社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－ＳＥ－160 (16)
2008－EMB－ 9 (16)
2008／6／20

サービス非依存インタラクションモデルを用いたシステム開発の提案

天川美那† 小形真平† 松浦佐江子‡
芝浦工業大学大学院 工学研究科 電気電子情報工学専攻†
芝浦工業大学 システム工学部 電子情報システム学科‡

ユーザとシステムの相互対話を司るインタラクション部には頻繁な仕様変更が考えられる．そこで，システムが提供するサービスを基点としてインタラクション部を分割し，MDAの手法に従ってモデルで表現することで，仕様変更への対応を簡易化する．本稿ではサービス非依存のインタラクションPIMとPIMをサービスに沿って具体化するノウハウを提案し，CUIおよびGUIを用いた図書管理システムの開発事例に対して適用した．開発の過程でサービスの仕様変更を行い，インタラクションとサービスの相互独立性を確認する．

A System Development Method based on a Service Independent Interaction Model

Mina Amakawa† Shinpei Ogata† Saeko Matsuura‡
Graduate School of Engineering, Shibaura institute of technology Department of electronic engineering and computer science†
Shibaura institute of technology Department of electronic information system‡

System requirements for a interactive part, which roles mutual conversation between the user and the system, have often changed. An association between a system and users is modeled by a MDA-based interaction model based on services which should be provided by the system, so that the specification can be easily changed. This paper proposes a system development method based on a service independent interaction model as a PIM, and the rule that transforms PIM into PSM.
We discuss the independence of an interaction model from the services by applying the development process to the books management system that will be implemented with CUI and GUI.

1. はじめに

昨今の技術革新に伴い，ソフトウェアの開発サイクルは日増しに短くなってきている．より効率的なソフトウェア開発のためには再利用性を向上させるような開発手法をとる必要がある．構造化プログラミングからコンポーネントとフレームワークに至るまで，ソフトウェア工学は段階的なプロダクトのブラックボックス化を進め，再利用レベルを向上させてきた[1]．
しかし，特定技術に合わせて作成したプロダクト＝ソースコードはすぐに使えなくなってしまう．実装技術が変更された場合，要求されるサービスが同じであっても，特定技術を実装したソースコードは変更せざるを得ないためである．このため，ソースコードという完成品の状態ではなく製品を作るノウハウを残しておいて，様々な技術に対応できるような開発手法が模索されてきた．
システムには異なる様々なシステムに共通する部分と，そのシステムで提供したい業務特有の部分がある．このうち再利用したいのは前者の多様なシステムに共通する部分である．共通部分を使いまわせるのであれば，システム固有部だけを作ることでシステムを完成させられるため，開発コストは減少すると考えられる．更に，異なる責務を与えられている固有部と共通部分をあらかじめ明確に分割してシステム開発を行うことで，サービスの分析や把握を簡易化することが出来ると考えた．
本稿では提供するサービスとして図書の賃借管理を，共通部分としてインタラクションを設定し，提案したサービス非依存のインタラクションモデルを図書管理システムの仕様に沿って変換して実装する手順を示す．

2. インタラクション

抽象的なモデルを具体化して実装するにあたり，開発にMDA(Model Driven Architecture)[2]の考え方を取り入れた．OMG(Object Management Group)の提唱するMDAは，モデルとノウハウを組み合わせて実現するシステム実装過程のモデル変更ノウハウの部分を定式化し，抽象モデルを具体モデルに自動変換することで完成する．これまでのMDAの開発事例はシステム全体を表現したPIM(Platform Independent Model)に対して変換規則を設け，実装技術に特化したPSM(Platform Specific Model)に変換するという手順を踏むことで，実装技術の仕様に依存しないシステム開発を行うものが多かった．例として浜口らの研究[3]

におけ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，システム要求と実装技術という異なる二つの要素を開発サイクルというプラットフォームで分割している．つまり，実装技術を変更してもシステム要求部に変更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．
同じシステムの中に開発サイクルの異なる部分が混在していることはままある．たとえば図書管理システムでは同じサービスを提供するシステムに対してバーコードリーダ，ICカード等の様々な入出力機器を用いることが考えられ，この部分の仕様変更はサービスを提供するロジック部よりも頻繁に行われることが予想される．更に，入出力機器の変更はシステム外部に対して取得・提示できるデータを制限する場合があり，システム要求部に対しても変更を余儀なくする場合がある．このため，単純に実装機器の違いを吸収するのではなく，入出力部とサービスロジック部を完全に分離し，二つのサブシステム同士の連携でシステム要件を達成する方法を提案する．更に入出力システムのメタモデルとそのモデルを特定のサービスシステムに対して特化させるノウハウを用意すれば，異なるサービスに対して同じ入出力機器の実装を一意に提供できるようになり，再利用性が向上すると考えた．
今回の事例のように共有部分と固有部分を切り離す，つまりシステムの扱う問題領域をプラットフォームと考えた上でプラットフォーム分割をするというフェーズを最初に取り入れる事で，MDAの狙うモデルの再利用という効果を得られると考えられる．

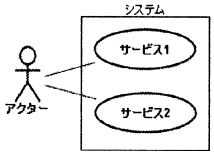

図 2.1　システムのメタモデル

本稿では異なるシステム間の共有部分としてインタラクションを分離する事を考えた．インタラクションとはアクターとサービスの間で行われるデータのやり取りを指し，図 2.1 のアクターとサービスを結びつける線，すなわち関連の部分に当たるものである．
メタモデルの表現からも分かるとおり，関連とサービスは一つのシステムを形成する部品であるが，担当する責務は相互に独立のものであり，開発サイクルも異なると考えることができる．インタラクションとサービスを初期段階で明確に分割しておくことで，システムの煩雑化を防げると考えられる．また，全てのシステムはシステム外部のアクターとなんらかのデータ交信をするので，インタラクションモデルとその具体化ノウハウを作成するのは再利用の側面から見て有効であると考えられる．ただし，全てのインタラクションを表現したモデルは抽象度が高くなりすぎてモデルとして意味のないものになってしまう．本稿では人間のユーザが画面に対して入力したなんらかのデータに対してシステムが返答するというインタラクションのみを想定して開発を行った．

3. ケーススタディ

3.1. 図書管理システム

今回作成したシステムは研究室所蔵の書籍の貸出を管理するものである．システムの機能として貸出・返却・図書の追加・図書の削除・検索の五種を考える．図書管理システムは入出力デバイスとしてバーコードリーダやICカード等様々なものが考えられる．また，ユーザに対しての情報提示を行うインターフェースも多様である．こうした仕様の変更に容易に対応できるようなインタラクションの開発として適しているという見解から，図書管理システムを事例として採用した．ここでは図書を「ISBN・タイトル・著者・出版社・貸出状態・帯出者」という六つの属性からなるオブジェクトと定義した．また，ISBN とは図書を識別するための 9 桁の正整数からなるコードである．

3.2. システムの作成手順

通常のシステム開発はユースケース分析からクラス図を定義し，実装方法を考慮してクラス図の詳細化を行ってから実装を行う．今回の実験ではインタラクションとサービスを分離するため，ユースケース分析で要求定義を行った後，アクティビティ図の作成を通してサービスのフローを分析すると共にインタラクションとサービスの責務を分割し，それぞれを異なるクラス図で定義・詳細化・実装する．ただし，インタラクションとサービスは最終的に組み合わせて一つのシステムを形成するので，インタラクションシステムのクラス図を詳細化する際は二つのサブシステムの相互作用に必要な振る舞いを併せて定義する．
具体的には下記の手順でシステム開発を行う．

- ユースケース図とシナリオを作成する
  システムの作成依頼者の要求分析を行い，システムが提供するサービスを定義する．
- シナリオに沿ってアクティビティ図を作成する
  定義した各サービスを実現するためのフローとシステムおよびユーザの振る舞い（アクション）を定義する．
- アクティビティ図にインタラクションパーティションを設ける
  ユーザとシステムの間のやりとりに注目し，受け渡されるデータの詳細や受け渡しのタイミング，データを渡すアクションを定義する．このやりとりをインタラクションシステムのフローとしてサービスパーティションから分離する．
- サービスパーティションを実装する
  インタラクションを除いたサービスを提供するロジックを実装する．これに伴い，提示するメタモデルにサービスを対応付けることで，システムがユーザに対して要求するデータの書式とそれを受け取るアクションを詳細に規定する．
- インタラクションパーティションを実装する
  サービスロジックにユーザからの要求を渡し，処理結果をユーザに提示するシステムを実装する．このとき，上で規定されたデータの書式に沿って入力精査を行う．また，データを受け取るアクションの規

定に基づいて，サービスロジック内の特定のアクションを呼び出す処理を実装する．これらの規定を実装することで二つのシステムの相互作用を可能にする．

以下，各手順について詳説する．

3.3. 要求分析

● ユースケース図・ユースケース記述の作成

図 3.1 のようなユースケースを作成した．また，例として「貸出」機能のシナリオを以下に示す．

1. ユーザは貸出を選択する
2. システムはユーザに ISBN の入力を要求する
3. ユーザは ISBN「4282282837」を入力する
4. システムは ISBN が「4282282837」である図書を検索する
5. システムは該当図書の貸出状態を調べる
6. システムは該当図書を貸出中にする
7. システムは該当図書の帯出者にユーザの ID を設定する
8. システムはユーザに貸出完了を通知する
9. ユーザは貸出完了を確認する

例外フロー

4. 該当図書が存在しない場合
   システムは図書が存在しない旨をユーザに通知し，2 に戻る
5. 該当図書が貸出中の場合
   システムは貸出状態の不正をユーザに通知し，2 に戻る

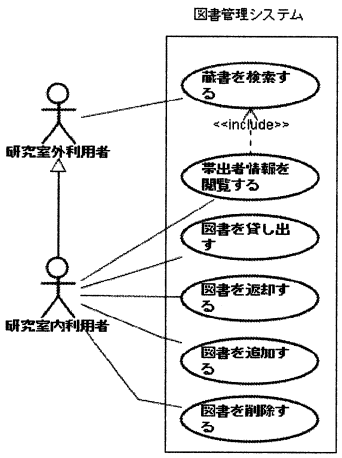

図 3.1 ユースケース図

● アクティビティ図の作成

我々は要求の妥当性に焦点を当て，要求分析プロセスで形式的なユーザインターフェースプロトタイプ生成を行う研究を進めている[4]．まずインタラクションを定義する際にユースケース記述を用いるが，自然言語の記述を読んだ場合に生じる解釈のばらつきを抑えるため，また視覚的な理解を促すために，ユースケースをアクティビティ図で表現するというフェーズを取り入れている．このとき，フローの主体や振舞いの対象を明確にするために，アクティビティ図の記述に際して以下のルールを設けている．本稿でもこのルールに則り，ユースケース記述からアクティビティ図を作成した．

1. 振る舞いの主体を明示するため，アクターとシステムの名前により構成されるパーティションを定義し，全ての要素をパーティション内に定義する．
2. アクターとシステムのインタラクションを逐次的な手順として定義するため，複数のパーティション間に跨る並列処理を定義しない．
3. アクターとシステム間で流れる入出力情報を明示するため，アクターとシステムの境界に入出力情報としてオブジェクトノードを定義する．明確な入出力情報がない遷移の場合はこの限りでない．

これに基づいて作成した貸出のアクティビティが図 3.2 である．

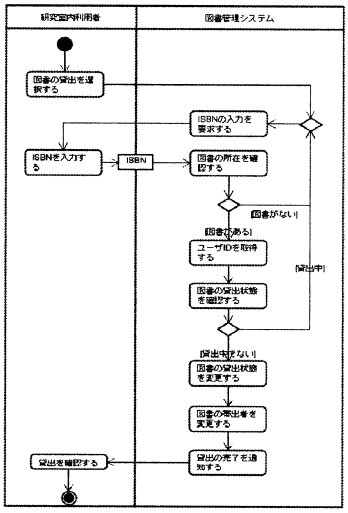

図 3.2 貸出のアクティビティ図

● 作成したアクティビティ図にインタラクションパーティションを追加

ここではユーザとシステムのやり取りを詳細化する．システムがサービスを提供するのに必要なデータを定義し，ユーザからの入力受付と精査，および入力に対応するアクションを呼び出すタイミングを決定してインタラクションのフローを作る．

通常，アクティビティ図で記述したアクションが対象領域のどのクラスの振る舞いであるかはこの時点でまだ定義されていない．しかし，インタラクションパーティションをサービスから独立させることで，インタラクションを実現するアクションを提案するインタラクション PIM に従って一意に実装することが出来る．ここでは抽出したサービス毎にアクション系列と入出力を分析しておき，3.5 項で詳述する手順に従ってインタラクションパーティションを実現する．

インタラクションパーティション追加の手順は以下の通りである．

1. システム内の外部アクターと関連するアクションを洗い出す．
2. アクター・システム間にオブジェクトノードを定義する．
3. オブジェクトノードに含まれる情報を定義する．

この定義に沿ってインタラクションが入出力の精査を行う．

4. インタラクションを通じて入出力を行うタイミングを決める．

手順に基づいて詳細化した貸出のアクティビティが図 3.3 である．これらのアクティビティ図作成作業を全てのサービスについて行う．

なお，インタラクションでは上下限や日付の制約，データ型等の定性的な制約条件についての精査を行う．システムの内部データの状態に関わらない普遍的な条件の精査をインタラクションの責務とした．

以降 3.4，3.5 項でそれぞれサービスパーティション，インタラクションパーティションのアクションについての実現を行う．

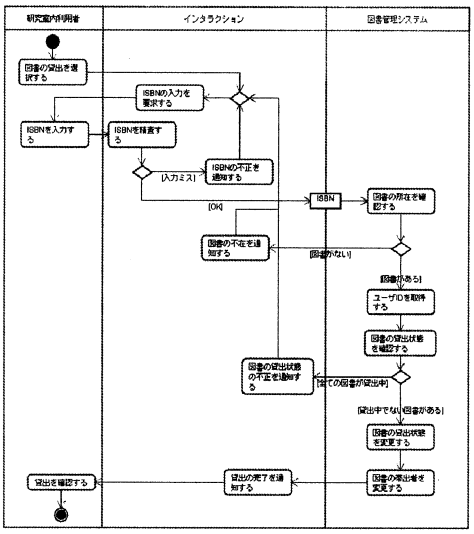

図 3.3　インタラクションを含む貸出のアクティビティ

3.4. 図書管理サービスロジックの作成

- 図書管理システムパーティションを実現するロジックの作成

ここでは図書管理システムのサービスを提供するシステムを構築する．サービスシステムのクラス分析手法は従来行われているものを採用したが，その過程で提供すべきサービスをサービスのメタモデル（図 3.4）に対応付けて分析する．サービスに外部から提供される値は正しいものとして，サービスを実現するために必要なアクションを洗い出す．それとともに，サービスに外部，内部から提供される値を取得するアクション，システムが外部に値を提供するアクションを定める．

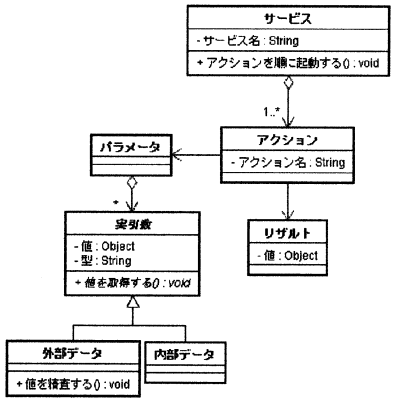

図 3.4　サービスのメタモデル

サービスは一つ以上のアクションの系列から構成され，アクションはパラメータとリザルトを一つずつ持つ．パラメータはシステム外から取得する外部データとシステム内部の値である内部データの集合である．内部データをパラメータとして用いる場合は，前回以前のアクションでシステム内部の値を取得し，それをパラメータとして与えるものである．

このメタモデルはシステム外の要因を考慮していない．つまり，システム外部から取得する必要のある外部データについては，予測した値を過不足なく取得できた場合のみを想定して表現されている．この条件下でサービスを提供するシステムの挙動について分析する．これに外部データの取得と外部へのデータ提示のために必要な処理を定義したインタラクションシステムを組み合わせることで図書管理システムを実現する．

たとえば貸出サービスの一部「本の所在を確認する」アクションはパラメータとして外部データ「ISBN」を持ち，リザルトとして処理結果を持つ．ISBN は double 型の 9 桁の整数値であり，処理結果は boolean 型の true / false の値である．こうした定義をアクティビティ図のアクション系列やオブジェクトノードをメタモデルに対応付けながら行う．サービスを実現するためのアクションはアクティビティ図作成時に洗い出してあるので，ここでは上述のパラメータおよびリザルトを踏まえたうえでそれを実現するためのメソッドを作成する．これを全てのサービスに対して行い，クラス図を作成する．

こうして作成したのが図 3.5 の図書管理システムのサービス部のクラス図である．蔵書は一冊ずつの図書の集合であり，図書は ISBN やタイトル等の属性として本を持つ．また，帯出者として研究室内利用者を持つ．

上述の通り，サービスロジックを構築する際の入出力はアクティビティ図で定義したオブジェクトノードの詳細を参照し，正確なデータが入力された場合についてのみのフローを作成する．

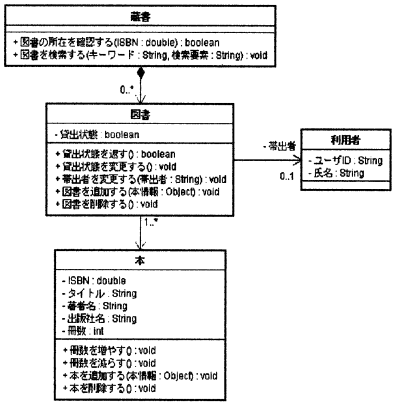

図 3.5　サービスロジックのクラス図

3.5. インタラクションシステムの作成

● インタラクション PIM について

続いて，図 3.3 におけるインタラクションパーティション部の実装を行う．図 3.4 のサービスのメタモデルをもとに，システム外部からのデータの取得と精査，およびシステム外部へのデータ提示を行うシステムを作成する．これにあたって提案するのが図 3.6 のサービス非依存インタラクションモデル（PIM）である．

インタラクション PIM はサービスのメタモデルのアクション・パラメータ・リザルト・実引数の四クラスに対応する構造を持っている．また，インタラクションクラス内にサービス名という属性を持つ．3.4 項の分析で洗い出したサービスを提供するためのアクション群およびその入出力データを対応する各クラスの中に格納する．外部データについてはインタラクションシステム内で精査を行うため，入力値に対する制約を定義しておく．

インタラクションシステムはサービスを実現するためのアクションを呼び出すことにシステム内部から必要なデータを収集し，外部入力が必要な場合は入力の要求と精査を行い，アクションに必要なパラメータを構成して起動する．これによって最終的に得られた結果をシステム外部に提示することで，サービスをシステム外部のユーザに提供するものである．

インタラクション PIM はサービスとそれを実現するアクションの系列を持つ．アクションはパラメータとリザルトを一つずつ持つ．パラメータはゼロ個以上の実引数の集合である．パラメータは図のように外部データ・内部データの集合からなるが，此処では外部データも内部データも同様に実引数という構造体で値を持つ．内部データとして値を持たせる場合はインタラクションが前回以降のアクションのパラメータおよびリザルトを当該アクションのパラメータに与え，初期値として設定する．このため，パラメータクラスは自身とリザルトクラスに事前条件として関連する．外部データを必要とする場合，インタラクションは Input クラスを通じて外部データ（今回はユーザからの入力）を取得し，当該アクションのパラメータに対して与える．全てのアクションを実行し終えた場合，また途中で処理を中断する場合に Output を用い，システム外部に対してデータを提示してサービスを終了する．

インタラクション PIM では詳細化が必要な箇所を抽象メソッドで表現してある．これらのメソッドを以下の手順に従って実装することで，システム固有のインタラクションを実現する．

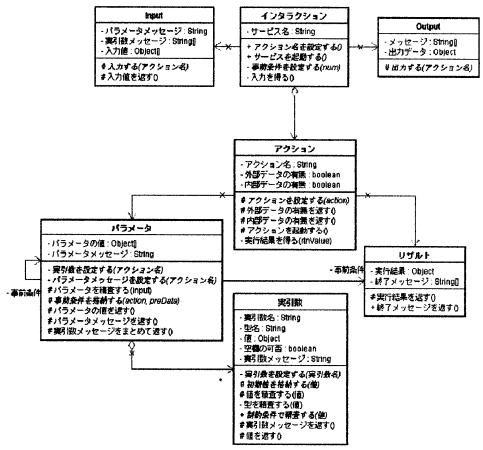

図 3.6　インタラクションの PIM

● 静的な値の格納

ロジック部の仕様に合わせてインタラクションシステム内の静的な値を定める．3.4 項で整理した各サービスの構成を参照するとよい．

➢ サービスの決定

図書管理システムの提供するサービスは「貸出」「返却」「検索」「追加」「削除」の五種である．まずはこの五種類のサービス名と，サービスを実現するために必要なアクションを定義する．

・ 「インタラクション」クラスの「アクションを設定する」メソッドを実装する．

たとえば「図書を貸し出す」という機能の場合，サービス名「lendBook」に対してアクション「chkBook（本の所在を確認する）」「stateBook（本の貸出状態を確認する）」「cngLendBookState（本の貸出状態と帯出者を変更する）」を設定する．この定義を全てのサービスについて行う．

・ 「インタラクション」クラスの「事前条件を決定する」メソッドを実装する

アクションクラスの属性「内部データの有無」が true である場合，当該アクションに対して前回以前のアクションのパラメータもしくはリザルトの値を与える必要がある．ここではアクション名からどのアクションのパラメータ及びリザルトを与えるかを指定する（3.4 項の最後に確認したアクションの流れを参照するとよい）．

たとえばアクション stateBook には前回アクション chkBook のパラメータである ISBN を与える必要がある．このとき，アクション名が stateBook であれば chkBook のパラメータを渡す，という処理を記述する．ここではパラメータクラスのオブジェクトをまるごと事前条件として与え，具体的な値の格納は「パラメータ」クラスの「事前条件を格納する」メソッドで行う．

➢ アクションの設定

・ 「アクション」クラスの「アクションを設定する」メソッドを実装する.

アクションは外部データと内部データの二種のデータを使った引数を与えて起動させる必要がある. この二種のデータはアクションによって要不要が異なるので, それを設定する.

この値について.「外部データ」はアクションがアクターからの入力を受け取る場合に true となる.「内部データ」は異なるアクションの入力/結果を入力として用いる場合に true となる.

たとえばアクション「chkBook」は外部データである ISBN を必要とするので, アクションの属性である「外部データの有無」の値を true に設定する. また,「stateBook」は前回のアクションで入力された ISBN を引数として必要とするので,「内部データの有無」の値を true に設定する. これを全てのアクションについて行う.

➢ パラメータの設定

アクションに対する引数全体を設定する.

・ 「パラメータ」クラスの「実引数を設定する」「パラメータメッセージを設定する」メソッドを実装する.

アクションに必要なパラメータは実引数の集合とパラメータを取得する際に提示するメッセージで構成されている.「実引数を設定する」メソッドでパラメータを形成する実引数の数と名前を設定する. また, パラメータメッセージをもう一つのメソッドで設定する.

たとえば「cngLendBookState」アクションには引数がないので, この場合は属性「パラメータの値」に null を設定する.「chkBook」アクションは引数に「ISBN」を持つので, 実引数に「chkBook_ISBN」を設定する.

・ 「事前条件を格納する」メソッドを実装する

アクションの「内部データの有無」が true である場合, 前回以前のアクションのパラメータもしくはリザルトがインタラクションクラスで与えられる. このデータを実引数名を元に実引数の初期値として格納する

➢ 実引数の設定

パラメータに含まれる値の詳細を設定する.

・ 「実引数を設定する」「実引数メッセージを設定する」メソッドを実装する

実引数名はパラメータクラスから「アクション名_引数名」という文字列で与えられる. これを元に実引数の型・空欄の可否を設定する.

・ 「制約条件を精査する」メソッドを実装する

システム固有だがシステムの状態によらない条件を設定し, 値がそれに合致しているかどうか調べる処理を記述する.

たとえば ISBN は「428 で始まる 9 桁の正整数」である. これはシステム内部のデータとは無関係に普遍の情報である. これを精査する.

➢ リザルトの設定

・ 「リザルトを設定する」「終了メッセージを設定する」メソッドを実装する

アクションの実行結果の型と, アクションがサービスを実現するアクション群の最後のアクションだった場合に出力するメッセージを設定する.

● 処理の呼び出し

➢ アクション呼び出しアルゴリズムの設定

静的な値の格納を終えた後,「インタラクション」クラスの「サービスを起動する」メソッドを実装する. ここではアクション呼び出しの基本・例外・代替フローを設定する.

基本フローは

・ インタラクションクラスでサービスを活性化
・ アクションとそれに連なるパラメータ・実引数・リザルトを生成
・ [内部データを与える/外部入力を受付/アクションを起動/リザルトを得る]を全てのアクションについて行う
・ 結果出力

となる. これにアクションの反復と分岐を加えてアクション呼び出しアルゴリズムを構成する.

➢ 反復

システムの仕様によっては同じアクションを引数だけ変えて複数回繰り返したい場合がある. たとえば図書の貸し出しを行う場合, 貸し出したい図書の ISBN を一度の入力で全て受け付け, データの個数分だけ同じ処理を行うというインタラクションが考えられる. この場合は入力された ISBN の数を数え, その数だけ同じ処理を行う必要がある.

➢ 分岐

分岐が発生する例外には二種類ある. インタラクションシステム内で発生する例外と, サービスロジック内で発生する例外である.

・ インタラクションシステム内での例外

つまり入出力の精査にミスがあった場合である. 図のグレーに塗られたアクションがこれに相当する.

この場合は直前の入力を精査が通るまでやり直すものとする. 実引数が複数あった場合は, 精査を通らなかったものについてのみ再入力を促し, その他のデータについてはシステム側で保持しておいて再入力の手間を省くものとした.

これに伴い, インタラクションシステム内で発生する精査中の独自例外クラスを定義しておく必要がある.

・ サービスロジックでの例外

こちらは例外という形では検知されず, 前回アクションのリザルトを元にインタラクションクラスで判断する. たとえば貸出処理の途中で, 入力された ISBN に該当する図書がなかった場合, システムはその旨を提示して ISBN 入力を要求するアクションに戻る.

アクションから得られた結果が例外フローに分岐するものであった場合, システムは以下のパターンで処理を行う

例外発生の旨を提示し,

1.フローをその場で終了

2.ある特定のアクションまでさかのぼり, 処理をやり直す

2 の場合, さかのぼるアクションはユーザからの外部データを入力するアクションに限られる. システム内部のデータで完結する処理の場合, 何度やり直しても結果は同じためである. このとき, 例外が発生する前に実行したアクションに対してユーザが与えた入力を再度与えるか否かについても設定する. これは一つの間違いのために全てのデータを再入力する手間が生じ

るのを防ぐことになるが，間違いが多い場合には逐次書き直しを行わなければならないという煩雑さの原因にもなる．

➢ 反復と分岐

更に，反復的に処理を行うために取得した複数の値の一部に例外が発生した場合を考慮する必要がある．
この処理には

1. 全ての値について処理を取りやめ，終了する
2. 全ての値について処理を取りやめ，特定アクションまでさかのぼって再開する
3. 正常な値についてのみ処理を行い，終了する
4. 正常な値についてのみ処理を行い，例外が発生した値に対してアクションをさかのぼる

というパターンがある．
つまり，複数入力がある場合に

・ 全ての入力を精査してから処理を行う
・ 一つずつ精査→実行のように繰り返す

という二種の対応があり，更にそれぞれに再入力して続けるかそのまま終了するかという違いがある．

➢ 入力デバイスの制限

この反復と分岐のアルゴリズムにはシステム制約と同時に入力デバイスの特徴も関わってくる．今回はCUIとwebブラウザGUIの二種の実装を考えているので，これらの特徴を考える．

・ CUI

入力欄は標準入力の一種のみ．一度の入力について処理を行い，また入力を要求するというインタラクションが一般的である．複数の値を一度に取得したい場合はユーザ側の入力形式に制限を加える必要がある

・ GUI

入力欄は複数種類のフォームからなる．データを一度に複数取得することが多い．一画面ごとに一つの情報を取得し，精査して次に進む，というインタラクションは一般的でない．
以上より，CUIでこのシステムを実装する場合は一度の入力に単数・複数の両方が考えられるが，GUIで実装する場合は一つの処理に必要な値を一度に取得するのが普通であると考えられる．つまり，CUIの場合は精査や実行の最中に例外が生じた時点でそれをユーザに提示するが，GUIの場合は全てを精査もしくは実行し，例外をまとめて通知するほうが自然である．
つまり

・システム自体の処理の利便性
・入出力デバイスの制限

という二つの要素を加味してインタラクションのアルゴリズムを決定する必要がある．
以上の点に考慮して「インタラクション」クラスの「サービスを起動する」メソッドを実装し，「Input」クラスと「Output」クラスを通じてユーザとのデータのやり取りを行う．

➢ 入出力

・ 「入力する」メソッドを実装する

Inputクラスではインタラクションクラスがパラメータ・実引数クラスから呼び出したメッセージを使い，適当にレイアウトしてユーザに提示する．

・ 「出力する」メソッドを実装する

Outputクラスでも同様に，インタラクションクラスから最終アクションのリザルトオブジェクトの終了メッセージを与えられるので，それを用いてユーザへのサービスの結果表示を行う．
以上の手順で図のPIMを図書管理システム用のPSM，およびソースコードに変換した．

4. 適用実験

作成したインタラクションPIMのサービスに対する独立性を検証する．サービスに依存しないインタラクションPIMの条件として

1. インタラクション仕様の変更に対して，サービスロジックに変更が生じない
2. サービスロジックの機能拡張に対し，インタラクションPIMおよび具体化手法に変更がない

ことが挙げられる．
1についてはCUI・webブラウザGUIの二種のユーザインターフェースについてPIMを具体化することで検証できたものとし，2の場合についてのみ適用実験を行う．
ここではサービス「図書を追加する」の機能拡張を行った．
図書を追加する場合，同じ図書が既に蔵書にある場合は所蔵冊数を増やすだけでその他の情報については変更されない．これに基づき，「図書の追加」というサービスに「同じ本が存在する場合は入力項目を省略する」という機能を設ける．この変更に対して3.4項の手順に添ってPIMを具体化しなおす．
まず，サービスを提供するためのアクション系列が変更になる．変更前は「図書を追加する」というアクションを呼び出していたが，変更後では「図書の所在を確認する」というアクションを先に呼び出し，該当する本があれば「既存の本を追加する」というアクション，該当するものがなければ「新規の本を追加する」というアクションにそれぞれ分岐する．これに従い，インタラクションシステム内のアクションを呼び出すアルゴリズムを変更する．更に追加・変更されたアクションに付随するパラメータとリザルトにも変更が生じる．

変更前：
「図書を追加する」アクションのパラメータ
外部データ

- ISBN
- タイトル
- 著者名
- 出版社名

結果

- 追加の成功／失敗

変更後：
「図書の所在を確認する」アクションのパラメータ
外部データ

- ISBN

結果

- 該当あり／なし

「図書の冊数を変更する」アクションのパラメータ
外部データなし

結果
- 追加の成功／失敗

「ISBN 以外の本情報の入力を要求する」アクションのパラメータ
外部データ
- タイトル
- 著者名
- 出版社名

結果
- 追加の成功／失敗

PIM のアクション系列を変更し，変更したアクションに対して以上の項目を書き加えることで，新しいインタラクションに対応することが出来た．サービスロジックの仕様変更によるインタラクション PIM への変更がなかったため，インタラクションとサービスが相互独立であるといえる．

5. 結論と問題点の考察

本稿ではシステム開発工程における再利用性を高めるための試みとして
- 多種のシステムに共通する機能の分離
- モデル＋変換という開発手順の提供

を行った．
これにより以下のような利点を得た．

- サービスロジック部の責務軽減

インタラクションをサービスロジックから分離することで，サービスロジックが実現すべき責務の内容が限定される．煩雑な入出力精査の条件と本質的なサービスの品質を分けて議論できるため，提供するサービスに抜けや欠けが生じにくくなる．

- 処理に必要なデータの明確化

インタラクション PIM を具体化する過程で，サービスを実現するために必要な入出力を厳密に定義する必要がある．これにより入出力が洗練され，半角全角や日付の規定違反等の入力ルール違反によるシステムエラーを起こりにくくする．

- ユーザインターフェースの品質向上

サービスロジックを図 3.4 のメタモデルに沿うように解析することで，提供したインタラクションモデルを異なるサービスを提供するシステムに対して一意に流用することができるようになる．
システムは単にサービスを提供するだけでなく，利用するユーザにとって使いやすいものである必要がある．このようなサービス以外にシステムに求められる要素を非機能要求と言い，以前から多く言及されている．最近は品質特性という形で分類・定義されており，ユーザインターフェースは使用性という観点で議論されている．サービスロジック部とインタラクション部はそれぞれ機能要求と使用性を満たすものであり，これらを相互独立に議論することで機能要求を満たしたまま非機能要求部の充実を図ることが出来る．また，ユーザの非機能要求を満たしたインターフェースを他のシステムに対してそのまま流用できる可能性もあり，使用性を高める開発手法として今後のソフトウェアシステム開発の品質向上に貢献できると考えている．
以下では問題点を述べる．

- インタラクション PIM の抽象度

先述の通り，インタラクションは全てのシステムに存在するものである．外部データのやり取りが機械との間で行われる組み込み開発などに今回の PIM がそのまま流用できるわけではない．モデルの粒度を定め，そのモデルを適用する対象となるシステムの範囲を区切る基準を作る必要がある．

- 具体化ノウハウの煩雑さ

本稿では抽象メソッドを含むクラス図で PIM を定義したが，メソッドを実装する過程については 3.4 項で記述したとおり，自然言語による定義にとどまっている．モデル内で定義したメソッドを用途に合わせて選択できるように細分化し，開発者による自由記述での実装範囲をなるべく少なくする必要がある．

- 実装技術のメタモデリング

図 3.4 ではサービスをメタモデリングして定義し，PIM との対応付けを分かりやすく示した．これと同様に CUI・GUI 等の実装技術の仕様についてもメタモデリングを行うことで，インタラクション PIM との一意な対応付けを行うことが望ましい．

6. 今後の課題

本稿では図書管理システムというケーススタディを通してインタラクション PIM を図書管理用インタラクション PSM に変換する過程を示し，インタラクションとシステムロジックの分離を図った．今後は別のシステム開発事例に対しても同様の PIM と具体化ノウハウを適用することで，今回用意したモデルの有効性を確かめたいと考えている．また，要求工学的なアプローチから得たユーザの非機能要求をシステムに反映する上で，インタラクションとサービスが独立であることがどう影響するかについて調査を行いたいと考えている．

7. 参考文献

[1] S.J. メラー，K. スコット，A. ウール，D. ヴァイセ：”MDA のエッセンス”，翔泳社 (2004)
[2] MDA(Model Driven Architecture), http://www.omg.org/mda/, Object Management Group
[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] 小形真平，松浦佐江子：UML の要求分析モデルからの Web アプリケーションプロトタイプ自動生成，情報処理学会研究報告，Vol.2008，No.29，pp9-16 (2008)