2007－SE－157 (15)
2007／9／28

再利用可能部品としてのPIMの適用性実験

天川 美那† 松浦 佐江子‡

MDAの考え方に則り，システムロジック，ユーザ-システム間のインタラクション，サービスとユーザ権限の関係性を定義した3つのPIMをサービスを介して接合するよう作成し，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構築する方法を図書管理システムを事例として提案した．ここではインタラクションモデルのPSMを標準入出力とFeliCa入出力により実現したが，その他の実現手段を当該モデルで表現できるか否かが未だ明確でない．そこで，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．

An Experiment of Applicability of Reusable PIM for the Interaction between a System and the Users

Mina Amakawa† Saeko Matsuura‡

Based on MDA we developed a library system by connecting three PIMs. Moreover, one of the PIMs was transformed into two types of PSMs so that the input/output mechanism was implemented by the standard I/O and the Felica I/O. These PIMs express the logic in the services, the interaction of the user with them, and the authority to use them, However, there are some problems in the reusability of the PIMs. This paper discusses refinements of the PIMs on these problems and the result of our experiment to apply the PIMs to Web based interaction.

1. はじめに

MDA(Model Driven Architecture)[1]の目的は開発抽象度の向上，および生成物の再利用性の向上にある[2]．かつて高級言語とコンパイラがそうしたように，開発抽象度の向上はモデルーモデルおよびモデルーソース間をマッピング関数で繋いだ自動変換によって達成される．

一方で再利用性の向上はプラットフォーム分割と呼ばれる手法により，特定の問題領域に特化した部分をシステムモデルの中から分離し，より抽象度の高いモデルで表現する事で開発環境に依存しないモデルを作成する事によって達成されるものである．

MDAがOMG(Object Management Group)によって提唱された当初に注目されたのが自動変換技術であったため，マッピング関数の策定については特に組み込みなどの分野で多く研究されており，Mentor Graphics社の「BridgePoint」，米IBM Corp.の「Rose Technical Developer（旧Rose RealTime)」など一部ツール化もなされている．しかし，ソフトーハードという切り分けの明確な組み込みソフトウェアと異なり，多種の実現技術が混在していながらその影響の及ぶ範囲が明確になっていない純粋ソフトウェアシステムに対するMDAの適用は限定的なものが多い．例として，浜口らの研究[3]における製品に関する技術資料の配布管理システムのMDAによる開発工程では，業務の要求と実装技術という異なる二つの要素を「変更サイクル」というプラットフォームで分割している．これはつまり，実装技術を変更しても業務の要求部分に変更の影響が及ばないようなシステムを作成したものである．しかしこの開発事例で作成されたモデルやマッピング規則（パターン）はあくまでこのシステムにのみ対応するものであり，再利用性に欠ける．

このように，ソフトウェアシステムにおける

† 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

MDAの適用事例には一つの業務に対しての保守管理の容易性について論じるものが多いが，OMGの主張によればMDAは異なる業務や分野の間で共有できるモデル資産を生む可能性のある技法である．今回の実験は，多くのシステムに考えうるユーザーシステム間のデータやり取り（インタラクション）をプラットフォームとし，分割した場合について考察するものである．

2. プラットフォーム分割

2.1. インタラクション

インタラクションとはアクターがシステム内のサービスを利用する際に行う入出力を指し，システム内に必ず存在するプラットフォームである（図 1）．しかし，多様なシステム全てに対応するモデルは抽象度が高すぎるため実用的でない．そこで，今回は適用範囲を純粋ソフトウェアシステムにおけるヒューマンユーザーシステム間のインタラクションに限定する．

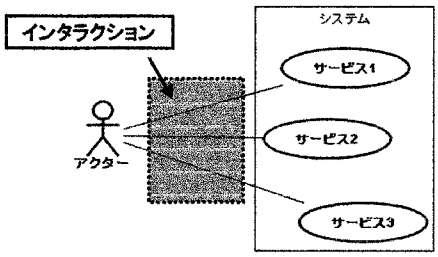

図 1 インタラクションの概略図

インタラクションをシステムのサービスに対するデータとアクションによってモデル化したPIM (Platform Independent Model)とインタラクションに関わる様々な実装技術に依存したPSM (Platform Specific Model)を作成する事により，システムがもつ複雑性をロジックから切り離す事を目的とし，再利用可能な部品としてのPIMを定義した．その適用事例として図書管理システムを用い，作成したインタラクションモデルの再利用性を論じる．

2.2. 図書管理システム

作成した図書管理システム[4]は，本の貸し出しや検索などのシステムが提供するサービスを実現するロジック部分，管理者や利用者などのユーザの役割に応じて利用できるサービスを切り替える部分（ユーザ認証部），そして前述のインタラクション部の三つの小システムから構成されている．これらのシステムは図書管理システムを「サービス本体」と「サービスを利用する外部要素を受け入れるために必要な部分」に切り分けた結果作成されたものである．ユーザ認証やインタラクションは図書管理システムに限らず他のシステムにも存在するサブシステムであるため，貸し出しや検索などの機能を「サービス」という抽象的な表現を用いたモデルで表す事により，異なる機能を持ち，異なるサービスを提供する別のシステムに対して応用する事で再利用可能な部品とする事が出来るのではないかと考えた．

同時にこのPIMを上述した関連研究と同様の「変更サイクル」をプラットフォームと捉え，インタラクションに関わる様々な実装技術に依存したPSMを作成する．前回の事例ではPSM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が，後述する二つのプラットフォームの観点からこのPIMは不十分であった．これを解決するため，本稿では構築したPIM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する事で改善を行い，その適用性を評価した．

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 2 は作成したインタラクション部のPIMである．

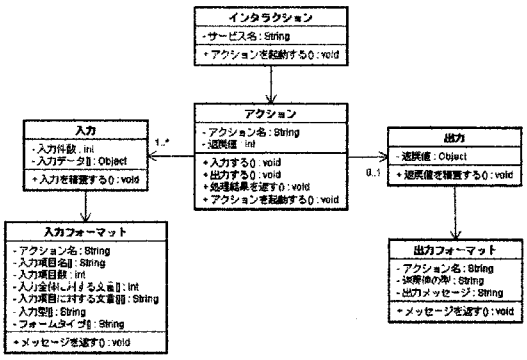

図 2 図書管理システム作成時のインタラクション PIM

「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」はサービスの「アクション」を呼び出し，ユーザからの「入力」を受付け，サービスの「アクション」に値を渡

し，更にその結果に応じて「出力」を行うという動作をするものである．

このPIMには以下の二つの問題点がある

- サービスが異なる入力を必要とする複数のアクション系列からなるケースに対応していない
- 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，ユーザがインタラクションシステムを用いて行った入力を全て一度にサービスに引き渡す仕様になっている．たとえば図 3 は「本を貸し出す」というサービスのアクティビティを表したものだが，サービス部への入力は一度しか行われていない．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「商品を注文する」サービスでのアクティビティ（図 4）を挙げる．

ユーザのアクション「商品を選択する」「個数を入力する」はサービスのアクション「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る．このようにサービスを実現するためにユーザがシステムとの応答を複数回繰り返さなければならないようなケースは，以前のモデルでは表現できない．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．

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 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．

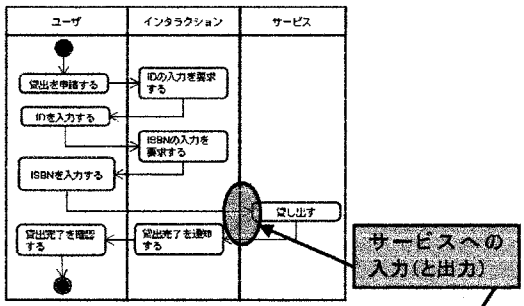

図 4 サービス「本を貸し出す」のアクティビティ

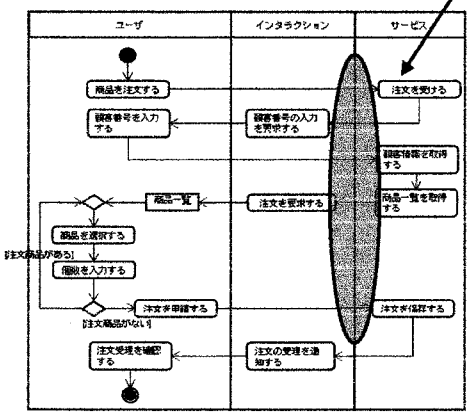

図 4 サービス「商品を注文する」のアクティビティ

表 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

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．

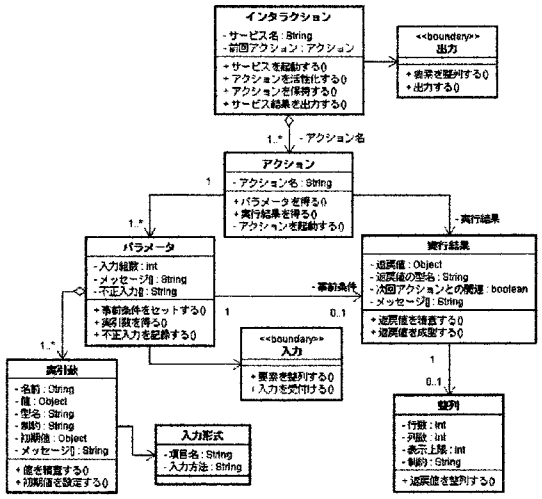

図 5 再構築後のインタラクション PIM

図 5 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,実際に「アクション」が処理に用いる値は「実引数」が表現する.「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」は渡されたメッセージなどの要素を整列し,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.

正当な「パラメータ」を得たら「アクション」を呼び出し,処理を行った後に返戻値を得る.「アクション」の返戻値を元に「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」がサービス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.

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,プラットフォーム実現に合わせた「出力」を行うために返戻値の調整を行う「整列」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.

- アクションに必要な入力の組を一度に複数得る事が出来る
- 複数種の入力形式がある

コマンドラインの場合,入力されるデータの

形式は文字列に限られており，一度の入力で得られるデータは必ず一つである．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，コマンドラインによる入力は，一度のアクション実行に対して必要なデータが揃った時点でアクションが実行されるため，同じアクションを複数回繰り返す場合は次回アクションとして同じアクションを指定する方法が一般的である．

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」の組ごとに分けて入力組数を数える必要がある．更に，入力欄はそれぞれ異なる形式を持つため，「入力形式」クラスでどのような入力フォーム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 5 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 6，図 7 はそのパラメータ取得に関する一部である．強調部分が各 PSM の独自要素である．

以下図 3 の「貸し出す」というサービスについての入力を考える．なお，ISBN とは世界共通の図書を特定するための番号を指す．

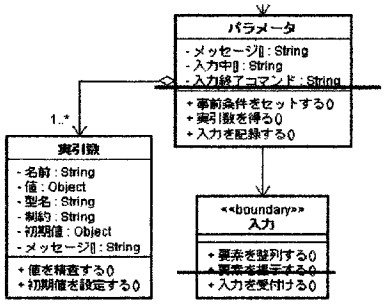

図 6 コマンドライン用 PSM(入力部)

コマンドライン入力の場合，「ID を入力する」や「ISBN を入力する」などのサービスの途中で処理を中止させたくなった時のための処理を付け加える必要がある．これを実現するために「入力終了コマンド」を設け，「実引数」での値の精査の際に「入力終了コマンド」が入力されていた場合，直ちにサービスを終了するという処理を行う．また，「入力形式」は一通りしかないため，この PSM では除去している．同じ理由でコマンドライン入力では入力欄によって入力すべき値を判断する事が出来ない．このため，ID を入力すべきなのか，ISBN を入力すべきなのかを入力を求める際に逐一ユーザに提示する必要がある．これを「要素を提示する」というメソッドを設ける事で実装している．

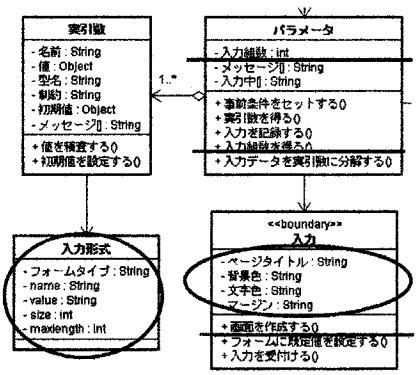

図 7 web ブラウザ用 PSM(入力部)

次に web ブラウザの場合を考える．web ブラウザでは一度に複数組の入力を扱えることから，一回の処理で複数の本を貸し出すための処理を行う必要がある．このため「入力組数を得る」というメソッドを使って貸し出し処理を行う本の冊数を数え，アクションに伝えておく．また，web ブラウザには入力フォームが複数種類用意されている．入力する「実引数」ごとに入力フォームを規定するため，「入力形式」クラスに必要な情報を記述しておく．ここで用意された情報を「パラメータ」クラスが「入力」クラスに渡す．一方「入力」クラスには入力内容の差異に関わらず web ブラウザを表示するのに必要な，「ページタイトル」や「背景色」などの情報が記述されている．これらの情報と「入力形式」クラスに記述された情報を総合して，「画面を作成する」メソッドを用いてアクション固有の入力画面を作成し，入力を受け付ける．

3.3.2. 出力

出力に関して行われる処理は，「貸し出す」「返却する」のような返戻値のない処理において「正常／異常終了をユーザに対して表示する」ものと，「検索する」などの処理において「検索結果を精査し，値を並べ替え，ユーザに対して表示する」という一連のいずれかである．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，つまり「出力」クラスである．

コマンドラインでの出力は入力と同様に出力形式が一通りしかなく，「整列」クラスで並べ替えたデータがほぼそのまま出力されるため，図 5 に改変を加えずに表現する事が出来る．

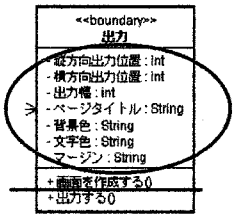

図 9 web ブラウザ用 PSM(出力クラス)

一方，web ブラウザによる出力を図 8 のようなクラスで表現した．この場合は「入力」クラス同様の画面を作成するのに必要な要素に加え，画面上のどこにどのような大きさで結果を表示するかを指定する「出力位置」や「出力幅」の要素が追加されている．これらの情報を総合し，「画面を作成する」メソッドを用いてアクション固有の出力を作成する．

以上のように，図 5 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 2 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 4 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 9 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 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 10)

図 10 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．

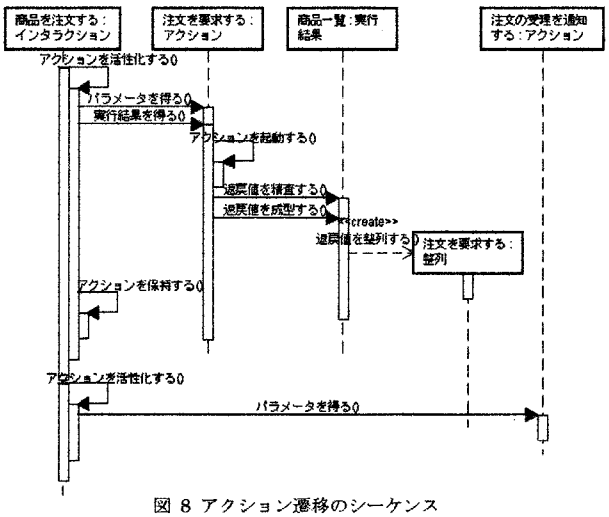

図 8 アクション遷移のシーケンス

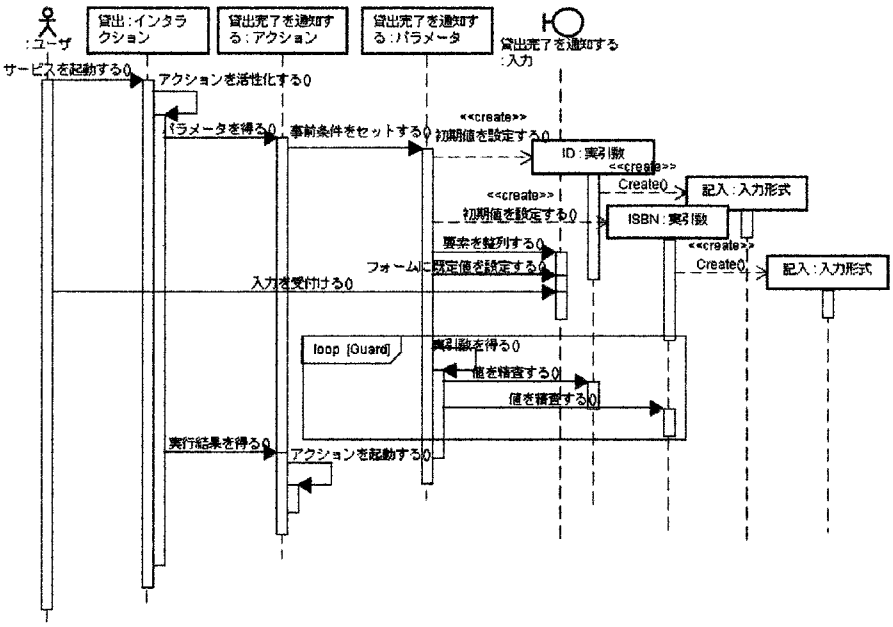

図 10 web ブラウザ用 PSM による貸出（入力）のシーケンス

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 項での二つの実験結果より，図 5 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適度に抽象化して表現し，3.1 項で述べた二つの問題を解決する事ができた．

一方，問題点として，このインタラクションシステムを使う事でシステムロジックに値を引き渡すのが煩雑になる事が挙げられる．コマンドラインからの入力であれば，パラメータを一組得るごとにアクションを呼んで処理を行うため，入力値を一意に解釈する事が出来る．しかしブラウザを用いた入力の場合，入力機構で得た値の組（複数）はインタラクション部の外部システムに対する接合の仕様上，精査を経たのちに全てを一つの配列に格納し，一括してシステムロジック部に引き渡さなければならない．このためシステムロジック側でその配列を解釈し，正しい形に分解する作業が必要になる．

パラメータの型や個数はシステムロジック側で想定しているとおりのものが格納されているため，一意的な分解作業の際にエラーが起こる事はないが，このインタラクション機構をシステム内に組み入れてシステムロジック部と接合する際，一つのオブジェクトとして引き渡される入力データを分解する作業をシステムロジック部に追加する必要が生じる．

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

システムロジックから剥離させる機能の解釈によっては，プラットフォーム実現の違いを一通りの PIM では表しきれない場合がある．たとえば同じユーザーシステム間のインタラクションでも，今回のインタラクションモデルを家電組み込み用システムに対して適用する事ができない事は明白である．

再利用性の高い部品を作るためにはその部品が異なる場面において多用される事を前提としていなければならない．しかし，あらゆるシステムのプラットフォーム実現の違いを吸収するような PIM は抽象度が高くなりすぎるため，開発資産として有用とは言えないだろう．当該 PIM を適用するシステムの範囲をどのような基準で限定し，どこまでモデルを抽象化すべきかという議論は今後 MDA が発展していく上で避けられないものである．

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用いた入出力を考慮し，その違いを吸収させる事で PIM の精細化を図った．今後は更にその拡張として Java Applet などの GUI (Graphical User Interface)による入出力を想定している．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し，モデルの適切な抽象度や正当な表現方法，PIM を適用すべきシステムの範囲の策定について考察する足がかりとしたいと考えている．

・ 参考文献

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

[2] 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.

[3] 浜口弘志，原口拓也，桐越信一，大場みち子，MDA によるコンポーネントベースモデリングの実例，情報処理学会研究報告，Vol.2005，No.86，pp1-8 (2005)

[4] 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