2008－SE－159 (2)
2008／3／17

# 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

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

我々は顧客による要求仕様の妥当性確認を目的として，UMLの要求分析モデルからのユーザインタフェースプロトタイプ自動生成ツールを開発した．本手法は，顧客がシステム全体のフローや情報を確認するためのプロトタイプを生成するのみならず，システムに入力・蓄積される具体的なデータを取り込んだプロトタイプを生成し，顧客のシステム用途の理解を高める．また，開発者がモデルとプロトタイプの対応を理解するため，段階的なプロトタイプ生成を提供する．段階的とは，アクティビティ図，クラス図，オブジェクト図を順次定義する各段階のプロトタイプ生成を指す．本稿は，図書検索Webアプリケーションを事例に本手法の有用性を議論する．

# An Automatic Generation of Web Application Prototypes based-on Requirements Analysis Models in UML

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

To support prototyping and validation, we have developed the tool that generates user interface prototypes from requirements analysis models in UML. The prototypes generated in our approach represent UI transitions and I/O items on the whole system and provide the concrete data on the workflow so that the customers can easily confirm the system behavior. It also supports an incremental prototype generation so that developers can understand correspondence of models and the generated prototypes. The incremental prototype generation means the prototype generation at each stage where activity diagrams, a class diagram, and object diagrams are defined. We discuss the effectiveness of the generated prototype by applying our approach to library search service development.

## 1 はじめに

システム要求はシステムが提供すべきサービスを定義し，システム運用に関する制約条件を規定するものである．このシステム要求に関する問題の一つとして，要求を提示する顧客と開発・保守を行う開発者間に誤解が存在することが挙げられる[1]．

この問題を解決するための手法として，プロトタイピングが提案されている．プロトタイピングは，実システムの模型(プロトタイプ)を通して，顧客が要求仕様の妥当性確認を行う手法である[2]．顧客は，プロトタイプ上で業務手順，入出力項目やデザインを確認することで，自身のイメージとシステムの整合性，追加要求や変更要求の必要性を確認する．また，どのステークホルダに対しても理解容易なプロトタイプを媒体に要求を議論することで，ステークホルダ間の誤解を効果的に解消する．

方で，要求仕様は如何に効果的な妥当性確認を行っても，正確に厳密に定義されなければシステムの仕様として意味を成さない．そのため，要求仕様の定義方法が重要である．オブジェクト指向分析・設計の標準記法であるUML(Unified Modeling Language)[3]は，システムのデータ構造や振る舞いを表現するための非曖昧な十数種類のモデルを提供する．近年では，要求仕様を厳密に定義するために，UMLのモデルからプロトタイプを生成する方法が提案されている[4,5,6]．我々も同様にUMLのモデルからプロトタイプを生成する方法を提案している[8]．プロトタイプの自動生成では，プロトタイプ開発の時間コストが軽減できる代わりに，開発者が生成元となるモデルと生成後のプロトタイプの対応を理解できるかが重要である．しかし，これらの研究では，一度に定義した複数種類のモデルからプロトタイプを生成するため，この対応の確認が困難である．本研究では，モデルを一種類ずつ追加定義する毎にプロトタイプが生成できるため，開発者はこの対応の確認が行いやすい．また，プロトタイプは，顧客が効果的な要求仕様の妥当性確認を行うため，システム用途をイメージできなければならない．本研究では，顧客がシステム用途をイメージできるプロトタイプとして，これらの研究では考慮されていない，システムに入力・蓄積される具体的なデータを取り込んだプロトタイプの生成方法を提供する．

本稿の適用事例である図書検索システムでは，検索機能の実現に焦点を当てている．本研究の有用性は，顧客が検索機能に対して確認すべき点をプロトタイプで表現できているかの観点で議論する．次節以降では，図書検索システム開発上のモデルを示す．

## 2 プロトタイプと要求分析モデルの定義

### 2.1 プロトタイプ

本研究では Web アプリケーションを対象とする．そのため，HTML(Hyper Text Markup Language)により構成されるユーザインタフェースプロトタイプを採用し，顧客は表 1 に示す五つの事項を確認する．

表 1 プロトタイプにおける顧客の確認事項

| 分類 | 確認事項 |
|---|---|
| 業務手順 | ・サービス実行手順としての画面遷移<br>・サービスに対する入出力項目の過不足 |
| 入出力項目 | ・入力項目の入力形式<br>・構造化すべき入出力項目<br>・出力項目の具体的なデータ |

「サービス実行手順としての画面遷移」は，目的のサービスを実行するために通過する画面遷移数や，各画面に割り当てられる情報の見やすさと使用性を確認する．「サービスに対する入出力項目の過不足」は，業務上必要な項目の過不足や，使用性を向上させる入力要素(検索機能の例であれば，ソートキーやソート順序など)の必要性を確認する．「入力項目の入力形式」は，情報の入力方法を確認する．例えば，テキスト形式の入力形式ならば，任意の文字列を手作業で入力することがイメージできる．また，ラジオボタンやチェックボックスならば，システムが表示する内部データを選択することがイメージできる．「構造化すべき入出力項目」は，理解の容易性や使用性の観点で入出力項目の構造の妥当性確認する．「出力項目の具体的なデータ」は，顧客がシステムを用いて業務を遂行する上で，システムに入力・蓄積されるデータの具体的な値を確認する．

### 2.2 要求分析モデル

要求分析段階で定義する UML のモデルを要求仕様として扱い，プロトタイプ自動生成に必要なアクティビティ図，クラス図，オブジェクト図を本提案用に拡張したものを要求分析モデルと呼ぶ．

#### 2.2.1 アクティビティ図の定義

一般的に，開発者が顧客から要求を獲得する方法として，要求文書，業務フロー図，顧客へのヒアリングなどが挙げられる．よって，要求分析プロセスでは業務手順が分析の一つの指標となる．そのため，開発者が業務手順に着目し，モデルを定義することはごく自然な流れである．そこで，本研究では開発者が既存の業務手順からシステム導入後の業務手順までの定義に最適なアクティビティ図を採用する．

一方で，オブジェクト指向分析では，ユースケースを定義し，ユースケース単位にアクターとシステムのインタラクションを表現するユースケース記述を定義する．このようなユースケースモデルの利用が一般的である．ユースケース記述はユースケース名，概要，アクター，事前条件，事後条件，基本フロー，代替フロー，例外フローにより構成される．

本研究では，プロトタイプ生成に必要なアクティビティ図の書式は，ユースケース記述を基礎に確立した．このアクティビティ図を用いる開発者の利点は二点ある．第一に，既存の業務手順の分析からアクターとシステムのインタラクションの定義までをアクティビティ図で一貫して定義できる点である．第二に，図はグラフィカルな表現なため，自然言語に比べフローのパスが直感的にわかりやすく，開発者がフローの矛盾や誤りを発見しやすい点である．

顧客による妥当性確認で提示される要求のうち，表 1 の「業務手順」全体と「入力項目の入力形式」に分類される要求はアクティビティ図へ反映する．

ユースケース記述では，各フローの主体とその振舞いおよび振舞いの対象を明確にし，システムに必要な入力とシステムの振舞いによって生成される出力を明らかにする必要がある．アクティビティ図によるユースケース記述はパーティションによって振舞いの主体を明示し，オブジェクトノードとアクションの主語によって振舞いの対象と入出力を明示するために，以下の制約に基づいて定義する．

1. 各フロー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，アクションなどのすべての要素をパーティション内に定義する．
2. インタラクションを表現する上で，アクターとシステムの並列的な処理を表現する必要はないため，複数のパーティション間に跨るフォークノード・ジョインノードを定義しない．
3. システムがアクターに表示する情報を明示的に表現するため，その情報をオブジェクトノードとしてシステムのアクションからアクターのアクションへの遷移の間に必要に応じて定義する．
4. 代替・例外フローとしての分岐条件を明確に定義するため，デシジョンノード直後の制御フローにガードを定義する．ただし，現状では代替・例外フローを区別する定義方法はない．
5. アクションには，ただ一つの振る舞いを持たせ，振る舞いの対象を明確にするため，アクション名の形式を「(<オブジェクト名>::)<目的語>を<動詞>」に統一する．<動詞>に処理を定義し，処理対象となる入出力項目名やシステムメッセージ<目的語>に定義する．入出力項目の構造を意識した振る舞いの対象を指定するため，必要に応じてオブジェクトノードの<オブジェクト名>を指定する．この必要性は，4.2 節で詳述する．
6. 4.1 節で詳述するプロトタイプ上の入出力項目の形式に対応した動詞を必要に応じてアクション名の<動詞>に定義する．

ただし，モデルを厳密に定義するため，一つの要素はただ一つの意味を持つように記述する．例えば，デシジョンノードとマージノードを一つのノードで同時に表現することを禁止する．

図書検索システムの要求として，キーワードによる検索と頭文字による検索の二つを実現する．また必要に応じて一覧表示件数，ソートキーとソート順を指定できる．ユースケース「図書を頭文字検索する」のアクティビティ図(図 1)について説明する．

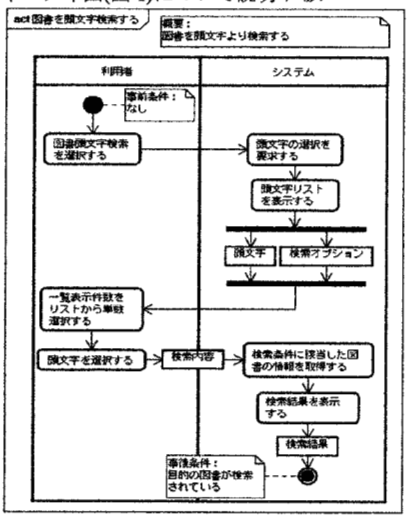
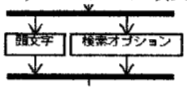



図 1　「図書を頭文字検索する」アクティビティ図

最初に利用者が図書頭文字検索機能を呼び出す．次に，システムは頭文字の選択を利用者に要求する．このとき，左図のように並列的にオブジェクトノードを定義する．これは，システムが「頭文字」と「検索オプション」を同時に利用者に表示することを示す．システムが表示するこれらの情報に対する利用者側の操作を利用者パーティションのアクションに定義する．ここでは，利用者は一覧表示数を設定し，頭文字を選択する．その「検索内容」を基にシステムは検索を行い，「検索結果」を表示してユースケースを終了する．



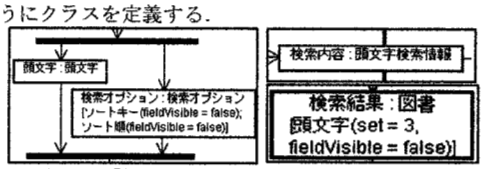

### 2.2.2　クラス図の定義

一般的に，システムで表示する入出力項目は，サービスの実行目的に沿って複数の項目を組にして表示することで，利用者にとってわかりやすい情報となる．この入出力項目の構造を仕様化するため，本研究では，クラスは一つ以上の入出力項目から構成される構造を表現するものとし，クラスの属性は一つの入出力項目を表現するものとする．

本来，クラス図は業務上必要な情報の実体関連に基づき，クラスを分割・定義し，そのクラスの責務に沿って属性や操作を決定する．しかし，要求分析段階において，開発者の要求文書や業務フロー図に基づく業務のイメージのみからクラスを定義することは，定義根拠が弱く，サービスに必要な情報の定義漏れが発生する可能性が高い．そこで，この段階では，開発者はユーザインタフェース上の入出力項目としてサービスに必要な情報を十分に抽出することに専念する．すなわち，この段階のクラス図では，後のクラス候補または属性候補を列挙する．このクラス図は，クラス間で重複する同義の属性を一つのクラスで統一し，この統一に応じてクラス間関連を定義するような洗練によって，要求分析段階の仕様レベルまでに発展する．

顧客による妥当性確認で提示される要求のうち，表 1 の「構造化すべき入出力項目」に分類される要求はクラス図へ反映する．また，入出力項目が増減する場合，アクティビティ図中の入出力項目に対する処理を記述するアクションも併せて修正する．

ユースケース「図書を頭文字検索する」の各オブジェクトノードに対し，下図のようにクラスを割り当てる．「：」の右側がクラス名である，同時に図 2 のようにクラスを定義する．




要求から「検索オプション」には「一覧表示数」，「ソートキー」，「ソート順」が属性として考えられる．しかし，属性「ソートキー」と「ソート順」は，このユースケースでは使用しないため，これらの属性に対してプロトタイプ上で項目自体を非表示にする命令記述(4.2 節を参照)をしている．また，検索結果では，「図書」に定義された属性に関する具体的に絞り込んだ値が見られる必要がある．そのため，検索対象となる図書のインスタンスを特定する必要がある．その実現方法を 2.2.3 節で説明する．

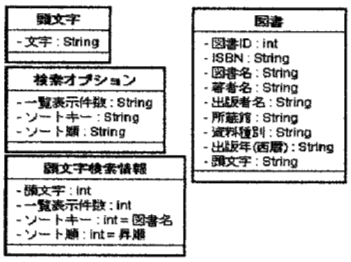



図 2　「図書を頭文字検索する」のクラス図

### 2.2.3　オブジェクト図の定義

本研究のオブジェクト図では，構造化された入出力項目の具体的なデータを表現するために用いる．開発者は顧客にプロトタイプを見せるための特定のシステムの利用シナリオに応じて，具体的なデータを定義する．開発者が具体的なデータを定義する利点は二つある．第一に，要求分析段階のシナリオに対して具体的なデータを明確に定義しておくことで，後のテスト工程で用いるデータとして再利用できる点である．第二に，顧客の妥当性確認時に，顧客が具体的なデータを挙げて状況を指定する場合，その具体的なデータを即時に仕様化できる点である．

顧客による妥当性確認で提示される要求のうち，表 1 の「出力項目の具体的なデータ」に分類される要求はこのオブジェクト図へ反映する．

図 3 はユースケース「図書を頭文字検索する」で使用するオブジェクト図の一部を示している．簡単のため，「頭文字」クラスと「図書」クラスのインスタンス仕様のスロットは非表示にしてある．なお，スロットへの複数の値はカンマ区切りで定義する．

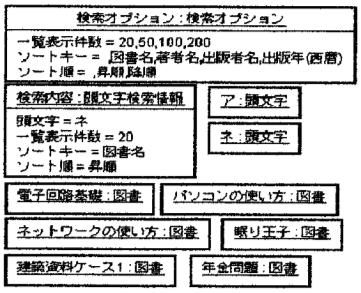



図 3　図書検索システムのオブジェクト図

2.2.2 節で説明したオブジェクトノード「検索結果」の状態には，属性「頭文字」のスロット値が互いに等しいインスタンス仕様を取得する命令記述(4.2 節を参照)をしている．「set = 3」は，その絞り込み方法であり，インスタンス仕様の数が三つに近いもの(ただし，三つ以下)の集合の一つを取得する．

### 2.2.4　ナビゲーションモデルの定義

ユースケース記述には事前条件，事後条件があり，これらの解釈によって全てのユースケースの関連を判断するが，整合性が取れているかの判断が難しい．そこで，本提案では，ユースケースの前後関係を明確にするため，ユースケース間の実行順序をナビゲーションモデルにより定義する．ナビゲーションモデルはアクティビティ図を基礎に記述する．ナビゲーションモデルは，ユースケース図のように，ユースケースと実行者であるアクターとの関係，ユースケース間の依存関係を示し，ユースケース図では示されない実行順序を示す．ナビゲーションモデルによりはユースケース単位で生成されたプロトタイプを統合する(統合方法は 4.4 節を参照)．

顧客による妥当性確認で提示される要求のうち，複数のユースケースに跨る実行順序に対する要求をナビゲーションモデルへ反映する．

以下にユースケースの実行者とユースケースの前後関係を明確にするためのナビゲーションモデルの制約について述べる．

1'. ユースケース単位のアクティビティ図と区別するため，図名を「navigation」とする．
2'. アクションはユースケースを表現するため，ユースケース単位のアクティビティ図名を用いる．
3'. ユースケースと実行者であるアクターの関係を明確にするため，パーティションはアクター名のみで構成し，アクションなどの要素はすべてパーティション中に記述する．
4'. ユースケース間の前後関係を明確にするため，事後のユースケースに対する事前のユースケースの終了状態を特定する必要がある．そのため，アクション間で出現するガードに対し，事前のユースケース(アクティビティ図)おいて，アクティビティ終了ノードへ到達するフローまでのシステムパーティション中のガードを含める．

図 4 は，図書検索システムのナビゲーションモデルである．一部を説明すると，開始ノードはシステムのルートを表現する．そして，ルートからは三つのユースケースが実行できることを表現する．「図書を頭文字検索する」または「図書をキーワード検索する」のユースケース後に「図書を表示する」のユースケースが実行できることを表現する．権限には「利用者」と「管理者」の二種類があり，「ユーザ認証する」を実行することで利用者は管理者として識別されることを表現する．

## 3　要求分析プロセスとプロトタイプの確認

本研究の要求分析プロセスを以下ように定義する．各段階で生成されるプロトタイプは 4 節で詳述する．

1. 既存の業務手順や要求文書，顧客へのヒアリングからユースケースを導出する．
2. 導出したユースケース単位でユーザとシステムのインタラクションをアクティビティ図に定義する．

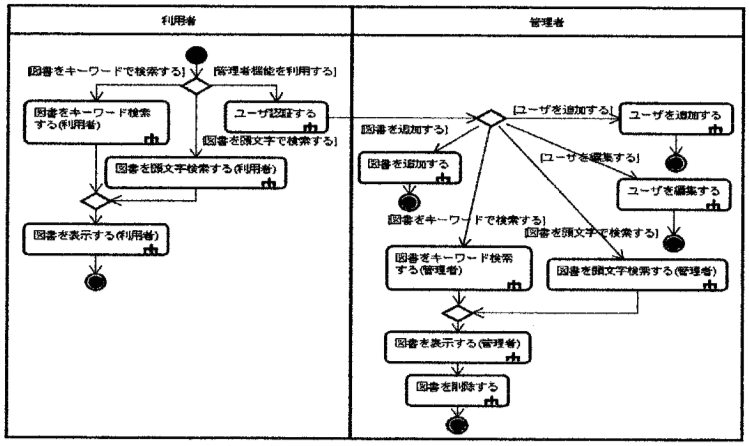



図 4　図書検索システムのナビゲーションモデル

顧客が全ての入出力項目を確認できるように，入出力項目を詳細に定義することが重要である．※ここで，初期プロトタイプが生成可能である．

3. 顧客が確認すべき入出力項目の構造を，クラスを用いてクラス図に定義する．※ここで，第二期プロトタイプが生成可能である．
4. 3.で定義したクラスに対し，顧客がプロトタイプで確認する特定のシステムの利用シナリオに応じたインスタンス仕様をオブジェクト図に定義する．※ここで，第三期プロトタイプが生成可能である．
5. ユースケース実行手順を定義するためにナビゲーションモデルを定義する．※ここで，プロトタイプ統合が可能である．また，権限を指定することで，権限別プロトタイプの生成が可能である．
6. 顧客はプロトタイプを通して要求分析モデルの妥当性を確認し，満足しない箇所を指摘する．
7. 指摘の要求レベルを表の確認事項にて判断し，対応するモデルを修正する．(以降，顧客の了承が得られるまで 2.-5.の成果物の修正と，6.の妥当性確認を反復的に繰り返す．)

## 4 生成プロトタイプとツールの設計と実装

本ツールは Java で実装しており，ツールへの入力として JUDE[7]ファイルを用いる．次節以降は，各段階のプロトタイプ生成，プロトタイプ統合，権限別プロトタイプについて述べる．先に本研究で用いている用語について説明する．システムパーティションへ遷移する直前の，システムを外部から操作するアクター(以下，ユーザ)パーティションのアクションを「画面遷移トリガー」と呼ぶ．ユーザパーティションへ遷移する直前のシステムパーティションの制御フローを「システム外部遷移フロー」と呼ぶ．

### 4.1 初期プロトタイプの生成

3 節のステップ 2.において生成可能なプロトタイプは，画面遷移と入出力項目の過不足及び入出力項目の形式を確認できる非常に簡素なプロトタイプである．このプロトタイプは顧客に提示できるレベルではないが，開発者にとっての利点が三つある．第一に，開発者自身がイメージする業務手順をモデル化できているかを確認できる点である．第二に，プロトタイプはモデルで表現される情報を基準として生成されるため，このプロトタイプが開発の初期段階におけるモデル定義の抽象度の指標となる点である．第三に，一種類の図のみから生成されるため，開発者はモデルとプロトタイプの対応が，複数種類の図から生成されるよりは理解しやすい点である．

初期プロトタイプは，次のルールでアクティビティ図から生成される．最初に画面の分割方法として，

A) 画面の開始点は，開始ノード，または画面遷移トリガー直後の制御フローである．
B) 画面の終了点は，アクティビティ終了ノード，または画面遷移トリガーである．
C) 生成される画面数は，　区間のシステムパーティション中のフローの通りうるパスの数と同数である．このとき，パスの通過順にガードを「&」で繋いだものが画面遷移条件となる．
D) フローにループのある場合，ループを一度通る場合と通らない場合の二通りの画面を生成する．

次に，各画面の入出力項目の表現ルールである．

A') ユーザパーティションのアクションの<目的語>(2.2.1 節の制約 5.)を入出力項目名として扱う．システムパーティションのアクションの<目的語>をシステムからのメッセージとして扱う．
B') アクションの<動詞>(2.2.1 節の制約 5.)に応じて，入出力項目に形式を与える．本提案では HTML 上の特定の形式に対応づいた<動詞>が，表 2 と表 3 に示すように用意してある．ユーザ側では，入力方法を確認するための形式を提供し，システム側では特定のシステムメッセージの強調を目的に装飾を提供する．

表 2　ユーザパーティションの動詞と形式の対応

| 動詞 | 形式の種類 |
|---|---|
| 入力する | テキスト入力形式 |
| 単数選択する | ラジオボタン形式 |
| 複数選択する | チェックボックス形式 |
| リストから単数選択する | プルダウンメニュー形式 |
| リストから複数選択する | リストボックス形式 |
| 参照する | ファイル参照形式 |
| 選択する | リンク形式 |
| 確認する | 項目表示 |

表 3　システムパーティションの動詞と形式の対応

| 動詞 | 形式の種類 |
|---|---|
| 要求する | 太字下線付き文字列 |
| 表示する | 見出し 2 レベル文字列 |

図 5 に「図書を頭文字検索する」のアクティビティ図(図 1)のアクティビティ図定義段階の　部と対応する初期プロトタイプを示す．

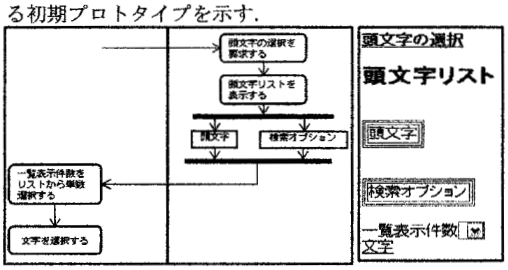



図 5　「図書を頭文字検索する」アクティビティ図の一部と対応する初期プロトタイプ

### 4.2 第二期プロトタイプの生成

3 節のステップ 3.において生成可能なプロトタイプは，初期プロトタイプの入出力項目を構造化し，テーブル形式で表現する．開発者にとってこのプロトタイプの利点は，プロトタイプ上のデータ構造が初期プロトタイプに比べわかりやすい点である．また，顧客にとってこのプロトタイプの利点は，システムが提供す

る本質的な情報(業務手順と構造化された入出力項目)のみを確認できる点である．

このプロトタイプ生成法では，最初にアクティビティ図のオブジェクトノードにクラスを割り当てる．アクティビティ図のアクションでは，入出力項目に対する処理を定義している．そのため，アクションの<目的語>とクラスの属性が一致した場合は，構造化された入出力項目に形式が与えられる．ステップ 2.でアクションが十分洗練されている場合，ステップ 3.ではアクティビティ図を変更する必要はほぼない．しかし，ここで二つの留意点がある．

第一に，互いに同名な入出力項目を持つオブジェクトノードを同時に表示する場合である．構造化された入出力項目を特定して形式を与える場合，アクティビティ図のアクションに対して，2.2.1 節の制約 5.で説明した「<オブジェクト名>::」を定義することである．これにより，オブジェクトノードを一意に識別し，同名の入出力項目があったとしても個々に形式を与えることができる．クラスではなくオブジェクトノードを指定する理由は，アクティビティ図に現れるオブジェクトノードはインスタンスレベルであり，アクションは元々インスタンスレベルの振る舞いを定義しているためである．

第二に，構造化された入出力項目の表示方法をフローの状況に応じて変化させたい場合である．それは，2.2.2 節，2.2.3 節において図書検索システムのモデルで説明したような場合である．デフォルトではプロトタイプ生成アルゴリズムはオブジェクトノードに対応するクラスの属性をすべて表現する．しかし，オブジェクトノードの状態に，属性の表示等に関する命令記述(表 4) を行うことで，ある程度の表示の加工が可能である．この命令記述はフローの状況に応じた情報の表示方法に対する仕様となる．

表 4　オブジェクトノードの状態に対する命令

| 命令の種類 | 命令記述 |
|---|---|
| 項目可視 | fieldVisible = “true” | “false” |
| 具体データ可視 | valueVisible = “true” | “false” |
| 集合選択 | set = “1” – “9” ( “0” – “9”)* |
| 新規欄 | set = “1” – “9” ( “0” – “9”)* newly |

図 6 に「図書を頭文字検索する」のアクティビティ図(図 1)のクラス図定義段階の一部と対応する第二期プロトタイプを示す．

## 4.3　第三期プロトタイプの生成

3 節のステップ 4.において生成可能なプロトタイプは，第二期プロトタイプの入出力項目の構造に具体データを挿入したものである．このプロトタイプの顧客にとっての利点は，業務上で扱う具体的なデータが確認できるため，システム用途が第二期プロトタイプに比べ，わかりやすい点である．システムに不慣れな顧客でも，普段，業務上で扱う具体的なデータを基に要求を議論することで，システムへの理解を深め，効果的な妥当性確認が行える．

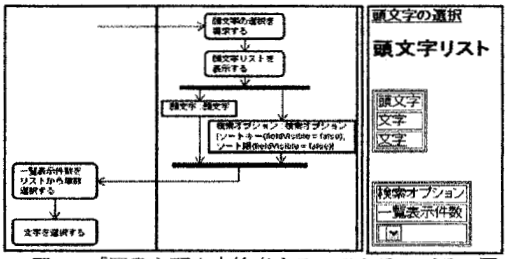



図 6　「図書を頭文字検索する」アクティビティ図の一部と対応する第二期プロトタイプ

このプロトタイプ生成法では，4.2 節で定義したクラス図のクラスに対するインスタンス仕様をオブジェクト図に定義する．そのインスタンス仕様をそれぞれ具体的なデータとしてプロトタイプに反映する．

図 7 に「図書を頭文字検索する」のアクティビティ図(図 1)のオブジェクト図定義段階の一部と対応する第三期プロトタイプを示す．オブジェクト図追加によるアクティビティ図，クラス図への影響はない．

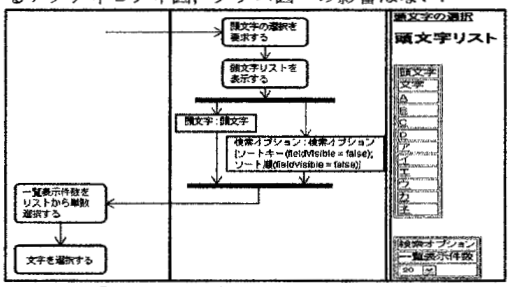



図 7　「図書を頭文字検索する」アクティビティ図の一部と対応する第三期プロトタイプ

## 4.4　プロトタイプの統合

本研究では，2.2.4 節で紹介したナビゲーションモデルを用いて，ユースケース単位のプロトタイプを統合する．この統合プロトタイプの顧客にとっての利点は，実システムと同じ画面遷移が確認できる点である．また，この統合による開発者の利点は二点ある．第一に，ユースケースの繋がったシステム全体の流れがプロトタイプ上で確認できるため，ユースケース間で意味的に重複するフローを発見しやすい点である．第二に，プロトタイプが表現するサービス実行順序をアクティビティ図一つで容易に変更できる点である．このナビゲーションモデルはユースケース名とアクティビティ図，及びアクター名にしか依存しないため，初期～第三期プロトタイプ間のいずれの段階でも適用する事が可能である．

プロトタイプ統合アルゴリズムでは，ナビゲーションモデルから前後関係にあるアクティビティ図間の開始ノードとアクティビティ終了ノードを仮想的に接合する．しかし，開始ノードは一つのアクティビティ図にただ一つであるが，アクティビティ終了ノードは複数存在する可能性がある．そのため，一つのアクティビティ図の開始ノードに対して，接合すべき事前のア

クティビティ図のアクティビティ終了ノードを特定することが必要となる．その方法は，2.2.4 節で述べた制約 4'.「アクティビティ終了状態へ到達するフローまでの遷移条件」により特定する．

図 4 で示した二つのユースケース「図書を頭文字検索する」と「図書を表示する」を統合した場合，ユースケース毎のプロトタイプでは一つの画面であったものが，図 8 のように一つに統合される．

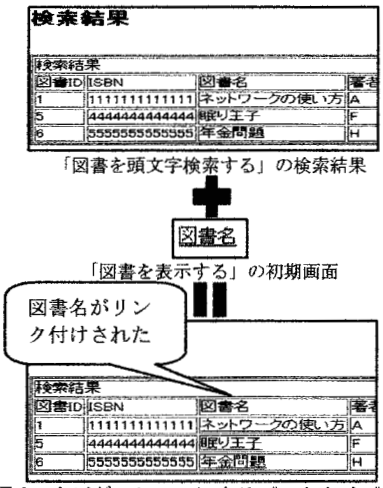



図 8　ナビゲーションによるプロトタイプの統合

### 4.5　権限別プロトタイプ

通常，実システムでは利用者の権限によって，提供するサービスが異なる．そのため，プロトタイプでも，権限別に情報が見える必要がある．本研究では，アクターを指定する事でユースケース単位のアクティビティ図，ナビゲーションモデルから権限別プロトタイプの生成が可能となる機能拡張を行った．

## 5　適用事例と考察

### 5.1　図書検索システム開発への適用

本提案は，図書検索システム開発に適用した．この事例を顧客が確認すべき点から考察する．最初に検索機能の特徴を以下の四点にまとめる．

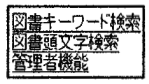
第一に，検索機能は検索対象の情報に即した様々な検索方法を提供する．第二に，検索機能は全体の情報から特定の情報の絞り込むサービスを提供する．第三に，検索機能は単独のサービスとしても有効だが，他のサービスからも利用される．第四に，検索結果の情報に対して行える操作はシステムの利用権限により異なる場合がある．これらの特徴を踏まえ，顧客の確認すべき点として，図書の検索方法，検索結果として得られる情報，検索後に実行できるサービス，権限別のサービスの差異を挙げる．これらの点に対する本提案のプロトタイプの表現力を考察する．また，開発の際にモデル定義上または，プロトタイプ表現上の問題点について説明する．

### 5.2　考察

最初に顧客の確認すべき点について考察する．

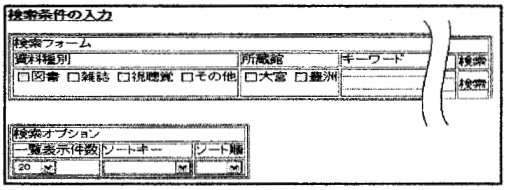
■　図書の検索方法

ナビゲーションモデル(図 4)からわかるように，図書の検索方法は二種類ある．一つはキーワードによる検索方法であり，もう一つは頭文字による検索方法である．システムのルートでは，ナビゲーションモデル




の開始ノード直後の三つのアクティビティ図に対する初期画面が統合され左図のメニューになる．

キーワード検索では下図のフォームが表現される．また検索実行時の画面では，検索結果に対して，絞り込み検索や再検索を行うフォームを表示する．




頭文字検索については，前述した通りのフォームを提供する．このように，プロトタイプは入力項目の選択系の入力形式に具体的なデータを表現しており，顧客は検索方法を十分に理解することができる．

■　検索結果として得られる情報

頭文字検索において，頭文字が「ネ」であるものの検索結果を下図に表現する．

開発者が定義したオブジェクト図(図 3)のインスタンス仕様のうち，オブジェクトノードの状態で指定されたもの(2.2.2 節を参照)を取得している．顧客は検索結果として表示される情報は確認できる．しかし，検索結果はオブジェクトノードの状態とインスタンス仕様により決定するため，顧客が様々な検索結果を確認するには開発者が複数のシナリオを用意する必要がある．

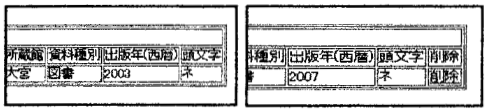
■　検索後に実行できるサービス

ナビゲーションモデル(図 4)では，検索後にユースケース「図書を表示する」が実行できる．通常，検索機能のユースケースでは，検索するまでが責務であり，その後のインタラクションは定義しない．図 8 からわかるように，検索結果に対してリンク付けされた項目「図書名」はユースケース「図書を表示する」のアクティビティである．このように，顧客は検索後のサービスを確認することができる．また，開発者側としては，検索機能と検索後のサービスを分割して定義することが可能である．

■　権限別のサービスの差異

ナビゲーションモデル(図 4)では，利用者と管理者の二つの権限があることがわかる．これらの権限に対して，似て非なる機能は，ユースケース「図書を

表示する」後にユースケース「図書を削除する」が実行できるかどうかである．




上図左は，利用者からユースケース「図書を表示する」を実行した場合であり，上図右は管理者から同ユースケースを実行した場合である．管理者側では項目「削除」が追加される．このように，顧客は権限別のサービスの差異を確認することができる．

次に開発の際にモデルの定義上または，プロトタイプの表現上の三点の問題点について説明する．

第一に，アクティビティ図のオブジェクトノードに対する入力方法をインスタンスレベルとクラスレベルで区別する方法の問題である．現状では，クラスの属性に対する形式付与を提供してきた．しかし，例えば，検索された複数の図書を選択し，一度に削除する場合，図書のインスタンス自体を選択する入力方法が必要である．この問題の解決方法として，アクションの<目的語>にオブジェクト名を指定することで，アクションがインスタンスレベルの処理を表現するという解釈機構の導入を検討する．

第二に，ナビゲーションモデルのパーティションの定義方法の問題である．権限別のプロトタイプは権限名を指定して生成する．現状の生成アルゴリズムでは，開始ノードから辿って指定外のパーティションのアクションを発見した時点で，そのアクション以降のフローを無視する．よって，図 4 のナビゲーションモデルで管理者権限のみを指定すると，何も生成されない．この解決方法として，権限別にナビゲーションモデルを定義する事を検討している．

第三に，同一ユースケースに対するフローが，権限により異なる場合の問題である．権限別プロトタイプを生成するために，アクティビティ図にはアクターの差異によるフローの差異が含まれていない事が必要である．このため，現状ではユースケース単位かつアクター単位でアクティビティ図を定義している．しかし，この方法では，権限別の同一ユースケース間に意味的に重複するフローが散在するため，洗練には適さない．代替案を検討する必要がある．

## 6 関連研究

Elkoutbi ら[4]はシステムの利用シナリオとしてコラボレーション図を定義し，クラス図，OCL(Object Constraint Language)から，ステートチャート図の生成を経てプロトタイプを生成する．Juan ら[5]はユースケースモデル，シーケンス図の拡張であるメッセージシーケンスチャート(MSC)から，ステートチャート図の生成を経てプロトタイプを生成する．

これらの研究はコラボレーション図または MSC において，いずれもオブジェクト間のメッセージングにより処理の流れを記述する．この場合，オブジェクトを定義するためにクラスを適切に定義する必要がある．しかし，開発者のクラスに対する定義動機は述べていない．本研究では，2.2.2 節で説明するように定義漏れを軽減するため，クラス図を業務手順中の入出力項目の構造化に沿って定義する．

また，これらの研究では，プロトタイプ生成には必ず複数種類のモデルが必要となり，開発者はモデルとプロトタイプの対応の理解が困難となる．本研究では，アクティビティ図のみからのプロトタイプ生成が可能であり，漸増的なモデルの各定義段階におけるプロトタイプ生成方法を提案している．これにより，開発者はモデルとプロトタイプの対応を学習しながら，開発を進める事ができる．

さらに Elkoutbi ら[4]は，プロトタイプのデザインについて課題を残している．デザインの向上は，顧客がシステム利用方法の直感的な理解を容易にする目的と，実システムと全く同じデザインを確認する目的がある．本研究でもデザインの向上は課題であるが，システム用途の直感的な理解を容易にするため，プロトタイプへ反映可能なシステムに入力・蓄積される具体的なデータの定義方法を提供している．

## 7 まとめと今後の課題

本稿では，段階的なプロトタイプ生成，プロトタイプ統合，権限別プロトタイプ生成を含む要求分析プロセスを提案した．適用事例よりプロトタイプ統合はプロトタイプ表現に有益であるが，モデルの定義方法についての問題点も多くあった．

今後の課題として，考察で述べた問題点の解決を含め，特定のシステムの使用シナリオに応じた実システムとしての情報量を持つプロトタイプの生成を実現する．また，クラス図の洗練プロセスに向けて，クラス間関連の定義プロセス及びツールへの解釈機構の導入を実現する．また，プロトタイプのデザインの向上については，テーブルレイアウトの補正等の処理をテンプレート化し，生成されたプロトタイプに対し，そのテンプレートを通して，補正をかける事でデザインを変更する方法を検討する．

## 8 参考文献

[1] I. Sommerville, P. Sawyer, 富野 壽 監訳, 要求定義工学プラクティスガイド, 構造計画研究所, 2000
[2] 大西淳, 郷健太郎, 要求工学, 共立出版, 2002
[3] Object Management Group : http://www.omg.org/
[4] M. Elkoutbi, I. Khriss, R.K. Keller, Automated Prototyping of User Interfaces Based on UML Scenarios, Automated Software Engineering, 13, 5-40, 2006
[5] J.S. Díaz, O.P. López, J.J. Fons, From User Requirements to User Interfaces: A Methodological Approach, CAiSE2001, LNCS2068, pp.60-75, 2001
[6] Cruz, A. M. Rosado, Faria, J.P., Automatic Generation of User Interfaces from Domain and Use Case Models, QUATIC 2007, 2007 Page(s):208 – 212, 2007
[7] JUDE : http://www.change-vision.com/
[8] 小形真平, 松浦佐江子, UML で記述された要求分析モデルからのプロトタイプ自動生成, SE-157-13, 2007