B-002

モデル駆動要求分析におけるエンドユーザの入力制約モデルの導入
An Input Constraint Modeling for Model-Driven Requirements Analysis

小形 真平† 松浦 佐江子‡
Shinpei Ogata Saeko Matsuura

1. はじめに

業務システム開発成功の鍵は，顧客の要求を満たすようにシステムを開発することである．そのため，開発者は，正確に要件を仕様化することと，要求仕様の要件が十分であることを顧客に保証することを両立しなければならない．さらに，要求仕様に対する開発者間の誤解を回避するために要求仕様の定義途上においても，適時に要求仕様の妥当性を確認できることが必要である．

我々はこれまで業務系 Web アプリケーション開発における要求仕様の妥当性確認の支援を目的に，プロトタイプ生成可能な UML(Unified Modeling Language)[1]要求分析モデルを提案してきた．モデルではアクティビティ図を主にアクターとシステムのインタラクションを定義し，システムのエンドユーザ視点から妥当性を確認するために，そこから Web ページビューのプロトタイプの自動生成方法を実現した[2]．提案手法は，非形式な自然言語を主に利用する要求分析手法に比べて，実システムに対する正確な要求仕様の定義に貢献したが，その反面，エンドユーザの入力項目数に比例して入力制約違反時の例外フローが増加し，可読性を維持した正確な例外フローの定義が困難となった[3]．そこで，本稿ではユーザ操作を表すアクション系列から Web ページ単位の入力値の取りうる状態を抽出し，そこに入力結果となるメッセージや遷移先の情報を加えて構成される入力制約モデルを提案する．

2. 要求分析と要求分析モデル

オブジェクト指向における要求分析手法として，ユースケース[4]分析がある．ユースケース分析では，ユーザや外部システムと言ったアクターから開発対象システムの機能を整理し，その関連する各々の境界におけるインタラクションから機能要求を分析する手法である．ユースケース分析では，一般に UML ユースケース図と非形式自然言語記述によるユースケース記述を成果物とする．ただし，ユースケース記述は標準化した形式はなく，当然成果物に対する妥当性確認方法も属人性が高い．しかし，ユースケース分析を踏襲した研究が多く提案されており[5,6,7]，実装技術を分離した要求の分析観点と，多種ドメインに対するインタラクションという分析観点の汎用性の高さから，ユースケースは開発者にとって親和性が高いと考えられる．

われわれの提案手法もまた，従来のユースケース分析を踏襲し，全ての成果物を UML を用いて準形式化している．提案手法の特徴としては，開発者が要件を十分に理解できるように，フローとデータの両側面を最も具体的なレベルで表現するモデルを成果物に導入することや，実システムのイメージに対する各種モデル図の責務を明確に理解できるように，モデルを 1 種類追加定義するごとに段階的にリッチ化する UI(User Interface)プロトタイプ生成を実現した．

図 1 はこれまでに提案してきた手法の要求分析プロセスを示している．各データは直前の各ステップにおける成果物や指摘を表す．また，図 2 に提案手法における「図書管理システム」の「図書を検索する」サービスを例として，要求分析モデルと自動生成プロトタイプの例を示す．次節以降で図 1 に示すステップごとにその作業内容を説明する．

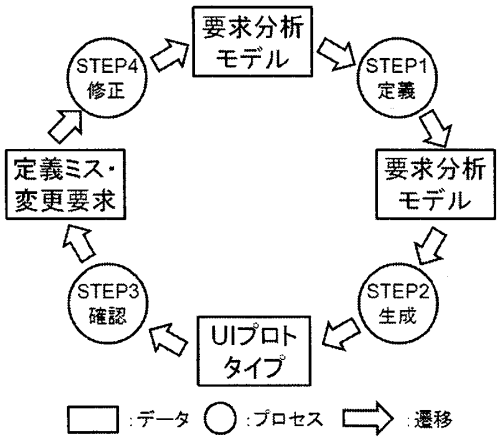

図 1 要求分析プロセス

2.1 定義

開発者は，UML モデリングツール astah*[8]を用いて要求分析モデルを定義する．最終製品となるシステムでは，ユーザに可視となる入出力データ，入力と出力の関係，ユーザ操作の容易性が顧客の要求を満たすための重要な要素となる．また，拡張開発に見られるように，開発コストの低減を目的として一部の要件の実現を既存システムの機能で代替するために，開発システムと既存システムとの連携も明確にする必要がある．この背景の下に機能要求の分析を主として，開発対象システムの処理，ユーザの操作，外部システムの機能のインタラクションの定義を目的に，アクティビティ図にシステムが提供するサービスを単位としてインタラクションを定義する．

アクティビティ図では縦パーティションには，図 2 の“利用者”といったアクターを定義し，横パーティションには，図 2 の“図書検索入力”といった入力状況を表す名前を定義する．また，アクションを用いて“図書名を入力する”といったユーザの操作や“分類を読み込む”といったシステムの処理を定義し，また，オブジェクトノードを用いてアクターとシステム間で入出力される“検索入力”や“図書”といったデータを表現する．

†芝浦工業大学大学院工学研究科・日本学術振興会特別研究員 DC，Graduate School of Engineering, Shibaura Institute of Technology・JSPS Research Fellow
‡芝浦工業大学システム理工学部電子情報システム学科，Department of Electronic Information Systems, College of Systems Engineering and Science, Shibaura Institute of Technology

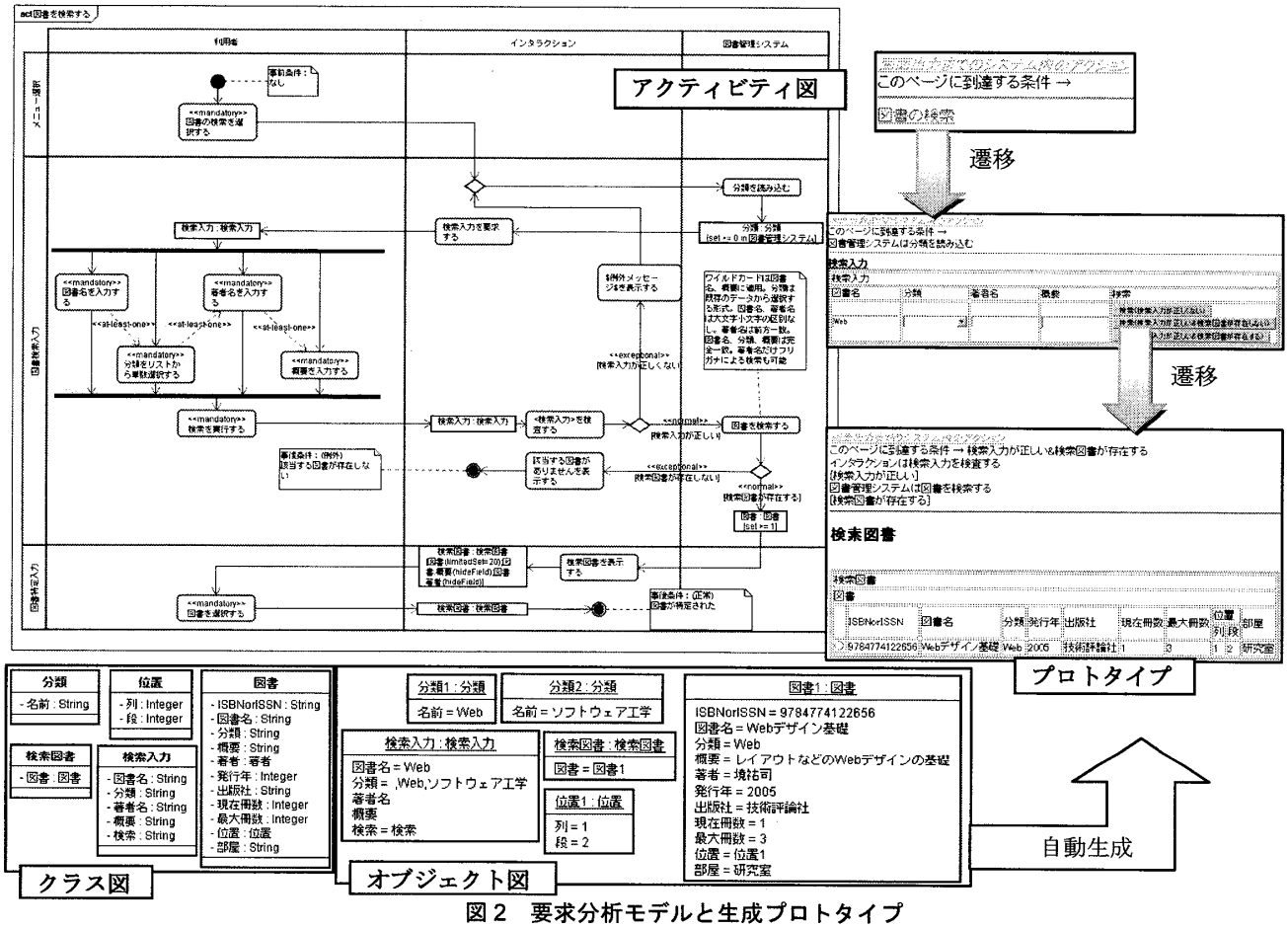

図 2　要求分析モデルと生成プロトタイプ

そして，クラス図にクラスとしてデータ構造を定義し，オブジェクト図にインスタンス仕様として具体値を定義する．astah*では，クラスを基点としてオブジェクトノードとインスタンス仕様を明確に対応付けることができるため，これら 3 種の図を正確に連携させることができる．例えば，図 2 におけるオブジェクトノードである“図書：図書”とインスタンス仕様である“図書 1：図書”は，コロン後の表記がクラス名を表している．提案手法では，具体値による値の形式や値の入力時期の十分な理解を促進するため，オブジェクト図の導入や，インタラクションを表現するアクティビティ図中の特定のパスに基づき，実際の操作例を具体値を用いて表現したシナリオの導入を実現している．

2.2 生成

開発者は，プロトタイプ自動生成ツールを用い，要求分析モデルから Web ページビューのプロトタイプを自動生成する．本ツールでは，機能を持たない紙芝居相当のプロトタイプをユーザビューとシステムビューの 2 種類のビューで生成することができる．ユーザビューのプロトタイプは，システムのエンドユーザの視点から入力項目や操作手順および具体値を含めた出力内容の妥当性を確認するために用いられる．また，システムビューのプロトタイプは，外部システムとの連携を含めてシステム内部の手続きとその途上で入出力されるデータの妥当性を確認するために用いられる．

プロトタイプのページ分割と入力項目は，アクティビティ図のアクション系列とその主体者によって決定される．ユーザビューのプロトタイプでは，図 2 の場合，基本として入力状況を単位としてページを生成し，“利用者”のアクション系列により入力項目を決定する．入力方法は，実装技術とは非依存なアクションの動詞に対応づく提案手法独自の Web ページの入力フォーム要素から決定される．例えば“入力する”を動詞に持つアクションはプロトタイプではテキストボックスに変換される．システムビューのプロトタイプでは，ユーザの入力を受け付けてから再度出力するまで単位としてページを生成する．

データ構造はオブジェクトノードに紐づくクラスから表現される．プロトタイプ上では，テーブル表現がデータ構造の表現に相当する．具体値はクラスに紐づくインスタンス仕様により表現し，テーブルに行単位で表現される．

また，プロトタイプ生成方法としては，アクティビティ図のフローを網羅するように生成する方法と，“開始から終了”または“開始から例外による再入力”までの特定のフローパスに基づいて生成する方法の 2 種類あり，後者がシナリオとなる．前者の網羅的なプロトタイプでは，ある入力状況から次の入力状況まで取りうるパスごとにリンクを生成し，パスごとに通過する分岐条件(ガード)を連結した内容をリンク名に併記する．後者のシナリオは，フローパスの抽出はプロトタイプ自動生成ツールが行い，そのパス中に逐次現れるオブジェクトノードに対する具体値を開発者が定義することで完成する．

2.3 確認

生成されたプロトタイプを通して，要求分析モデルを確認する．その状況は 2 種類ある．第一に，開発者が，要件を正確にモデル化できたかどうかを確認する．この場合，開発者間の共通理解を図るためのモデルレビューにプロトタイプを用いる．第二に，顧客が，要求を満たすかどうかの観点からモデルの内容を確認する．

2.4 修正

開発者または顧客の確認作業から発見した定義ミスや変更要求に基づいて，開発者が要求分析モデルを修正する．例えば，入力項目不足の修正や，操作手順の改善の要求はアクティビティ図に反映し，構造化されたデータの過不足の修正はクラス図に反映する．また，値の形式や値の生成時期に対する想定の誤りの修正はオブジェクト図やシナリオに反映する．

3. 現状の提案手法の利点と問題点

3.1 利点：システムイメージと整合したモデリング

われわれが UML モデルの基礎知識をもった本学学部 4 年生を対象に行った要求分析実験[2]において，提案手法と，非形式な自然言語記述によるユースケース記述を主とした仕様と手動作成画面イメージを用いた方法(以下，便宜上従来手法)と比べた場合，従来手法では，仕様中のデータ記述と画面イメージ上の入出力項目の不整合だけでも 4〜6 割程度存在した．例えば，画面イメージではデータ登録完了後に当該データを表示する想定であっても，仕様中にはそのような定義は存在しなかった．提案手法ではモデルの定義がそのままプロトタイプに反映されるため，イメージ通りにプロトタイプが生成されるまでモデルを洗練することになる．従って，プロトタイプで保証された内容はモデルでも保証できているため，このような問題は発生しない．

また，作業時間として，基本的には提案手法では学習時間も含めて従来手法よりも低減できた．これは，モデリング時間は提案手法の方がかかっていたが，画面イメージ作成時間をほぼ排除できたためである．

これらの結果から，提案手法ではモデルの基礎知識は持っているが運用に不慣れな開発者でも，生成されるプロトタイプを確認しながらシステムのイメージに沿って要件をモデルに定義しやすくなった．また，アクティビティ図やクラス図といった振舞いと構造の観点から分離されたモデルについて，統合した観点の妥当性をプロトタイプにより確認しやすくなり，モデル間の整合性を図りやすくなったことが結果として表れていた．

3.2 問題点 1：モデルの複雑化

前節の利点から，提案手法では実システムのイメージに沿ってモデリングするように開発者を誘導することができた．しかし，正確にモデリングを行う場合，図 3 に示すようにアクティビティ図のフローが複雑化し，著しく可読性が低下する問題が発生した．これは，1 入力状況における入力項目数に比例して，その入力検査と例外への条件を正確にモデル化した場合に発生しやすい．図 3 の場合，入力項目を項目ごとに全て検査してから，例外が発生する項目全てに対するメッセージを出力するようなフローとなっているため，長大なフローとなっている．

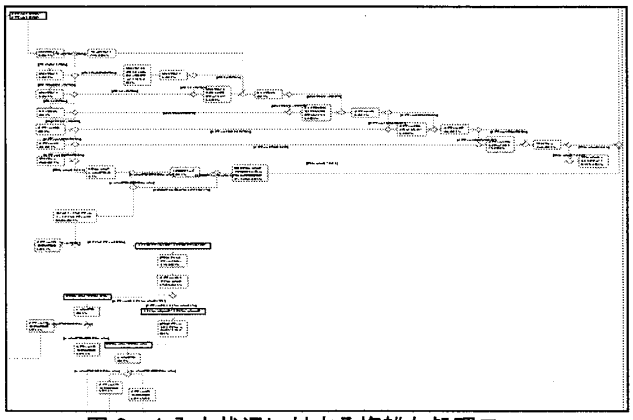

図 3　1 入力状況に対する複雑な処理フロー

3.3 問題点 2：非形式な条件記述による曖昧性

これまで提案手法では，インタラクションフロー中に必要となる分岐には，必ず条件を記述することを義務付けていた．しかし，分岐条件は非形式な自然言語で記述されるため，分岐条件で想定した値の制約とシナリオで想定した具体値が整合するかどうかの形式的な検証が行えなかった．

4. 入力条件の形式化と入力制約モデルの導入

前章の問題を解決すべく，システムが受け付けられるユーザの入力方法と入力値の制約を明確化し，そこから取りうる入力項目の状態(以下，入力状態)と，入力状態ごとにフローの遷移先や正常/例外といった種別や例外となる入力状態に対する出力メッセージを入力状態表として定義する．入力制約モデルはそれらの制約定義と入力状態表から構成される．そして，入力制約モデルをアクティビティ図と適切に連携できるように規定することで，モデルの複雑化を回避し，かつ形式的な条件記述を導入する方法を提供する．

また，入力制約モデルの導入により，ユーザが取りうる全入力状態を明確化することで，ビジネスロジックで対処すべき正常入力時の入力状態を開発者間で正確に共通理解でき，ビジネスロジックに現れる処理の条件を抽出する基礎とすることが期待できる．

開発者が入力制約モデルを定義する動機として，2 つの状況を想定する．第一に，開発者がビジネスロジックとして想定した処理やその事前条件といったサービスが成立する要件を満たすように，ユーザが入力すべき項目の定義や，入力値の限定を目的とする．第二に，システムへの入力方法や入力値の形式または入力の既定値に係わる要求がある場合に，開発者が当該要求を仕様としてモデルに定義する．

最終的には，入力制約モデルに基づき，具体値レベルまで定義した要求分析モデルのシナリオの妥当性に対する形式的な検証を可能にし，また，入力状態の正常/例外を判定するためのバリデーションコードを生成することで下流工程における開発支援の実現を目指す．本稿では入力制約モデルまでを説明する．

新提案では，単に漠とした要求から制約を形式的に定義するのではなく，これまでの提案手法により開発者が各サービスを具体レベルまで理解することを前提とすることで，従来の形式仕様記述の導入の難しさを緩和できると考える．

4.1 入力制約モデルの形式と制約の種類

入力方法の制約の種類は，以下の 4 種類を想定する．

(1) 必須・任意入力：項目に必ず入力しなければならない場合と必ずしも入力しなくても良い場合である．サービス成立の観点ではシステムの処理に用いるデータや出力すべきデータは必須入力となる．任意入力は，サービスの実行結果をリッチ化するために用いられるデータである．例えば，検索機能に付随する一般的なソート機能における昇順・降順やソートキーへの入力は任意入力となる．

(2) 選択入力：1 グループの入力項目の中で，いずれか 1 つの入力項目が必須入力となれば，他の入力項目が任意入力となる場合である．サービス成立の観点では最低限 1 つの入力がなければ，要件が満たされない場合である．例えば，図 2 では図書を検索するために“図書名”，“分類”，“著者名”，“概要”の少なくとも 1 つに入力がなければ，検索処理が成立せず，また，入力された項目数に応じて，検索したデータを絞り込む機能を想定している．

(3) 同時入力：ある項目に入力があった場合，別の項目にも入力が必要となる場合である．サービス成立の観点では常にセットで入力されるべき項目群を識別する．例えば，図書の“著者名”に入力があれば“著者名フリガナ”にも入力を必要とする場合である．

(4) 値依存入力：ある項目の値に応じ，別の項目の値の入力値が決定する場合である．サービス成立の観点では複数項目間において満たすべき値間の関係がある場合である．これらの値間の関係は既定値の抽出の基礎となる．例えば，“年月”が決定して初めて，“日”の入力範囲が決定するような場合がある．

一方，入力値の制約の種類は，以下の 2 種類を想定する．

(1') 値の範囲：典型的なブラックボックステストの技法である“同値分割”や“境界値分析”に見られるように，分岐条件に係わるような処理を決定するための値の範囲を指定する．数値型であれば数値の範囲を指定し，文字列型であれば，文字数を指定する．

(2') 値の形式：特定の文字列を内容に含む必要がある場合に指定する．例えば，郵便番号では 3 桁目と 4 桁目の間にハイフンが必要な場合に，値の形式を指定する．

4.2 入力方法の制約定義

4.2.1 必須・任意入力

図 4 に示すように必須入力の場合は，ユーザ操作となるアクションに対し，<<mandatory>>ステレオタイプを定義する．また，任意入力の場合は，<<optional>>ステレオタイプを定義する．本制約においては，必須入力の項目に入力が無い場合に，例外が発生する．なお，ステレオタイプ未定義時は，アクションは必須入力とみなす．

4.2.2 選択入力

図 5 に示すように選択入力の場合は，ユーザ操作となるアクションに対して，関連のある項目に対するアクション群に依存を定義する．依存には<<at-least-one>>ステレオタイプを定義する．そして，本制約においては依存の方向は意味を持たないものとする．

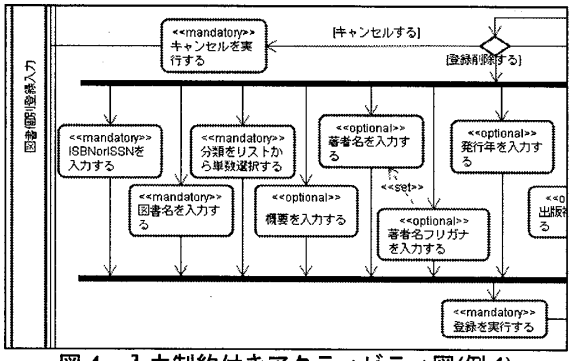

図 4　入力制約付きアクティビティ図(例 1)

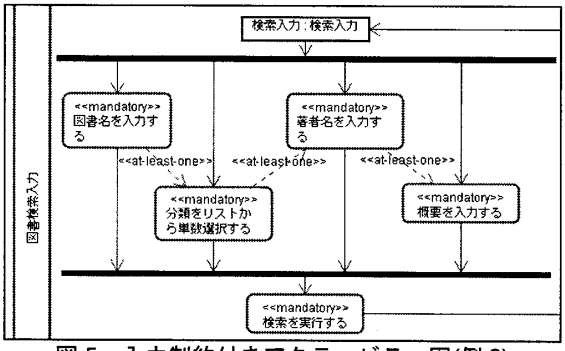

図 5　入力制約付きアクティビティ図(例 2)

4.2.3 同時入力

図 4 に示すように同時入力の場合は，ユーザ操作となるアクションに対して，関連のある項目に対するアクション群に依存を定義する．依存には<<set>>ステレオタイプを定義する．そして，本制約においては依存の方向は意味を持たないものとする．

4.2.4 値依存入力

図 6 に示すように値依存入力の場合は，ユーザ操作となるアクションに対して，関連のある項目に対するアクション群に依存を定義する．依存には<<value-specified>>ステレオタイプを定義する．依存の方向については，依存先(矢印の終点)の値が決定すると依存元(矢印の始点)の値の範囲が決定するという関係とする．値の範囲の定義は，4.3 節において入力値の制約として説明する．

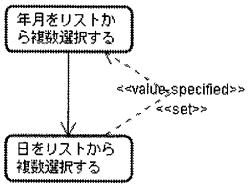

図 6　入力制約付きアクション系列(例 3)

4.3 入力値の制約定義

値の範囲と値の形式については，図 7 に示す表形式により表現する．図 7 は図 8 のユーザ操作のアクション系列（一番左のパーティション中）を例とした入力値の制約である．以降，表の各項目について説明する．

項目名(自動)	型(自動)	形式(手動)	下限値(手動)	上限値(手動)	依存項目(自動)	依存項目値(手動)
年月日	String	III/II/II	10	10		
検索	String					

図 7　入力値の制約定義

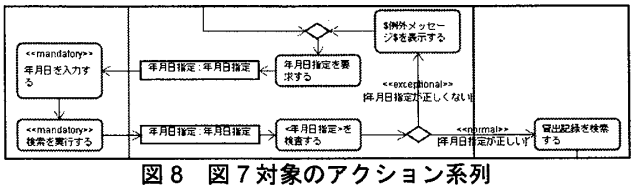

図 8　図 7 対象のアクション系列

“項目名”はアクションにおいて入力対象となっている項目である．“(自動)”は要求分析モデルから値が形式的に抽出できるものである．

“型”は UML 標準の形式仕様記述言語である OCL(Object Constraint Language)[9]の基本型である String, Integer, Real, Boolean の 4 種類を想定する．型は入力フォームのデータ構造を表すクラスの属性から抽出される．

“形式”は値が取るべき形式であり，I:整数値，R:実数値，S:文字をメタ文字として，形式を規定する．これ以外の文字は文字または数の実値とみなす．例えば，”IIII”は 4 桁の整数値を表す．エスケープ文字としては'\'を利用する．値が未入力である場合は，制約が存在しないことを表す．以降の項目についても未入力の場合は同様とする．また，“(手動)”は開発者が手動で入力する項目である．

“下限値”はシステムが受け付け可能な数値の下限値または文字列の下限文字数を表現し，同様に“上限値”は数値の上限値または文字列の上限文字数を表現する．

“依存項目”は，前節の入力方法の制約として依存関係にある入力項目の一覧が表示される．複数項目ある場合は，セミコロンによって区切られる．“依存項目値”は，依存関係にある項目に対する具体値である．この値も，依存項目の順番に併せてセミコロンによって区切られる．この値は，値によって当該入力項目の“形式”，“下限値”，“上限値”のいずれかが変化する場合のみに定義する．

4.4 入力状態の明確化と入力状態表

入力形式及び入力値の制約がモデル化された後に，当該モデルに基づき，図 9 に示す入力状態表を半形式的に抽出する．以降，各項目について説明する．

年月日指定:年月日指定				年月日				検索		
	入力状態種別	例外メッセージ	遷移先条件(ガード)	入力	形式	範囲	入力	形式	範囲
パターン1	正常	-	年月日が正しい	○	○	○	○	-	-
パターン2	例外	年月日はyyyy/mm/ddの形式で10桁で入力してください	年月日が正しくない	○	×	×	○	-	-
パターン3	例外	年月日は10桁で入力してください	年月日が正しくない	○	○	×	○	-	-
パターン4	例外	年月日はyyyy/mm/ddの形式で入力してください	年月日が正しくない	○	×	○	○	-	-

図 9　入力状態表

クラス分類済オブジェクトノード名(1 行 1 列目)，入力項目名(1 行目右方)，パターン識別子(3 行目以降 1 列目)，“入力状態種別”，“入力”，“形式”，“範囲”は 4.2 節と 4.3 節の制約定義及び要求分析モデルから抽出できる．一方で，“例外メッセージ”，“遷移先条件(ガード)”は開発者が手動で入力する．

ここで主な項目について詳細に説明する．“入力状態種別”は，入力項目の制約が全て満たされた入力の場合に“正常”となる．また，それ以外では“例外”となる．

“例外メッセージ”は，例外となる各入力状態時に出力されるメッセージである．この項目の導入は，入力項目の増加に比例して例外となる入力状態が増加することを背景に，入力状態ごとの例外メッセージを忠実にアクティビティ図に定義した場合のモデルの複雑化を回避するためである．例外メッセージの値の受け皿は，図 8 のアクティビティ図中の“$例外メッセージ$を表示する”アクションとなる．

“遷移先条件(ガード)”は，入力状態ごとに次の遷移先を指定するために定義する．これは，アクティビティ図上における次の入力状況までのフローパスに現れるガード群を“&&”で連接したものを定義する．

各入力項目に与えられる“入力”は入力の有無を表す．“形式”は入力値の制約を満たしているかを表し，“範囲”は下限値・上限値の範囲を満たしているかを表す．値は“○”“×”“－”の 3 種類であり，それぞれ“満たしている”“満たしいていない”“無関係”を表す．これらの入力状態は入力方法及び入力値の制約から形式的に抽出することができる．

パターンの状態爆発を防ぐために，不要なパターンを排除することを目的として制御フローの形態や，入力方法の制約を利用する．例えば，図 8 のアクションの“年月日を入力する”と“検索を実行する”は逐次の関係であり，必須入力である．この場合，前アクションの実行なくして，後アクションの実行が実行できないため，必須入力である“年月日”が入力無しであり，かつ“検索”が入力有りであるパターンを抽出する必要はない．

入力値の制約は，“形式”と“範囲”の入力状態を決定するために用いられる．このとき，制約が未定義の場合に“－”の値が定義される．

5. 関連研究

三部ら[6]は，われわれと同様にユースケース駆動によるプロトタイプ自動生成を含めた要求獲得方法を提案している．本手法では，シナリオパターンを組み合わせてシナリオを定義し，シナリオパターンと対応づいた画面パターンからプロトタイプを自動生成する．本手法の方針はシナリオベースの仕様の定義とプロトタイプ生成による仕様の視覚化を実現することで，低い作業時間コストで仕様の確認効果を向上することをねらいとする．しかし，これまでの提案手法と同様に，どの項目をどの場面で入出力するかまでは指定できても，各項目の入力制約は定義することはできない．提案手法では，これまでの要求分析モデルの拡張という形で，当該モデルと接点を持たせながら入力制約モデルを定義できる．また，具体値を明示的に導入している時点で，条件の形式化後におけるシナリオに基づくモデルの妥当性検証の実現可能性がある．

Raza ら[10]は，テスト支援を目的に UML2.0 の相互作用概要図を主としたモデルから表現したシナリオに沿ってテストパスを自動生成する手法を提案している．相互作用概要図では，システムの処理フローを定義し，アクションまたはシーケンス図で表される操作に対する事前条件・事後条件の定義に OCL 式を用いる．本手法では，開発者が OCL を定義するまでにモデルの妥当性をどのように確認できるかについては言及していない．しかし，形式仕様記述言語の導入を容易にするためには，形式仕様記述言語で記述する対象を具体レベルで理解し，妥当性を確認できることが重要であり，そのような理解を持たない状態で完全に十分に制約を定義することは難しい．提案手法では，開発者がプロトタイプによりモデルを実システムのイメージと対比しながら具体値レベルまで理解していることを前提とした入力制約モデルの導入を想定している．そして，入力制約モデルでは入力方法の観点と入力値の観点から制約を

整理することによって，条件の形式仕様記述の導入が容易となることが期待される．

6. 結論

本稿では，われわれはこれまで提案してきた要求分析モデルに対し，条件値の形式化を目的とした入力制約モデルを新たに提案した．これまでの提案ではシステムのエンドユーザに可視部分，即ちシステムとユーザの境界を中心に，具体値レベルまでのアクターとシステムのインタラクションを定義し，そこからのプロトタイプ生成を可能にすることで，開発者・顧客の両者に対するモデルの理解性を向上し，正確なインタラクションのモデリング支援の実現をコンセプトにしてきた．

本稿の入力制約モデルも同様に，開発者が具体値レベルまでモデルを十分に理解していることを前提として，ユーザの入力方法及び入力値の制約を形式化することで，形式仕様記述言語の導入の難しさを緩和することを目指す．

今後の展望として，第一に，入力状態表の生成アルゴリズム，OCL 言語による制約への変換，Web アプリケーションにおける 1 つの実装技術として JavaScript によるバリデーションコードの生成までをツールとして実現する．第二に，これらの入力制約モデルの定義の容易性や定義内容の十分性を含めた有効性を評価する．

参考文献

[1]UML：http://www.uml.org/
[2]小形 真平, 松浦 佐江子, “UML 要求分析モデルからの段階的な Web UI プロトタイプ自動生成手法”, コンピュータソフトウェア, Vol. 27, No. 2 (2010), pp.14-32.
[3]Ogata, S. and Matsuura, S., “Evaluation of a Use-Case-Driven Requirements Analysis Tool Employing Web UI Prototype Generation”, WSEAS TRANSACTIONS on INFORMATION SCIENCE and APPLICATIONS, Issue 2, Vol. 7 (2010), pp.273-282.
[4]Jacobson, I., Christerson, M., Jonsson, P. and Övergaard. G., “Object-oriented software engineering: A usecase driven approach”, Addison-Wesley Publishing (1992).
[5]中鉢 欣秀, 小林 孝弘, 松澤 芳昭, 大岩 元, “シナリオの図解化によるユースケースモデリング”, 電子情報通信学会論文誌, Vol.J88-D-I, No.4 (2005), pp.813-828.
[6]三部 良太, 河合 克己, 竹内 拓也, 石川 貞裕, 福士 有二, “Web アプリケーションのユースケース駆動プロトタイプによる要求獲得方法”, 情報処理学会論文誌, Vol.49, No.4, 1669-1679 (2008).
[7]Somé, S.S., “Use Case based Requirements Validation with Scenarios”, Proc. of the 13th IEEE International Conference on Requirements Engineering (RE’05), (2005), pp.465-466.
[8]astah*: http://www.change-vision.com/
[9]OCL：http://www.omg.org/spec/OCL/2.0/
[10] Raza, N., Nadeem, A. and Iqbal, M.Z.Z., An automated approach to system testing based on scenarios and operation contracts. The seventh International Conference on Quality Software, (2007) pp.256-261.