B-028

再利用可能部品としての PIM の適用性実験

Applicability Experiment of PIM as Reusable Parts

天川 美那† 松浦 佐江子‡
Mina Amakawa Saeko Matsuura

1. はじめに

MDA(Model Driven Architecture)の特徴であるプラットフォーム分割による問題領域の明確化を用い，組み込みやユビキタスなどの多様なハードウェアデバイスによる実装が考えられる場面での開発や，自動変換技術に着目して抽象度の高い領域で開発を行う事でモデルを資産とする試みなどが行われている．本研究はそれらの研究を成立させる大前提である適正なプラットフォーム分割という問題に注目し，システムの汎用部品であるインタラクション部（ユーザ・システム間のデータ入出力を司る部分）を適切にロジックから分離して独立させる事で，インタラクション部の仕様変更に対応する事はもとより，ロジック部分の本質の更なる明確化を試みるものである．これにより開発対象の本質部分を正確に把握し，質の高いシステムを構築する事を目指す．

2. プラットフォーム分割

2.1. 図書管理システム

事例として図書の管理を行うシステムを開発した．これはシステムの提供するサービスのロジックモデルと，システムがユーザにサービスを提供する際のインタラクションに着目したモデル，サービスとユーザ権限の関係に着目したモデルの 3 つのPIM(Platform Independent Model)をサービスを接合点として構築し，これらのモデルから 1 つのアプリケーションを複数のインタラクションに対して提供するものである．この事例ではインタラクションモデルのPSM(Platform Specific Model)を標準入出力とICカードR/WシステムのFeliCa入出力により実現した[1]が，その他の実現手段に対しても表現が十分であるか未だ明確でない．そこで今回は同システムを汎用的な実装技術の 1 つであるwebブラウザを介したインタラクションに対応するように拡張し，生じるモデルの変更点を観察して作成したPIMの適用性を実験する．

2.2. インタラクション

プラットフォーム分割の事例として挙げたインタラクションはすべてのインタラクティブシステムに存在するである．ゆえに，体系化されたモデルは今後の開発にも有効であろう．ただし全てのシステムにある部品を一般化する事は難しく，抽象度が高すぎて実用性が低下する恐れがあるため，何種類かの代表的なインタラクションを考えて PIM を組む事で，ある程度流用可能で現実的なモデルを描く事を考える．

また，本質的なロジックの開発はロジックに与える値の取得方法や実行結果の出力方法であるインタラクションに依らない．人はインタラクション部を通じてシステムに触れるため，インタラクション部の責務はロジック部の責務との混同を招きやすい．これを明確に分離し独立させる事でロジック部の問題領域を見極めやすくする事が肝要である．

†芝浦工業大学大学院工学研究科電気電子情報工学専攻，
Graduate School of Engineering, Shibaura institute of technology
Department of electronic engineering and computer science
‡芝浦工業大学システム工学部電子情報システム学科，
Shibaura institute of technology Department of electronic information system

3. PIM の再構築

3.1. 図書管理システムの PIM の問題点

以上を踏まえ，図書管理システムのインタラクション部分のモデルをブラッシュアップする事を考える．図 3.1 は作成したインタラクション部の PIM である．「インタラクション」はロジック部のあるサービスをユーザに対して提供する際に呼び出される．「インタラクション」はサービスを実現するための「アクション」を呼び出し，ユーザからの「入力」を受付け，ロジック部に値を渡し，更にその結果に対応する「出力」を行うという動作をするものである．

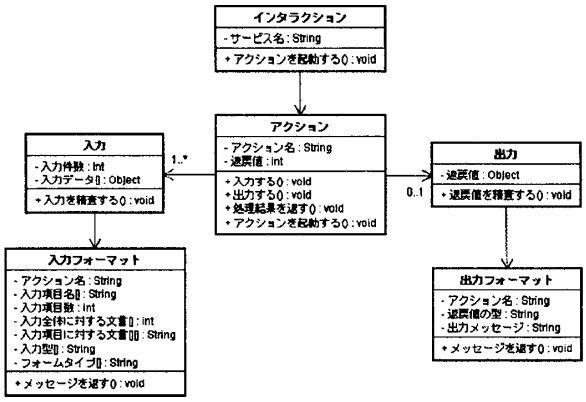

図 3.1 図書管理システム作成時のインタラクション PIM

この PIM には以下の二つの問題点がある

- 図書管理システムのインタラクションに特化している
- 標準入出力・FeliCa の二種の実装のみを想定して定義された

これらの問題点を改善するために，二通りの観点からモデルの洗練を行う．

3.2. インタラクションの再定義

作成した図書管理システムは提供するサービスとそれを解決するロジックのメソッドが 1 対 1 になっている．たとえば図 3.2-1 は「貸出」というサービスのインタラクションであるが，ここでのロジックメソッドは「貸し出す」という一つのみである．

これを別システムのインタラクションに流用しようとした場合に齟齬が生じる．例として，商品の発注管理を行うシステムにおける商品の注文を受付けるインタラクションでのアクション系列（図 3.2-2）を挙げる．ユーザ入力「商品を選択する」はロジック「商品一覧を取得する」の

出力として得られる「商品一覧」がユーザに提示されている前提で行われるものであり，以前のモデル図ではこのような複数のシステムロジックを介したアクション同士の前後の関連を表現できていない．これを解決するためにインタラクションモデルのオブジェクトの関係性を洗い直す必要がある．

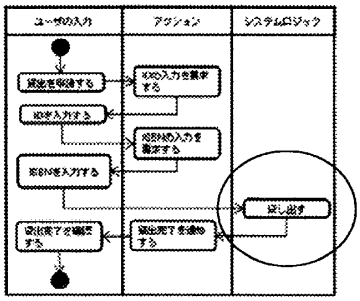

図 3.2-1 貸出のインタラクション

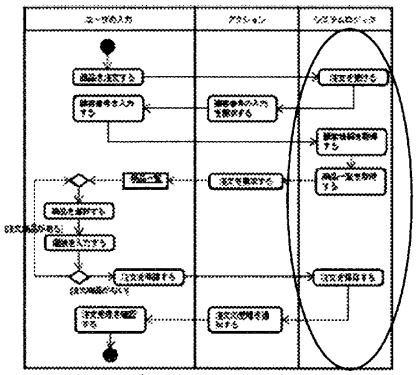

図 3.2-2 商品注文のインタラクション

以下，インタラクションとアクションの再定義を行う

- インタラクションとは
  サービスに対する一連の入出力処理の事
  →サービスを達成するためのアクション系列を定義したもの
- アクションとは
  パラメータを得て，処理を行い，実行結果を返すもの

表 3.1 インタラクションの実行系列

アクションに必要なパラメータを得る
情報を提示する
入力値の説明
前回アクションの結果
入力方法(入力欄)
新たな情報を得る
値を入力する
記入する
｜選択する
｜送信する
入力値が適正か調べる
適正値を設定する
入力された値と比較する
アクションを実行する
アクションの実行結果を提示する
結果の整合性を調べる
返戻値の予測値を設定する
実際の返戻値と比較する
結果を並べて表示する
並べ方を設定する
並べる
出力する

表 3.1 はインタラクションを実現するために必要な動作を一覧にしたものである．

何を入力する必要があるのかをユーザに説明し，入力欄を提示して入力を促す．システムに対する入力は値の記入，値の選択，値の送信の三種に大別され，そのようにして入力された値がアクションのパラメータとして事前に設定した条件に反していないかを精査する．返戻値についても同様に精査し，一覧しやすい形に整列して出力する．この時，同サービスのアクションの返戻値が次回アクションの入力における事前条件になりうる事に注意しなければならない．

図 3.3 は上述の定義を踏まえて構築し直したインタラクションの PIM である．「インタラクション」はサービスを満足するための「アクション」群を定義しており，順次呼び出していく．活性化された「アクション」は処理を行うのに必要な情報を得るため，「パラメータ」を活性化する．「パラメータ」は「実引数」の集合であり，実際の値は「実引数」が持つ．「実引数」は「入力形式」として項目名と記入・選択・送信のいずれかの入力方法を持っている．「パラメータ」は入力を得るために提示する情報を全て把握しており，それを「入力」に渡す事でユーザに対して入力を促す責務を持つ．この時，「入力」に必要な情報の一つに前回アクションの実行結果である「事前条件」がある．前回のアクションはインタラクションクラスが保持しているため，「アクション」及び「パラメータ」を活性化させる際に値を渡しておく必要がある．

「入力」は渡された要素を整列し，入力欄を交えてユーザに提示する．得られた入力は「パラメータ」が受け取り，一つずつの値を「実引数」に引き渡す．「実引数」は渡された値が決められたフォーマットに則っているかどうかを精査し，不正があれば「パラメータ」が記録する．不正がある場合は不正値についてのみ「入力」を繰り返す．

「パラメータ」を得た「アクション」はロジック部分を呼び出し，処理を行った後に返戻値を得る．「アクション」は返戻値を元に「実行結果」を活性化し，返戻値が予測されるフォーマットに則っているかどうか精査する．値が空でなければ返された値を整列する．整列された値は今回のアクションごと「インタラクション」に保持され，次回のアクションの入力の事前条件となる．次回のアクションが存在しない場合は「実行結果」がインタラクションそのものの結果となるため，整列された返戻値と「実行結果」の持つメッセージを「インタラクション」が「出力」に渡し，ユーザに対する結果の表示を行う．

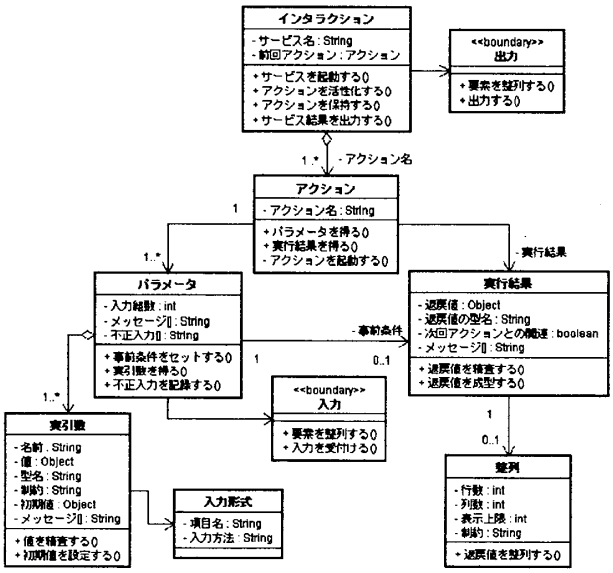

図 3.3 再構築後のインタラクション PIM

3.3. 入出力機構に対する PSM の構築

次に入出力機構について一般化を図る．前回の実験では標準入出力と FeliCa による入力を想定した上で，二つの入力デバイスについて一般化を行った．しかしシステムの仕様上 IC カードに対しての出力は行わず，入力もユーザ ID の取得のみにとどまったため，結果的に一度の入力について一つの入力値を解釈するというインタラクションモデルが構築されている．PIM の汎用性を考える上で，一部のデバイスや入出力機構に特化した構成が好ましくない事は自明である．そこで，前回の実験で使用したコマンドラインを用いた CUI(Character-based User Interface)による入出力に対し，システムに対する汎用入出力機構である web ブラウザについて拡張する事でモデルを洗練し，改めて PIM の正当性について論じる．

なお，インタラクションシステムには入出力機構によって変更を生じる部分と生じない部分がある．変化があるのは「パラメータ」にあたる情報を取得する一連の入力機構と，「出力」を行うために返戻値の調整を行う「実行結果」の一部である．他のサブシステムやロジック部分に接合する責務を担うインタラクションクラス・アクションクラスについては「得られたパラメータについて処理を行い，返戻値を得る」というアルゴリズムが実現されればよいので，入出力機構の変更があってもこの部分に変化は生じない．

3.3.1. 入力

コマンドラインと web ブラウザによる入出力での入力における大きな差異は以下の二つである．

- 一度に複数の値の入力を得られる
- 複数種の入力形式がある

コマンドラインの場合，入力される値は必ず一度に一つずつである．故に実際の入力値を持つ「実引数」の集合である「パラメータ」クラスで任意の回数入力要求を繰り返し，データが揃った事を確認してからアクションが実行される必要がある．また，コマンドラインからの入力方式は常に一通りである．このため，「入力形式」の違いは入力する値についての説明の提示方法の変化にのみ現れる．

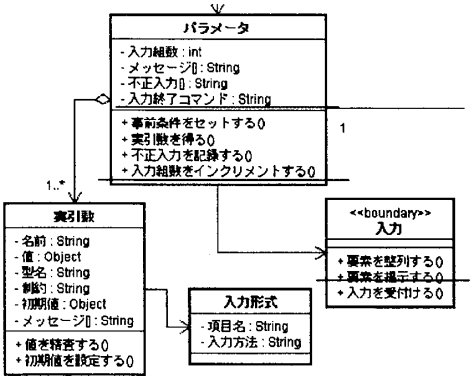

図 3.4-1 コマンドライン用 PSM(入力部)

一方 web ブラウザの場合，ユーザからの入力は複数の入力欄を備えたフォームを使って行われるため，一つのアクションに対するパラメータが複数同時に入力される事が考えられる．これに対して「パラメータ」クラスは受け取った入力を「実引数」ごとに分け，それぞれについて値の精査を行う必要がある．更に，入力欄はそれぞれ異なる形式を持つため，「入力形式」クラスでどのインプットを用いるのかを具体的に指定する必要がある．

以上を踏まえて図 3.3 の PIM をコマンドライン用 PSM，及び web ブラウザ用 PSM に変換した．図 3.4-1，図 3.4-2 はそのパラメータ取得に関する一部である．下線部が各 PSM の独自要素である．

図 3.4-1 についてはユーザからの明示をもって入力動作を終了するため，入力終了をシステムに通知するためのコマンドを定めておく必要がある．また，全ての入力動作が終了するまでアクションに対してのパラメータの個数が分からないため「パラメータ」クラスに「入力件数」を逐次数え上げて記録するためのメソッドが必要である．更に，コマンドライン入力では入力ラインと入力する値についての説明が混在する事がないため，これを明文化する目的で「入力」クラスに「要素を提示する」というメソッドを設けた．

一方図 3.4-2 については一度の入力で複数の「パラメータ」及び「実引数」が得られるため，入力値を「パラメータ」単位に分解して数え上げる「入力件数を得る」メソッドと，「実引数」単位に分解してそれぞれを精査に回す「入力データを実引数に分解する」メソッドを設けた．また，web ブラウザ固有の入力フォームに対応するため，「入力形式」クラスを入力フォームを構成する要素で表現し，「入力」クラスの「フォームに規定値を設定する」メソッドで実際の入力欄を作成するものとした．

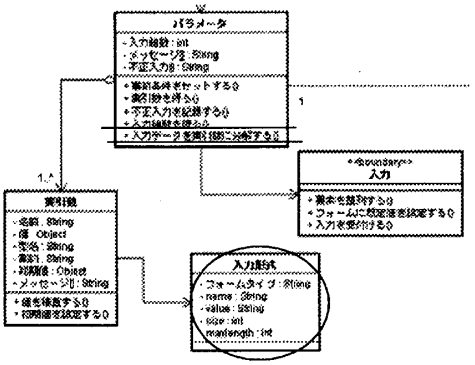

図 3.4-2 web ブラウザ用 PSM(入力部)

3.3.2. 出力

出力に関して行われる処理は「返戻値を精査し，値を並べ替え，ユーザに対して表示する」という一連である．このうち出力機構によって変更が生じるのは「ユーザに対して表示する」ための部分である．

コマンドラインでの出力は並べ替えたデータがほぼそのまま出力されるため，図 3.3 に改変を加えずに表現する事が出来る．web ブラウザによる出力は図 3.5 に見えるように，画面上での表示位置の要素が加わる．

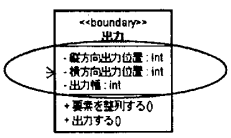

図 3.5 web ブラウザ用 PSM(出力クラス)

以上のように，図 3.2 の PIM に独自要素を加える事で各機構の実装を実現するものとした．

4. 適用実験

図 3.1 の PIM の問題点として挙げた二点が 3 章で構築した PIM・PSM を用いて解決された事を実験で確認する．

4.1. 実験内容

まず，図 3.2-2 のインタラクションにおけるアクション「注文を要求する」から「注文の受理を通知する」に移行する過程を図 4.1 に示す．「注文を要求する」の実行結果として得られる「商品一覧」は，インタラクションクラスのメソッド「アクションを保持する」で「注文を要求する」アクションごと保持される．この結果を踏まえてインタラクションクラスは次のアクション「注文の受理を通知する」を活性化し，「パラメータを得る」メソッドを呼び出す際に整列済の「商品一覧」の属性「返戻値」を渡す事

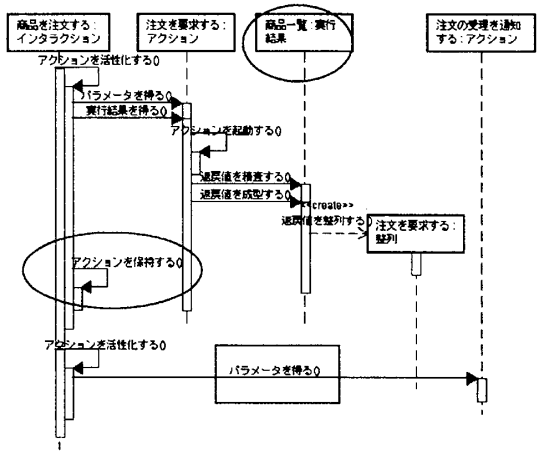

図 4.1 アクション遷移のシーケンス

で次回入力に前回アクションの結果を反映させる．

続いて図 3.2-1 のインタラクションを web ブラウザ用 PSM で表現する事を考える．（図 4.2）

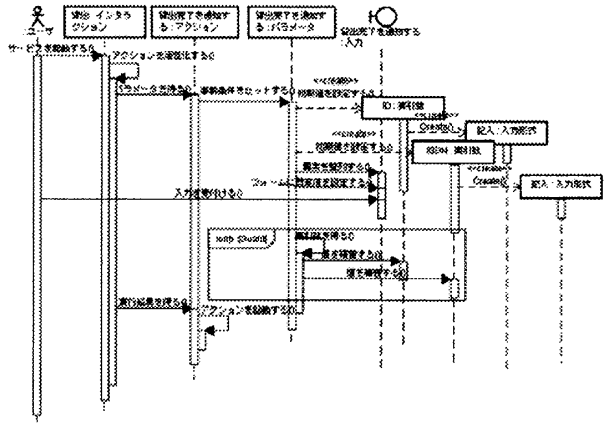

図 4.2 web ブラウザ用 PSM による貸出（入力）のシーケンス

図 4.2 における「ISBN」とは図書を一冊ずつ識別するためのコードである．図書の貸出サービスはユーザを識別するための「ID」と「ISBN」の二種の情報を必要とする．

web ブラウザによる入力を受けられるように拡張されたシステムは，アクションに対しての複数のパラメータを一度の入力で受け取る事ができる．よって「ID」と「ISBN」の組を同時に受け取ると同時に，借りたい本が複数ある場合の入力，つまり「実引数」の集合である「パラメータ」を一度の入力で複数受け取る事もできる．

システムに渡された入力はパラメータごと，実引数ごとに分解されて精査にかけられる．データがなくなるまでループを繰り返した後，インタラクションが「貸出完了の通知」を起動させ，実行結果を得て出力に移行する．貸出完了通知には返戻値がないため，処理が正常に終了すれば「実行結果」クラスの要素である「メッセージ」に記述された完了通知の文言が呼び出され，インタラクションクラスの属性として保持された後，「出力」クラスで出力される事になる．

4.2. 考察

4.1 章での二つの実験結果より，図 3.2 の PIM がアクションを複数持つ商品を注文するというサービスを表現できる事，また本を貸し出すという前回は CUI を用いて行われていたサービスを，PSM を介する事で，web ブラウザを用いても表現できる事が分かった．これにより，複数システムのインタラクションを表現できる PIM の構築，及びインタラクションの仕組みの中で入出力機構をある程度抽象化して表現する事に成功し，4 章の最初で述べた二つの問題を解決する事ができた．

インタラクションは本来システムに求められるロジックに左右されない部分である．これは商品管理と図書管理という二つの異なるシステムに対して同じ PIM を利用する事で対処できる事からも明白である．インタラクション部の責務を図書管理や商品管理というシステムの本質である貸出や返却・商品の注文処理や発注管理といった問題を考える際に，たとえば処理に必要な情報とは異なる入力がなされた場合などの例外処理を省く事が出来る．このように問題領域の区分，すなわちインタラクションというプラットフォームの分割を行う事で，入出力というシステムに不可欠でありながら処理の本質に携わらない問題をそれ単体で完結させる事が出来る．このようなサブ課題をそれぞれに完結させていくと，システムロジックの問題領域が明確になり，システムを精細化する際に有効であると考えられる．

5. 今後の展望

今回の実験ではコマンドラインと web ブラウザによる入出力という代表的な二つの機構を比較する事で PIM の精細化を図ったが，これらが適用される事の多いシステムで汎用的に用いられる別の入出力機構に Java Applet などの GUI(Graphical User Interface)がある．GUI では入出力にグラフィックを多用するため，キャラクタによる入力を主に用いる既存の PSM とは大きく異なる PSM を用意する必要があると予想される．今回作成した PIM がその差異を吸収しうるものか，しないのであればどこを変更すればよいのかについて検討する事が，PIM の再利用性をより向上させる事に繋がると考えている．

6. 参考文献

[1] 天川美那，松浦佐江子：MDA におけるプラットフォーム分割手法の妥当性の検証，情報処理学会第 69 回全国大会，6L-6，2007

[2] MDA(Model Driven Architecture), http://www.omg.org/mda/, Object Management Group

[3] スティーブ・メラー, テクノロジックアート: "MDA のエッセンス", 翔泳社, 2004.