2.2 節で述べた「機能分割」の問題については，フローデータを処理する制御オブジェクトを (PCU として) 分離して，状態機械を持つクラス (Controller) と，データを持つクラス (Model) を分離した結果，データ・状態に注目した責務分離が出来ている．最後に，2.3 節で述べた「レイヤー化の不自然さ」の問題についても，凝集度が高すぎるアプリケーションや UI クラスなどは存在せず，依存の方向についても MVC の利点を活かす形に秩序が維持されている．

オブジェクト指向，制御，状態機械の 3 種のモデルが扱えるため，モデルベース開発ツールとの相性はよいはずである．しかしながら，5 章で示した例においては，ツールを使ってコード生成したわけではなく，すべてハンドコーディングでの動作確認である．ソース生成ツール適用との相性の確認が今後の課題である．加えて，アーキテクチャ適用範囲とトレードオフポイントの明示も今後の課題である．

7. ま　と　め

本研究の目的は，組込みシステムにおける，オブジェクト指向，制御，状態機械の 3 種モデルベース開発に対応した，モデリング技法のガイドラインを示すことであった．まず，既存の組込みモデルを分析し，真性実世界モデル化，機能分割，レイヤ化などの論点があることを指摘した．次に組込みシステムとオブジェクト指向の横断的関心事の問題は Shaw(1995) から引き続く問題であることを指摘し，横山らや野呂らなどによる議論のレビューを行った．その議論に基づいて PCA と MVC の複合アーキテクチャスタイルを用いたモデリング技法を提案した．目標値の保持を Model に，センサからのフィードバックの受信を View，並行状態機械を Controller として，責務の整理を行った．社会人を対象とした教育カリキュラムにおける 2 つの課題に対して適用した結果，オブジェクト指向，制御，状態機械の 3 種のモデルが素直に扱えて，理解容易性が高く，モデリングがしやすいアーキテクチャスタイルであることを確認した．

参　考　文　献

1) 土樋祐希, 上江洲吉美, 北井翼, 田村純一, 樋口博史, 藤本英基, 細田健人: モデル駆動開発 (MDD) を ET ロボコンで実践, NIKKEI ELECTRONICS 5 月号, pp. 81–109 (2009).
2) SESSAME WG2: 組込みソフトウェア開発のためのオブジェクト指向モデリング, 翔泳社 (2006).
3) ET ロボコン実行委員会: ロボットレースによる組込み技術者養成講座, 毎日コミュニケーションズ (2008).
4) 磯田定宏: 実世界モデル化有害論 : オブジェクト指向モデル化技法の解明, 電子情報通信学会論文誌. D-I, 情報・システム, I-情報処理, Vol. 83, No. 9, pp. 946–959 (2000).
5) Douglass, B.: *Real-Time UML: Developing Efficient Objects for Embedded Systems, Second Edition*, Addison Wesley Longman, Inc (2000).
6) Hassan Gomaa: *Software Design Methods for Concurrent and Real-Time Systems*, Addison-Wesley (1993).
7) Shaw, M.: Beyond Objects: A Software Design Paradigm Based on Process Control (1995).
8) M.Shaw and D.Garlan: *Software Architecture - Perspective on an Emerging Discipline*, Prentice Hall (1996).
9) 横山孝典, 納谷英光, 成沢文雄, 倉垣智, 永浦渉, 今井崇明, 鈴木昭二: 組込み制御システムのための時間駆動オブジェクト指向ソフトウェア開発法, 電子情報通信学会論文誌. D-I, 情報・システム, I-情報処理, Vol. 84, No. 4, pp. 338–349 (2001).
10) 吉村健太郎, 宮崎泰三, 横山孝典: オブジェクト指向組み込み制御システムのモデルベース開発法 (分析・設計技法), 情報処理学会論文誌, Vol. 46, No. 6, pp. 1436–1446 (2005).
11) M.Noro, A.Sawada, Y.Hachisu and M.Banno: E-AoSAS++ and its Software Development Environment, *Proc of APSEC2007*, pp. 206–213 (2007).
12) Matsuzawa, Y., Noguchi, Y., Mori, T., Shima, S. and Shiomi, A.: ESAD: An Intensive Retreat Program for Embedded System Architect Developing, *Proc of APSEC2010*, pp. 90–97 (2010).