情 報 シ ス テ ム 58－7
（1996. 3. 19）

ユースケース法の経験をとおして
＝OO－DESIGNの開発＝

高橋　富夫　*1 富士通
宮崎　比呂志 *2 富士通
吉原　哲宏　*3 OSL

概要
オブジェクト指向に初めて取り組む事務処理のアプリケーションの技術者にとっては，適切なオブジェクトを抽出することは容易ではない．その点，オブジェクトを抽出するための明確な手順を提案しているユースケース法が有効である．我われはオブジェクト指向システムの分析・設計にユースケース法を適用してきた．ユースケース法は他の方法論に対して幾つかの優位な点を持つが，適用しづらい点もある．たとえば，
◇ ユースケースのみの情報では，必要なすべての実体オブジェクトが抽出できない．
◇ ユースケースの記述の深さや範囲について，明確なガイドラインを示していない．
◇ クラス，継承，関係，多重度などの表記法にスペースを多く必要とする．
我われは，ユースケース法をベースとして，他の方法論やオブジェクト指向の経験報告から多くのアイディアを借り，さらに，我われの経験を追加して，オブジェクト指向開発方法論（OO－DESIGN）を開発した．
本論文では，最初に，OMT，ユースケース法での適用経験，OO－DESIGNの概要，ユースケース法との相違とその主要理由を述べる．

Experiences with Use-Case Method and OO-DESIGN

Tomio Takahashi　　*1 Fujitsu
Hiroshi Miyazaki　　*2 Fujitsu
Tetsuhiro Yoshihara *3 Oita Software Laboratory

OO paradigm shift is tough. Appropriate objects must be selected at the early phase of OO analysis. However, this job is not easy for a navice OO analysist, even though he or she is well-experienced in traditional paradigm. Use-Case method has clear-cut concepts and procedures to support this process. We have four-year experience in this method. We recognize that it has various advantagesover the other OO analysis and design methodologies. But at the same time, we have some difficulties in applying this method to business application. For example,
◇ Information described in use cases are not sufficient enough to build OO model.
◇ It does not have any guideline on how to define boundary of system and precision of use case. And,
◇ It requires much space to describe class, inheritance, relation, and multiplicity.
So, we have developed our methodlogy, OO-DESIGN, based on our experience, by borrowing many ideas from Use-Case method and other methodologies.
This paper describes, at first, our experiences with Use-Case method and OMT, then, the basic ideas of OO-DESIGN, difference among them and the reasons.

1. はじめに
オブジェクト指向に初めて取り組む事務処理のアプリケーションの技術者にとっては，適切なオブジェクトを抽出することは容易ではない．その点，オブジェクトを抽出するための明確な手順を提案しているユースケース法が有効である．我われはオブジェクト指向システムの分析・設計にユースケース法を適用してきた．ユースケース法は他の方法論に対して幾つかの優位な点を持つが，適用しづらい点もある．たとえば，
◇ ユースケースのみの情報では，必要なすべての実体オブジェクトが抽出できない．
◇ ユースケースの記述の深さや範囲について，明確なガイドラインを示していない．
◇ クラス，継承，関係，多重度などの表記法にスペースを多く必要とする．
我われは，ユースケース法をベースとして，他の方法論やオブジェクト指向の経験報告から多くのアイディアを借り，さらに，我われの経験を追加して，オブジェクト指向開発方法論（OO－DESIGN）を開発した．
本論文では，最初に，OMT，ユースケース法での適用経験，OO－DESIGNの概要，ユースケース法との相違とその主要理由を述べる．

---

*1,*2 261 千葉市美浜区中瀬　1-9-3　富士通　株式会社
261 Chiba-shi Mihama-ku Nakase 1-9-3 Fujistsu Ltd.
*3 870 大分市東春日町　17-58　株式会社 富士通大分ソフトウェアラボラトリ
870 Ooita-shi Higashi-Kasuga-machi 17-58 Fujitsu Ooita Software laboratory

2. OMTとユースケース法の経験

1993年に Intelligent Pad [01]，1994年に ODB II [02]，Objectpower [03]が出荷された．それ以前にも Smalltalk や C++，Fortran KR[04]でオブジェクト指向システムの実現が可能であったが，AIや CAD など一部の部門に限られていた．我われ事務処理系の技術者がオブジェクト指向に着目したのは，1994年に入ってからであった．当時，すでに多くのオブジェクト指向分析・設計方法論が提案されていた．

(1) OMTでの経験

OMT[05]はすでにデファクト標準になりつつあった．いくつかの実プロジェクトで適用し，プロトタイピングや教育や自由研究の場で記述実験をした．OMTは概念が豊富で多くの技法と指針をもつことを確認した．しかし，構造化技術には経験豊かであるがオブジェクト指向には初めての技術者の多くが，最初のフェーズ，すなわち，オブジェクトの抽出とオブジェクト間の関係の記述で躓いた．

OMTでは，問題文の中にある名詞を暫定的なクラスの候補にする．そして，この候補の中から誤ったクラスを排除することにより適切なクラスを抽出する．しかし，この方法では，適切なクラスを抽出できるか否かは，適切な問題文を作成できるか否かに依存することになる．OMTは問題文の作成の方法を示してはいない．さらに，誤ったクラスを排除するとは，冗長なクラス，不適切なクラス，曖昧なクラス，属性，操作，役割，実装の構成要素を排除することである．しかし，この排除するための判断には，オブジェクト指向の経験とセンスを必要として，初心者には難しい．（図 01）

◇ 豊富な概念をもつ．
◇ 多くの技法と指針をもつ．
◇ デファクト標準になりつつある．
◆ 要求仕様（問題文）の作成の方法を示していない．
◆ 問題文から適切なオブジェクト（クラス）の抽出が難しい．

図 01 OMTでの経験

(2) ユースケース法での経験

ユースケース法[06]では，最初にシステムの境界を設定する．次に，そのシステムを利用するアクタ（利用者の分類）を列挙する．そして，アクタがシステムをどのように利用するか，システムがどのような機能を提供するかをユーザの視点から記述する．この記述をユースケースと呼ぶ．ユースケースは，要求仕様として，システムの開発者と発注者の契約として利用したり，顧客が望んでいることをシステムの開発者に理解させることにも利用できる．すべてのアクタが列挙されているか，個々のアクタのすべてのユースケースが記述されているかを見ることにより，要求仕様にモレがないかのチェックが簡単にできる．また，ユースケース法では，オブジェクトをそのライフサイクルに着目して，インタフェースオブジェクト，実体オブジェクト，制御オブジェクトの三種類に分類している．特に，制御オブジェクトの導入により，ライフサイクルの短いオブジェクト，および，複数のオブジェクトに跨がる操作をオブジェクトを制御オブジェクトにすることによって，変更に強い頑強なシステムが構築できる．なお，ユースケースは要求仕様としての意味だけではなく，分析，設計，実装，テストの全プロセスの作業と検証のベースとなる．

OMTでは，オブジェクト指向の概念を理解した（と思っている）初心者が，最も難しいとされるオブジェクトの抽出とオブジェクト間の関係の表記を最初に行う．一方，ユースケース法では，構造化技術の経験者なら比較的容易にユースケースを記述できる．この最初の作業が容易であるということは，オブジェクト指向に初めて取り組む初心者への心理的抵抗を小さくし，ユースケース法に親近感を抱かせる．

一方，ユースケース法の問題点としては以下のような問題がある．

(1)ユースケースの記述の範囲

ユースケースの記述の範囲や深さについては，しばしば，メンバ間で意見が別れる．これは，現状システムの問題点，新システムの目的，システムの制約などの認識の相違によって起こる．ユースケース法は，この点の明確なガイドラインを示していない．

(2)オブジェクト抽出の問題

ユースケースのみの情報では，必要なすべての実体オブジェクトが抽出できない．これは，ユースケースがシステムを外部からみた利用者の観点から記述されていることによる．

(3)制御オブジェクトの問題

オブジェクト指向の初学者には，オブジェクト間の関係の表現が難しい．制御オブジェクトの導入はユースケース法の特徴であるが，ユースケースが増えてくると，ライフサイクルの短い制御オブジェクトが多くなる．頑強なオブジェクトモデルを作成するためのユースケース法の特徴ではあるが，やはり，初学者には制御オブジェクトの発見が難かしい．また，クラス，継承，関係，多重度などの表記するとOMTに比べてスペースを多く必要とする．

◇ 要求仕様［ユースケース］の記述方法を示している．
◇ ユースケースに抜けがないかどうかのチェックが容易である．
◇ ユースケースからオブジェクトを見つける方法を示している．
◇ 制御オブジェクトにより，変更に強い頑強なモデルが作成できる．
◇ ユースケースは，要求定義，分析，設計，実装，テストのベースとなり，一貫性がある．
◆ ユースケース記述の段階で，システムの範囲や深さについて，メンバ間で意見が別れることがある．
◆ ユースケースからはすべての実体オブジェクトが抽出できない．
◆ 初心者には，制御オブジェクトの発見は難しい．ライフサイクルの短い制御オブジェクトが多くなる．
◆ 表記法に冗長な部分がある．

図 02 ユースケース法での経験

3. OO−DESIGN

バジリは，経験が成長する仕組みとして経験工場[07]を提案している．1994年10月に，前章での経験をもとに事務処理アプリケーションを対象としてオブジェクト指向開発方法論「OO-DESIGN」の枠組みを発表した[08]．

(1) OO-DESIGN の基本的考えかた

本項では，以下の観点よりOO-DESIGN の基本的な考え方を整理する．

【対象アプリケーション】

事務処理アプリケーションを対象とする．理由は，事務処理アプリケーションの需要が多いことと，我われが事務処理以外の領域には暗いことによる．

【OO-DESIGN の利用者】

構造化パラダイムからオブジェクト指向パラダイムへの円滑な移行に焦点を当てる．構造化パラダイムの分析，設計経験をもつが，オブジェクト指向には初めての技術者を対象とする．

【OO-DESIGN の成長の仕組み】

経験の蓄積，洗練をとおして，方法論が成長する仕組みに重点を置く．図 03 参照．

① すでに多くの方法論が提案され，適用報告がされている．車輪の再発明を防止し，NIH (Not Invented Here) 症候群に陥らないため，先駆者のアイディアや経験を素直に受け入れる．

② 手順，表記法，ノウハウを評価し，整理する．
その際，用語や表記法はできるだけ，デファクト標準を用いる．

③ プロジェクトへ技術移転をする．また，プロジェクトからの経験を収集し洗練する．

④ 事例や部品を蓄積する．
具体的なソフトウェア製品に依存した実際的な事例や部品を蓄積する．

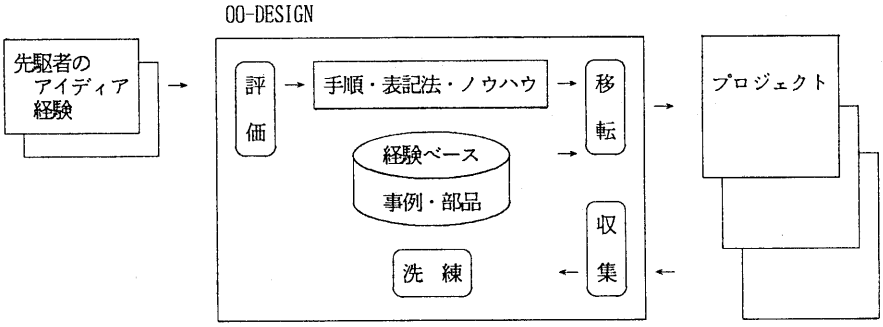

図 03　経験が成長する仕組み

(2) OO-DESIGN 開発フェーズ

OO-DESIGN の開発フェーズを図 04 に，その主要作業内容を図 05 に示す．

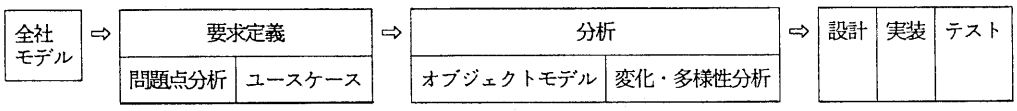

図 04　OO-DESIGN　開発フェーズ

フェーズ	主要作業内容
全社モデル	対象システムと関連する他のシステムを含む全体モデルを作成する．
問題点分析	対象システムの目的，主要な問題点，制約を明確にする．
ユースケース	システムの範囲とアクタを明確にし，ユースケースを作成する．
オブジェクトモデル	オブジェクトモデルを作成する．
変化・多様性分析	予想される変化と多様性を分析し，オブジェクトモデルを洗練する．
設計移行	設計，実装，テストを行う．

図 05　OO-DESIGN　開発フェーズの主要作業内容

(3) オブジェクトモデル作成の手順

本節では，ユースケースからどのようにしてオブジェクトモデルを作成するかを事例を用いて説明する．

【事例：SEコンベンション支援システム】

某社では，年に一回，関連会社，関係会社のSEを含め，SE活動の成果のまとめと技術交流も目的として，SEコンベンションを開催する．

開催に先立って，組織委員会が結成される．組織委員会は，方針，スケジュール，予算などを決定し，プログラム委員会を設立する．プログラム委員会は，募集する論文のカテゴリを決め，カテゴリ毎に論文審査委員（以降，審査委員と呼ぶ）を任命する．プログラム委員会は，SEに対して論文募集を行う．SEは決められた期日までに論文を審査委員に送付する．審査委員は論文を査読して，そのコメントを論文投稿者に送る．論文を投稿したSEは適当に修正して再び担当審査員に送る．審査委員は論文の評価をする．プログラム委員会は評価結果を考慮してコンベンションのプログラムを作成する．プログラムは複数のセション（同一カテゴリの論文を発表する単位）よりなり，セションごとにチェアパーソンが任命される．また，プログラム委員会は複数のチュートリアルセションを企画する．チュートリアルは，技術解説であり，解説を担当する適当な担当者が任命される．

組織委員会は，論文とチュートリアルのプログラムを，SEに広報する．参加を希望するSEは，参加申込み書を組織委員会に送付する．参加が認められると，組織委員会はコンベンション参加証とチュートリアル参加証を送付する．

参加者は，前日，または，当日，レセプションをする．すなわち，受付で，参加証と引換えに，論文集，または，チュートリアルの資料を受け取る．

【問題点分析】

SEコンベンション支援システムを開発するに際して，組織委員会経験者と情報システム開発室のメンバからなる「分析チーム」が設立される．分析チームは，問題点分析によって，長年，慣習的に開催されてきたSEコンベンションの問題点と新システムの目的を明確にする．

［現行システムの問題点］

(a) 大量の紙（論文，コメント，申込み書など）の複写，送付が行われる．
(b) しばしば，特定の審査員の査読，評価が滞りる．そのことがその時点ではわからない．
(c) 特定のチュートリアルに参加者が集中し，超満員の会場と閑散とした会場に別れる．

［新システムの目的］

(a) 紙の洪水を防ぐ．
(b) スケジュール，状態を監視し，円滑かつ効果的な進行を図る．
(c) 論文，チュートリアル，審査員コメントなどの技術情報，および，論文作成技術の情報を蓄積し，交流を図る．

【システムの範囲とアクタの決定】

分析チームは，このシステムの主要な利用者（アクタ）とそのアクタがシステムをどのように利用するか（ユースケース）を列挙する．

全体システムの概要から素直に，アクタとユースケースをリストする．

組織委員会	プログラム委員会	審査員	ＳＥ
・方針，スケジュール，予算の決定	・論文カテゴリの決定	・論文査読	・論文の執筆
・プログラム委員会の設立	・審査員の任命	・コメントの送付	・論文の投稿
・参加募集	・論文募集	・論文評価	・修正論文の再投稿
・参加証交付	・プログラムの作成	・評価結果の送付	・参加申込み
	・チュートリアルの企画		・レセプション

分析チームは，これらをすべて対象システムで行うべきかを検討する．例えば，組織委員会が方針，スケジュール，予算を決定するが，このシステムは直接その活動を支援しないことにし，この活動はこのシステムの対象範囲外とする．プログラム委員会の論文カテゴリ決定の活動に対しても同様とする．しかし，問題点分析で明確にされたスケジュールの管理は非常に重要なのでこのシステムの対象範囲内とし，組織委員会が行うことにする．現行システムの問題点やシステムの目的を考慮し．図 06 のアクタとユースケースを決定する．

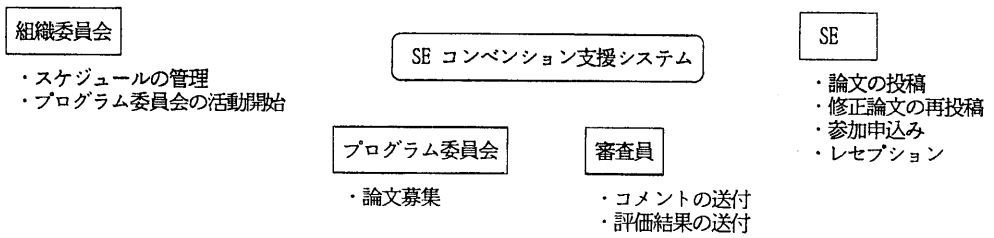

図 06 アクタとユースケース

【詳細ユースケース】
紙面の都合ですべてを記述できない．ここでは，スケジュールの管理，論文募集，および，参加申し込みの詳細ユースケースの記述例を示す．

ユースケース番号: U01
ユースケース名 ：スケジュールを管理する．
アクタ ：組織委員会
事象 ：スケジュールが決定された．
前提条件 ：操作員は組織委員会から入力の許可が与えられている．
内容 ：操作員は，以下のイベントの日程入力を行う
・プログラム委員会構成決定日 ・コメント送付締切日 ・参加募集日
・審査委員決定日 ・修正論文再投稿締切日 ・参加募集締切日
・論文募集日 ・審査締切日 ・コンベンション前日
・論文投稿締切日 ・プログラム作成日 ・コンベンション当日
システムは，進捗状況を監視し，適当な催促と督促を行う．
例外事項 ：システムは，イベント期日が過ぎて，アクションが取られていない場合は，毎朝，督促する．

ユースケース番号: U03
ユースケース名 ：論文を募集する．
アクタ ：プログラム委員会
事象 ：論文カテゴリが決定された．
前提条件 ：査読委員，SEの登録がされている．操作員は操作資格をもつ．
内容 ：操作員は，以下の入力をおこなう．
・論文カテゴリ
・審査員名
システムは，SEに対して論文を募集する．
審査員に対して，SEから投稿された論文を査読しコメントを送付するよう依頼する．
例外事項 ：--

ユースケース番号: U08
ユースケース名 ：参加を申し込む．
アクタ ：SE
事象 ：--
前提条件 ：参加募集がされている．SEは登録されている．
内容 ：SEは，以下の入力をおこなう．
・論文セション名
・チュートリアルセション名
システムは，申込みを登録する．
例外事項 ：--

【オブジェクトモデルの作成】
U01 のユースケースから，システムが長期にわたって記憶しておく必要のある［実体］オブジェクトを検討する．各種「イベント」を知っている「スケジュール」オブジェクトが必要である．また，進捗管理の対象として「組織委員会」や「プログラム委員会」「審査員」「論文」「プログラム」などがオブジェクトの候補として列挙される．また，これら複数のオブジェクトの状態を知っている「誰か」がいて，イベントを監視し，必要な督促をする．
ユースケース法では，一つ，または，複数のユースケースに対して，一つの制御オブジェクトを導入することを勧めている．この「誰か」を制御オブジェクトとして，ここでは「SEコンベンション」オブジェクトとする．これらのオブジェクトとオブジェクトの関係をＯＭＴ表記で記述する．図 07-1.
U03 のユースケースは「論文」「SE」「審査員」「コメント」の関係が得られる．図 07-2.
U08 からは事象として「登録」オブジェクトが必要であることがわかる．また，「プログラム」「論文セション」「チュートリアルセション」の関係が得られる．図 07-3.
最初から完全なユースケースは記述できない．しばしば，ユースケースの記述とオブジェクトのモデルの作成は平行して，あるいは，ラウンドトリップ的に行う．例えば，投稿された論文には審査員からコメントが付けられることになっている．SEコンベンションオブジェクトはスケジュールオブジェクトと協力して期日までにコメントが付いていない論文の審査員には，督促をする．しかし，督促にも係わらずコメントの付かない審査員，および，当該論文をどのようにあつかうかという例外事項は，ユースケースに書かれていないかもしれない．論文オブジェクトの状態遷移図を書く時点でこのことは判明する．その時点で，システムの範囲内であればユースケースが修正される．システムの範囲外の場合は，ビジネスルール集（後述）に記入する．

【相互作用図】
ユースケース毎に相互作用図を作成する．論文募集の相互作用図を図 08 に記す．
この図より，操作員と論文カテゴリのオブジェクトがオブジェクトモデルに漏れていることが分かる．

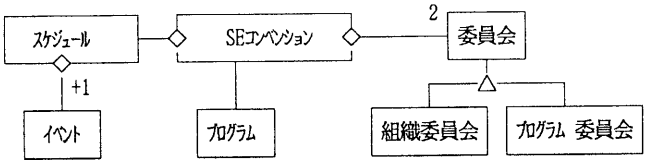

図 07-1
ユースケース U01から
作成された
部分的オブジェクトモデル

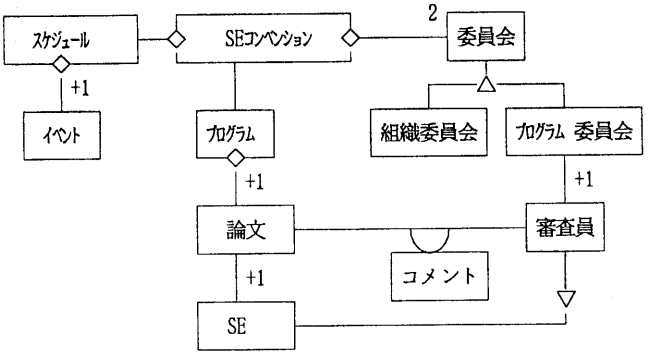

図 07-2
ユースケース U03から
追加された
部分オブジェクトモデル

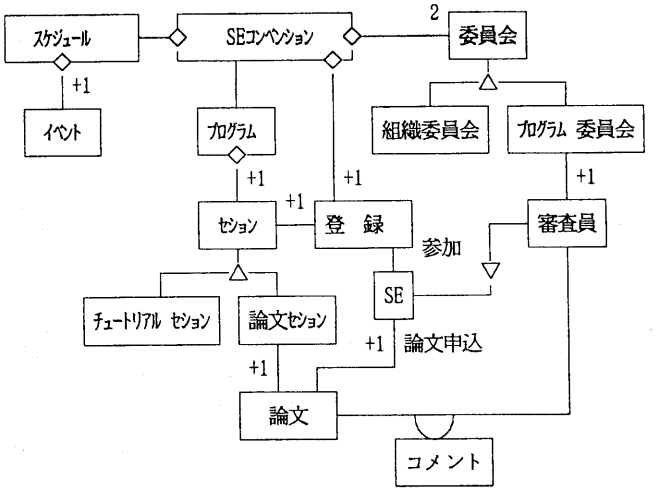

図 07-3
ユースケース U08から
追加された
部分オブジェクトモデル

4. 議論

本章では，幾つかの技術トピックについて議論する．

(1ユースケースの記述範囲

ユースケースを書いていてすぐ気づくことは，たとえ，メンバ間でシステムの境界についてコンセンサスを得ていてもいろいろなレベルのユースケースが書けることである．何のためにそのようなユースケースが必要であるのか，どのユースケースがシステムにとってもっとも重要であるのかで，議論が堂々巡りをすることがある．これは，伝統的な開発パラダイムの時代に我われが既に経験済みのことである．ユースケースは要求仕様であり，どのように要求仕様を決めるかはメンバ間の価値観やメンバの属する組織の利害などに左右される．OO-DESIGN では，ユースケースの前に問題点分析のフェーズをおいている．SEコンベンション支援システムのユースケースの記述の前に，現行システムの問題点や新システムの目的を明確にしている．このことによって，メンバ間で共通の価値観を共有でき，個々のユースケースを検討する時，なにが本質的成功要因 (CSF: Critical Success Factor) であるか，あるいは，そのようなユースケースが問題の解決やシステムの目的に貢献するかを検証することで，効率的に，また，効果的なユースケースを決定できる．この問題点分析

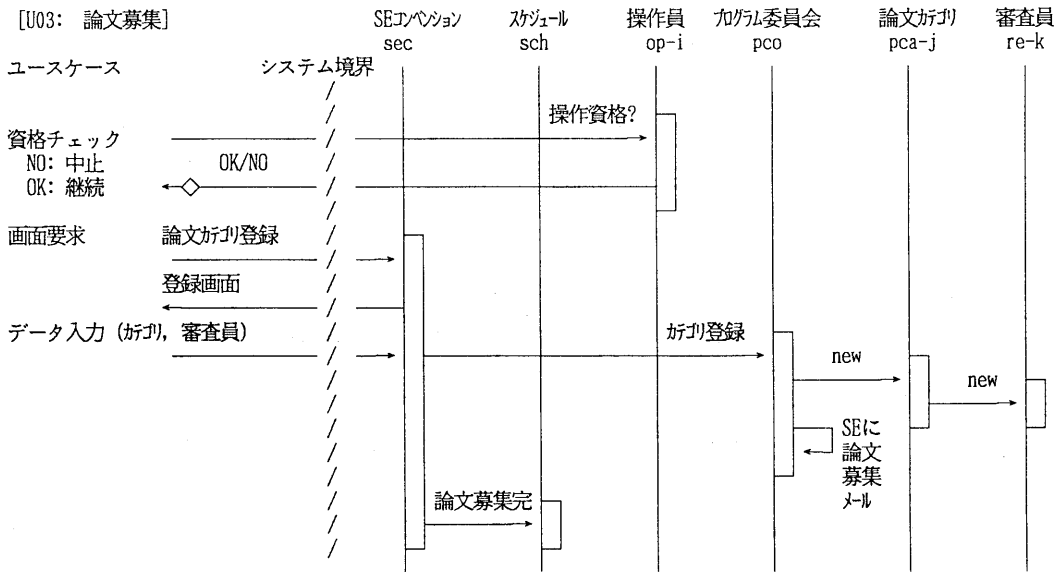

図 08 論文募集の相互作用図

には，EPG や C-NAP[09]などの既存の問題解決技法を用いる．さらに，システムが複雑な場合は，その前のフェーズで企業モデルなどを作成し，現在検討中のシステムの位置づけを明確にする．この成果は，アクタが適切に選定されているかの検証にも有効である．

(2) ドキュメンテーション

利用者がシステムをどのように使用するかの観点から記述されるユースケースだけでは要求仕様ドキュメントとしては不十分である．分析が進むみ，属性や操作の設計をするには，アプリケーションドメイン知識が必要となる．また，インタフェースの論理的な情報はユースケースに記述されるが，レイアウトや操作手順などの物理的な情報を記述するとユースケースが膨大となり，ユースケースを理解するのが大変となり，また，変更の影響が大きい．さらに，問題点分析で明確になった現行システムの問題点，システムの目的，主要な制約は，別のドキュメントとした方が理解しやすく，変更もし易い．

要求仕様書として，

・システムの目的，問題点，主要制約
・ユースケース
・アプリケーションドメインのビジネスルール
・インタフェースの物理仕様

に分割する．

(3) 変化と多様性分析

パルナス[10]は，設計とは，要求仕様の変化と設計決定を隠蔽し，設計変更の頻度と設計変更の容易性をバランスさせることである，と述べている．オブジェクト指向の効果に，拡張性/ 変更容易性が強調されるが，分析段階での要求仕様の変更を予測せずに，オブジェクト指向の概念と仕組みだけでは，この効果は享受できない．

三章で紹介したように，OO-DESIGN では，変化と多様性の分析フェーズを特に強調している．ここで，変化とは，時間の経過とともに変更が予想される要求仕様である．多様性とは，一時点で，顧客ごとに変わる要求仕様である．例えば，同じシティホテルの顧客管理システムでも，TホテルとNホテルでは仕様が異なる．

要求仕様の主要な変化と多様性を分析し，その頻度が高い仕様に対して，変更の影響を局所化できるようにオブジェクトの設計をする．

(4) 制御オブジェクト

ユースケース法では，変更に強い頑強なオブジェクトモデルを設計するために，複数のオブジェクトに関連するオブジェクトやライフサイクルの短いオブジェクトを制御オブジェクトとすることを勧めている．しかし，ユースケース法ではそのために一般的なガイドラインしか示していない．一つの方法論ですべての問題を解くことは難しい．個々のユースケースによりオブジェクトを抽出したあと，オブジェクト間を関係づける作業は初心者を悩ませる．この関係づけにコッド[11]のオブジェクトパターンやガンマ達[12]のデザインパターンは，この作業の大きなヒントを与える．まず，これらの定石を適用し，その後，定石を修正して，アプリケーションの文脈に最っとも適した関係を見つける．

(5) 表記法

ユースケース法は，要求分析，分析だけではなく，設計，実装，テストをカバーした一貫した方法論である．ユースケース法独自の表記法も用意されている．しかし，ユースケース法でオブジェクトを記述しようとすると，スペースを沢山必要とする．また，ＯＭＴが実質的にデファクト標準になろうとしている．多くの報告がＯＭＴ表記でされており，これからオブジェクト指向を学ぼうとする人には，ＯＭＴの知識が必須である．OO-DESIGN では，二つの表記法を学習する労力を省くために，表記法はＯＭＴとした．

しかし，このことは，少し混乱を招くことになる．ユースケース法では，ＯＭＴとは異なる用語を使っている．しかしその副作用よりも，ユースケース法のもつ効果の方が遙に大きい．

5. まとめと課題

本論文では，最初に，オブジェクトを抽出する際に初心者が遭遇するＯＭＴの問題を示した．そして，それがユースケース法を要求定義のフェースで用いることで解決できることを示した．しかし，ユースケース法の適用経験をとおして，ユースケース法にも幾つかの問題があることが判明した．OO-DESIGN は，これらの経験を蓄積していく仕組みである．さらに，事例を通して，いかにユースケースを記述するか，いかにユースケースからオブジェクトを見つけ，オブジェクトモデルを作成するかを示した．

我われは，体験を通して，ユースケース法を教科書どおり忠実に実行するだけでは，不十分であることを学んだ．そして，ユースケース法の前フェーズの必要性，要求仕様ドキュメントの構成，拡張性/ 柔軟性のあるオブジェクトモデルの作成方法，各種パターンの重要性，表記法などの問題について，OO-DESIGN がどのように考えているかを紹介した．

OO-DESIGN の枠組みを作成したのは 1994 年である．その後，いわゆる，第二世代ＯＭＴ[13]や統一方法論 (Inified Method) [14]が提案され，ユースケース法への認識が増している．

オブジェクト指向の技術は発展過程にある．経験や事例の蓄積・交流をとおして，事務処理アプリケーションのより実際的な開発方法論として，OO-DESIGN を洗練していきたい．

参考文献

01 オブジェクト指向ベースとしたビジュアル環境: IntelligentPad; 神田ほか; Fujitsu Vol 44 No 6; Nov. '93
02 オブジェクト指向データベース: ODB Ⅱ; 神田ほか; Fujitsu Vol 44 No 6; Nov. '93
03 Objectpower 使用手引書; 富士フェコム制御; June '94
04 いかにしてオブジェクトを見つけたか; 小川ほか; 情報処理 情報システム研究会; May '93
05 オブジェクト指向方法論ＯＭＴ; ランボー; 羽生田 訳; トッパン; July '92
06 オブジェクト指向ソフトウェア工学 OOSE; ヤコブソン; 西岡ほか 訳; アジソン・ウェスレイ・トッパン; Sep. '95
07 A Reuse-Oriented Software Evolution Architecture and Life Cycle Model; V. Basili; The Third International Workshop on Software Quality Improvement; Jan. '91
08 オブジェクト指向による新しいアプリケーション開発のスタイル<OO-DESIGN>; 富士通; Oct. '94
09 システム計画技法 EPG と C-NAP; 高橋ほか; Fujitsu Vol 41 No 3; May '90
10 Enhancing Reusability with Information Hiding; D. L. Parnas; '83
11 Object Models; Peter Cod; Prentice Hall; '95
12 Design Pattersn; E. Gamma ほか; Addison-Wesley; '95
13 OMT: The development process; J. Rumbaugh; JOOP; May '95
14 Unified Method: Notation Summary; OOPSLA'95