RB-001

Web UI プロトタイプ自動生成ツールを用いたユースケース駆動要求分析の評価実験
An Evaluation of the Use Case Driven Requirements Analysis using an Automatic Web UI Prototype Generation Tool

小形 真平† 松浦 佐江子‡
Shinpei Ogata Saeko Matsuura

1. はじめに

業務システムの要求分析段階では，要求を十分に正しく仕様化することが不可欠である．しかし，業務には複雑な業務手順や業務データが存在するため，要求も複雑である．

オブジェクト指向分析は，開発者が複雑な要求を容易に分析できる観点として実世界のオブジェクトに基づいて，システムの振る舞いや構造を分析する方法であり[1,2]，UML(Unified Modeling Language)[3]が利用される．

典型的なオブジェクト指向分析であるユースケース駆動要求分析は，ユースケース[4]単位でアクターとシステムのインタラクション(以下，単にインタラクション)に着目し，振る舞いとデータを分析・定義する方法である．本方法では，ユースケースモデル(ユースケース図，ユースケース記述)を用いて振る舞いを定義し，クラス図を用いて概念レベルのデータを定義することが一般的である[2,5]．

ユースケース駆動要求分析には，2 つの利点がある．第 1 に，ユースケース記述では，実装に非依存な顧客視点の用語を用いて，自然言語によりインタラクション中の振る舞いを定義するため，顧客がモデルを理解し易い点である．第 2 に，分析観点であるインタラクションは，開発者が UI(User Interface)要求，パフォーマンス要求，データ形式，業務ルールのような様々な要求を分析するための核となる点である．このことから，ユースケースの概念は多くの開発者に受け入れられており，ユースケースの抽出[6]，分析[7,8]，検証・妥当性確認[7,9,10]の各プロセスに着目した研究が盛んに行われている．

ユースケース駆動要求分析では，実現すべき UI に対する振る舞いとデータがモデルに正確に表現される場合，当該モデルは顧客の要求を正しく仕様化した要求仕様となり得る．しかし，要求仕様レベルのモデルを定義する上で，以下の 3 つの問題がある．

1. 実装に非依存な顧客視点の用語を重用するユースケース記述では，顧客は実際の UI におけるインタラクションを想起しづらい．そのため，顧客を満足させるために重要な UI の使用性に係わる要求の十分な定義が困難である．
2. 一般に，定義されたユースケース記述をもとにオブジェクトの抽出を行い，クラス図を定義する．ユースケース記述に登場する単語はクラスの様々な状況におけるインスタンスの表現であり，これを自然言語で一貫して定義することは困難な作業であることから，オブジェクトの抽出作業を難しくする一因となっている．
3. 自然言語記述の仕様では，読み手の知識背景によって単語の解釈が異なり，個々人で異なる実装イメージを持つため，誤解を生じ易い．

本研究では，これまでに業務系 Web アプリケーション開発を対象に，ユースケース駆動要求分析の利点を活かせるような，UML モデルを中心とした要求分析モデルから UI プロトタイプを自動生成する手法(以下，プロトタイプ自動生成手法)の研究とツールの開発を行ってきた[11]．ここで，要求分析モデルとは，アクティビティ図，クラス図，オブジェクト図，シナリオを用いてインタラクションを分析・定義するモデルである．プロトタイプ[12]とは，顧客のみならず開発者が要求やシステム像を最も容易に理解できる実システムの特定の面を実現した模型であり，本研究では UI のみを実現した機能を持たないプロトタイプを扱う．プロトタイプ自動生成手法では，自動生成したプロトタイプを要求分析モデルの検証・妥当性確認に利用する．

プロトタイプ自動生成手法では，前述した問題点を以下のように解決する．

- モデルからプロトタイプを自動生成することで，モデルと実装イメージを対応付け，1.と 3.の問題を解決する．
- アクティビティ図，クラス図，オブジェクト図が規定するオブジェクトノード，クラス，インスタンス仕様を用いて，データとなるクラスやインスタンスを定義する．そして，クラスを中心に各図を連携し，データを一貫して定義することで，2.の問題を解決する．

本稿では，プロトタイプ自動生成手法を適用したユースケース駆動要求分析方法(以下，提案方法)が，ユースケースモデルを適用したユースケース駆動分析方法(以下，従来方法)の問題点を解決し，かつ作業時間コストを軽減することを検証するため，従来方法との比較評価実験を行った．

第 2 章では，従来方法の概要および問題点と限界を説明する．第 3 章では，提案方法の概要を従来方法と比較しながら説明する．第 4 章では，比較評価実験の概要と結果を述べ，第 5 章では，結果から提案方法の有効性を議論する．第 6 章では，関連研究と比較し，第 7 章で結論を述べる．

2. ユースケースモデルによるユースケース駆動要求分析

2.1 概要

ユースケース駆動要求分析の典型的な方法として，ユースケースモデルを用いる方法がある．以下にその分析手順を説明する．

ユースケース図の定義：顧客要求の分析時，開発者は最初に，誰がどのような目的をシステムで達成し得るべきかを整理する．そこで，開発者は要求からシステムに必要な外部機能の候補を列挙し，システムに係わるユーザの種類(以下，権限)や外部システムを洗い出す．

†芝浦工業大学大学院工学研究科機能制御システム専攻，Functional Control Systems, Graduate School of Engineering, Shibaura Institute of Technology
‡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

ユースケース図は，アクターとユースケースの関連を定義することで，権限に応じて実行可能なユースケースを明確に定義することができる．

ユースケース記述の定義：ユーザは入力に対するシステムからの応答によりサービスを受けることができる．すなわち，顧客の要求はアクターとシステムのインタラクションに集約される．インタラクションを定義する上で，最も重要な点は，ユーザが業務を正しく遂行できる範囲で必要な全てのデータを何らかの方法で入力でき，必要なデータを適切なタイミングで確認できることである．このことから，顧客がインタラクションの妥当性を確認するために，まずインタラクションを理解できなければならない．

ユースケース記述は，顧客視点の実装に非依存な自然言語を用いてインタラクションを定義するため，ユーザの理解性が高い．ユースケース記述には，標準形式はないが，一般に表 1 に示す構成要素を持つ．基本・代替・例外フローでは，ユーザの入力操作とシステムの応答処理を，順序だてたステップを単位として記述する．

表 1　ユースケース記述の構成要素

構成要素	説明
ユースケース名	外部機能の名前
アクター	ユースケース実行に係わる権限と外部システムの総称
事前条件	ユースケース実行開始時に満たすべき条件
基本フロー	最も基本となる正常フロー
代替フロー	基本フローから分岐する正常フロー
例外フロー	不正入力等により正常フローから分岐する異常フロー
事後条件	ユースケース実行終了時に満たすべき条件

クラス図の定義：ユースケースモデルによるインタラクションの分析において，開発者はシステム化する範囲に沿って業務データを想定する．想定した業務データの構造は，クラス図として，ユースケースモデルとは別途に定義する．

要求分析段階のクラス図は，概念レベルのクラスを定義するものであり，設計段階のクラス候補となる．

2.2 ユースケースモデルの問題と限界

ユースケースモデルでは，以下の 3 つの問題点がある．

第 1 に，実装に非依存な顧客視点の用語を重用するユースケース記述では，顧客は実際の UI におけるインタラクションを想起しづらいため，顧客を満足させるために重要な UI の使用性に係わる要求の十分な定義が困難な点である．ここで UI の使用性に係わる要求として，フロー，データ，レイアウトの 3 種が考えられる．フローは，一度に操作可能な項目量がユーザに高負荷とならないようにするための操作手順の分割と項目の操作方法及びフローの分岐と結合のタイミングに係わる．データは，一度に視認できる分量の調整と操作状況に応じた表示データの選定に係わる．レイアウトは，項目の入力フォームに対するサイズのようなプロパティや，項目の位置に係わる．ただし，文献[7]にも述べられるように，ユーザの目に付きやすいレイアウトや装飾を優先的に定義する場合，業務の本質であるフローとデータの要求から逸れた議論がなされる可能性が高いため，本稿では，フローとデータに対する使用性に係わる要求を対象とする．

入出力データの性質を正確にモデルに定義するため，例えば，上記の 2 種の要求はユースケース記述で明記されるべきであるが，フローのステップの記述が複雑になるため，理解性の低下を招く原因となる．そのため，インタラクションを正確に定義し，かつ理解性の高い自然言語記述の実現には限界がある．また，業務ノウハウを活かした代替フローや，不正入力時の例外フローは，フローの分岐と結合のタイミングとして使用性に係わる重要な要素である．しかし，ユースケース記述では，直列的に分岐フローを記述するため，多数の分岐があるフローに対して，フロー全体を俯瞰するという目的では可読性が低いという欠点がある．

第 2 に，一般に，定義されたユースケース記述をもとにオブジェクトの抽出を行い，クラス図を定義するが，このオブジェクトの抽出作業が困難な点である．これは，ユースケース記述に登場する単語がクラスの様々な状況におけるインスタンスの表現であるため，これを自然言語で一貫して定義することが困難であることに起因する．

上記の理由から，ユースケース記述とクラス図を形式的に対応づけることは不可能である．仮に形式的な対応付けを可能にするような記法をユースケース記述に設ける場合，自由記述による開発者の定義容易性の利点が損なわれ，自然言語を用いる必要性がなくなる．

第 3 に，自然言語記述の仕様では，読み手の知識背景によって単語の解釈が異なり，個々人で異なる実装イメージを持つため，誤解を生じ易い点である．ユースケース記述を開発者と顧客の間のコミュニケーションツールとした場合，実装に非依存な用語を用いて記述されたことを原因として，開発者や顧客の各々が自由に UI の実装イメージを想起してしまうために，誤解が生じる．ここで UI の実装イメージは，入力項目の入力形式，画面単位の表示データや画面の分割単位のような UI の構成を指す．

上記の理由から，ユースケース記述のみを用いて，開発者と顧客の間で UI の実装イメージの共通的に理解することは不可能である．仮にユースケース記述に対して，シナリオとして特定の状況下の具体的な振る舞いとデータを定義し，UI に対するユーザの操作手順や，UI 上に現れる具体データを確認できるようにしても，開発者と顧客の間で実装レベルの UI イメージの共通理解を得られるわけではない．

3. プロトタイプ自動生成手法適用の要求分析

3.1 要求分析プロセスとモデリング

提案方法では，図 1 に示すようなプロセスを繰り返す．各データは直前の各プロセスに対する成果物や指摘を表す．

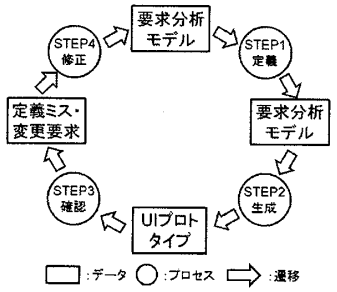

図 1　要求分析プロセス

定義：開発者は，UML モデリングツール JUDE[13]を用いて各種要求分析モデルを段階的に定義する．提案方法では，従来方法と同じく，インタラクションに着目して，振る舞いとデータを分析・定義する．JUDE では，クラス図のクラスを，アクティビティ図のオブジェクトノードとオブジェクト図のインスタンス仕様に対して明確に対応づけることができ，これら 3 種の図の正確に連携できる利点がある．

生成：開発者は，プロトタイプ自動生成ツールを通して，定義した各種要求分析モデルを統合したプロトタイプを自動生成する．生成されるプロトタイプは要求分析モデルの検証・妥当性確認に利用される．

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．その確認の形態は 2 種類ある．第 1 に，開発者が自身の意図を正確にモデルに定義できたかどうかを検証する観点からの確認である．第 2 に，顧客が要求の通りにモデルが定義されているかどうかの妥当性を確認する観点からの確認である．

修正：開発者が検証により発見した定義ミス，または顧客が妥当性確認により指摘した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．応答速度のような非機能要求は直接プロトタイプへ反映できないが，プロトタイプに対応する箇所の要求分析モデルに注釈として定義する．

プロトタイプ自動生成手法では，開発者が複数種類のモデル定義によるモデルの複雑化以前に要求分析モデルの確認と洗練を行えるように，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 プロトタイプ自動生成を実現している．

3.2 要求分析モデルの定義と検証・妥当性確認

プロトタイプ自動生成手法では，図 2 に示すように要求分析モデルを手動で定義する．そして，各種要求分析モデルの定義段階に応じてプロトタイプの自動生成が可能であり，自動生成されるプロトタイプを通した要求分析モデルの確認がいつでも行える．要求分析の入力情報としては，システム化要求や既存業務の業務フロー図と言った顧客の要求文書とそこから開発者が分析したアクターと初期ユースケースを想定する．

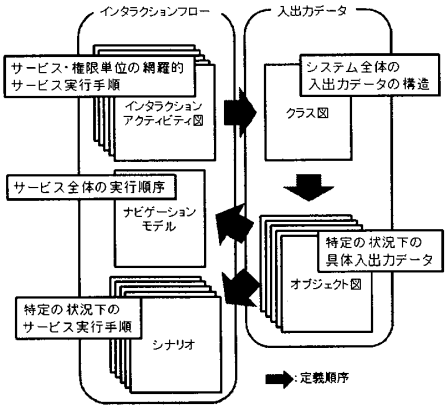

図 2　要求分析モデル

インタラクションアクティビティ図の定義：インタラクションに着目して，振る舞いをアクティビティ図(以下，インタラクションアクティビティ図)により定義する．本図では，最初にユースケース単位で定義する．ここで，ユースケース単位は顧客から見た外部機能の単位である．しかし，実際にはユースケース単位のフローを分割・統治する必要がある．例えば，登録系ユースケースにおいて，単独実行と検索系ユースケース実行後の実行の双方が可能である場合，検索結果を利用するか否かでフロー開始時のデータフローの差異が現れることが一因となる．この差異は，設計や実装に影響するため，仕様に明記されるべきである．分割・統治後の本図の定義単位をユースケース単位とは区別して，サービス単位と呼ぶ．また，複数の権限が同様なサービスを実行できる場合，例えばサービス中で表示可能なデータが権限によって異なる場合がある．このような差異は機密性の観点から明確に区別すべきであるため，本図では，サービス単位に加えて権限単位で定義する．

本図は，実装に非依存な用語の利用を踏襲しつつ，従来のユースケース記述の定義観点を全て網羅し，かつ 2.2 節で述べた 3 つの問題を改善する．以下に，ユースケース記述の要素と本図の要素を対比しながら説明する．

ユースケース記述のユースケース名は，本図の名前に相当する．ここで，本図の名前はサービス名と呼ぶ．

ユースケース記述のアクターは，本図のパーティションとして定義する．1 アクターが 1 パーティションに相当し，加えて開発対象システムのパーティションを設ける．

ユースケース記述の事前条件は，本図の開始ノードのノートとして付記する．一方で事後条件は，本図のアクティビティ終了ノードにノートに付記する．アクティビティ終了ノードは複数存在する可能性があり，各ノードは正常終了と例外終了の 2 種類に分類される．ノートには正常終了と例外終了の識別を明記する．これは，終了の仕方によって，後に続くサービスが異なる場合があるためである．

ユースケース記述の基本・代替フローは，開始ノードから正常終了のアクティビティ終了ノードまでのフローの内，業務ルールや法律のようなルールを常に満たすフローに相当する．本図では，正常フローとして基本・代替フローを統一的に扱い，分岐後のフローに<<normal>>ステレオタイプとして明記する．一方で例外フローは，業務ルールや法律のようなルールを満たさない場合に分岐するフローに相当する．本図でも，例外フローと呼び，分岐後のフローに<<exceptional>>ステレオタイプとして明記する．

ユースケース記述では，動作，オブジェクトやその状態を表す全ての用語が，自然言語としてフラットに定義される．そのため，属人的にオブジェクトとなる用語を抽出しなければならないことが問題であった．インタラクションアクティビティ図では，UML アクティビティ図の記法に則って，各ノードまたはエッジに意味づけがなされている．例えば，動作はアクションに相当し，オブジェクトはオブジェクトノードに相当する．

また，正常・例外フローを網羅的に定義しても，フローを二次元グラフとして表現できるため，ユースケース記述に比べて可読性が高い．

図 3 は授業管理システムの新規学生の追加のインタラクションアクティビティ図である．本サービスは授業管理システムの学生管理機能におけるサービスの 1 つである．新規学生の追加時に管理者が既存学生と重複しないように，既存学生の一覧を確認しながら，新規学生の情報の入力を行えるようにする．学生の情報として，学籍番号，姓，名，学部，学科，学年，連絡先を扱う．既存学生と重複しない

新規学生の情報を確認し，正しければ新規学生を追加するフローとなっている．

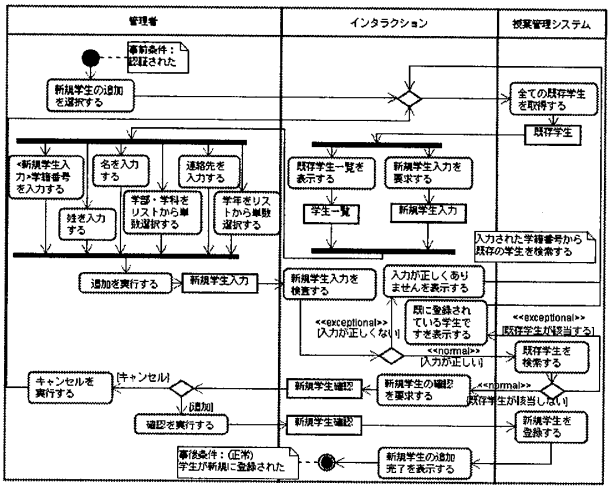

図 3　インタラクションアクティビティ図

図 4 は，自動生成されるプロトタイプ中の新規学生入力画面であり，必要な入力項目を画面単位に確認できる．UI 上の要素は基本的にフローを逐次的に解釈した順序で出力される．

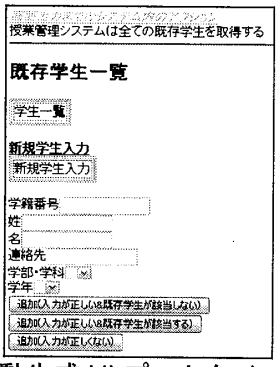

図 4　自動生成 UI プロトタイプ(段階 1)

ツールには UI 上の入力形式と対応づいた動詞が規定されている．その動詞をインタラクションアクティビティ図のユーザのアクションに用いることで，プロトタイプ生成時にアクションから入力形式を持った入力項目に変換することができる．例えば，「入力する」はテキスト入力形式に対応し，「実行する」はボタン形式に対応する．これにより，従来方法の問題点として，実装に非依存な自然言語を用いることで，1)顧客は実際の UI におけるインタラクションが想起しづらい，2)顧客または開発者が個々に UI の実装イメージを自由に想起し，誤解が生じるという 2 つの問題点を解決することができる．

クラス図の定義：インタラクション中の入出力データの項目に対して正しい構造を定義するため，クラス図を定義する．業務を正しく遂行するフローでは，最低限定義すべきデータが必ず UI に表れる．そのため，概念レベルのクラス図しか扱わない従来方法に比べ，提案方法では入出力単位で全サービスを通して一貫に定義することで，定義すべきデータの漏れを回避することができる．

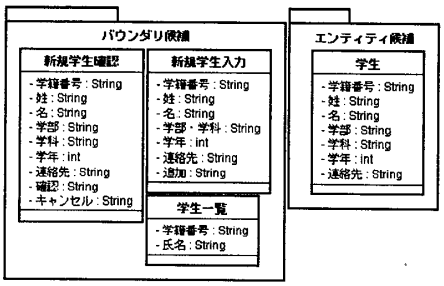

図 5　クラス図

図 5 は，図 3 のオブジェクトノードに付与されるクラスの一覧を示す．エンティティ候補のクラスは，システム内部を流れるオブジェクトノードを分類するクラスを対象とする．例えば，学生クラスは，学生情報を保持する．また，バウンダリ候補のクラスは，ユーザの入出力に直接係わるオブジェクトノードのみを分類するクラスを対象とする．例えば，新規学生入力クラスは，新規の学生情報を入力するフォームを表す図 6 は，クラス図までを定義したモデルから生成された図 4 部分のプロトタイプである．この時点でインタラクションアクティビティ図とクラス図の検証・妥当性確認および修正が行える．

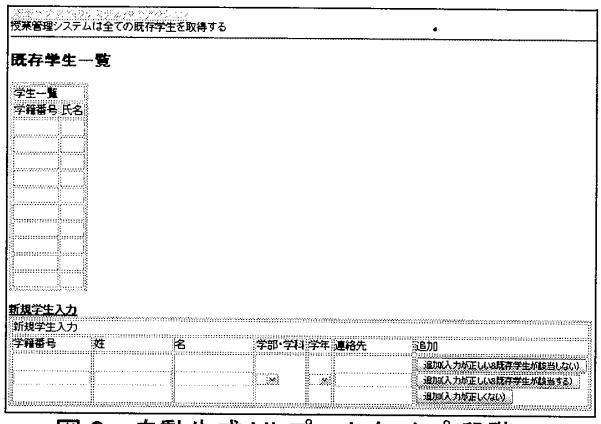

図 6　自動生成 UI プロトタイプ(段階 2)

オブジェクト図の定義：クラスに定義された入出力項目名とその実体の関係を明らかにし，検証・妥当性確認を行うために，オブジェクト図を定義する．オブジェクト図では，特定の利用状況に沿ってクラスのインスタンス仕様を定義する．このインスタンス仕様をプロトタイプに反映することで，プロトタイプに対する顧客の理解性を向上できる．

なお，クラス図のクラスを分類子としてインスタンス仕様に割り当てることで，両モデルの連携が可能である．

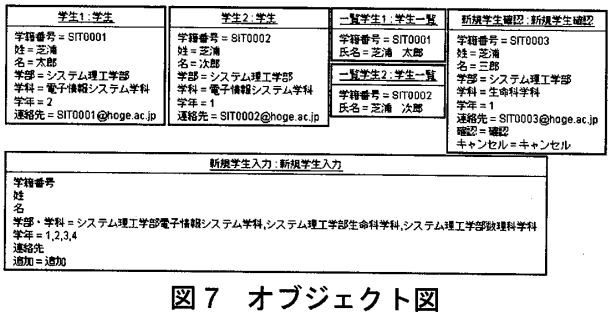

図 7　オブジェクト図

図 7 は SIT0001 と SIT0002 の既存学生が登録されている状況で，新しく SIT0003 の新規学生を入力することを想定する．図 8 はオブジェクト図までを定義したモデルから生成された図 4 部分のプロトタイプである．この時点でインタラクションアクティビティ図，クラス図，オブジェクト図の検証・妥当性確認および修正が行える．

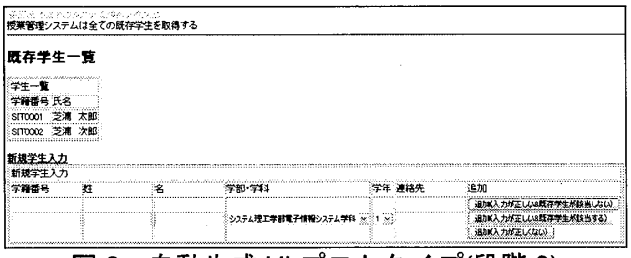

図 8　自動生成 UI プロトタイプ(段階 3)

シナリオの定義：網羅的なインタラクションフローが実際に業務が遂行できることの検証・妥当性確認を行うために，さらに具体的な利用状況を想定したフローをシナリオとして定義する．シナリオでは，インタラクションアクティビティ図に対して，具体フローを定義したものであり，具体データを指定することもできる．シナリオはサービス名，シナリオ名，シナリオ開始直前のシステム内部のデータを表す事前データ，シナリオ手順，シナリオ終了直後のシステム内部のデータを表す事後データにより構成される．

シナリオは特定のフォーマットにより定義された自然言語記述であるが，そのほとんどはインタラクションアクティビティ図から自動生成される．シナリオは，シナリオ名を定義し，事前データ，シナリオ手順，事後データのそれぞれに具体データを割り振ることで完成する．

プロトタイプ自動生成手法では，シナリオに沿って一路的なフローを表現するプロトタイプを生成できる．シナリオでは，オブジェクトノード単位で具体データを詳細に割り振ることができるため，あたかも動的にデータが変化するようなビューを持つプロトタイプが生成可能である．

ナビゲーションモデルの定義：サービスの接合の観点から個々のサービスのフローの検証・妥当性確認を行うために，サービス全体のフローを表現するアクティビティ図(以下，ナビゲーションモデル)を定義する．ナビゲーションモデルでは，インタラクションアクティビティ図のサービス名をアクションとし，システムに対する権限をパーティションとして，サービス実行順序フローを定義する．

そして，前後関係にあるアクションで前位置の事後条件と後位置の事前条件が合致する画面を接合することで，ユーザがサービス全体のフローを確認できるようなプロトタイプを生成できる．

4. 要求分析の比較評価実験

4.1 目的

提案方法が，従来方法に欠けていた分析観点を補い，かつ作業時間コストを軽減して，高品質な要求分析モデルを定義することを検証するために，従来方法との比較評価実験を行った．ここで，高品質な要求分析モデルとは，顧客が妥当性を判断したインタラクションの振る舞いおよびデータを正確に反映した要求分析モデルを指す．以降，便宜的に従来方法で定義されるモデルも要求分析モデルと呼ぶ．

4.2 実験手順と開発事例

UML の基礎知識と業務系 Web アプリケーション開発経験を有する本学 2008 年度の学部 4 年次の学生 3 名(A,B,C)を要求分析者とし，および大学院 2 年次の学生 2 名を顧客として，図書管理システム開発(α)とスケジュール情報共有システム開発(β)における要求分析を行う．

提案方法を適用する要求分析者には，要求文書，実験ルール定義書，UI プロトタイプ自動生成ツールとそのマニュアルを配布する．また，従来方法を適用する要求分析者には，要求文書，実験ルール定義書を配布する．

要求文書とは，図書管理システムまたはスケジュール情報共有システムにおいて，顧客の要求するサービスの概要と既存の業務データを定義する非形式文書である．サービスの概要は，主に 2 種類の情報を記述する．第 1 に，要求するサービス名やその要求動機である．第 2 に，業務で満たすべきルールである．一方で，既存の業務データは，システム化以前の業務データを記述する．

実験ルール定義書とは，実際の開発を想定した状況作りや，実験データの収集のためのルールを記述するものである．以下にその根幹となるルールを説明する．

1. 一般に顧客は，システムに不慣れであり，要求分析モデルを直接理解できる可能性は低い．そこで，実際の開発を想定して，要求分析者はプロトタイプのみを用いて顧客との要求分析モデルの妥当性確認を行う．従来方法では，プロトタイプに代替するものとして画面イメージを自由形式により作成するものとする．以降，便宜的に画面イメージもプロトタイプと呼ぶ．
2. 実験期間は，顧客が要求分析モデルを妥当と判断し，かつ，要求分析者がモデルとプロトタイプが整合したと判断するまでの期間とする．
3. 作業の公平性を期すため，要求分析者間の情報共有を禁止する．

また，評価のために要求分析者に残してもらう実験データとして以下のものがある．

作業時間として，要求分析モデルの定義時間，プロトタイプの開発時間，顧客との要求分析モデルの妥当性確認に要した時間を要求分析者が記録する．要求分析モデルの履歴として，プロトタイプから要求分析モデルの修正箇所を発見した時点を区切りとして，JUDE ファイルを記録する．プロトタイプの履歴として，要求分析モデルの履歴を記録するタイミングで，そのときのプロトタイプを記録する．UI プロトタイプ自動生成ツールに対する改善要望を，自由記述形式の文書で記録する．

定義義務を課す要求分析モデルは，提案方法と従来方法を対比するために，インタラクションの網羅的な振る舞いとシステム全体のデータを定義するモデルとする．提案方法では，インタラクションアクティビティ図とクラス図を義務とする．従来方法では，ユースケースモデルとクラス図を義務とする．なお，提案方法では，要求分析モデルの品質の向上のために，要求分析者がオブジェクト図，ナビゲーションモデル，シナリオを定義することを許容する．従来方法でも，同様に自然言語記述によりサービス全体の流れやシナリオ，具体データを定義することを許容する．

以下に，各開発事例の概要を説明する．

図書管理システムは研究室の図書の貸出管理を目的としたシステムであり，図書管理とユーザ管理の 2 種を柱としたサービスを提供するシステムである．要求文書として，要求サービスは 8 種あり，既存の業務データは 12 個ある．一方で，スケジュール情報共有システムは，研究室内のゼミや会議等のスケジュール情報を研究室メンバー間で共有することを目的として，スケジュール管理，ユーザ管理，グループ管理の 3 種を柱としたサービスを提供するシステムである．要求文書として，要求サービスは 16 種あり，既存の業務データは 23 個ある．

4.3 評価方法

業務を正しく容易に遂行するためのインタラクションの振る舞いとデータは，要求分析モデルに正しく定義すべきである．そこで，プロトタイプの画面遷移フローおよび画面単位の入出力データ名について，要求分析モデル中のモデルとの整合率を測る．プロトタイプ自動生成手法では，上記の整合性は自動生成で保証できるため，従来方法における不整合率を測ることで提案方法の有効性を評価する．

そして，高品質な要求分析モデルが定義可能であっても，作業時間コストが高い場合，納期の観点から問題である．そこで，要求分析モデルの定義時間とプロトタイプ開発時間を測る．なお，提案方法の定義時間では，プロトタイプ自動生成手法の学習時間を含む時間となる．

4.4 実験結果

表 2 は，要求分析モデルの定義時間と UI プロトタイプの開発時間である．プロトタイプ自動生成手法では，基本的にプロトタイプ開発時間が 0 時間となるが，要求分析者 C の開発事例 β においては，ツールの不具合により多少の時間がかかっている．

表 2　要求分析の作業時間

開発事例		作業時間					
		A		B		C	
		M	P	M	P	M	P
α	各種作業時間[時]	19.4	0	20.6	0	15.4	14.1
	合計作業時間[時]	19.4		20.6		29.5	
β	各種作業時間[時]	15.4	23.8	24.0	33.4	86.3	1.1
	合計作業時間[時]	39.2		57.4		87.4	

M:モデル定義時間,P:プロトタイプ開発時間,
■:提案方法,□:従来方法

表 3 は，従来方法において，プロトタイプとユースケース記述の間の入出力データ名の整合率を測ったものである．計測画面数は全てのユースケースを対象としており，画面上データ総数は各画面に定義されたデータの総数である．また，ユースケース記述定義の画面対応データ総数は，プロトタイプ上に定義されたデータ名を基準として，ユースケース記述のデータ名と完全に一致するものを手動で計測した．そして，逆に不整合率とはデータ名が完全に一致しなかった割合である．

表 3　従来方法の入出力データ名の不整合率

項目	A-β	B-β	C-α
計測画面数[個]	117	190	74
画面上データ総数[個]	1202	1740	379
ユースケース記述定義の画面対応データ総数[個]	438	1018	189
不整合データ総数[個]	764	722	190
不整合率(不整合データ数/画面上データ総数) [%]	63.56	41.49	50.13

表 4 は要求分析モデルのクラス数または属性数を計測した結果を示す．

表 4　要求分析モデルのクラス数と属性数

項目	A		B		C	
	α	β	α	β	α	β
クラス[個]	13	4	18	5	4	98
属性[個]	61	35	89	37	9	383

また，表 5 に提案方法におけるインタラクションアクティビティ図のアクション数と従来方法におけるユースケース記述のフローのステップ数を示す．なお，従来方法においてユーザとシステムのステップ数の和が，ステップの合計計測数と一致しない原因として，主語の欠如したステップがありユーザまたはシステムに分類できないためである．

表 5　要求分析モデルのアクション数またはステップ数

開発事例		A		B		C	
		U	S	U	S	U	S
α	各種計測数[個]	106	78	112	94	59	130
	合計計測数[個]	184		206		245	
β	各種計測数[個]	103	174	197	190	757	432
	合計計測数[個]	362		480		1189	

U:ユーザのアクション数またはステップ数,
S:システムのアクション数またはステップ数

そして，表 6 に要求分析者とユーザの間の要求分析モデルの妥当性確認回数とその作業時間を示す．

表 6　要求分析モデルの妥当性確認回数と作業時間

		α	β
A	回数[回]	2	4
	合計時間[分]	67	315
B	回数[回]	3	6
	合計時間[分]	55	539
C	回数[回]	4	7
	合計時間[分]	105	514

5. 考察

5.1 プロトタイプ自動生成手法の有効性

表 2，表 3 より開発規模が大きく，モデル定義時間が短い程，要求分析モデルとプロトタイプ間のデータの不整合率が高くなることが明らかとなった．この結果に対し，提案方法ではモデルの定義時間の観点から，両成果物の整合性を低コストで保証できることと，複雑なシステムに対して

も要求分析者が質の高い要求分析モデルを定義できるまでモデル定義を継続するように誘導できたと考えられる．

提案方法では，図書管理システム開発では学習時間も含めているにも係わらず従来方法の要求分析よりも短期間の要求分析が完了している．また，本実験で義務ではないナビゲーションモデルとオブジェクト図を要求分析者が積極的に定義していたことから，プロトタイプ自動生成手法の要求分析に対する適用容易性が評価できる．

表 3，表 5 から従来方法では，データの不整合率が高く，かつユーザのアクションの定義が少ないため，業務遂行可能性と業務遂行容易性に係わるユーザのアクションを十分に分析できていない．

例えばユースケース記述では「アクターは借りる図書が該当した図書でよいか確認する」というステップがあり，これはアクターが「該当した図書」の情報を視認しながら確認操作を行うものと考えられる．しかし，該当した図書の情報が具体的に何であるかはこのステップからは読み取れない．一方で，画面イメージでは「該当した図書」とは記述されずに，「図書名」「著者名」「出版社名」のように粒度の細かい単位で「該当した図書」に相当するようなデータが表現されていた．つまり，プロトタイプと要求分析モデルの間で記述されるデータの粒度にギャップが存在した．このギャップを解消する場合，最終的には実際の UI を想定したプロトタイプの方に粒度を合わせるべきである．

提案方法では，要求分析モデル中の表現がプロトタイプに直接反映される．そのため，要求分析者はプロトタイプ上に表現すべき粒度を意識しながら，入出力データを要求分析モデルに定義できるため，上記のギャップは生じない．そして，一意に解釈できるような適切な粒度で定義した入出力データに対するユーザのアクションを定義することになるため，当該アクションを十分に分析することができる．

また，従来方法では，プロトタイプを Excel や Word により 1 シートまたは 1 ページを 1 画面として作成するか，紙ベースで作成していたが，プロトタイプの画面遷移については，要求分析モデルとの対応が明確に理解できる仕様が皆無であったため，画面遷移とユースケース記述のフローの整合性を検証することは不可能であった．提案方法では，網羅的な画面遷移フローはインタラクションアクティビティ図およびナビゲーションを確認することで検証できた．

さらに従来方法のプロトタイプでは，具体データをプロトタイプに定義して説明を行っていたが，仕様としての具体データの定義は皆無であった．このことから，提案方法の具体データの明確な仕様化とプロトタイプへの反映の実現は，要求分析者がプロトタイプを開発する思考と適合していると考えられる．

表 6 に示すように自動生成されたプロトタイプと手動作成されたプロトタイプの違いによる要求分析モデルの妥当性確認の回数や時間に差が見られなかった．これは，自動生成プロトタイプが，自由な表現が可能な手動作成プロトタイプと同程度の確認効果があると考えられる．

5.2 問題点

第 1 の問題点として，提案方法におけるクラス数及び属性数(表 4)が異常に多数である点が挙げられる．この原因として主に以下の 2 つに分類される．第 1 に，クラス図の定義方法を誤解することである．通常，クラスとインスタンス仕様の関係は 1 対多である．しかし，要求分析者 B はクラスとインスタンス仕様の関係を 1 対 1 と誤解したため，意味の重複するクラスをインスタンス仕様の数に合わせて定義していた．この問題は要求分析者にプロトタイプ自動生成手法の学習の徹底を行うことで解決できる．ただし，ここで挙げた問題は，一般的な UML の知識に係わる問題であるため，プロトタイプ自動生成手法の適用の難しさを指摘する問題ではない．

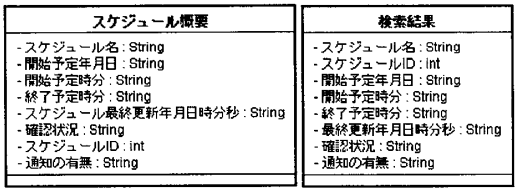

図 9　同義のデータ構造をもつクラス定義

第 2 に，統合可能な同義のデータ構造が散在することである．図 9 は，統合可能であるような同義のデータ構造をもつクラスが定義された例である．要求分析モデルの履歴によると，スケジュール概要クラスの方が定義時期は先であり，その後の要求分析モデルの洗練時に検索結果クラスは定義されていた．この原因は，要求分析者が最初に別々のデータ構造を持つクラスとして両クラスを定義したが，ユーザとの要求分析モデルの妥当性確認を経て，クラスを洗練した結果，同義のデータ構造が定義されたことにある．このような冗長なデータ構造の定義は排除すべきである．

第 2 の問題点として，従来方法および提案方法では，要求分析者が入力例外時の例外フローへの分岐条件を明確に仕様化するように誘導できない可能性がある．例えば要求分析者 A は入力に対して「全てに不備がない場合」という分岐条件を定義していた．また，プロトタイプ自動生成手法において，要求分析者 C は入力例外に対する分岐条件を十分に洗い出して定義したインタラクションアクティビティ図を定義していたが，フローが複雑になり，可読性が著しく低下したという別の問題もある．

第 1，第 2 の問題点を解決するために，サービス成立時にデータが満たすべき性質を明確に定義する方法が必要である．第 1 の問題点の解決方針について，当該性質は統合すべきデータを洗い出す観点の 1 つとなる．例えばエンティティとしての「図書名」と図書検索時に入力を要する「図書名」は性質が異なる．具体的に前者では空値入力が許容されないが，後者では空値入力が許容される．このように同様な概念のデータであったとしても，特にバウンダリとエンティティの間でクラスごとに制約が異なる場合がある．従来方法ではエンティティレベルのクラス分析のみを行うため，この議論自体が難しい．そこで，バウンダリ候補クラスの定義も含めるプロトタイプ自動生成手法において，例えば形式言語である OCL(Object Constraint Language)式によりクラス単位で満たすべき性質を明記し，性質の等しさとデータ名を判断基準として統合可能なクラスの検出することで，この問題の解決を図る．

第 2 の問題点の解決方針について，インタラクションにおけるシステムへの入力データは，ユーザのあらゆる入力操作を想定したものでなくてはならない．そのため，特に入力例外時の入力操作のパターンを分析する必要がある．

しかし，そのパターン数は入力項目数に比例して大きく増加するため，モデルの可読性を考慮して，既存モデルへの追記ではなく，別モデルとして定義すべきである．そこで，プロトタイプ自動生成手法で，ユーザのアクション系列から入力パターンを網羅的に検索し，前述の OCL 式を基準に入力例外時のパターンを自動的に検出できるようにする．このように入力例外における例外フローへの分岐条件を明確化することで，問題の解決を図る．

第 3 の問題点として，改善要望の記録から現状のプロトタイプ自動生成手法では，データやメッセージを自由にレイアウトすることができないことへの問題点が挙がった．一方，従来方法では，自由なレイアウトが可能であるが，当該レイアウトの明確な仕様が定義されることはないため，要求とレイアウトの対応付けの観点から望ましい状態とはいえない．そこで，プロトタイプ自動生成手法では，要求分析モデルを基礎とした UI レイアウト仕様を実現し，その仕様を UI プロトタイプに反映することで解決する．UI レイアウト仕様では，複数データの配置やテキストボックスの幅の入出力項目の性質の仕様化に向けた内容を検討し，当該仕様をプロトタイプへ反映することができるようにすることで問題の解決を図る．

6. 関連研究

ユースケース駆動要求分析の 1 つのポイントとして，システム化する顧客業務に対して，必要なユースケースを如何にして抽出するかの問題がある．中鉢ら[6]の研究では，SBVA(Scenario-Based Visual Analysis)法により業務シナリオからユースケースを導出する手法を提案している．提案方法では，本手法によりアクターとユースケースが明らかになった時点から適用可能であると考えられる．このような他の手法との連携は，提案方法がユースケース駆動要求分析を踏襲している利点の 1 つである．

三部ら[7]は，我々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．この観点では，提案方法も同様である．提案方法では，本方法がサポートしない具体データを定義することで，シナリオをより鮮明に理解することができる．例えば，検索系サービスにおける絞込み機能では，絞込み前後の検索結果の動的な変化がシナリオの理解性の向上に不可欠である．また，シナリオベースによる分析では，シナリオの統合が実現できなければ，システムの全体像の把握が困難である．提案方法では，全体像や網羅的なフローを表すナビゲーションモデルやインタラクションアクティビティ図，具体フローを表すシナリオのどちらの観点もモデルとして定義可能である．

Jayaraman[9]らはユースケースの検証の観点において，ユースケースチャートを利用したユースケースのシミュレート方法を提案している．しかし，ユースケース駆動の要求分析では，インタラクションに対する顧客の理解が必須であり，十分にインタラクションの妥当性を議論できるモデルを定義した上で，ロジックを検証できる必要がある．

Somé[10]らは，ユースケースの妥当性確認の観点において，テキストベースのシナリオによるユースケースのシミュレート方法を提案している．しかし，テキストベースのシナリオでは，顧客はインタラクションの理解が可能であるが，個々人によって異なる UI の実装イメージを持つために誤解が生じる．また，具体データを定義しないため，顧客のシミュレーション結果の理解性に疑問が残る．

これらの研究[9,10]に対して，提案方法では，インタラクションを議論する十分なモデルとプロトタイプを提案しており，具体データを含めたシナリオに基づいてプロトタイプを自動生成することで，シナリオによるモデルの妥当性確認を実現している．

7. まとめ

本稿では，評価実験を通して，提案方法が，従来方法の定義観点を網羅し，かつ従来方法よりも低作業コストで高品質な要求分析モデルを定義できることが明らかとなった．また，従来方法と並列して手動作成された画面イメージに具体データが定義されていたことから，開発者は具体理解性を考慮して画面イメージを作成したことは明らかであるが，従来方法では理解容易な確認方法も含めて具体データの形式的な定義方法が欠落していた．提案方法では，モデルとして形式的に具体データを定義し，プロトタイプに反映することができるため，従来方法で補えなかった定義観点も補えている．

今後の課題として，業務ルールをインタラクションアクティビティ図やクラス図と連携できる形で別モデルとして形式的に定義する方法を検討する．また，UI の体系的な分析・設計を実現するためのレイアウトの仕様化方法と仕様の内容を検討する．

参考文献

[1] 大西 淳, 郷 健太郎, 要求工学, 共立出版 (2002).
[2] 玉井 哲雄, ソフトウェア工学の基礎, 岩波書店 (2004).
[3] UML : http://www.uml.org/
[4] Jacobson, I., Christerson, M., Jonsson, P. and Övergaard. G., “Object-oriented software engineering: A usecase driven approach”, Addison-Wesley Publishing (1992).
[5] Cockburn, A., “Writing Effective Use Cases”, Addison-Wesley Publishing (2000).
[6] 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, pp.813-828 (2005).
[7] 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).
[8] Elkoutbi, M., Khriss, I. and Keller, R.K., “Automated Prototyping of User Interfaces Based on UML Scenarios”, Automated Software Engineering, Vol.13, No. 1, pp.5-40 (2006).
[9] Jayaraman, P.K., Whittle, J., “UCSIM: A Tool for Simulating Use Case Scenarios”, Companion to the Proc. of the 29th International Conference on Software Engineering (ICSE’07 Companion), pp.43-44 (2007).
[10] Somé, S.S., “Use Case based Requirements Validation with Scenarios”, Proc. of the 13th IEEE International Conference on Requirements Engineering (RE’05), pp.465-466 (2005).
[11] 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”, ソフトウェアエンジニアリングシンポジウム 2008, pp.79-86 (2008).
[12] ACM SIGSOFT, “Special Issue on Rapid Prototyping”, ACM SIGSOFT Software Engineering Notes, Vol.7, No.5 (1982).
[13] JUDE: http://www.change-vision.com/