Vol.2015-SE-188 No.2
Vol.2015-EMB-37 No.2
2015/6/4

モデルカタログを用いたモデル駆動開発の実践

菊池雄太郎[†1]　　力武克彰[†1]

モデル駆動開発(MDD:Model Driven Development)は組込みシステム開発の手法として注目されている．しかし，MDD が求める点として，再利用性や拡張性に富んだモデルを作り出すことが挙げられる．再利用性，拡張性に富んだモデルは，モデルの記法を学んだだけでは作れるものではなく，訓練と経験が必要である．訓練や経験が少なくても，優れたモデルを作成するための参考として UMTP によってモデルカタログが作成されている．モデルカタログとは，組込みシステムの機能や製品ごとに UML モデルを記してあるモデル集である．それを使えば，モデル製作の未経験者であっても再利用性，拡張性に優れたモデルを作ることができる．しかしながら，モデルカタログではプラットフォームに依存しない，抽象的な段階のモデルまでしか提供していない．そこで，本研究では事前にベルトコンベアによる組込みシステムのハードウェアを作り，モデルカタログのインライン装置の項目を参考に，PSM を設計した．実際に装置を作り，それに合わせて設計することで，モデルカタログの有用性を示す．

The practicing Model Driven Development with Model catalog

YUTARO KIKUCHI[†1]　RIKITAKE YOSHIAKI[†1]

The purpose of this work is providing a practical example of Model Driven Development (MDD) using Model catalog. A model catalog, which was published by UMTP, is a collection of model has reusability and expandability for embedded systems.// In MDD processes, one have to design a platform independent model (PIM) for a system and then design a platform specific model (PSM). In this work, we applied the MDD process for the development of a color sorter. When designing the PIM of the color sorter, we intended to refer a model provided in the Model catalog. We discuss the effectiveness of using Model catalog in a development of embedded system.

1. はじめに

組込みシステムの開発手法の一つとして，モデル駆動開発（Model Driven Development）[1]が知られている．モデル駆動開発とは，主に Unified Model Language（UML 記法）などに代表されるモデル記述言語または，形式によって決められたモデルを利用した開発である．MDD の利用により，現在開発しているソフトウェアの全体を把握することが容易になる．さらに，MDD はチームでソフトウェアを開発することにあたってもメリットがある．ソフトウェアの構造をモデルにして表すことは，開発の方針やメンバー間との意思の共有が容易になることが期待できる．MDD の特徴の 1 つとしては，実際の開発で使うモデルには，拡張性や再利用性が必要とされるということである．拡張性とは，既存の機能に加えて新たな機能を付け足すことができる特性のことを指す．再利用性とは，プログラミング言語やハードウェア環境など様々な開発環境に対応できる特性を指す．この 2 つの特性を満たすモデルを生み出すには，訓練と経験が求められ，モデルの描き方を学んだだけでは作成できるものではない．

モデリング技術を学んだばかりであっても，拡張性，再利用性があるモデルを作成するには，モデルのお手本が必要である．お手本となるモデルがあれば，それを基にモデルを作成することができ，モデリングへの敷居が下がると考えられる．

作成するモデルのお手本として，モデルカタログ[2]というものを利用する．モデルカタログとは組込みシステム開発に役立つ UML モデルを提供するモデル集である．モデルカタログを用いた MDD は，拡張性や再利用性に富んだ優れたモデルによる開発を容易にする．しかし，モデルカタログの認知度は低く，モデルカタログを用いたシステム開発の事例は数少ない．

こうした背景から，モデルカタログによる組み込みシステムの開発事例を提供する研究が進められている[3]．しかし，開発事例の数はまだ多いとは言えず，新たな種類の事例を増やすべきである．

本研究は，モデルカタログを参考にして，カラーソーターの開発を行う．ハードウェアの実装には LEGO MINDSTORMS EV3 を用いる．カラーソーターの開発は MDD による開発である．よって本研究において，モデルカタログをモデル駆動開発に適用した実践例を示す．

2. モデル駆動開発

1.1. 概要

モデル駆動開発（以下 MDD と示す）とは 2001 年に Object Management Group が公式に発表したソフトウェア設計手法である[4]．UML などのモデル記述言語を用いてモデル

* †1 国立仙台高等専門学校
Sendai National College of Technology

を作成し，それを設計の中心として開発を進めていく．

MDD を適用することのメリットとしては，モデルを使うことでソフトウェアの構造を視覚的に把握することが見込める．さらにモデルを開発チーム内で共有することにより,開発の方針やメンバー間との意思の伝達も容易になる．また，あらかじめモデルを利用してシステムのシミュレーションを行うことができ，効率的な開発が見込めるといった点があげられる．

以下に MDD による開発手順を示す．

(1) **要求分析・分析モデル作成**

開発するシステムの要求の分析を行う．そして分析の結果をモデルで表し，仕様の視覚化を図る．本研究では具体的に，ユーザや外部機器などのアクターとの関連をユースケース図に直し,ケースごとのシナリオを設定した.また，開発対象の振る舞いを設計し,ステートマシン図に起した．

(2) **PIM 作成**

PIM は Platform Independent Model の略称である．プラットフォームから独立したモデルを指し，システムの処理方式にのみ着目したモデルである．複数のプラットフォームに適用できる形である必要がある．

前段階で作成した分析モデルを基に，プラットフォーム（ここでは開発に使用する言語や，LEGO MINDSTORMS EV3 のことである）に依存しないモデルを作成する．本研究では，開発するソフトウェアのクラス図を作り，操作や属性をプラットフォームに依存しない範囲で設定していく．

(3) **PSM 作成**

前段階で作成した PIM を基に,プラットフォームを考えたモデルを作成する．そのモデルを PSM と呼ぶ．PSM は Platform Specific Model の略称であり，プラットフォームに特化したモデルを指す．本研究では，PIM のクラス図を基に，開発環境に適したクラス図を作成する．

PSM を作成するにあたって，システムの処理方式だけではなく，開発言語に適した形に直したり，ハードウェアとの関連を定義したりする必要がある．

(4) **実装**

前段階で作成した PSM を基に，Java ファイルのプログラムを作成する．本研究では，ツールで PSM から，メソッドの内容やコンストラクタを定義していないスケルトンコードを生成し，コンストラクタやメソッドの内容を実装していく．

1.2. モデル駆動開発に必要な点

実際の MMD を用いたシステム開発には，拡張性，再利用性に富んだ優れたモデルが求められる．そして優れたモデルを作成するには訓練と経験が必要であり，そのようなモデルを作成するにはモデルの描き方を学んだだけでは難しい．

実際，情報処理推進機構の調査では，企業でのモデルベース開発への取り組みは浸透していないとの結果が出ている．[5]この原因は優れたモデルを作成することが難しいせいであると考えられる．

上記の点を満たすため,モデリングの初心者であっても，優れたモデルを作り出すことができる必要がある．ここ述べる優れたモデルとは，新たな機能拡張性と再利用性に富んだモデルであるとする．

3.1 モデルカタログの概要

優れたモデルを作成するための参考として，モデルカタログが挙げられる．

モデルカタログとは，特定非営利活動法人 UML モデリング推進協議会(UMTP)によって発行されたモデル集である．モデルカタログは組込み系システムの開発を支援するために発行されたものである.UML を利用することによる開発ソフトウェアの生産性・品質の向上が目的で，いくつかの製品を例に挙げ，その製品の UML モデルを提示することによって，開発者のモデリングのヒントになる．

モデリング初心者でも優れたモデルを作成するには，手本となるモデルが必要である.手本となるモデルがあれば，開発者はそのモデルに手を加えていくことで，開発システムに適した形に直していき，システム設計を行うことができる．しかし，モデリングの参考になるほど優れたモデルには価値があり，企業秘密などで隠蔽されることが多い．そこでモデルを公に公開することにより利用性の高いモデルの共有を目指している．

また，モデルカタログの作成者は，実際にシステムを開発している企業の社員が主であり，メンバー間で「モデルの品質向上」を掲げた活動を積極的に行っている[2].

3.2 モデルカタログの問題点

モデルカタログは 2010 年に第 1 版が発行され,現在もバージョンアップが進んでいるが，知名度は低く，開発事例も少ないため，有用性は周知されていない．

また，モデルカタログは MDD における PIM までしか示しておらず，PSM は開発者自身で作り出す必要がある．さらに,その記載されている PIM は様々な開発事例にも適用できる必要があるため，実際に開発するハードウェアには不可能な機能が存在することが多い．したがって，開発者は自分が開発したい製品に適した PIM に直す必要がある．

以上の点から，モデルカタログには問題点が存在し，それがモデルカタログ知名度低下の一因となっていると考えられる．そこで本研究は,モデルカタログを利用した MDD を行うことで，モデルカタログの有効性を示した事例の提供し，モデルカタログによる開発の支援となる．

4. モデルカタログを用いたモデル駆動開発の実践

本研究は MDD にモデルカタログを適用し，組込みシステムを開発する．具体的な方法としては，実際に開発する製品を定め，モデルカタログに記載されている項目でそれ

Vol.2015-SE-188 No.2
Vol.2015-EMB-37 No.2
2015/6/4

にもっとも近い製品を選出する．そしてモデルカタログ内のモデルを参考にしながら PIM の設計を行う．

本章の構成は，4.1 節に開発対象の仕様と開発環境の説明，さらに参考にしたモデルカタログの項目の説明を述べる．4.2 節はモデルカタログ内の PIM を分析しながら，実際の開発対象に適した PIM を作成する．4.3 節は PIM を基にした PSM の作成工程を述べる．4.4 節は PSM からの実装工程を述べ，4.5 節に開発対象の動作チェックの詳細を述べる．

4.1 開発システム

4.1.1 カラーソーターの概要

実際に開発する装置として本研究ではカラーソーターを開発した．カラーソーターとは，導入した試料を，特定の色とそうでない色の 2 つに仕分けする装置である．次の図 4.1 にカラーソーターのイメージ図を示す．

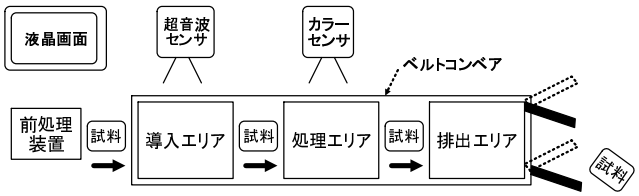

図 4.1 カラーソーターのイメージ図

図 4.1 では，ベルトコンベアによって，試料を左端から右端に搬送する．また，試料はベルトコンベアに乗せられる際，前処理装置から引き渡される．ベルトコンベアには 3 つのエリアが存在している．3 つのエリアの詳細を次に示す．

- 導入エリア

前処理装置から渡された試料を置く場所である．問題なく試料が設置されたかどうかを超音波センサによって判定する．

- 処理エリア

カラーセンサによって処理エリア上に試料があるかどうかの判定と試料の色の識別を行う．

- 排出エリア

処理エリアにて色の読取が終わった試料が最後に運ばれるエリアで，色によってレールの切り替えを行う．

さらに，処理エリアでの処理の記録を表示するための液晶画面を実装する．表示する内容は，読み取った試料の ID と色の名前である．カラーソーターが停止中の場合は”Color Sorter is stopped”という文字列を表示する．

また，振る舞いの定義を状態遷移図で示す．次の図 4.2 に状態遷移図を示す．

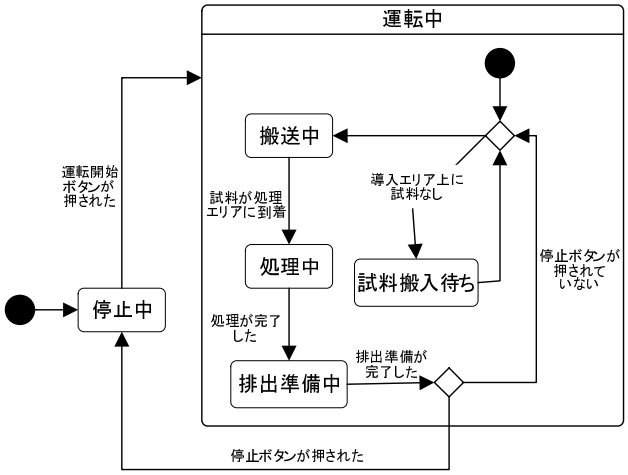

図 4.2 カラーソーターの状態遷移図

インライン装置が停止中，ユーザがスタートボタンを押下すると，超音波センサによって，試料が導入エリアに存在するかどうかを確認する搬入チェックを行う．試料がなかった場合，前処理装置が搬入処理を行う．この動作によって試料がベルトコンベア上に導入される．もし，搬入に失敗し，試料が導入されていないと判定した場合は再び前処理装置による搬入処理を行う．試料が正常に搬入された場合，試料の処理と排出の準備を行う．それらの処理が終わり，かつ運転停止ボタンが押下されていなかったら，再び試料搬入待ちの状態に遷移する．

4.1.2 開発環境

カラーソーターのハードウェアは LEGO MINDSTORMS EV3（以下 EV3）によって実装する．EV3 とは LEGO 社が開発している教育用ロボットキットであり，マイクロプロセッサが組み込まれたブロックに対してプログラミングをすることでロボットの動きを制御できる．

EV3 の構成要素は，マイコンとバッテリーを組み込んだインテリジェントブロック，超音波センサ，カラーセンサ，そして 3 つのサーボモータである．これらを利用して前節で示したカラーソーターを実装する．次の図 4.3 に実際の画像を示す．

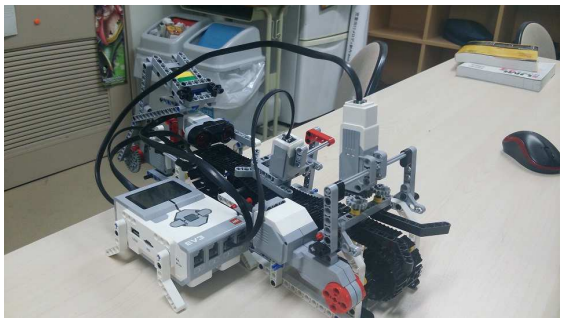

図 4.3 カラーソーター

液晶画面と操作用のボタンは EV3 インテリジェントブロックに付属している LCD とボタンを利用して実装した．また，ベルトコンベア，排出エリアにあるレール，そして前処理装置を動かすためのモータは EV3 付属のサーボモータで動かす．超音波センサとカラーセンサは EV3 の付属

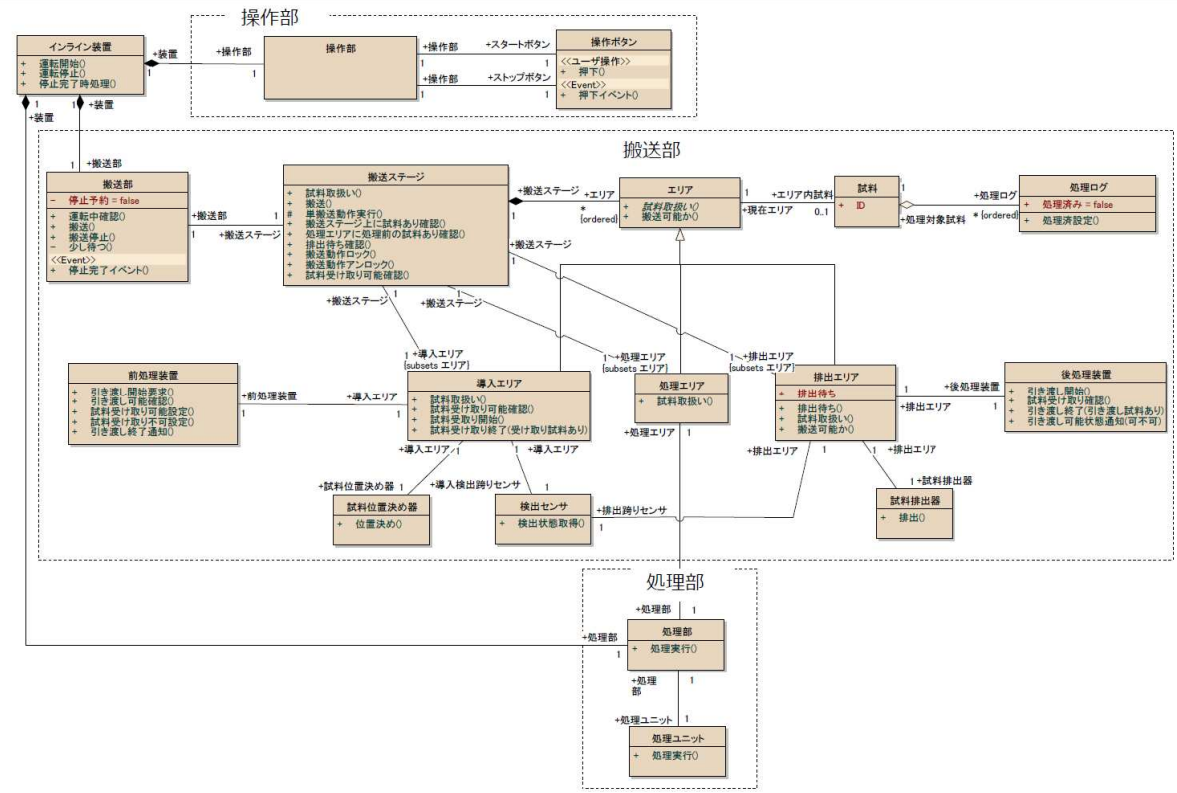

図 4.4 モデルカタログ内の PIM(文献[2]より引用)

品を利用した.

また, EV3 を Java で制御するためのファームウェアとして leJOS EV3 を用いた. leJOS EV3 とは，インテリジェントブロック上で動作する Java バーチャルマシンである. Java の基本的な仕様に加え，EV3 の各種デバイスを制御するための API を実現している. 本研究では，leJOS EV3 を用いてシステムを実装した.

環境について，本研究では Java にて開発を行うため，PC から EV3 まで jar ファイルを転送する必要がある Jar ファイルを EV3 インテリジェントブロック内で動作させるため EV3 インテリジェントブロックに leJOS EV3 をインストールする必要がある. また, ソースコード作成にあたって，統合開発環境として eclipse を使用した. 各種モデルはモデルダイアグラム作成ソフトウェアの Enterprise Architect を使用した.

4.1.3 モデルカタログの PIM モデル

カラーソーターを開発するにあたって，そのモデルを作るため，モデルカタログからカラーソーターにもっとも近い項目を選び，モデルの参考にする.

今回はカラーソーターの機構に最も近いインライン装置の項目を参考にした

インライン装置の PIM を図 4.4 に示す.

インライン装置は，工場の生産ラインなどで使用される上流から下流に向かって，製造物や検査品を搬送・処理する装置のことである. 次の図 4.5 にインライン装置のハードウェアのイメージ図を示す.

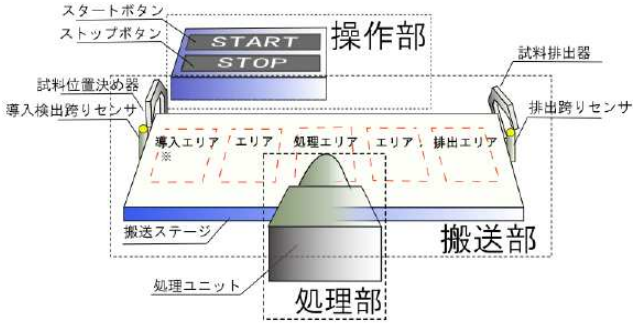

図 4.5 インライン装置のイメージ図(文献[2]より引用)

この装置は操作部，搬送部，処理部の 3 つのユニットから構成されている. 操作部では，インライン装置の運転状態と停止状態を切り替える役割を持つ. 搬送部は前処理装置からの試料の受け入れと試料の排出，ベルトコンベアなどで試料を搬送するという機能を持つ. 処理部は試料への加熱処理など，試料への処理を行う部位である. 以下に各部が持つ機能を示す.

- 操作部
  - スタートボタン，ストップボタン

インライン装置全体の停止状態と運転状態を切り替える. カラーソーターでは EV3 インテリジェントブロックに付属しているキーを用いる.

- 処理部
  - 処理ユニット

加工や検査など試料に対する処理を行う. カラーソーターでは，光センサを用いて試料の色

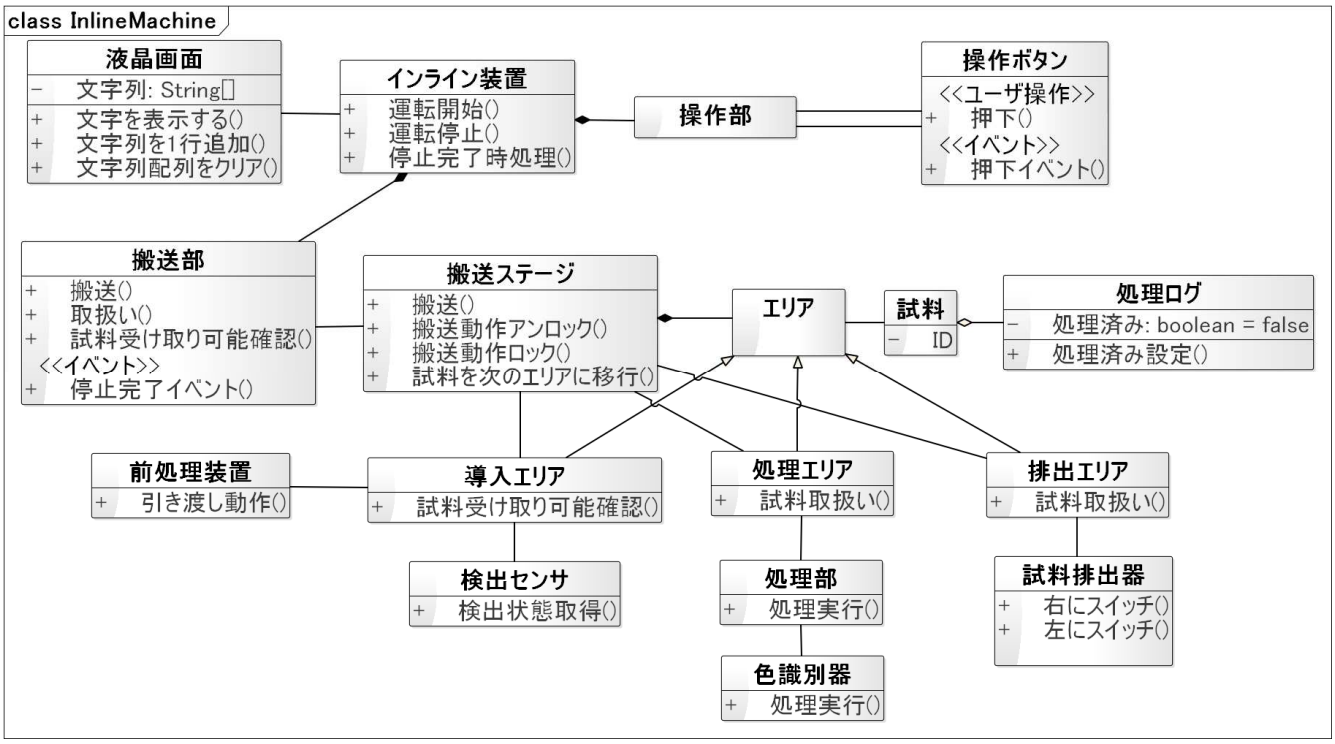

図 4.6　カラーソーターの PIM

を判定する.

- 搬送部
  - 搬送ステージ
    試料を搬送する領域全体の部分．カラーソーターではレゴブロックのパーツで実現する.
  - 導入検出跨りセンサ
    前処理装置から正しく試料を受け取れたかどうかを検出する．カラーソーターでは超音波センサを用いて試料を受け取れたかどうかを判定する.
  - 試料排出器
    試料を後処理装置へ排出する機構．カラーソーターではレゴブロックのパーツで実現する.
  - 導入エリア
    前処理装置から受け取った試料が一時的に置かれ，位置決めを行う場所.
  - 処理エリア
    搬送ステージのうち，試料に対し処理を行う場所.
  - 排出エリア
    試料を後処理装置へ排出する場所.

4.2 PIM の作成

プラットフォームに依存しないモデルとして PIM を作成する．前節にて説明したカラーソーターの処理方式に注目してモデルを考える．プラットフォームに依存しないことが求められているため，ファームウェアや Java 言語での実装については考慮しない.

4.2.1 クラス図の作成

図 4.2 に示した状態遷移図の振る舞いに合わせて，各構成要素の動作内容を定義する．モデルカタログ内で示されていたインライン装置のクラス図をより具体的にしていくことで，PIM を実現する．次の図 4.6 により具体的な処理内容を記述したクラス図を示す.

モデルカタログ内のクラス図を次のように変更している.

- 構成要素の削除

モデルカタログ内でのインライン装置は汎用的に対応するため，多くの種類の構成要素を用意した．しかし，本研究ではカラーソーターに対応させる必要があったため，いくつかの構成要素を減らす必要があった．クラス図から削除した構成要素は，排出跨りセンサ，試料位置決め器，後処理装置である.

- 構成要素の追加

モデルカタログ内でのモデルに新たな構成要素を付け足した．モデルカタログ内では存在しなかった機能を加えることで，モデルカタログ内のモデルの柔軟性を示すことが目的である．追加した構成要素は液晶画面である．表示する文字列を配列で管理して,画面に表示する機能を有する．液晶画面クラスはインライン装置クラスに関連させた.

- 搬送ステージ，前処理装置，導入エリアの処理方式

前述したふるまいを実現するにあたって，いくつかのクラスの処理を変更した.初めに,搬送ステージに関しては，本開発対象において，搬送ステージは試料をエリアからエリアへ適当な操作量で搬送することが役割である．そのため，その役割に直接関係ない操作を削除した．前処理装置や導入エリアに関しても，同様に仕様の実現に必要な操作

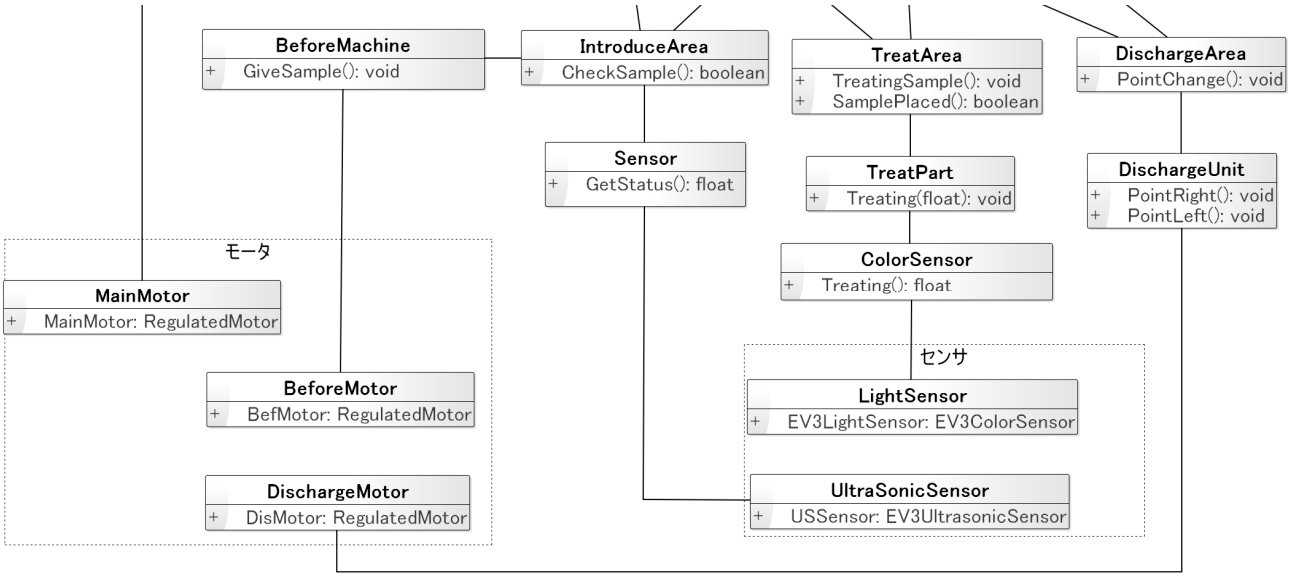

図 4.8 カラーソーターの PSM の一部

だけを残し，より処理内容をシンプルにした．具体的な変更点を次の表 1 に示す．

表 1 搬送ステージ，前処理装置，導入エリアの具体的変更点

クラス名	変更前の操作	変更後の操作
搬送ステージ	試料取扱い 搬送 単搬送動作実行 搬送ステージ上に試料あり確認 処理エリアに処理前の試料あり確認 排出待ち確認 搬送動作ロック 搬送動作アンロック 試料受け取り可能確認	搬送 搬送動作ロック 搬送動作アンロック 試料を次のエリアに移行
前処理装置	引き渡し開始要求 引き渡し可能確認 試料受け取り可能設定 試料受け取り不可設定 引き渡し終了通知	引き渡し動作
導入エリア	試料取扱い 試料受け取り可能確認 試料受け取り開始 試料受け取り終了	試料受け取り可能確認

- 処理ログの記録項目の具体化

本開発対象は処理エリアにおいて，試料の色を判別し色名を記録する役割を持っている．そのため，処理ログクラスの属性に色の名前を追加した．

- 試料排出器の処理の具体化

モデルカタログ内でのモデルにおいて，試料排出器の動作内容は抽象的であった．カラーソーターにおける試料排出器の役割は，レールを左右に切り替えて，試料が出ていく方向を制御することである．この役割を踏まえ，PIM 内での試料排出器クラスの動作内容を具体化した．

インライン装置の PIM からカラーソーターの PIM への変換のより具体的な内容を示す．インライン装置の PIM の一部とカラーソーターの PIM の一部を見比べることにより，作成の具体例を示す．次の図 4.7 にインライン装置の前処理装置クラスと，カラーソーターの前処理装置クラスを示す．

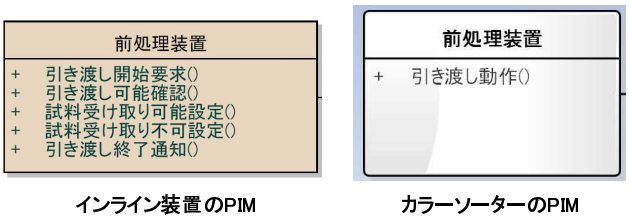

図 4.7 インライン装置とカラーソーターの PIM

実際の開発において上図右のクラスを左のクラスに書き換えた．インライン装置の PIM は引き渡し開始要求メソッドなど，5 つものメソッドが用意されているが，カラーソーターの PIM は引き渡し動作メソッドの 1 つのみである．このような書き換えを行った理由は，モデルカタログ内で想定されていた前処理装置の仕様が，カラーソーターでの前処理装置の仕様と離れていたためである．前処理装置とはインライン装置本体とは別のシステムである．そのため，処理体系も異なることがあるため，引き渡し可能確認メソッドなどで，前処理装置とインライン装置とのハンドシェークを意識する必要がある．カラーソーターでは，前処理装置はシステムの 1 つとして組み込んであるため，ハンドシェークを意識する必要はない．そのため，カラーソーターの前処理装置クラスはモデルカタログの PIM よりシンプルなものとなった．

このように，モデルカタログと実際に開発するシステムとはハードウェア的にも違いが大きく出てくる場合がある．そのことを意識して PIM 作成を行った．

4.3 PSM の作成

PSM（Platform Specific Model）を作成する．PSM とは前節で作成した PIM を基に作られるモデルであり，よりプログラム言語やハードウェアなど，プラットフォームに依存

Vol.2015-SE-188 No.2
Vol.2015-EMB-37 No.2
2015/6/4

した形のモデルである．図 4.6 に示した PIM を基に作成した PSM を図 4.8 に示す.

前節に示した PIM 図からの主な変更を示す.

- 各センサ，モータのハードウェアラッパー

実際にコーディングするにあたって，EV3 インテリジェントブロックのハードウェアポートと各種センサ，モータとのつながりを定義する必要がある．そこで，前処理装置クラスや実行ユニットクラスなど，ハードウェアに深くかかわるクラスそれぞれに新たにハードウェア用のクラスを関連させた．図 4.8 内での BeforeMotor クラス，MainMoto クラス，DischargeMotor クラス，Button クラス，LCD クラスに該当する.

- 各クラスの名称の変更

PIM の段階では処理の方針などは具体的に定義しなかったため,クラスの名前は抽象的なものになっている. PSM では処理の方針は決定したので，一部のクラスの名称を具体的なものに変更した.（実行ユニットクラスをカラーセンサユニットに変更するなど）さらに，実際のコーディングでは英語でのコーディングであるため，日本語ではなく英語で PSM を作成した.

そのほかにも，操作用のボタンのクラスにスレッドを適用するなど,ハードウェアや使用言語に合わせた PSM を構築した.

4.4 実装

PSM を基にコーディングを行う．各制御はモデルを実装されるため，クラスや構造は考えず，モデルに沿って実装するのみである．そのため，コーディングの段階で考えるべきなのはメソッドの具体的内容のみとなる.

たとえば，処理エリアはカラーセンサから RGB 値を取得し，試料の色を判断するが，どのような判定に判定するかをプログラムに反映させていく.

また,コーディング途中で PSM に誤りがあった場合はその都度 PSM を修正する.

更に，実装の際に活用した API に関する説明を付加する．EV3 を Java で制御するために，Lejos の API を用いて各センサやモータを制御した．使用した API の一部を，例を用いて示す．カラーソーターを構成する要素の 1 つである DischargeMotor クラスを例にとる．DischargeMotor クラスは排出ユニットに使われるモータを制御するためのクラスである．有するメソッドは，モータを動かす準備をするための init メソッドと，与えられた引数の角度だけモータを回す Rotate メソッドである. 次の図 4.9 に DischargeMotor クラスのソースを示す.

```
1  package InlineMachine;
2
3  import constants.Constants;
4  import lejos.robotics.RegulatedMotor;
5
6  public class DischargeMotor {
7
8      private RegulatedMotor dismotor;
9
10     DischargeMotor(){
11         dismotor = Constants.DISP;
12     }
13
14     public void init(){
15         dismotor.resetTachoCount();          //モータ内の速度計を初期化
16         dismotor.rotateTo(0);                //モータの角度を0度に直す
17         dismotor.setSpeed(Constants.DISSPD); //モータの回るスピードを設定する.
18     }
19
20     public void Rotate(int in){
21         dismotor.rotateTo(in);               //モータを引数の角度だけ回転させる.
22     }
23
24 }
```

図 4.9 DischargeMotor クラスのソースコード

上記のソースコードを作るにあたって，leJOS で定義されているメソッドを使用した．以下に使用したメソッドとその内容を示す.

表 2 使用したメソッドの詳細(文献[6]を基に作成)

返り値	メソッド名とその説明
void	resetTachoCount() Reset the tachometer count
void	rotateTo() Causes motor to rotate to limitAngle; Then getTachoCount should be within +− 2 degrees of the limit angle when the method returns
void	setSpeed(int speed) Set motor speed

4.5 カラーソーターの動作結果

カラーソーターを，モデルカタログを基に開発した．その実行結果を今節で示す.

実装したカラーソーターを実際に動作させ，問題なく試料を仕分けることができたかどうかをテストした．カラーソーターに 8 つの試料を設置し，試料が問題なく特定の色とそうでない色の 2 つに仕分けられたかどうかを調べた．表 3 に結果を示す.

各試料の色と実験結果を示した．仕分けることが出来た試料は実験結果の項に○書き，仕分けできなかった試料は×を書いた．また，分別に失敗した試料については，なぜ失敗したのか原因を示す.

表 3 動作テストの結果

	1回目		
ID	資料の色	実験結果	原因
1	青色	○	
2	緑色	○	
3	黄色	○	
4	青色	○	
5	赤色	○	
6	赤色	○	
7	黄色	○	
8	緑色	×	前処理装置からベルトコンベアへの伝搬ミス

ほとんどの試料を仕分けできた．また，前処理装置からベルトコンベアへの伝搬ミスがあった場合でも，システム

は正しく導入できなかったと判断し，再び再搬入動作を行ったため正常な動作であるといえる．

5. 考察

5.1 モデルカタログを利用することによる効果

本研究では PIM の設計にモデルカタログを利用することで，効率的な開発を試みた．

カラーソーターを設計した際，PIM の設計段階では，モデルカタログ内のインライン装置のモデルをひな形としてクラス図を設計した．この段階で行ったことはインライン装置のモデルの分析と，カラーソーターの仕様に適した形にモデルを変更するということである．その手法は一般的なモデルの設計とは大きく違い，モデルの根幹的な方針が決まっているため，必要な構成要素や操作などを削除したり，追加したりするなどで PIM 設計を行うことが出来た．これにより，PIM を初めから設計することよりも短時間で済み，よりスムーズに次の段階に進むことが出来た．また，PIM を作成することが容易であるため，MDD の経験がある人員がチーム内にいなくとも，MDD による開発に臨みやすくなる．

さらに，MDD の経験がない開発者でも，モデルを使って開発するということを通してモデリング技術のノウハウを学ぶことが出来ると考えられる．したがって，モデリング技術を学ぶ参考書としてもモデルカタログを利用することが出来る．

5.2 モデルカタログの問題点

モデルカタログ内のモデルには，あまり詳細な説明が記載されていないモデルも存在する．それにより，モデル内で，何に使うのかわからないクラスやメソッドが存在するなど，使用者を混乱させることがあった．例えば，モデルカタログ内のクラス図における搬送ステージクラスのいくつかのメソッドは使用用途のわからないものがあった．各メソッドに説明を加えるなど，詳細な説明が求められるだろう．

5.3 今後の課題

5.3.1 他の開発環境への適用

モデルカタログで示されている PIM はプラットフォームに依存しない形である．すなわち本研究での開発環境以外にも適用できると考えられる．EV3 のファームウェアは leJOS EV3 以外にも複数存在する．今後は本稿で示した PIM を他の開発環境にも利用し，変わらない動作を実現できることを検証する必要がある．

5.3.2 機能変更による柔軟性の検証

モデルカタログで記されているモデルは，抽象的であり，様々な動作に対応できる．それを検証するため，本研究で開発した装置とは別なものを設計実装したい．たとえば，処理ユニット以外は変えずに処理ユニットの内容のみを変更するなどが開発例として挙げられる．具体的な変更内容としては PSM 内での，TreatUnit クラスの下にある Colorsensor クラスを書き換えることによって実装できる．このことで，機能変更による柔軟性をより強調できる．

6. おわりに

本研究では，初めに組込みシステム開発手法の 1 つとして，モデル駆動開発について述べ，その問題点を論じた．次に問題の対応策として，モデルカタログの存在を示し，モデルカタログの利点と問題点を挙げた．

そして，モデルカタログの有効性を示すため，実際に組込みシステムを設計・開発することでモデルカタログを用いた開発事例の提供を行った．これによって，モデルカタログを用いた MDD の支援となる．

今後の課題は，ほかのシステムの開発事例の提供などが挙げられる．

参考文献

1) OMG MDA http://www.omg.org/mda/
2) UMTP/Japan 組込みモデリング部会：組込み分野のための UML モデル解説書，(2013)
3) 新村祐太，組込みシステム開発における UML モデルカタログの実践研究，(2014)
4) John D. Poole Model-Driven Architecture: Vision, Standards and Emerging Technologies, (2001)
5) 独立行政法人 情報処理推進機構：「組込みシステムの先端的モデルベース開発実態調査」調査報告書，p60，(2012)
6) LeJOS, Java for Lego Mindstorms, Overview (leJOS EV3 API documentation) http://www.lejos.org/ev3/docs/