ソフトウェア工学 107－10
（1996．1．18）

オブジェクト指向方法論OMTにおける静的モデルと動的モデルの一貫性についての一考察

山井 和博† 吉髙 淳夫‡ 平川 正人‡ 市川 忠男‡

†広島大学大学院工学研究科

‡広島大学工学部

近年，ドメイン分析モデルとしてオブジェクト指向方法論 OMT が注目されている．OMT ではオブジェクトモデル，動的モデル，機能モデルを用いてモデリングを行うが，これらのモデル間の関係が明確でないため，記述されたモデルの一貫性が必ずしも保証されない．また，モデルの曖昧性も一貫性の検証を困難にしている．

本稿では OMT の 3 つのモデルのうち，オブジェクトモデルと動的モデルとの一貫性について考察を行う．モデルの曖昧性を，オブジェクト間のインタラクションをインスタンスレベルで多様に記述できるように動的モデルを拡張することで解消する．また，両モデルで満たされるべき制約に着目して一貫性の検証を行う手法を提案する．

A Study on the Consistency between Object and Dynamic Models of OMT

Kazuhiro Yamai† Atsuo Yoshitaka‡ Masahito Hirakawa‡ Tadao Ichikawa‡

†Graduate School of Hiroshima University

‡Faculty of Engineering, Hiroshima University

1-4-1 Kagamiyama Higashi-Hiroshima 739, Japan

Object Modeling Technique(OMT) has recently become attractive as a domain analysis model. In OMT, three models are used to describe a system: object, dynamic and functional. But the consistency between those models is not ensured, because the relationship between models is not clear and the dynamic model contains serious ambiguity. This paper discusses the consistency between the object model and the dynamic model. To eliminate the ambiguity of the dynamic model, we introduce an extended dynamic model which allows analysts to describe the various interactions between objects more flexibly. Furthermore, the consistency is guaranteed by the restriction caused on these models.

1 はじめに

近年, ドメイン分析モデルとしてオブジェクト指向分析方法論 [1, 2, 3] が注目されている. そのなかでも, 方法論として完成度が高い OMT [1] が広く用いられている.

OMT ではオブジェクト間の静的な構造的関係を表すオブジェクトモデル, 時間にともなう変化を表す動的モデル, オブジェクト間の機能を表現する機能モデルの 3 つを用いて対象ドメインを分析する.

しかし, OMT ではこれら 3 つのモデル間の関係が明確にはされておらず, モデルの一貫性を検証することが困難である. また, OMT では, 現実世界の実体をオブジェクト (インスタンス) という実体に写像して分析することを前提としているが, 実際は各モデルをインスタンス単位ではなくクラス単位で記述している. このため, モデル, 特に動的モデルに曖昧性が残っている. この点もモデル間の一貫性の検証を困難にしている原因となっている.

一貫性が保証されていないような分析モデルには矛盾が存在している可能性がある. 分析モデル中にこういった矛盾や誤りが存在すると, 以後の設計や実装の段階での後戻り行程に必要以上の時間や労力がかかり, これがシステムの生産性を低下させる原因となる. そこで分析モデルの結果に一貫性を保証することが望まれる.

OMT において, モデル間の一貫性の検証をしようとする研究として [4, 5] が, また, 実装言語に Eiffel を前提とし, オブジェクトモデルと動的モデルとの一貫性を保証した独自の方法論として BON [6] がある. これらの研究では, 独自にモデルを改良して一貫性を検証する手法 [5, 6] や, モデル間で満たすべき制約に注目して一貫性を検証するという手法 [4] をとっている. しかし, これらの研究ではモデルの記述能力に改善の余地がある.

本研究では, OMT で最も重要な役割を果たすオブジェクトモデルと, オブジェクト間の制御関係を表す動的モデルとの一貫性を検証することを目的とする. 動的モデルのイベントトレース図と状態遷移図を, オブジェクト間の多様なインタラクションが柔軟かつ形式的な記法で記述できるよう拡張することによって, モデルの曖昧性を解消する. さらに, 動的モデルで記述されたオブジェクト間のメッセージの送受信がオブジェクトモデルと矛盾しないことを検証することによって, 両モデル間の一貫性を保証する手法を提案する.

2 研究概要

この章では, 簡単な例題を用いて従来の OMT での分析手順, モデルを示し, 問題となるモデルの曖昧性について述べる.

2.1 例題

文章編集用の複数の Textwindow から構成される Window システムの作成を考える. Window は複数の Textwindow を所有し, ユーザは Window にメッセージを送ることで新しく Textwindow を開くことができる. Textwindow は, ユーザがテキストを書き込むことができる複数の Page を持っている. Page は現在表示されているページの前のページや次のページへ自由に表示を変えることができる. ユーザは新しく Page を開いて文章を編集したり, そのページを save ならびに close することができる. また, Window を close すれば全ての Textwindow は自分自身が所有する Page を save して終了する.

2.2 OMT に基づくモデル化

2.1 に与えた例題を OMT に基づいて記述する.

2.2.1 オブジェクトモデル

Window オブジェクトは複数の Textwindow オブジェクトを構成要素として所有する. よってこれらのクラス間の関係を, 多重度が 1 対多の集約を用いて関連付ける. 同様に, Textwindow オブジェクトは複数の Page オブジェクトと関係を持つので, 多重度を 1 対多, 関連名を own として関連付ける. Page は現在表示されているページの前後の Page へ表示を変えることができるので, Page クラス間に change という関連名で, ロール名を next, back として関連付ける (図 1).

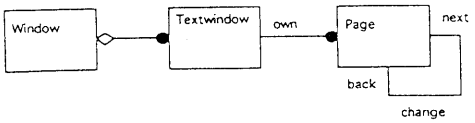

図 1: オブジェクトモデル

2.2.2 動的モデル

新たに Textwindow を開きたい場合, ユーザは Window オブジェクトに open_Text というメッセージを送る. このメッセージを受けた Window オブジェクトは Textwindow に create というメッセージを送り, Textwindow を生成する. さらに, 生成された Textwindow は 1 ページ目として Page オブジェクトを生成する (図 2).

Page は, Open と Close の二つの状態を持ち, さらに Open は display と disappear の二つの状態を持つ. Page は display の状態の時にのみ edit メッセージを受け取り, 文書を編集できる. さらに next や back というメッセージを受け取ると, next, back でたどられるオブジェクトに display メッセージを送り, 自分自身の状態は disappear に遷移させる (図 3).

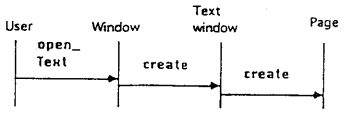

図 2: イベントトレース図

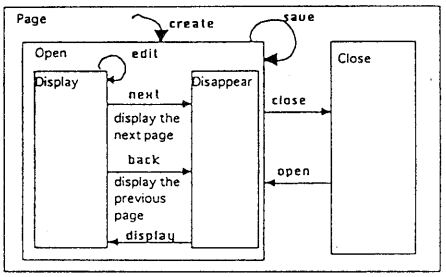

図 3: 状態遷移図

2.3 モデルの曖昧性

本節では, モデルの一貫性を保証するうえで問題となるモデルの曖昧性について説明する.

オブジェクトモデルでは, 対象ドメインから抽出されたオブジェクトを形式的なセマンティックスをもつ表記法で記述するため, モデルに曖昧性はない.

一方, 従来の動的モデルでは, シナリオごとに, オブジェクト間の時系列に沿ったイベント授受関係をイベントトレース図として記述する. しかし, 先ほど述べたとおり, 実際はクラス間の授受関係を記述しており, インスタンスレベルの関係が明示されない. このため, 同一クラスの異なるインスタンスにメッセージを送るといったことや, 一度に複数のオブジェクトに同じメッセージを送るといったことなどが明確に表現できない.

例えば, Widnow オブジェクトが close メッセージを受け取った場合を考える. まず Window オブジェクトは, 自分が現在開いている全ての Textwindow オブジェクトに close メッセージを送る. 次に, これらの Textwindow は, 現在自分が保持している Page オブジェクト全てに save メッセージを送って Page を close する.

この一連の流れを従来のイベントトレース図で記述した場合, 図 4のようになる. この図では, close や save というメッセージを全ての Textwindow と Page オブジェクトに送るという状況をうまく表現できていない. また逆に, 全ての Textwindow, Page オブジェクトを close したいのか, あるいは, どれか特定の Textwindow とそれに関連する Page だけを close したいのかが区別できない.

このように, 従来のイベントトレース図では, インスタンスレベルでのインタラクションをうまく表現できない. そのため, 次に記述する個々のオブジェクトの状態遷移図においても, メッセージ送信先を陽に記述できず, モデルに曖昧性が残る. インスタンスレベルでの記述能力の欠如が, モデル中に曖昧さを残す原因となっている. このようなモデルの曖昧さがオブジェクトモデルとの一貫性の検証を困難にしている.

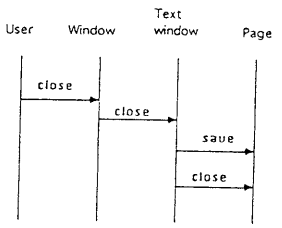

図 4: 曖昧なイベントトレース図

2.4 本研究のアプローチ

以上をふまえたうえで本研究のアプローチを述べる.

本研究では,動的モデルの曖昧性を解消するために

- オブジェクト間のインタラクションが正確かつ多様に記述できる
- モデルに明確なセマンティックスをもたせる

という点に着目してイベントトレース図,状態遷移図の両モデルにいくつかの記法を新たに導入し,モデルの拡張を行う. さらに,次章で述べるオブジェクトモデルと動的モデルとの間で満たされるべき制約に着目して,モデルの一貫性を検証する.

3 モデルの一貫性

モデルの一貫性について説明する前に,まずオブジェクトモデルと動的モデルとの関係について述べておく.

3.1 オブジェクトモデルと動的モデルの関係

オブジェクトモデルと動的モデルとの関係は次のように言える.

- 動的モデルで記述されるメッセージはオブジェクトモデルの操作に対応する.

イベントトレース図や状態遷移図で記述されるオブジェクト間のメッセージのやりとりは,他オブジェクトに対しての操作の依頼である. よって,メッセージを受け取ったオブジェクトには,そのメッセージを実行することができる操作が定義されていなければならない.

- オブジェクトモデルで記述されたオブジェクト間の関連は,動的モデルにおけるメッセージの通信経路である.

オブジェクトモデルで記述されたオブジェクト間の関連はメッセージ通信の経路である. つまり,オブジェクトモデルにおいて直接/間接の関連が存在しないオブジェクト間には,イベントの授受関係(メッセージ送受信)は存在しえない. そこで,動的モデルで記述されたメッセージ送受信は,オブジェクトモデルで記述されたオブジェクト間の関連に制約されると考えることができる.

3.2 モデルの満たすべき一貫性

前節で述べたオブジェクトモデルと動的モデルとの関係は,両モデルで満たされるべき制約と考えることができる. このモデル間の制約が,両モデル間で満たされるべき一貫性である. その一貫性として次の 3 項目を挙げる.

1. イベントトレース図でメッセージを受け取るオブジェクトは,そのメッセージに対応する操作を持っている.
2. イベントトレース図で記述されたオブジェクト間のメッセージ通信が可能である.
3. 同モデルのメッセージ通信とオブジェクトモデルでのオブジェクト間の多重度との間に矛盾がない.

1 は両モデルの操作名とメッセージ名,引数,返値の signature が一致しているかを,また 2, 3 はメッセージ通信がオブジェクトモデルで記述されたオブジェクト間の静的な関係と矛盾しないということを保証するものである. 本研究では 2, 3 を重点的に扱う.

3.3 メッセージ通信

3.3.1 オブジェクトの視界

あるオブジェクトから見てメッセージを送ることができる世界を,オブジェクトの視界と呼ぶ. オブジェクトは自分の視界に入っているオブジェクトとのみ直接的なインタラクションを行うことができる. つまり,オブジェクト間のメッセージ通信が可能であるかどうかは,メッセージ送信先オブジェクトが送信元オブジェクトの視界に入っているかどうかで決まるといえる.

オブジェクトの視界に入っているオブジェクトは,

- オブジェクトモデルにおいて,関連によって直接結合されているオブジェクト
- 動的モデルにおいて,他のオブジェクトからの操作の引数や返値として与えられたオブジェクト

の2つの場合である.

オブジェクトモデルにおいて,送信元オブジェクトと直接結合されていないオブジェクトは視界には入っておらず,直接認識されることはできない.これらのオブジェクトは,関連をたどること,または,操作の引数や返値として直接送信元オブジェクトに送られることによってのみ認識される.

3.3.2 オブジェクトの視界とオブジェクトモデルの関係

本節では,オブジェクトの視界とオブジェクト間の多重度の関係について述べておく.オブジェクトの視界にはオブジェクトが1つしか入っていないか,複数入っているか,または全く入っていないかのいずれかである.視界にオブジェクトが全く入っていない場合は,ここでは考慮しない.

2章に挙げたオブジェクトモデルでは,Windowオブジェクトと Textwindow オブジェクトの間を1対多の関係として記述している.この場合,1つのWindow オブジェクトは複数の Textwindow オブジェクトと関係を持つ.よって Window オブジェクトの視界には複数の Textwindow オブジェクトが入っている(図5 (a)).逆に1つの Textwindow オブジェクトはただ1つの Window オブジェクトとしか関係を持たないので, Textwindow オブジェクトの視界には1つの Window オブジェクトしか入っていないことになる(図5 (b)).

このように,視界中のオブジェクト数は,それらのオブジェクト間の多重度と密接な関係がある.このことはまた,外部からオブジェクトが直接与えられる場合,そのオブジェクトが単数であるか複数であるかということとも同様に関係する.

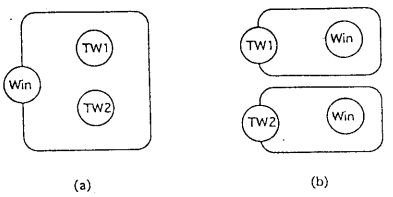

図5: オブジェクトモデルと視界の関係

3.3.3 個々のオブジェクトのメッセージ通信

前節に述べたように,視界に入っているオブジェクトの数は,1つの場合と複数の場合に分類される.送信側のオブジェクトは,視界中のオブジェクトと,ある特定の1つのオブジェクトにメッセージを送る場合(1対1通信)と,複数のオブジェクト全てにメッセージを送る場合(1対多通信)の2通りのインタラクションを行うと考えられる.

これらのメッセージが送信可能であるかどうかは,送信先を一意に特定可能であるかどうかと等価である.

以下に,オブジェクトがメッセージ送信先を特定でき,メッセージ送信が可能である場合を整理しておく.

1. オブジェクト間の視界に1つしかオブジェクトが入っていない.
2. 視界中の複数のオブジェクトと1対多通信を行う
3. 視界中の複数のオブジェクトのうちのある特定のオブジェクトと1対1通信を行う場合で,一意にオブジェクトを特定することのできるデータが操作の引数,返値として与えられている.

項目3でいう,オブジェクトを特定することのできるデータとは,クラスにオブジェクト識別子として定義された属性や,限定子で定義された,一意にオブジェクトを特定できる属性など指す.

4 イベントトレース図の拡張

従来のイベントトレース図ではオブジェクト(クラス)を縦線で,オブジェクト間のメッセージを横線で表現していた.イベントトレース図はオブジェクト間の多様なインタラクションを記述するものであるから,これらの記述を明確に行えるように以下の点を考慮してモデルを拡張する.

- メッセージ通信が1対1通信か1対多通信かを明確に記述できる.
- メッセージ送信先をインスタンスレベルで記述できる.

- メッセージ送信先をオブジェクトモデルの関連やロールを用いて記述することができる.
- 操作の引数や返値としてオブジェクトが渡される場合, それらの数が 1 であるか複数であるか区別することができる.

また, オブジェクトはメッセージを受け取った際に自分の状態によって動作を決定するので, このような状況を条件分岐として記述できるように拡張する.

次節以降で具体的な記法を説明する.

4.1 Object Line(縦線)

Object Line は, 送信側のオブジェクト数と受信側のオブジェクト数を明確に記述できるようにする.

1. **I(ndividual)-Line:** 細線で表す. 受信オブジェクト数が 1 であることを意味する. Name にはオブジェクト名かクラス名を記述する (図 6 (a)).
2. **M(ultiple)-Line:** 太線で表す. 受信オブジェクトが送信オブジェクトと関連を持つ同一クラスの全てのオブジェクトであることを意味する. Name には, オブジェクト名+(role 名または関連名) クラス名か, クラス名+(role 名または関連名) クラス名を記述する (図 6 (b)).

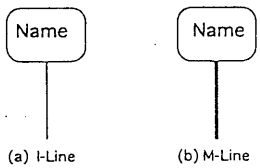

図 6: Object Line

4.2 Message Link(矢印)

Message Link にはメッセージ名とその引数や操作の返値などを記述する. メッセージ通信は 1 対 1 か 1 対多かで区別されるが, オブジェクト間の多重度によってインスタンスレベルでの通信状況が異なるため, 多重度を考慮した記法を導入する. また, メッセージにはオブジェクトに対して操作を要求するメッセージと操作の結果を返すだけのメッセージがあるので, これらも区別して扱う.

4.2.1 操作を要求するメッセージ

メッセージの送信側のオブジェクト数と受信側のオブジェクト数, さらに多重度を考慮した 4 種類の Message Link で記述する. 以下にその記法を示す. 引数が複数のオブジェクトである場合は, オブジェクト名の後に"*"をつける (図 7).

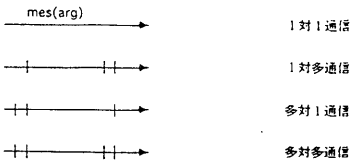

図 7: Message Link

4.2.2 返値を返すメッセージ

メッセージを受け取ったオブジェクトは, どのオブジェクトからそのメッセージが送られてきたのかを知る必要がないので, その返値をどのオブジェクトに対して返すかということは知らなくて良い. よって, 返値を返すだけのメッセージに関しては送受信オブジェクトの数は考慮する必要がない. そこで, このメッセージは以下のような記法で統一する. 返値として複数のオブジェクトが返されると予想される場合には, オブジェクト名の後に"*"をつける (図 8).

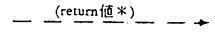

図 8: 返値メッセージ

4.3 記述例

本節では, オブジェクト間のインタラクションを定義した上記の表記法を用いて, 2 章で挙げた例題の記述法を示す (図 9,10).

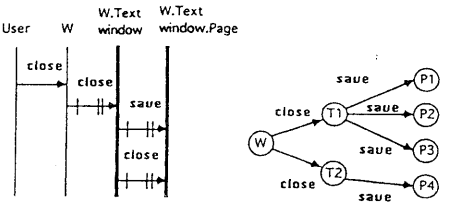

図 9: 実際のインタラクションとその記述例 (1)

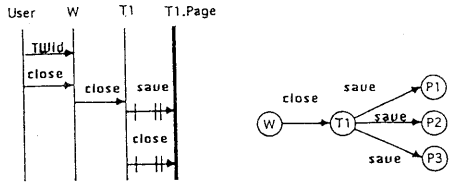

図 10: 実際のインタラクションとその記述例 (2)

以上のようにイベントトレース図を拡張することでインスタンスレベルでのインタラクションが記述でき,モデルにおける曖昧性が解消できる.

5 一貫性の検証

イベントトレース図の縦軸は時間軸,横軸はオブジェクト間の結合関係を表す空間軸とみなすことができる.

イベントトレース図全体の一貫性は,空間軸と時間軸の 2 つの軸について保証されなければならない.

3 章で述べたように,オブジェクトは自分の視界中のオブジェクトとのみインタラクションを行うことができる.このことより,空間軸についての一貫性は,メッセージ通信が行われるオブジェクト間に 3 章で述べた制約が満たされていれば保証される.また,オブジェクトは自分の属するクラスで定義された状態遷移図に従って振舞う.このことより,時間軸についての一貫性は,個々のオブジェクトがイベントトレース図で記述されたイベントの送受信の順序通り振舞えるかどうかを,状態遷移図と比較することによって検証される.イベントトレース図が状態遷移図上でトレース可能であれば時間軸についての一貫性は保証される.

空間軸についての検証によって,2 つのオブジェクトの間のメッセージ通信の可能性を保証でき,時間軸についての検証によって,個々のオブジェクトのイベントの送受信順序を保証できる.これら 2 つの軸についての検証を行うことで,モデル全体としての一貫性が保証される.

以下では,イベントトレース図とオブジェクトモデルとの一貫性を検証するためのものである,空間軸についての検証法を説明する.

5.1 手順

1. 受信メッセージをリストアップし,それらがオブジェクトモデルでメソッドとして定義されているかをチェックする.(4.1 節の 1 についての一貫性)

2. メッセージ送信先のオブジェクトが視界に入っているかどうかをチェックする.(4.1 節の 2 についての一貫性)

3. Message Link がオブジェクトモデルでのオブジェクト間の多重度と矛盾しないか,また,多重度が 1 対多や多対多の場合で,オブジェクト間に受信側のオブジェクト数が 1 であるような通信が行われているような場合には,"多"の中から"個"を特定できるデータがそろっているかどうかをチェックする.(4.1 節の 3 についての一貫性)

この手順によって,個々のオブジェクトがオブジェクトモデル上でメッセージ通信を行えるかどうかが検証される.

空間軸についての検証時に考慮しなければならない事項を次節以降にまとめる.

5.2 メッセージ送受信の時間的な関係

2, 3 において,外部から与えられたオブジェクトとインタラクションを行う場合,送信元オブジェクトは,それらのオブジェクトが与えられて初めてそれらを認識し,インタラクションすることができる.よって,外部からのオブジェクトは,それらとのインタラクションが開始される前に送信元オブジェクトに与えられていなければならない.

同様に,複数のオブジェクトのうち,特定のオブジェクトとのみインタラクションを行う場合も,オブジェクトを特定することができるデータがインタラクション開始以前に送信元に与えられていなければならない.

5.3 M-Line の扱い

M-Line は,あるオブジェクトと関係するあるクラスの複数のオブジェクトが並列的に振る舞っている様子を表している.よって,この M-Line が受けたメッセージを,同じクラスで I-Line として記述され

たオブジェクトが受けているという状況が生じる. 例えば, 図 11では, A1.B は A1 から m2 というメッセージを受けとっている. B1 は A1 と直接関係を持つので, B1 は A1 から m1 だけでなく, m2 というメッセージも受信することになる.

このような場合は, Object-Line につけられたオブジェクトの名前からオブジェクトの結合関係を求め, クラス名で記述された部分を具体的なオブジェクトに置き換えていくことで, M-Line を I-Line にマージする (図 11).

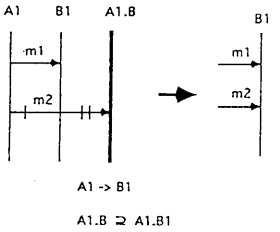

図 11: M-Line の扱い

5.4 検証結果のフィードバック

一貫性の検証を行うことによって, オブジェクトモデルと動的モデルで定義, 記述された操作が一致していることを保証できる. この他, オブジェクトモデルで定義されているが動的モデルでは使用されていないような冗長な操作や, 逆にオブジェクトモデルで未定義の操作などを検出できる. さらに, 一貫性検証時に, メッセージ経路 (オブジェクトへのアクセスパス) や通信状況を調べることによって, オブジェクト間の冗長な関連や, 新たに必要となる関連, オブジェクト間の多重度など, より適切なオブジェクトの静的な構造へモデルを精錬するための指標を得ることができる.

6 おわりに

OMT のオブジェクトモデルと動的モデルに関して, オブジェクト間の構造やイベントの授受に着目して両モデル間の一貫性を保証する手法について述べた. イベントトレース図と状態遷移図を拡張することによって動的モデルの記述能力を高め, モデルに曖昧性をなくす. また, オブジェクトモデルとの一貫性を検証することによって, 両モデルの一貫性を保証する.

今後の研究課題としては, 本手法を様々な例題に適用して, その有効性を実際に確認することが挙げられる.

参考文献

[1] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, Object-oriented modeling and design, Prentice Hall, 1991 (羽生田栄一監訳:オブジェクト指向方法論 OMT, トッパン, 1992).

[2] P. Coad and E. Yourdon, Object-Oriented Analysis 2nd ed, Yourdon Press, 1991 (羽生田栄一監訳:オブジェクト指向分析 (OOA)[第 2 版], トッパン, 1993).

[3] S. Shlaer and S. J. Mellor, Object lifecycles, Prentice Hall, 1992 (本位田真一, 伊藤潔訳:続オブジェクト指向システム分析, 啓学出版, 1992).

[4] 大西 淳, 岩倉 次郎, 野呂 一仁, 宇野 勝, "オブジェクト指向分析におけるモデルの検証支援", オブジェクト指向 '95 シンポジウム論文集, 情報処理学会, 1995, pp. 189-196.

[5] Fiona Hayes and Derek Coleman, "Coherent Model for Object-Oriented Analysis", Proc., OOPSLA'91, 1991, pp. 171-183.

[6] J.M.Nerson, "Applying Object-Oriented Analysis and Desgin", *Communications of the ACM*, Vol. 35, No.9, Sep., 1992, pp. 63-74.

[7] Richard Helm, Ian M. Holland, and Dipayan Gangopadhyay, "Contracts, Specifying Behavioral Compositions in Object-Oriented Systems", Proc., OOPSLA'90, 1990, pp. 169-180.