ソフトウェア工学 101－3
（1994．11．18）

階層表現を利用するオブジェクト指向設計

小林大輔 (koba@ms.kagu.sut.ac.jp)
大森晃 (ohmori@ms.kagu.sut.ac.jp)

東京理科大学工学部

オブジェクト指向設計で用いる表記法として，インスタンス表記とクラス表記を区別しインスタンス間の集約を階層的に表記する表記法を提案する．従来の開発手法で用いられてきた表記法では，主にクラス図を利用してオブジェクト図が記述されてきたために，クラスを中心とした視点でシステムがとらえられ，システム全体の構造は不明瞭になりがちであった．そこで，今回提案する表記法はシステムの構造を記述するためにインスタンス図に着目した．クラス図とインスタンス図を明確に区別し，それぞれの記載内容を区別した．本稿ではまず提案する表記法について説明し，OMT 法および FUSION 法における表記法との比較・検討を行なう．

A Study on Object-Oriented Design with Hierarchical Object Diagram

Daisuke Kobayashi, Akira Ohmori

Faculty of Engineering, Science University of Tokyo

This paper describes a new diagram for Object-Oriented Design, and compares it with the diagrams of OMT and FUSION. The new diagram is a kind of object-diagram that uses a class-diagram and an instance-diagram individually, in which the aggregation of instances is hierarchically expressed. Since the diagrams used in conventional OO-methods describe a system as class structure, it is not easy to see the whole system structure. So, we focus on an instance-diagram to make the whole system more visible. In our notation, a class-diagram and an instance-diagram are clearly separated, and concepts described in each diagram are different.

1. はじめに

従来，オブジェクト指向 (Object-Oriented) の技術を利用する開発手法として，様々な分析・設計の方法論が提案されてきた．代表的なものに OMT 法，Coad-Yourdon 法，Booch 法，FUSION 法といったものがある．

これらの開発手法の特徴として，図表を用いた表記をあげることができる．一般的には，システムの静的な側面の記述としてオブジェクト図が，動的な側面の記述として状態遷移図や事象トレース図（あるいはオブジェクト交信図）が描かれる．しかし，これらの表記ではシステムの全体像を把握するためには問題がある．また，そのまま実装に結びつくものとはいいがたく，検討の余地を残していると言える．

そこで，そのような問題点を解決するようなオブジェクト・インスタンスの階層的な表記を中心とした，より実際的な設計手法を提案する．そして，従来の表記法との比較を行なう．そして適用例として，図書館問題への適用を行なう．

2. 従来の開発手法の問題点

従来提案されてきた開発手法には，次のような問題点がある．

(1) システム全体の構成が見えにくい

従来の開発手法ではオブジェクト図の中でも特にクラス図が用いられ，クラスの内部構造やクラス間の関係を中心に表記している．そして，インスタンスの表記は，インスタンス図としてクラス図の補助に利用されるか，あるいはクラスでは表記に不都合が生じるような場合に便利のためにクラス図中にインスタンスを表記するといった，補助的な利用に留まっている．

しかし，クラスはオブジェクトの実体を表わしていないことからシステムの全体像をとらえるためにはインスタンスを中心とした表記が必要であると考える．

また，表記モデルの数や表記モデル間における概念の一貫性にも問題がある．この点は手法によって異なるが，表記モデルの数が多いと，モデル間の相互参照が非常に困難になる．また，表記モデル間で概念が一貫していない場合も，同様に相互参照が困難になる．

(2) 実装できない概念が用いられている

従来の開発手法でよく利用される「関連」という概念は，分析段階で有効な概念として分析中心の手法で特によく用いられてきた[2]．しかし，関連は OOP 言語において実装する際に対応する機構がない概念である．そのため，関連は何らかの形で実装可能な概念に変換することが必要となる．しかし，そのような変換が必要であることが明示されている開発手法は少なく，変換が行なわれる工程についても言及されていない．実装可能な概念による補完はある程度試みられているが，関連との対応関係が不明瞭で，実装に十分とは言えない．

(3) 複数の表記モデルを利用することによる混乱

従来の開発手法の中には，複数の表記モデルを併用するものがある．この表記モデルはそれぞれシステムの異なった側面を表記するものとされるが，表記モデル間の対応関係が不明瞭なままのものが多く，相互参照をする際に混乱を生じやすい．

3. 提案する表記法

上記三点の問題点を解決するために次のような点に留意した表記法を提案する．

- インスタンス図の積極的な利用
- なるべく少ない表記モデル数と，概念の一貫性
- 実装できる概念を用いた表記

提案する表記法は，オブジェクト図をベースに，クラス図とインスタンス図を用いる．そして，それぞれの図で表記する内容を分離する点に特徴がある．それぞれの図で表記する内容は次の通り．

インスタンス図

- インスタンス（属するクラス）
- メッセージパッシング
- 集約

クラス図

- クラス（属性，サービス）
- 継承

なお，今回提案する表記法は対象 OOP 言語として Smalltalk を考えている．

3.1 インスタンス図の表記法

インスタンス図の表記法を図1 に示す．

インスタンスは角の取れた四角で表記し，四角の中にクラス名とインスタンス名を表記する．同一クラスに属するインスタンスの集合はインスタンスの四角を重ねて表記し，右肩の部分に数を表記する．このときの数の書式は図1. に示したとおりである．

集約はインスタンスの四角の中にインスタンスを階層的に表記する．

メッセージパッシングは矢線でインスタンスの四角を結ぶ．矢線の横にメッセージ名と戻り値を表記する．

このような階層的な表記の問題点として，集約の階層が深くなった場合に表記が困難になるというこ

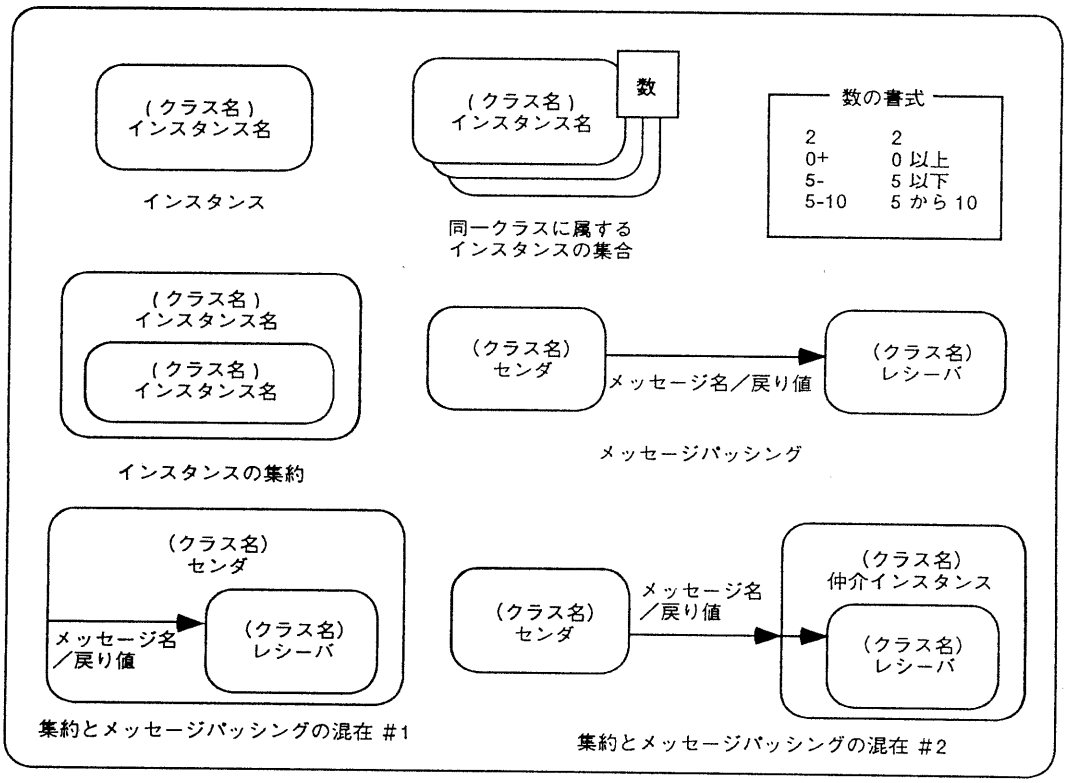

図1. インスタンス図の表記法

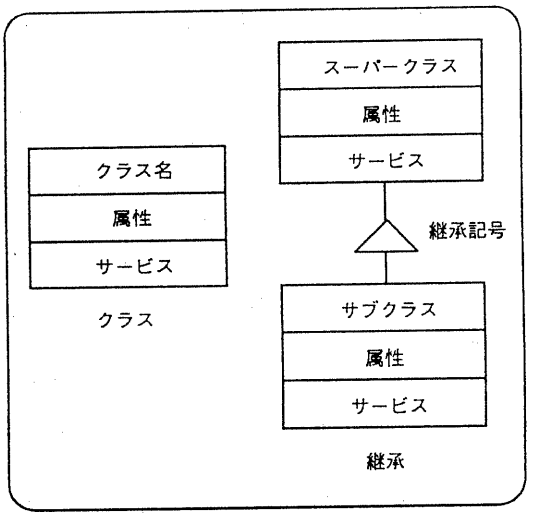

図2. クラス図の表記法

とが挙げられる．そこで，それを回避するために，集約の詳細は別紙に記載しても良いものとする．

3.2 クラス図の表記法

クラス図の表記法を図2 に示す．このクラス図の表記法は，OMT 法のクラス図の表記法から必要な部分を引用したものである．

クラスは四角で表記する．そして四角の中を三分し，上からクラス名，属性，サービスを表記する．継承は二つのクラスを無向線で結び，その線上に継承記号を表記する．継承記号は三角形で，クラスを結ぶ線上に三角形の頂点がくるように配置する．その継承記号の頂点が向いているクラスがスーパークラスであり，辺が向いているクラスがサブクラスである．

4. 従来の開発手法との比較

この表記法を従来の開発手法で用いられてきた表記法と比較すると，次のような点に特徴があるといえる．

- 集約の階層的な表現
- クラス図とインスタンス図の明確な分離と表記内容の区別

比較の対象として取り上げる開発手法は，OMT 法，FUSION 法の2手法である．OMT 法は提案する表記法と最も距離のある開発手法として，FUSION 法は最も類似点の多い開発手法として取り上げた．

例として用いたのは図書館問題[4]であるが，ここでは紙面の都合上，比較に用いた部分だけを図で示す．

提案する表記法を用いた例を図3. に示す．（図書館問題への適用例の詳細については文献 [5] を参照．）

図3 中の「図書館職員インターフェイス」はユーザインターフェイスを示す．ユーザインターフェイスをどのように扱うかは実装系によって異なるが，ここでは表記の便宜上インスタンスと同じように表記した．

4.1 OMT 法との比較

OMT 法はオブジェクトモデル，動的モデル，機能モデルという3つの表記モデルを併用するところに特徴がある．これらの表記モデルは分析で得られるものであり，設計では新たな表記モデルは導入されない．

オブジェクトモデルはクラス図とインスタンス図からなっており，クラス図が主に利用される．インスタンス図はクラス図の補助として利用することとされている．このクラス図の特徴としては，

- 集約，継承をそれぞれ集約記号，継承記号を用いて表記する，
- 集約，継承以外のクラス間の関係は関連として記述する，

といった点が挙げられる．

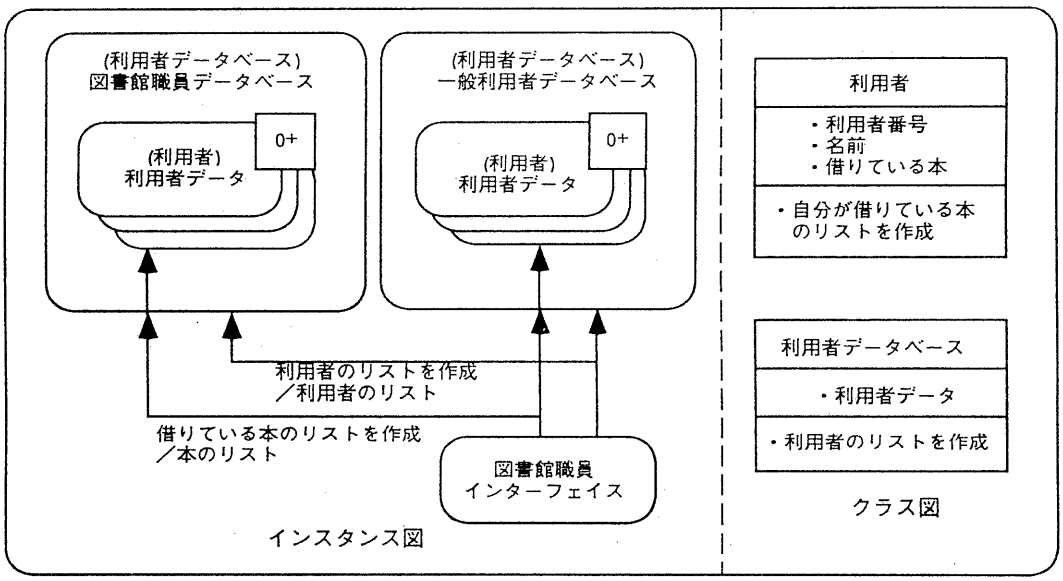

図3. 提案する表記法を用いた例

このオブジェクトモデルをサポートする意味で併用されるのが動的モデルである．動的モデルでは，まずメッセージの流れを時系列で表わす事象トレース図が作られ，次にオブジェクト間のメッセージパッシングを表わす事象フロー図そして各クラスの状態遷移図が描かれる．

機能モデルもオブジェクトモデルをサポートする形で利用される．しかし，機能モデルは DFD を用いており，他の二つのモデルとの対応関係が不明瞭なので今回は比較の対象としては取り上げない．

また，今回提案する表記法ではオブジェクトの状態遷移は取り扱っていないので，ここでは対象をクラス図と事象フロー図および事象トレース図に限定して比較する．

OMT 法を用いた例を図4 に示す．

ここで図3 と図4 を比較するとまず，OMT 法の問題点として，次の各点を挙げることができる．

- 全ての表記がクラス中心に行なわれるので，システムの構造がとらえにくい．いいかえれば，クラス図からインスタンスの構造が想像できない．図4 では利用者データベースがシステム内部で一般利用者データベースと図書館職員データベースという二つのインスタンスとして存在していることや，その集約で内部にそれぞれが独自に利用者クラスのインスタンスを持つことなども理解するのが難しい．
- 表記モデル間で概念の対応が不明確である．図4 では，利用者データベースと利用者の間にメッセージパッシングがあるが，クラス図には集約が表記されている．つまり，クラス図上における集約や関連と動的モデルにおけるメッセージパッシングの対応関係が不明確である．クラス図で表記される関連はそのまま実装できないので，実装に際して動的モデルとの相互参照が必要となるが，モデル間の対応関係が明らかでなく，実装が非常に困難である．
- 集約がクラス間の関係で表記されるので，メッセージパッシングを把握することが困難である．

図4 では図書館職員インターフェイスクラスと利用者クラスの間のメッセージパッシングは利用者クラスを介する必要がある．しかし，利用者クラスのインスタンスが全て利用者データベースクラスのインスタンスに集約されるかどうかはこのクラス図からはわからない．クラス図が巨大化し関連や集約，継承の線分が増えた際には，更にメッセージパッシングの把握が難しくなることが予想される．

一方，提案する表記法の問題点としては次の各点が挙げられる．

- メッセージパッシングの動的な系列が不明確である．これは今回提案した表記法ではメッセージパッシングの時系列の組を表記しないので，メッセージ間の連携が不明確になっている．
- 拡張性に乏しい．集約を階層的に表記し，クラス図とインスタンス図を分離しているので，ある関係を他の関係で置き換えるような修正や，クラスおよびインスタンスの追加，削除を行な

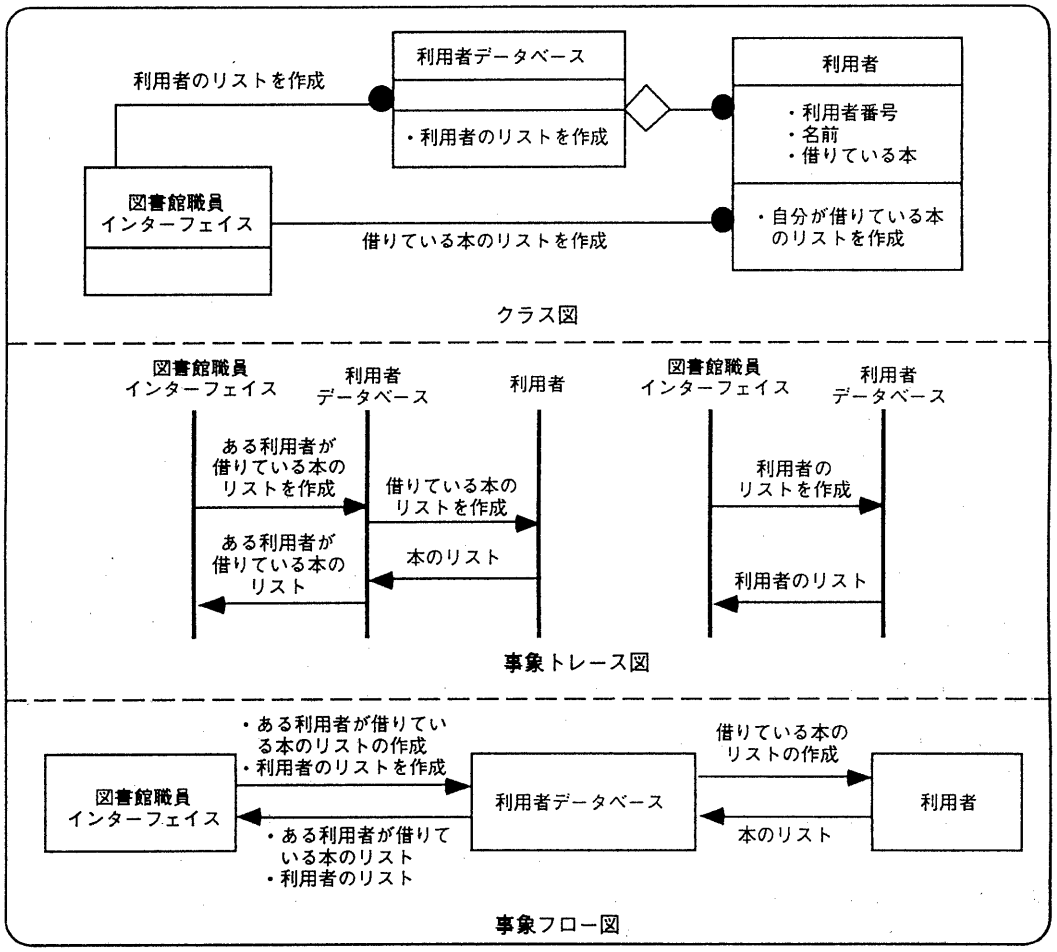

図4. OMT 法を用いた例

う際に図の修正が困難である．

- インスタンスが固定して表記されているので，インスタンスの生成，破棄が頻繁に起こるようなシステムの記述には不適である．

4.2 FUSION 法との比較

FUSION 法は，提案する表記法に最も近い表記モデルを提案している手法である．類似点としては，(1) 設計における実装可能な概念の重視，(2) インスタンス中心の表記モデルを採用，という二点が挙げられる．

また，FUSION 法は分析と設計で表記モデルを分けている点でも特色があり，分析と設計で表記法を区別しており，モデルの変換が明示的な数少ない手法でもある．

表記法は分析で描かれる System Object Model を中心に，設計では３つの図が描かれる． System Object Model では，クラスを中心に関連，集約，継承およびクラスの属性について表記する．Object Interaction Graph ではメッセージパッシングを，

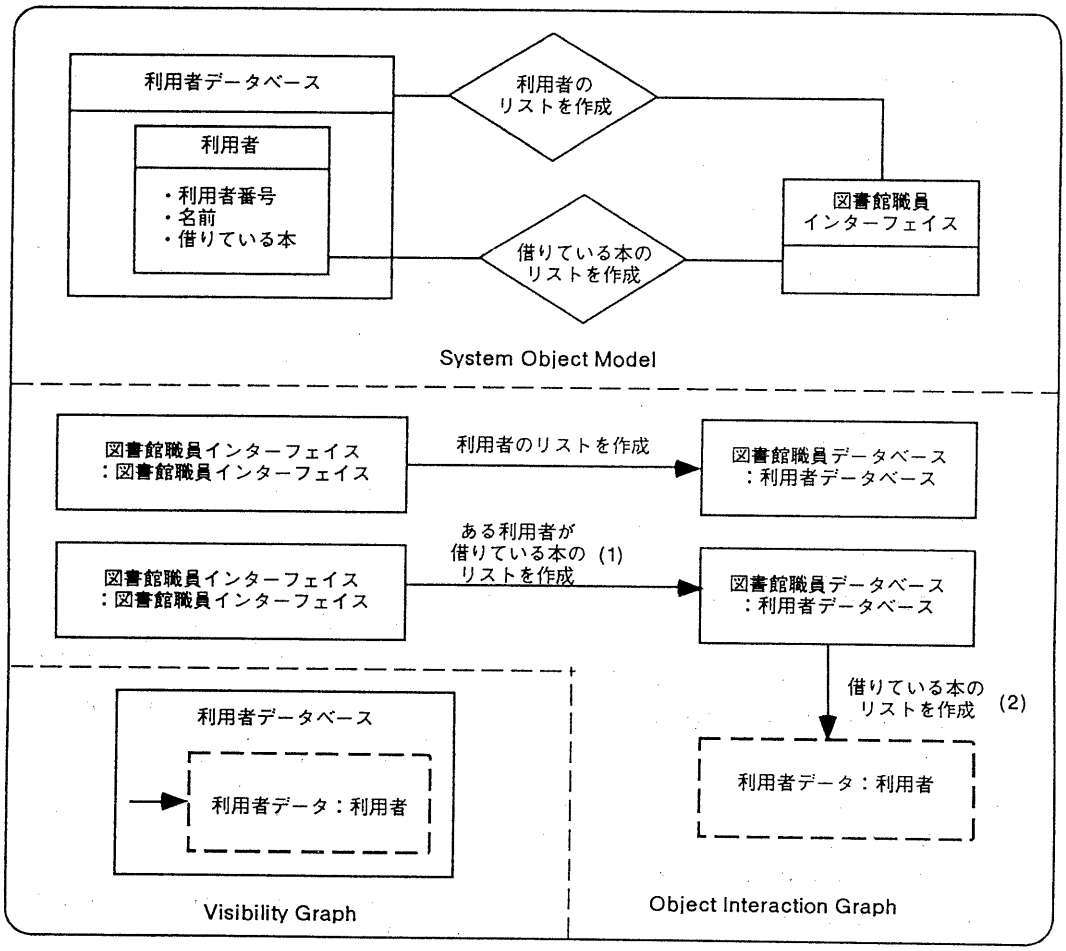

図5. FUSION 法を用いた例

Visibility Graph では情報隠蔽の構造を，Inheritance Graph ではクラス間の継承をそれぞれ表記する．

FUSION 法を用いた例を図5 に示す．今回取り上げた例では継承はなかったので，Inheritance Graph は省く．また，Object Interaction Graph は一般利用者データベースと図書館職員データベースに対してそれぞれ同じものを描く必要があるが，ここでは図書館職員データベースに限定して表記する．

図3 と図5 を比較すると，FUSION 法の問題点として次のことが挙げられる．これらの問題点はOMT 法における問題点と基本的には一致していると言える．

- System Object Model はクラスを中心に記述されており，OMT 法と同様にシステムの構造がとらえにくい．Object Interaction Graph, Visibility Graph ではインスタンスが記述されるが，システムの一部分しか表記していない．
- 集約がクラス間の関係として表記されるので，集約の構造に当てはまらないインスタンスの記述ができない．例えば，利用者クラスのインスタンスで利用者データベースと集約の関係にないものは表記されない．
- System Object Model で関連を利用しているが，メッセージパッシングなど実装可能な概念との対応関係が不明瞭である．

提案する表記法の問題点としては，次の点が挙げられる．

- メッセージパッシングの動的な系列が不明確である．前述の通り，メッセージ間の連携が記述されていない．

OMT 法との比較で明らかになった拡張性の乏しさについては，階層的な表記と，クラス表記とインスタンス表記の併用が原因であることから，FUSION 法にも同様の問題があると言える．

4.3 比較のまとめ

以上の比較の結果から，提案する表記法の利点は次のように指摘できる．

- インスタンス図を積極的に利用したことにより，システム全体の構成が見通しやすい．
- インスタンス図における集約の階層的な表記によって情報隠蔽の構造が見通しやすくなった．
- 実装可能な概念だけを用いた表記で，従来の開発手法と比較して容易に実装できる．

問題点としては，メッセージ間の動的な系列が不明確である，表記の変更に対して拡張性に乏しい，といった点が挙げられる．これら表記上の問題点については，CASE ツール化によって解決できるものと考える．

5. おわりに

オブジェクト指向設計の表記法として，集約の階層的な表現と，インスタンス図とクラス図の分離を特徴とする表記法を提案した．また，従来の開発手法として OMT 法，FUSION 法による表記との比較を行なった．

今後の課題としては，(1) 仕様変更などによって生じる表記上の変更点の明示，(2) メッセージの動的な系列の記述，が挙げられる．

参考・引用文献

[1] 小林大輔，大森晃：「多階層オブジェクトを利用するソフトウェア設計手法の提案」，計測自動制御学会第20回システムシンポジウム．

[2] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen, 羽生田栄一（監訳）：「オブジェクト指向方法論 OMT」，トッパン (1992).

[3] D. Coleman, P. Arnold, S. Bodoff, C. Dollin, H. Gilchrist, F. Hayes, P. Jeremes. "Object-Oriented Development The FUSION Method", Prentice Hall (1994).

[4] Probrem Set for the Fourth International Workshop on Software Specification and Design.