MDA によるリアルタイム制御機器の開発手法、及び UML を利用したオブジェクト指向開発手法について

日本工営パワー・システムズ株式会社
システム・ソリューション統轄部 新規事業グループ

技術課長 石田 哲史・技師 近藤 正俊

1．MDA に基づく組み込み制御システム開発

(1) MDA 開発における現状

MDA（モデル駆動アーキテクチャ）とは OMG（Object Management Group）がソフトウエア全ライフサイクルを通じてソフトウエアと開発モデルを高度に同期させる手法として 2000 年に紹介されました。その手法の根幹においては **図 1-1** に示すようにオブジェクト指向による分析・設計を導入する企業の増加によって、JAVA などのオブジェクト指向開発環境が充実し、さらにはネットワーク技術との融合によって CORBA による動作プラットホームを意識せず開発することが可能な環境が整備されたことにあります。

企業のビジネス情報システムの多くは、膨大なソースコードから構築され、ソフトウエア部品を利用した開発が必須でした。そのため、生産性改善を目的としてオブジェクト指向開発が普及しました。

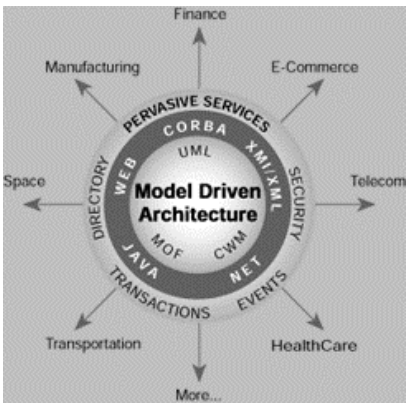

図 1-1 MDA のアーキテクチャ(OMG のロゴ)

ビジネス情報システムは、組み込みシステムのコード量の数十倍にも及ぶため、過去の資産を新しいアーキテクチャへ移行する過程では、さまざまな試行錯誤が繰り返され、レガシーシステムからサービス（機能）単位に XML+J2EE へ移行させ、新しいアーキテクチャへの移行を MDA を利用して進めています。

一方、組み込みシステムはソフトウエアの動作環境（メモリサイズ・CPU 性

能)や開発に投入できる費用の問題から、OS 搭載やオブジェクト指向言語（C++・JAVA）環境の搭載すら実施されていないのが現状です。

理由としての多くは、処理性能が低いことや、組み込み CPU に対応した開発環境が整っていないこと、ハードウエアが PLC などのようにメーカ指定の環境しか利用できないことなどの制約条件によります。

近年は動作周波数 100MHz 以上の RISC CPU が主流となりつつあり、SDRAM も大容量で低価格なものが普及し始めました。これにより、OS 搭載が現実的のもとなりつつあります。さらに、開発環境においても C++ コンパイラが CPU メーカから提供され始めています。

組み込み装置を取り巻く環境もネットワーク・インフラの普及により“オンライン”から“ユビキタス”へと変遷しており、常時ネットワークと接続し装置機能を自律したサービスとして提供することが次世代への必要条件となっています。

今までは“非現実！！” “どこかがやり始めたら・・” “動作環境が？”などという理由をつけ、オブジェクト指向開発の導入が遅れていましたが、最近はネットワーク協調連係機能が多く含まれてきており、開発コード量も飛躍的に増加し、オブジェクト指向を代表とした部品化開発が必要となってきています。

(2) PIM と PSM モデル

MDA はモデル駆動型のシステム開発をソフトウエア全ライフサイクルに適用することにより、ソフトウエア開発の効率化と信頼性を向上させる目的で OMG を中心として進められています。特に、UML 2.0 の仕様においては MDA を前提にして仕様がまとめられつつあります。

MDA のモデルには PIM（Platform Independent Model）と PSM（Platform Specific Model）の 2 つのモデルが定義されています。PIM とは特定の実装環境に依存しないでシステムをモデル化する方法を示し、MDA で述べられている 20 年後も動くシステムを実現するための基本となっています。PSM は、PIM の実行環境（プラットホーム）への実装に関するインターフェイスをモデル化する方法を示しています。

そして、組み込みシステムのリアルタイム制御に関しても、MDA 技術が必要であると、UML Forum/Tokyo 2002 にて米アイロジックスのチーフエバンジェリスト，ブルース・パウエル・ダグラス氏が以下のように語っています。

「リアルタイム／組み込みシステムの構築において最も重要なのは，プラットフォームに依存しないモデル PIM を採用することだ」、「PIM を利用することで，企業システムの効果的な IT 化が可能になる。これにより長期的な IT の活用が可能になるだけでなく，再利用性

も高く，費用対効果（ROI）の向上も期待できる」。

先に述べたように、ビジネス情報システムにおいては PIM ＝ UML ＋ XML(JAVA) でモデリングし、PSM＝ J2EE+CORBA で実装をすることで MDA のモデルを具現化する方法が確立し始めています。PIM と PSM の関係は **図 1-2** のような関係にあり、PIM と PSM のインターフェイスを確立することで、モデルと実装ソフトウエアとを同期させるため、MDA では必ず両方のモデルが必要となります。

(3) 組み込み開発への MDA 導入の課題

・非プラットホーム依存のモデリング

組み込み環境における設計方法は様々ですが、実装が複雑であることにより、従来はモデリングよりも実装に重点が置かれ、モデリングツールよりも製作・デバッグツールの方が多く利用されています。そのため、モデリングの分析・設計があいまいで、設計情報が実装の段階で乖離してしまうケースが多くありしました。さらに、業務（ビジネス）モデルの抽象化定義に適した XML のように、リアルタイムで変化する振る舞いと制御機構をモデル化する手法がありませんでした。

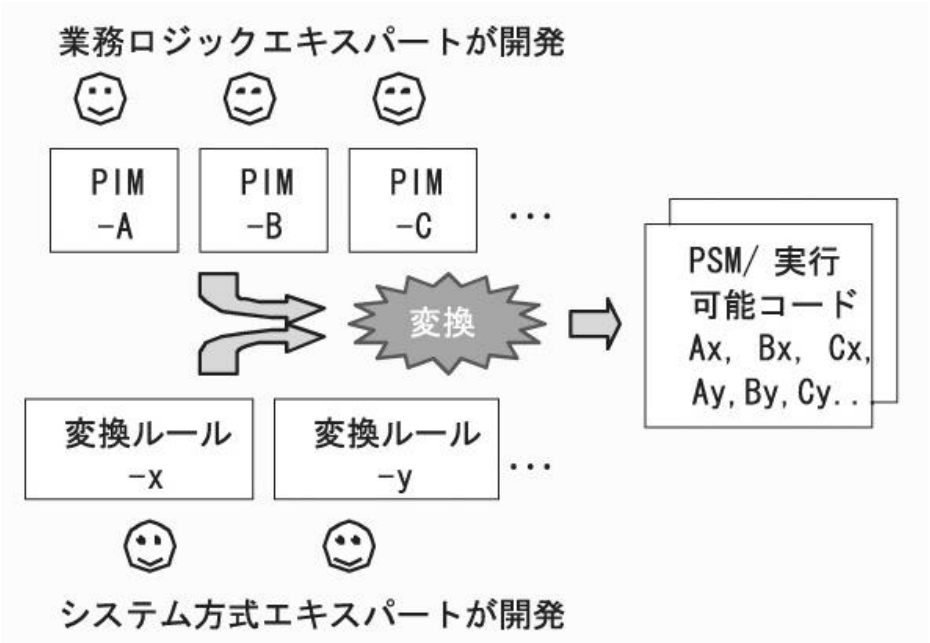

図 1-2　PIM と PSM による実装イメージ
（http://www.bcm.co.jp/mycareer.htmlUML 基礎と応用―第 17 回―より引用）

・ＰSM への対応

組み込み開発環境は、CPU やハードウエアの構成が様々で Windows PC のような標準的なハードウエア構成がないため、PIM から PSM への展開をモデル化することが難しい状況でした。しかしながら、OS＋TCP I/IP の実装が一般的になりつつあり、組込みシステムの実装のモデル化が近年は可能になりつつあります。

(4) MDA 開発支援環境 DVC fai（Φ）

上記 (3) の課題を踏まえた DVCfai（Φ）の MDA イメージを示します。

DVCfai（Φ）モデリングの特徴は、**図 1-3** に示す通り構造化（C インターフェイス）モデルによる開発も MDA に組み入れていることを特徴としています。これにより、開発システムに必要な要件や環境に合わせたモデリングを可能とします。また、PIM においては状態遷移表を採用しており、リアルタイム制御の振る舞い（動的）を厳密に定義可能としています。さらに制御 I/O などの制御定義用の PIM として日本語で定義可能な制御仕様記述言語（**図 1-4**）により、組み込み制御システムを明確にモデリングすることを可能としました。オブジェクト指向開発においても、PIM に UML ツールを利用可能とし、ツールでモデリングしたクラス情報からプログラム動作特性に合わせリアルタイム処理部と業務処理部に分割したモデリングを可能とします。

PSM は **図 1-4** のアーキテクチャにより、OS＋Dvc-Ware によりプラットホームに依存しないインターフェイスを PIM に提供します。さらに PSM

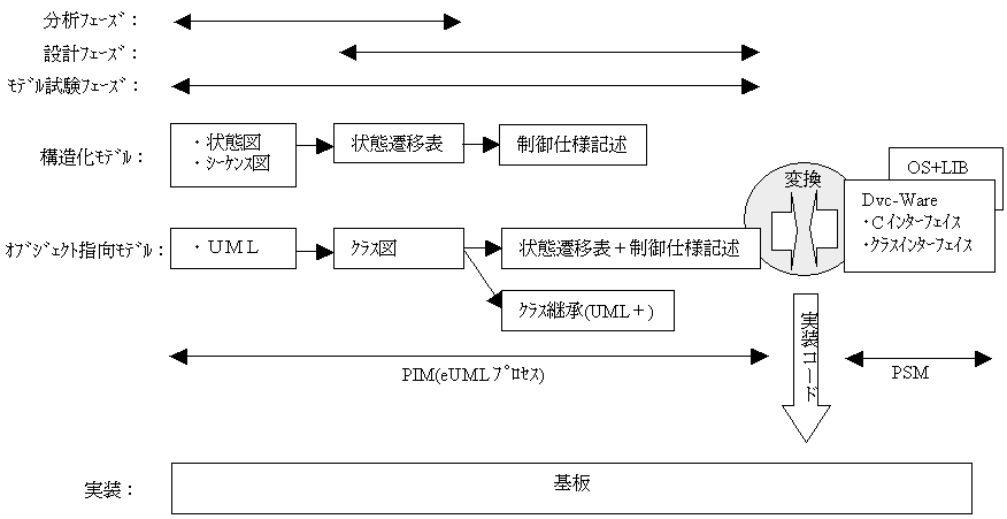

図 1-3　DVCfai（Φ） の MDA イメージ

は制御システム構築時において、ネットワークによる分散（ユビキタス）構成を可能とするため、CORBA 同様のオブジェクトインターフェイスを用意しています。

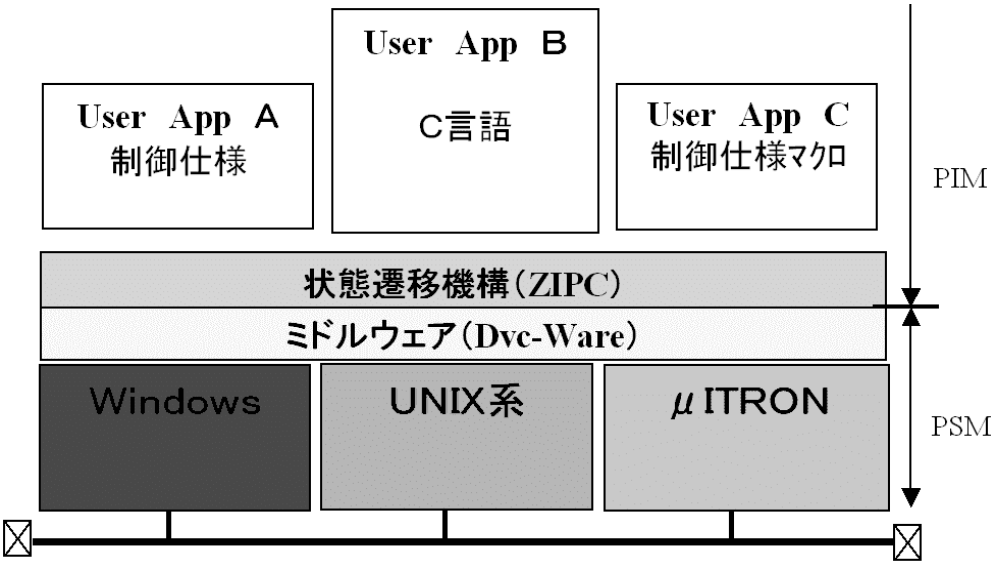

図 1-4　DVCfai（Φ）アーキテクチャ

2. PIM（ZIPC）＋PSM（Dvc-Ware）半導体製造装置通信プロトコルの開発例

(1) 半導体製造装置通信プロトコルとは

半導体製造装置と製造ラインを監視・制御・情報収集するためのホストとの情報交換手順を規定したもので、半導体製造業界の世界標準策定機構（SEMI）にてリリースされています。プロトコル構造の概要は **図 2-1** のようになっており、今回は斜線部（データリンク層）レベルのプロトコル、SECS-1 を ZIPC と Dvc-Ware によって実装を行いました。

(2) SECS-1 仕様書

図 2-2 に SECS-1 の SEMI 仕様書を示します。

(3) 状態遷移表（PIM）とミドルウエア（PSM ）によるモデリング

図 2-3 はモデリングした結果です。**図 2-2** に比べ処理仕様の定義がより明確になっていることがお判り頂けると思います。さらに、PSM インターフェイス（関数）を PIM （状態遷移表）に貼り付けることで、モデリングの段階から実装と同期させることができます。その結果、ZIPC の持つ C ソースコード生成機能により SECS-1 プロトコルを実装コードとして生成しています。

参考までに今回製作したプロトコル部分については、モデリングに約 4 日、製作・試験を 3 日で実施しており、RUP の成果が十分に確認されました。また、試験中に発生した不具合に関しても、PIM の修正から実施することで、常にモデルと実装コードが同期し、ソフトウエアの全ライフサイクルを通じたモデル開発が可能となることが確認できました。

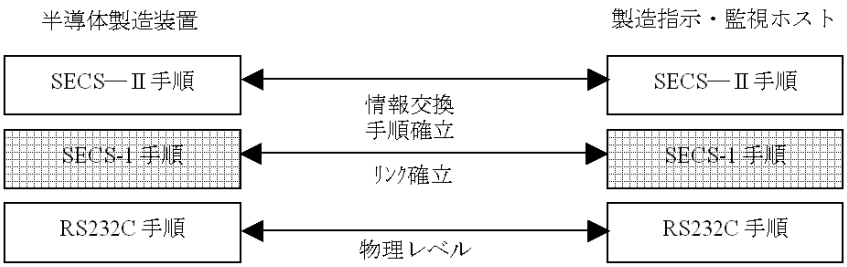

図 2-1　SEMI 規格のプロトコルスタック

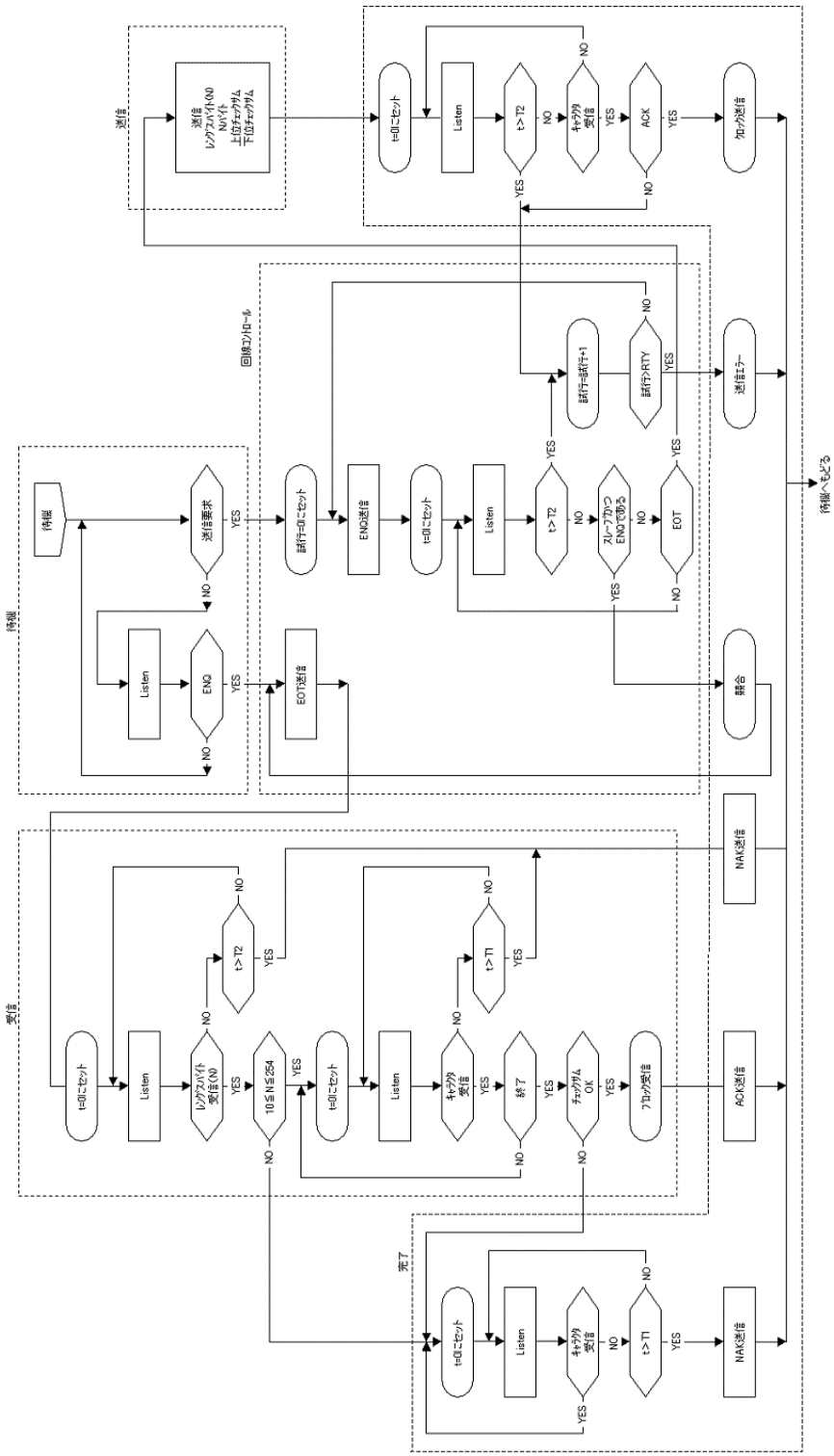

図 2-2　SECS-1 プロトコル仕様（SEMI E4-0699）

図 2-3　SECS-1 のモデリング結果

３．組込み制御システムのオブジェクト指向開発環境

(1) 概要

システム構造を抽象化して、統一した表記方法で記述するための言語であるUML(Unified Modeling Language)は、ソフトウェア開発分野への適用によって開発効率の改善性が期待されることから、その必要性が高く求められています。
オブジェクト指向によってシステム構造を出来る限り抽象化して、対象のシステム領域を単純化して表現できるため、全体の理解が深まり、手戻り削減や再利用性が向上します。

組込みシステム開発環境への UML の利用要求も高まっており、そのためのガイドラインとして、eUML が策定されています。eUML は組込みシステム向けに UML を応用拡張したもので、開発プロセスをモデリング（分析）・設計・アーキテクト／メカニズム設計・テストの４つの役割に分類し、それぞれに必要な作業プロセスを規定しています。

ここでは、組込みシステムにおけるオブジェクト指向開発環境とその有効性を理解して頂く目的で、簡単な組込みシステムの開発を例にして、特に分析と設計フェーズでの作業をご紹介します。簡単な事例を参考にして UML によるシステム開発を論ずる、という資料は多く見掛けるのですが、少し趣きを異にしてお話しを致します。ハードウェア資源の制約を多く受ける組込みシステムの開発環境において、どのようにしてオブジェクト指向開発を進めていくことが有効であるのかを焦点としたいと思います。

(2) 開発システムの概要

図 3-1 のような装置（デバイス部）と PC（HMI 部）を用いて、PC 画面からの設定により PLC の位置決め制御ユニットからサーボモータを制御することで、以下のように １軸の位置決め動作をするシステムを開発します。

図 3-1　位置決め装置概観図

＜＜装置概要＞＞

- オペレータが PC で動作距離と動作回数を設定する
- 装置の低速／高速ボタンを押下する
- 装置は設定された速度にて動作距離と動作回数だけサーボモータを動作させる
- 設定した距離だけ動作すると、装置は位置決め完了として動作方向を反転させる
- 一往復するたびに、その回数をデジタルメーターがカウントする

(3) 分析

まず、システムの外部仕様を明らかにします。システムの外部から見える機能をユース･ケースとし､その機能へアクセスするアクタを設定してユース･ケース図を作成したものが 図 3-2 です。

また、この装置の状態遷移図は図 3-3 のようになります。システムが外部からイベントを受け付け、その際のシステムの振る舞いを定義します。これにより、イベントによって装置がどのような状態に遷移していくかが明確になります。

これまでで要求分析が整いました。UML はオブジェクト指向に基づいて記述するため、次にオブジェクト構造の分析を行ないます。オブジェクトは対象領域から名詞を抽出する名詞抽出法が良く用いられますが､そこで 1つの工夫するポイントとしてシステムのメカ部品に着目をして抽出を行ないます。これによって、クラスを生成させる際に部品化が比較的容易に行なうことができます。

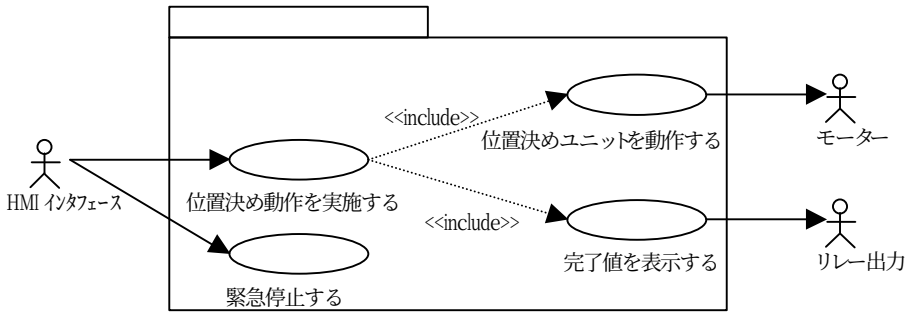

図 3-2　ユース・ケース図

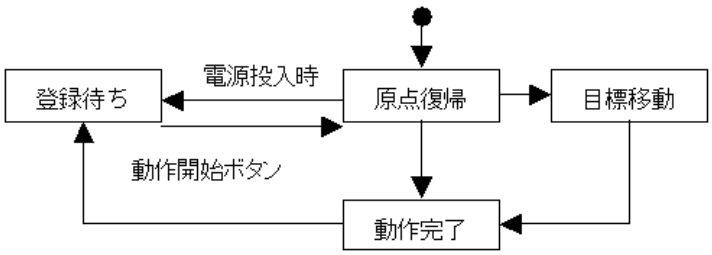

図 3-3　装置状態遷移図

この分析を基に、UML で最も多用されるダイアグラムであるクラス図を作成することとします。各クラスを生成する際、どのようなステレオタイプがあるかを意識しながら、各オブジェクトをクラス化していきました。**図 3-4** に本システムの概念モデルのクラス図を記します。

[コントローラ]―[個別タスク]―[機能ユニット]―[メカ部品]といった階層構造を意識して記述していることがお分かり頂けると思います。

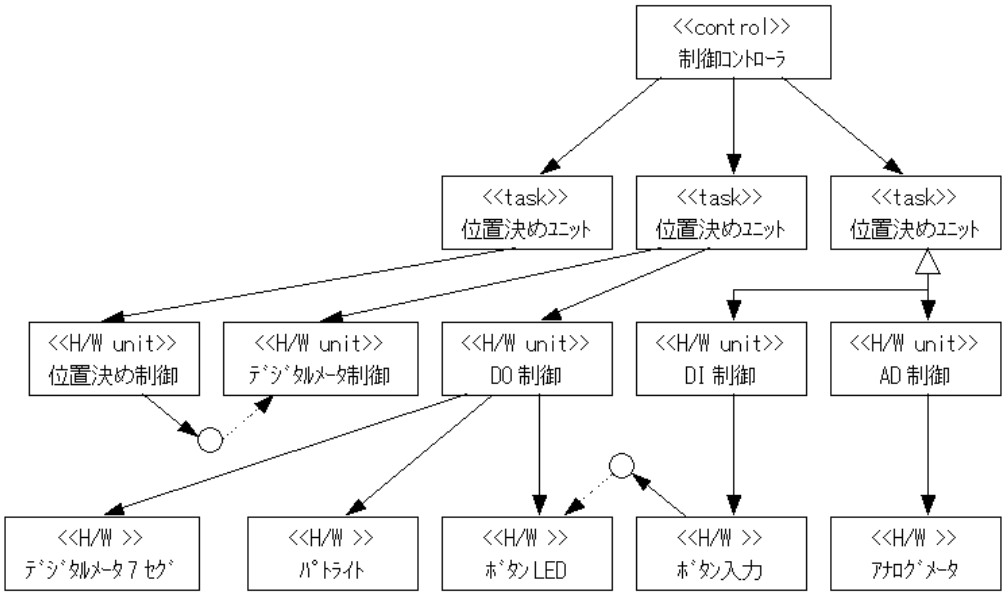

図 3-4　概念モデルクラス図

(4) 組込みシステムにおけるオブジェクト指向開発について

さて、ここからが重要な点です。PLC・開発ボードなどの組込み環境においては、Java や C++ などオブジェクト指向環境が実装できないケースが多いはずです。その際はどうすれば良いでしょうか。最近では携帯電話にも Java 環境が実装されるなど環境は整いつつありますが、組込みシステム全体の現状では、以下の理由によりオブジェクト指向環境の利用が困難であると思います。

- リアルタイム性・高速制御制御が求められる場合には、処理性能を厳密に規定できなくてはならない
- H/W や OS 資源の制約があることが多く、実装サイズが大きくなってしまうため取り込めない
- オブジェクト指向によって開発効率が向上するための具体的なツールや方策が示されていない

この解決策として、組込みシステム開発環境においては ZIPC for DVC の利用による状態遷移表と、DVC fai(Φ)による産業用ミドルウェアでの開発が有効となります。これらをうまく利用することで、MDA による開発が可能になります。

(5) ZIPC for DVC による状態遷移表の利用

オブジェクト指向によって組込みシステムを開発する際に、ZIPC for DVC が提供する状態遷移表を利用することのメリットを **表 3-1** にまとめます。

表 3-1　ZIPC for DVC が提供する状態遷移表によるメリット

利点	内容
拡張性	オブジェクト指向が利用できない環境にも部品化・再利用性に優れたシステム開発が適用できる
保守性	処理の呼び出し操作を明確化し、各セル間の関係を疎結合化することができる
信頼性	動的に変化するターゲットマシンの動作処理内容を状態遷移表によって厳密に定義できるため、シーケンス図やビヘイビア分析では発見が困難な、処理モレ・抜けを抑制できるプログラムコードを自動生成することで、人的ミスなく個人差のないプログラムが開発できる
機能性	PC と連携する場合等は、PC と組込み基板間の処理の分担ができる

状態遷移表の割り込み事象（表の行名の部分に相当）には、処理タイミングを意識した厳密な規定が可能ですので、リアルタイム性や高速制御が求められるような開発の場合にも、組込み装置の H/W・OS 環境と処理性能を切り離した設計が可能となります。また、状態遷移表の各セル内には、ST 言語に基づく制御仕様記述言語＊で処理内容を記述することで C ソースプログラムを自動生成することが可能となり、人的ミスや個人差のないプログラムが利用できます。

組込みシステムの開発環境として定評のある ZIPC は、既に大きな実績を上げており、理想的な PIM 環境を構成します。

＊ Structured Text 言語....IEC61131-3 にて規定されている Pascal ライクな構造化設計言語

(6) DVC fai(Φ) による産業用ミドルウェアの利用

先に述べた状態遷移表を利用することのメリットは、DVC fai(Φ) が提供する産業用ミドルウェア:DVC-Ware を組み合わせることで、更に飛躍的に高まります。プラットフォームの実装環境に依存しない PSM 環境となる、DVC-Ware を利用することの主なメリットは以下のとおりです。

- H/W や OS 操作をミドルウェアが仲介するため、ハードウェア依存を抑制し、メモリ等の資源の不正使用を防止できる
- OS や開発言語などとアプリケーションの関係を疎にすることができる
- プログラムの移植性が向上する

これらミドルウェア利用のメリットは、各種 I/O や通信プロトコルスタック、ファイル I/O 等のライブラリ群を豊富に兼ね備えることで、初めて実現します。また、ライブラリ群はターゲットの組込みシステム環境に応じて取捨選択しながら実装することで、必要最小限な構成も可能となります。

(7) 設計の例

UML ツールの多くは、クラス図から自動的にオブジェクト指向に準拠した言語によるプログラムコードを発行するものがあります。

これと連携しながら、デバイスに特化した処理やリアルタイム性が必要な処理などの部分のクラスについては、ZIPC for DVC を利用します。状態遷移表にて処理性能を意識しながら、プラットフォームに依存しない開発を行ないます。これによって、装置開発と処理機能の開発を分けて行なうことが可能となります。**図 3-5** に状態遷移表の例（ZIPC のプロジェクト画面を一部抜粋）を記します。UML のツールから生成された Java プログラム

（PC 上で HMI 処理）と、組込み環境上に実装した産業用ミドルウェアが情報連携してシステムを構成します。

これらによって、従来のウォーターフォールモデルによる開発から、ラピッド開発環境によるインクリメンタルアプローチが可能になります。UML による開発の利点として、仕様策定の早い段階からシステム開発者間での理解が深まる点が挙げられますが、この点からも状態遷移表を用いた開発が有効である、というわけです。

また産業用ミドルウェアを利用することで、システム全体をオブジェクト指向によって開発し、必要なクラスの部分だけを抽出して組み込み装置上に実装する設計を行なうことができ、将来的にオブジェクト指向環境が提供された時点で、部品化したクラスを移植していくことも可能となります。

図 3-5　状態遷移表の例

(8) まとめ

状態遷移表は、UML の正式なダイアグラムにはなっていないものの、組込みシステム開発環境においては、非常に重要だと考えます。それは、装置の様々な状態に応じて発生し得る事象をすべて網羅して、信頼性の高いシステムを構築できることのメリットのみならず、開発者間のコミュニケートのツールとして利用できるからです。また、ZIPC for DVC の持つ以下のコア機能によって、組込みシステム開発においてもオブジェクト指向を適用し、開発効率を向上させることができることを実現性の高いレベルで確認することができました。

- 状態遷移表の各処理内容を制御仕様記述言語で記述することによる C ソースプログラム自動生成機能
- 産業用ミドルウェアによる OS やハードウェアに依存しないソフトウェア設計実現機能

今後、システム開発は更に短納期・かつ低コストでの開発が求められることが予想されます。冒頭で述べたとおり、その必要性から近い将来オブジェクト指向による開発が当然の時代が来ると予想しています。