ユースケース記述のためのフレームワークとメタモデル

中谷　多哉子[†]　玉　井　哲　雄[††]

オブジェクト指向開発では，ユースケースが要求仕様記述に用いられている．定型的なユースケースでは，記述に枠組みを与えることによって，記述者による差異を減少させ，生産性を向上させることができる．本稿では，ユースケースフレームワークを紹介し，そのメタモデルを提案する．メタモデルには，システムの不変な属性を共有オブジェクトとして管理する構造を組み込んだ．これによって，仕様情報を共有するユースケース間の関連を明記することができるようになり，要求仕様書におけるユースケース間の一貫性を保つ構造を与えることができるようになった．本稿の最後に，メタモデルに基づいて試作したユースケース記述支援システムを紹介する．

Metamodel and Frameworks for Use Case Description

Takako NAKATANI [†] and Tetsuo TAMAI[††]

In object-oriented developments, use cases are defined as requirements specifications. In our previous works, we have proposed use case frameworks for typical use cases. The frameworks are effective for reducing the differences between expert engineers and novice engineers and improving productivity. In this paper, we also propose a metamodel of use case frameworks. The metamodel defines relationships between use cases via invariant objects, including entity objects and interface objects, in the system. Applying the metamodel for defining use case frameworks, we have come to the conclusion that it is useful for helping us maintain use cases.

1. はじめに

1.1 研究の背景

オブジェクト指向が開発現場に導入されるようになり，システムの要求定義にユースケースやユースケース図が用いられるようになってきた．ユースケースはJacobson の開発方法論で提案された要求仕様記述の形式であり，オブジェクト抽出，設計，テストケースなど，すべての後工程の情報源となる[7]．ユースケース には，システム外部にあるアクターの働きかけと，それによって起動されるシステムの動作が定義される．その記述には自然言語を用いることが多いため，ユーザが要求を確認するのも容易だといわれている[5]．

Jacobson が定義したユースケースは，事前条件，事後条件，基本系列と代替系列といった 4 つの項目から構成される[5),6]．しかし，その構造は，要求を仕様化するために十分であるとは言えない．たとえば，ユースケースの構造に関して，次のような問題がある．

- システムの不変項目の取り扱いが考慮されていない．
- 情報を介して制約を与え合うユースケース間の関連について言及されていない．
- ユースケースの数が爆発することがあるが，それに対する考慮がなされていない[1]．
- 基本系列と代替系列の対応関係を管理する構造がない[1]．

これらの問題を解決するためには，UML のメタモデル[12]よりも詳細な構造が必要である．

1.2 関連研究

ユースケースの構造化を検討するにあたって，ユースケース記述の内部構造と，ユースケース間の関連を表すユースケースの外部構造を考える．

Jacobson の後，ユースケースの内部構造を検討した研究がいくつかある．Cockburn は，ユースケースの数の爆発へ対処するために，多様性項目を導入した[1]．また，ユースケースの優先順位，非機能要求の取り込みを提案し，ユースケースを記述するためのテンプレートを提供した[2]．Coleman は，ユースケースの中に，他のユースケースの呼び出し，系列中に現れる条件分岐や繰り返しを定義するための構文規則を明示的に取り込んだ[3]．Larman[9]や Schneider と Win-

† SLagoon
e-mail: tina@slagoon.to
†† 東京大学大学院総合文化研究科広域科学専攻
Graduate School of Arts and Sciences,
University of Tokyo
e-mail: tamai@graco.c.u-tokyo.ac.jp

ters[11]も，ユースケースの優先順位，条件分岐におけるユースケースの記述方法などを提案している．実務面の適用では，吉田らが，要求仕様を管理するための識別子を提案した[14]．

ユースケースの外部構造として，UML ver.1.1 は extends 関係と uses 関係を提供している[12]．前者は基本ユースケースと，それに新たな系列を付加した拡張ユースケースとの間の関連を表し，後者はユースケースと，そこから呼び出されるサブユースケースとの間の関連を表す．これらのユースケース間の関連は主にユースケース図で参照されるが，Coleman は，ユースケーステンプレートにユースケースの拡張関係と呼び出し関係を明記する項目を設けた[3]．

Collins-Cope は，ユースケースを要求，インタフェース，サービスに分類し，要求モデルとして，個々のユースケースがどのように関係づけられるかを明らかにした．その目的は，ユースケースの適切な粒度と記述内容を定義するための記述の指針を示すことにあった[4]．

ユースケースの外部構造には，このような直接的なユースケース間の関連の他に，情報を共有することによって互いに制約を与え合う間接的な関連がある．たとえば，登録ユースケースで定義されるオブジェクトは，更新ユースケースや参照ユースケースでも参照される．特に参照ユースケースには，参照されるオブジェクトの属性がすでに定義された属性か，あるいは，それらの派生属性でなければならないという制約がある．先に示したいずれのユースケースも，共有するオブジェクトによって関連付けられるといったユースケースの間接的な外部構造については議論していない．また，このような間接的なユースケース間の関連は，個々のユースケースの記述だけでなく，ユースケース図でも把握することができない．

要求仕様の品質を向上させるためには，このような間接的なユースケース間の関係も，記述者が把握しやすい構造にして，ユースケースへ取り込む必要がある．

1.3 研究の方針

要求抽出作業では，多くの時間がユースケース間の一貫性の検証に費やされ，ユースケースの品質向上に努力が払われる．本稿では，ユースケース自体に一貫性を保てる構造を与えることで，一貫性検証の作業効率の改善を目指す．

ユースケースに一貫性が求められる一方で，ユースケース記述時の煩雑な作業の繰り返しが問題となっている[13]．ビジネス系システムでは，複数のユースケースで記述の重複が観察されている[13]．記述の重複は，記述の生産性を低下させるだけでなく，保守性も低下させる．記述の重複の中には，ユースケースが定型的な処理を定義していることに起因するものもある．筆者らは，これまで，これらの定型的なユースケースに対して，予めユースケースの記述の枠組みを与えることによって，記述の重複を解消する方法を提案してきた[10]．

本稿では，ユースケース記述の品質向上と生産性向上を目標として，次の問題解決方針を提案する．

(1) ユースケースの内部と外部の構造を与え，ユースケース間の制約関係を明確に定義する．
(2) 共通の系列を持つユースケースに対して，ユースケースの枠組みを提供する．
(3) ユースケースを記述するための支援システムを提案する．

本稿の構成は以下のとおりである．次の第 2 節でユースケースの枠組みとしてユースケースフレームワークを紹介する．第 3 節では，適用事例を示し，第 4 節では，ユースケースフレームワークから導いたメタモデルを示す．第 5 節では，本研究で試作したユースケース記述支援システムの概要を紹介し，メタモデルの有効性を考察する．

2. ユースケースフレームワーク

2.1 定　　義

ビジネス系システムにおける定型的なユースケースには，共通の基本系列や代替系列が定義される．そこで，定型的なユースケースの記述を構造化し，記述の枠組みを提供する．この枠組みをユースケースフレームワークと呼ぶ．

ユースケースフレームワークは，アプリケーションのユースケースで具体化されるべき情報をパラメータとして持つ．これらの情報に具体的な記述を与えることで，アプリケーション固有のユースケースを生成することができる．このユースケースをアプリケーションユースケースと呼ぶ．

ユースケースフレームワークから生成されたか，スクラッチから記述したかに関わらず，単一の目的を持つユースケースを基本ユースケースと呼ぶ．また，要求抽出で直接定義されるユースケースなど，複数の基本ユースケースを組み合わせて構成されたユースケースを複合ユースケースと呼ぶ．

2.2 例

以下に，ユースケースフレームワークの例として，登録ユースケースフレームワークを示す．ゴチック文字で表示した部分がアプリケーションユースケースで具体化するパラメータである．パラメータに対応する参照オブジェクトを定義すると，アプリケーションユースケースが生成される．例えば，プログラム委員長登録ユースケースでは，「登録対象オブジェクト」な

どのパラメータに「プログラム委員」といった固有のオブジェクトを与えることで，アプリケーションユースケースを得ることができる．

ユースケースフレームワークの例

- システム名：**システム名**
- ユースケース種別： 登録ユースケースフレームワーク
- ユースケース名：**登録対象オブジェクト** 登録ユースケース
- アクター：**オブジェクト 登録希望者**
- ゴール：**登録対象オブジェクト** を登録する
- 事前条件： アクターが**登録に必要な情報**を得ている
- 正常終了における事後条件：**登録対象オブジェクト** が登録済みとなったという結果をアクターが得ている
- 例外終了における事後条件：**登録対象オブジェクト** の登録が中止され，副作用が残っていない
- ユースケース起動動作：アクターはシステムに**登録対象オブジェクト** の登録希望を通知する
- ユースケース全体にかかる代替系列：
  アクターが登録希望を取りやめたときは，**登録対象オブジェクト** の登録を中止し，副作用を残さない
- 基本系列：
  1．システムはアクターに**登録対象オブジェクト** を登録するための**登録インタフェース**を提示する．
  :::代替系列 1-1: **登録インタフェースが提示されない**
  2．アクターが入手した**登録インタフェース**を介してシステムに**登録対象オブジェクト** の**登録に必要な情報**を渡すと，システムは渡された**データが正当である**という条件を満たし，かつ同じデータが登録されていないことを確認し，**登録対象オブジェクト** を**永続化方法**によって永続化する．
  :::代替系列 2-1: 正当なデータと判定できなかった場合は次のことを行う．
  ::::2e1-1．システムはデータが不当である理由を明らかにする．
  ::::2e1-2．システムはアクターにデータの不当個所と不当理由を，**データの不当個所と不当理由通知インタフェース**を介して通知する．
  ::::2e1-3．基本系列 2 を繰り返す．
  :::代替系列 2-2: 登録しようとした**登録対象オブジェクト** が既に登録済みである場合は，次のことを行う．
  ::::2e2-1．システムは，アクターに登録希望された**登録対象オブジェクト** が登録済みとなっていることを，**登録済みエラー通知インタフェース**を介して通知する．
  ::::2e2-2．基本系列 2 を繰り返す．
  3．システムはアクターに**登録対象オブジェクト** が新たに登録されたことを，**登録完了通知インタフェース**を介して通知する．
  :::代替系列 3-1: **通知に失敗した**
  4．アクターは，**登録対象オブジェクト** が登録されたことを，**登録確認インタフェース**によって確認する．
  :::代替系列 4-1: **確認に失敗した**

2.3 適用過程

アプリケーションシステムのためにユースケースを記述する開発者は，ユースケースフレームワークを利用し，アプリケーションユースケースを定義することになる．ビジネスシステムにおけるユースケースフレームワークの適用過程を以下に示す．

(1) アクターのアクティビティ図を記述し，各アクティビティ毎にシステムの開発対象となるか否かを識別し，要求を抽出する．
(2) 開発対象となるアクティビティについて，自由な形式でユースケースを記述する．
(3) 自由な形式で記述したユースケースを，データ登録，更新，削除などの基本的なデータ操作を組み合わせて再構成し，複合ユースケースを定義する．
(4) 複合ユースケースのレビューを行い，抽出した要求を確認する．
(5) 複合ユースケースを構成する基本ユースケースがユースケースフレームワークから生成できる場合は，レビューによってユースケースフレームワークの妥当性を確認する．
(6) 基本ユースケースを，ユースケースフレームワークのパラメータに情報を付加したアプリケーションユースケースとして生成する．
(7) ユースケースフレームワークのパラメータに与えた情報ををレビューし，アプリケーションユースケースの妥当性を確認する．

3. 例題への適用

図 1に「国際会議のプログラム委員長の業務[8)]」のユースケース図を示した．図中の右列に並ぶ二重丸で囲まれたユースケースが 6 種類のユースケースフレームワークである．図中の中間列に並ぶユースケースがユースケースフレームワークから生成されたアプリケーションユースケースであり，左列のユースケースが，アクターから直接起動される複合ユースケースである．

図 3に準拠して記述したアプリケーションユースケー

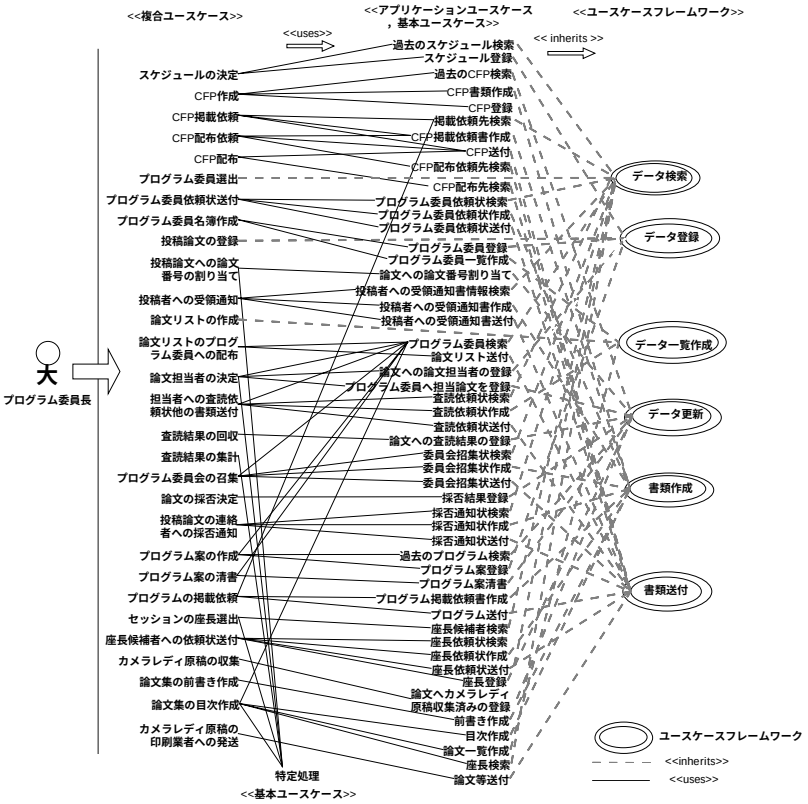

図 1 プログラム委員長業務のユースケース図とユースケースフレームワーク
Fig. 1 Use cases for a program chair and use case frameworks

スの例として,「国際会議のプログラム委員長の業務」問題から,「プログラム委員の登録ユースケース」を挙げる.

開発者が登録ユースケースフレームワークのパラメータに対して定義した参照オブジェクトは，システムのデータ辞書の項目として登録される．以下の例は，プログラム委員登録ユースケースで定義されたパラメータの内容を示したものである.

（開発者が定義する事項）
システム名 プログラム委員長業務
ユースケース種別 登録ユースケースフレームワーク
登録対象オブジェクト プログラム委員
登録に必要な情報 プログラム委員名（文字列），所属（文字列），役職（文字列）*，連絡先（文字列），専門分野（文字列），関係国（文字列）(*はオプション)
登録インタフェース プログラム委員登録インタフェース
:::代替系列 1-1: 登録インタフェースが提示されない なにもしない
データが正当である 関係国はすでに登録された国名であり，かつ，重複して同じ人を登録しない.（ただし，人の同一性は，名前と連絡先によって判断する）
永続化方法 XX データベースへの登録
データの不当個所と不当理由通知インタフェース 未定インタフェース
登録済みエラー通知インタフェース 未定インタフェース
登録完了通知インタフェース 未定インタフェース
:::代替系列 3-1: 通知に失敗した なにもしない
登録確認インタフェース 未定インタフェース
:::代替系列 4-1: 確認に失敗した なにもしない

開発者がユースケースフレームワークに対して与えた上記の情報によって，次のようなユースケースの形式でレビューを進めることができる.

（レビュー関係者が参照するユースケース）
- システム名：**プログラム委員長業務**
- ユースケース種別：登録ユースケースフレームワーク
- ユースケース名：**プログラム委員**登録ユースケース
- アクター：**プログラム委員長**
- ゴール：**プログラム委員**を登録する
- 事前条件：アクターが**プログラム委員名（文字列），所属（文字列），役職（文字列）*，連絡先（文字列），専門分野（文字列），関係国（文字列）(*はオプション)**を得ている
- 正常終了における事後条件：**プログラム委員**が登録済みとなったという結果をアクターが得ている
- 例外終了における事後条件：**プログラム委員**の登録が中止され，副作用が残っていない
- ユースケース起動動作：アクターはシステムに登録対象の登録希望を通知する
- ユースケース全体にかかる代替系列：アクターが登録希望を取りやめたときは，**プログラム委員**の登録を中止し，副作用を残さない
- 基本系列：
  1. システムはアクターに**プログラム委員**を登録するた

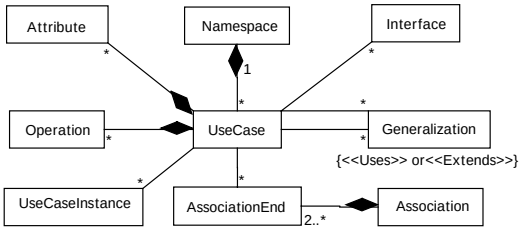

図 2 UML によるユースケース記述のためのメタモデル
Fig. 2 Use case metamodel by UML ver.1.1

めの**プログラム委員登録インタフェース**を提示する．
2. アクターが入手した**プログラム委員登録インタフェース**を介してシステムに**プログラム委員**の**プログラム委員名（文字列），所属（文字列），役職（文字列）*，連絡先（文字列），専門分野（文字列），関係国（文字列）**(*はオプション) を渡すと，システムは渡された**関係国はすでに登録された国名であり，かつ，重複して同じ人を登録しない．（ただし，人の同一性は，名前と連絡先によって判断する）**という条件を満たし，かつ同じデータが登録されていないことを確認し，**プログラム委員**を XX **データベースへの登録**によって永続化する..
..... 中略.....

- 代替系列 :
  1. 基本系列 1 において,
  ::代替系列 1-1:登録インタフェース**プログラム委員登録インタフェース**が提示されない場合は，**なにもしない**
  … 以下省略

3.1 ユースケースフレームワークの評価

図 1に示すように,「国際会議のプログラム委員長の業務」の問題記述から 29 種類の複合ユースケースを抽出したところ，これらのユースケースは，56 種類の基本ユースケースの組み合わせで構成できることがわかった．さらに，これらの基本ユースケースのうち，51 種類のユースケースについては，データ登録，更新，削除，検索，一覧表作成，書類作成という 6 種類のユースケースフレームワークから生成できるアプリケーションユースケースとして定義することができた[10]．

ユースケースフレームワークを適用することによって，適用しない場合よりも新たな記述量を減らすことができる．このように，ビジネス系システムでは，生産性向上への効果は大きい．また，各ユースケースで新たに開発者が記述する量が減るため，重点的にレビューする箇所も減り，レビューの効率を向上させることもできる．

4. ユースケースメタモデル

4.1 ユースケースのメタモデル

UML の ver.1.1 は，図 2に示すユースケースの記述構造を表すメタモデルを提供している．

このメタモデルは，ユースケースの記述に関するモデルを提供する．そのため，

- ユースケース間の関連を uses，extends という二つのステレオタイプによって定義できること，
- 複数の利用者インタフェースがユースケースに定義される可能性があること

は表されているが，複数のユースケースが，共有するオブジェクトを介して互いに制約を与え合うといった関連まではモデル化の対象になっていない．

山浦らは，ビジネス系システム開発で定義したユースケースを 4 種類に分類し，そこからユースケースメタモデルを導出した[13]．山浦らのメタモデルはユースケースの内部構造について検討したモデルとなっており，ユースケース間の直接的な関係は定義されているものの，UML のメタモデルと同様に，ユースケース間の間接的な関係は考慮されていない．

我々は，山浦らのメタモデルに

- ユースケース間の間接的な構造
- システム全体の不変属性

を導入し，図 3に示すメタモデルを構築した．

メタモデルの構成要素を以下に説明する．

契約 事前条件，事後条件，アクター，記述内容を属性として持ち，例外状態をキーとして他の契約との関連を持つ．

関与者 契約の関与者．

システム ユースケースが開発対象としているアプリケーションシステム．契約の関与者となる．

アクター 契約の関与者．

条件 契約から事前条件，事後条件として参照される．条件の記述は，パラメータに対して具体的な参照オブジェクトが与えられて完成する．

単純契約 記述内容が 1 項目である契約．

複合契約 記述内容が二つ以上の子契約の系列になっている契約．

一般ユースケース 複合契約の一種である．記述内容は，定義者がスクラッチから記述した一般的なユースケース．明確な構造上の制約が与えられないため，ここから参照されるオブジェクトが他のユースケースからも共有されるという制約が守られる保証はない．

ユースケースフレームワーク 複合契約の一種である．具体的な参照オブジェクトを与えるパラメータを持つユースケース．フレームワークの名前，起動動作，例外時の終了条件[1)]を属性として持つ．

参照オブジェクト 契約や条件から参照されるオブジェクト．システムからはデータ辞書の項目として管理される．

エンティティオブジェクト 分析モデルで抽出モデル化の候補となるオブジェクト．Jacobson のエン

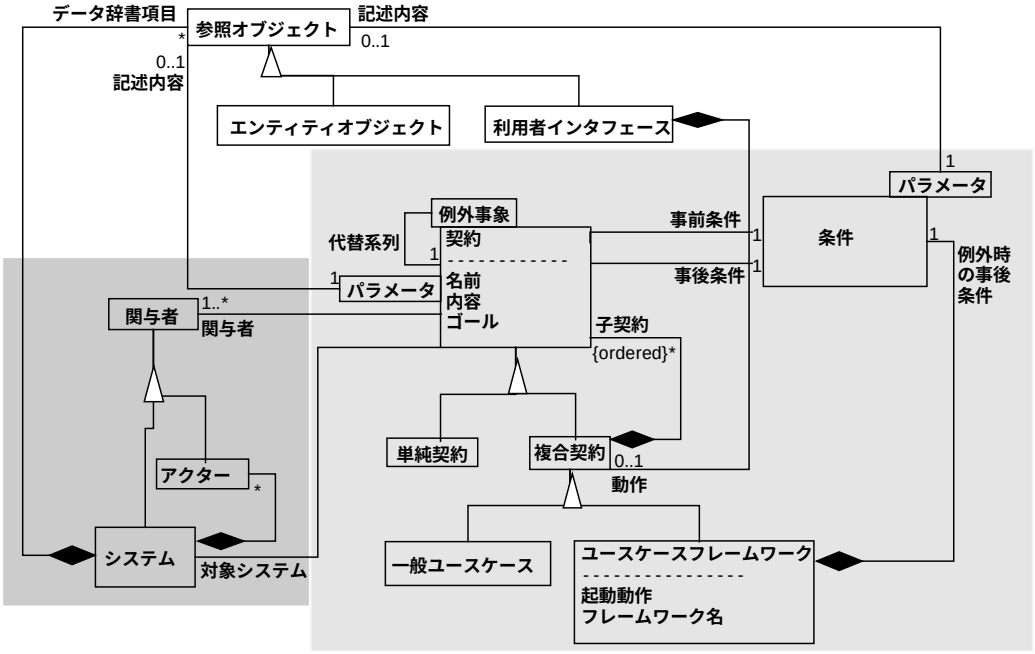

図 3 ユースケースメタモデル
Fig. 3 Use case metamodel for use case frameworks and application use cases

ティティオブジェクトと同義である．

利用者インタフェース 契約から参照されるウィンドウやコンポーネントなどの利用者インタフェース．

操作方法は，ユースケースで記述されることもあるため，複合契約と関連を持つ．

メタモデルでは，起動動作をユースケースの新しい項目として設けた．ユースケースは，アクターのシステムへの働きかけによって起動される以外に，タイマー事象など，システム自身によって起動される[2]．従来のユースケースでは，ユースケースの起動動作が基本系列の一項目として記述されていたため，起動動作を明記する強制力に欠け，記述漏れや曖昧に記述されてしまう場合もあった．この問題を解決するために，起動項目を事前条件や事後条件のように，記述すべき項目として追加した．

さらに，代替系列は，例外事象をキーとして主たる契約に関連付けられる別の契約として構造化した．

一般のユースケースでは，基本系列と代替系列が別の項目に定義されていたため，その対応関連を把握するには，自然言語で記述された内容から読みとるしかなかった．このような記述方法は，レビューをユースケースから生成されるシナリオに沿って行う際には便利であるが，ユースケースのメタモデルや記述する手順が，レビュー時の制約を受ける必要はない．代替系列を注意深く読むと，基本系列の特定の項目に対応して発生する例外事象に対して記述される代替系列と，ユースケース全体にかかる例外事象に対して記述される代替系列とに分類できることがわかる．

これは，契約には，契約が履行されないような例外事象が起きたときに何をすべきかという別の契約への関連が存在すると一般化して考えることができる．メタモデルによる構造化によって，前者の代替系列は，基本系列を構成する各項目の契約オブジェクトに，例外事象をキーとして代替系列に相当する契約を関連付ける構造を与えることができる．同様に，後者の代替系列は，ユースケースという契約オブジェクトに，例外事象をキーとした代替系列が関係づけられる構造に取り込まれる．

記述されたユースケースがメタモデルに基づく構造を持っていれば，従来のユースケースの記述に従うユースケースを表現することは容易である．

5. 考　　察

5.1 メタモデルに基づく記述支援

ユースケースフレームワークのメタモデルに基づいて，ユースケース記述支援システムを試作し，メタモデルを検証した．試作したシステムの操作結果を図 4 を示す．

操作方法は次のようになる．

(1) 開発対象システム名，ユースケース名，利用するユースケースフレームワーク名を指定する．

(2) 記述したいユースケースとユースケースフレー

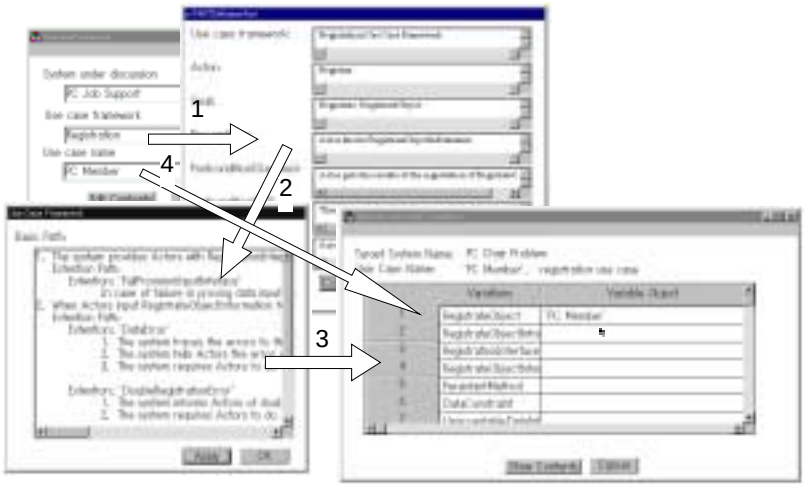

図4 ユースケースフレームワークを活用するユースケース記述支援システム
Fig. 4 A Concrete use case "PC Member Registration" can be created by the Registration use case framework

ムワークの内容が合うか否かを確認するために，ユースケースフレームワークの構造が表示される．

(3) 続いて，ユースケースフレームワークの基本系列が表示される．

(4) 多様性項目を登録する．この例では，すでにユースケース名で登録対象オブジェクト名を与えているので，表には「PC Member」が表示される．

各項目は，すでに該当するデータがデータ辞書に登録されている場合は，ユースケース名と与えたパラメータの名前（たとえば「登録対象オブジェクト」）を指定することによって，他のユースケースからも参照できるようになる．

メタモデルによって与えられたユースケース間の間接的な関連によって，共有されたオブジェクトが変更されても，そのオブジェクトを参照するユースケースに反映させることが容易となり，ユースケース間の一貫性が保持できる．

このツールは試作段階であるが，メタモデルに期待されたユースケース間のオブジェクト共有を実現することができた．今後は，XML の導入などを検討し，開発現場への適用によってメタモデル導入によるユースケースの信頼性の向上を評価していきたい．

5.2 メタモデルの有効性

要求分析では，Collins-Cope が提案した要求ユースケース，インタフェースユースケース，サービスユースケース[4]のうち，要求ユースケースの抽出に，より多くの時間をかけるべきである．そのためには，ユースケース記述とレビューの生産性を向上させ，さらにユースケースの品質を向上させる仕掛けが必要である．本稿で提案したユースケースフレームワークは，問題領域分析以降のユースケース記述の生産性向上とレビューの効率向上を目指す手段として適用することができる．また，ユースケースフレームワークから求めたユースケースのメタモデルは，品質向上のための仕掛けとなる．

ユースケースフレームワークのメタモデルは，Jacobson が提案した古典的なユースケースの構造だけでなく，Cockburn によるユースケーステンプレート[2]など，ユースケースの構造化に関する議論を踏襲することができた．特に，

- ユースケース間の直接的な関連である uses 関係，includes 関係，extends 関係，
- Coleman が提案している連接，分岐，反復構造を持った基本系列[3]の構造，
- Collins-Cope の三種類のユースケース[4]間の関連

といった構造は，契約の入れ子構造として表現されている．

定型的なユースケースフレームワークは実際にどの程度有効であろうか．「国際会議のプログラム委員長の業務」の例題を解いたユースケース図（図1）からわかるように，ビジネス系システムで定義されるユースケースの多くは，定型的なユースケースを組み合わせて作ることができる．段階的に詳細化しなければ記述できないような複雑で大規模なユースケースも，ユースケースが複合契約の一種であることによって構造を定義できる．

ユーザから抽出される要求は，基本的な要求を複合化していることが多い．本稿で紹介したユースケース

フレームワークは，アクターとシステムとの単純な相互作用の枠組みを記述したものである．そのため，要求抽出の段階では，複合ユースケースをユースケースフレームワークから生成できるアプリケーションユースケースに分割する必要がある．このようなフレームワークの適用過程は，構造化手法における段階的詳細化と，オブジェクト指向におけるボトムアップのクラスの再利用の過程を組み合わせた過程とよく対応する．図 1に示したユースケース図は，複合ユースケース，アプリケーションユースケース，ユースケースフレームワークの三層で表現したが，要求が複雑になれば，アプリケーションユースケースに至るまでに段階的に詳細化される複合ユースケースの層が存在することもあるだろう．

本稿で提案したユースケースのメタモデルは，ユースケースに明確な構造を持たせることによって，ユーザと開発者との対話を円滑化させる自然言語の良さを残しながら，開発効率を向上させる利点を持った要求仕様書として，ユースケースを適用していくためのモデルである．今後は，メタモデルに準拠したユースケースフレームワークを実用化していく過程で，度々現れる複合ユースケースに対して，より複雑なフレームワークも提案していく必要がある．

6. ま と め

本稿では，ユースケースに明確な構造を定義し，定型的なユースケースについて，そのユースケースフレームワークを与えた．ユースケースフレームワークを用いることによって，アプリケーションユースケースの記述に関する生産性を向上させることができる．また，ユースケースの構造に基づき，メタモデルを提案した．このメタモデルには，複数のユースケース間の制約関係が表現されている点で，UML が与えたメタモデルよりも強力である．メタモデルにユースケース間の制約関係を定義したことによって，要求仕様書におけるユースケース間の一貫性と整合性を保つ構造を得ることができた．

謝　　辞

本研究を進めるにあたり,（株）情報技術コンソーシアムが情報処理振興事業協会より委託を受けて実施した研究に従事されていた山浦直人氏，安達隆氏，黒田健司氏から多くの意見を頂き，メタモデル開発に協力を頂きました．ここに感謝いたします．

参 考 文 献

1) Cockburn, A. : “Structuring Use Cases with Goals,” http://members. aol.com/acockburn /papers/usecases.htm.
2) Cockburn, A. : “Basic Use Case Template,” http://members. aol.com/acockburn /papers/uctempla.htm.
3) Coleman, D. : “A Use Case Template: draft for discussion,” *EDOC'99 (Tutorial material),* May,1999.
4) Collins-Cope, M. : “The RSI Approach to Use Case Analysis,” *C++ REPORT*, July-August 1990.
5) Jacobson, I., Christerson, M., Jonsson, P. and Overgaard, G. : *Object-Oriented Software Engineering: A Use Case Driven Approach*, Addison-Wesley, 1992.
6) Jacobson, I., Ericsson, M. and Jacobson, A. : *The Object Advantage*, Addison-Wesley, 1994.
7) Jacobson, I. , Booch, G. and Rumbaugh, J. : *The Unified Software Development Process,* Addison-Wesley, 1999.
8) 情報処理学会ソフトウェア工学研究会 要求工学ワーキンググループ共通問題 : http://www.selab.cs.ritsumei.ac.jp/ohnishi/RE /problem.html.
9) Larman, C. : *Applying UML and Patterns: an Introduction to Object-Oriented Analysis and Design,* Prentice Hall, 1998. （今野睦, 依田智夫監訳, 実践 UML, プレンティスホール, 1998.）
10) 中谷多哉子, 玉井哲雄 : “ユースケースフレームワークの検討”, ソフトウェアシンポジウム, 2000 年 6 月.
11) Schneider, G. and Winters, J. P. : *Applying Use Cases,* Addison-Wesley, 1998.
12) UML Semantics version 1.1, Rational Software Corporation, September 1997. http://www.rational.com/uml/index.html
13) 山浦直人, 安達隆, 黒田健司, 中谷多哉子 : “上流工程における再利用部品としてのユースケース”, 情報処理学会第 60 回全国大会論文集, 2000, pp.1-203–1-204.
14) 吉田裕之, 山本里枝子, 上原忠弘, 田中達雄 : UML によるオブジェクト指向開発実践ガイド , 技術評論社, 1999.