ソフトウェア工学 73－5
（１９９０．７．１９）

# 多視点に基づくオブジェクト指向表現システム

片山 佳則

富士通(株) 国際情報社会科学研究所

多数の開発者が異なった視点で開発を行なえる表現システムを提案する．これは，オブジェクト指向概念を基本にし，多視点という観点で知識表現／データベース／プログラミング言語それぞれの独特な考え方や手法を導入したものである．この多視点に基づく表現システムは，情報表現に２つのレベルを持つ．オブジェクトの共有化を進めるための格納レベルの情報と，開発者の視点を反映した開発者のViewsレベルの情報である．これらのレベル分けによって，多視点の開発を効果的に実現する．このレベル分けは，オブジェクトの関係表現を厳密にし，その処理方法に従った意味や働きを規定することで得られる理解レベルの分割でもある．

# An Object-Oriented Representation System Based on Multi-View Concept

Yoshinori KATAYAMA

International Institute for Advanced
Study of Social Information Science
FUJITSU LIMITED
140 Miyamoto, Numazu-shi, Shizuoka 410-03, Japan

In this paper, we propose a system which incorporates the Multi-View approach to an object-oriented representation system. This system has two levels of representing information. The first is an internal level of information, which has reusability and commonality of objects. The second is a view level of information, which depends on the developer's concept and his problem domain. This system effectively accomplishes the multi-view approach by level division, a mechanism for understanding objects. Levels are divided by restrictions including semantics and the operating processes of the object.

## 1．はじめに

ここで提案する表現システムは，多数の開発者が異なった視点で開発を進める際に，効果的機構を提供するものである．この多視点の機能を実現するために，提案するシステムの基本概念にはオブジェクト指向概念を用いている．これは，この概念が重要な役割を果たす多数の機能を備えている[1]からである．

この表現システムは，知識表現／データベース／プログラミング言語それぞれの独特な考え方や手法を，多視点という観点に基づき改善し，組み入れた形態を取っている．ここでは，プログラミングの際の視点に限って，表現システムの挙動や扱い方を述べる．

この表現システムは，内部では共同利用するためのオブジェクト環境を提供しながら，個々の開発者が個人的な視点で開発を進めることができるものである．

プログラミングにおいては，開発対象が複雑化，大規模化することにともなって，流動的に利用可能になる要素や，共有可能な要素などの実現の必要性が強調され，設計段階だけでなく，実現および分析(解析)の段階でも，広範な利用が実現できるための方法論の開発が進められている．これらの共有機能を実現するためには，個々の開発者の視点を重視し，それらの違いを的確に処理できる開発環境の考え方が重要になる．

本稿での表現システムも，Productivityを重視することから，オブジェクト指向概念におけるEncap-sulationの機能をベースに得られるReusabilityの実質的な能力向上を狙ったもの[2]でもある．これを実現するために，開発者のViewsの考え方とそのための特種な表現方法の導入を提案している．

このため，一般にオブジェクト指向システムとして代表にされるSmalltalk-80などのオブジェクトの取り扱い(表現方法,関係表現)とは異なり，より制約の強い記述方法を採用している部分もある．

ここでは，はじめにオブジェクト指向プログラミングの特徴及びその環境を簡単にまとめる．次に，開発者の視点の考え方及びそれを実現する表現システムの表現形式，Views操作について述べる．その後に，多視点の考えに基づき，広範な対象領域での開発において，実質的なReusabilityを向上する表現システムの効果をまとめる．

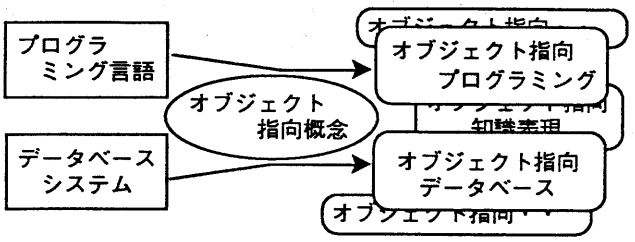



図1　オブジェクト指向概念の導入

## 2．オブジェクト指向プログラミング

最近では，オブジェクト指向概念の有意性を活用することを目的に，融合／導入を図る分野が目立っている(図1)．これは，オブジェクト指向概念が単なるプログラミングパラダイムという範疇だけでなく，計算機における問題解決に関して，その情報処理の基礎から応用まで，広い範囲に適用できる考え方であることを示している．

### 2.1　オブジェクト概念とプログラミング環境

オブジェクト指向という考え方は，開発者がその対象問題領域において，描いている概念体系をモデルとして表現する方法を与えるもの(図2)である．

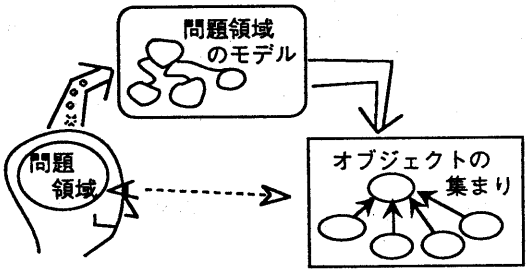



図2　オブジェクト指向の考え方

したがって，オブジェクトは，対象問題領域での実行に必要な内部状態または，その集まりと操作群(手続き群)をまとめたものである．このオブジェクトの集まりによって構成対象であるモデルや問題領域を表そうとする考え方がオブジェクト指向である．オブジェクトがこのように実現されることから，個々のオブジェクトが持っている操作を起動するためのメカニズムとしてメッセージが考えられ，さらにオブジェクトが持っている内部状態や操作において，オブジェクト間

に共通なものを扱うための構造的関係表現が生まれ，その階層構造に基づく継承機能が実現される．

これらオブジェクト指向概念の基本部分を考えても，内部状態と操作の集まりとしてオブジェクトをとらえる際には，その規範として状態に注視した方向や，操作の実現方法としてとらえることなどいくつかの候補が考えられる．これらのとらえ方が必ずしも一致するとは限らない．また，オブジェクト間の構造的関係表現においても，問題領域における内部状態の階層(内部状態に注視した概念体系に基づく階層)と操作ベースでの階層の作成方法が考えられる．

このように，表現システムへの実現方法に多数の選択肢が得られる可能性をオブジェクト指向概念の基本部分自身が含んでいる．したがって，オブジェクトの構成法についての何らかの指針を表現システム側で持つことも一つの必要な方向でもある[3].

オブジェクト指向プログラミングにおける特徴には，抽象化，カプセル化，継承，ポリモルフィズムなどが挙げられたり[4]，カプセル化を強調して，インスタンシェーションやインヘリタンス，ポリモルフィズム，強いタイプ付けなどがとらえられたりもする[2]．特にソフトウェアの再利用性と結び付けて，継承機能が取り上げられることも多い[5]．しかし，オブジェクトの関係階層構造によって継承機能を利用することに関しては，よほど整理された概念体系をもつ対象問題領域でないかぎり，そのオブジェクト集合の階層構造が多段になることはまれである．実際に表現されたオブジェクト集合を洗練し，幾度もの改良を行うことで始めて価値のある階層構造が実現できるようになる．

概念としての階層構造の実現や継承機能に効果があることは認められるが，現実の対象領域でこれらの機能がどれだけ効果を発揮するか，これらの機能を利用して実現されたオブジェクトに対して，それを解釈するためにかかるコストとのバランスはどうであるかを考えるとそれほど効果的な機能とはいえない点もある．

問題領域において考えられる対象をオブジェクトとしてとらえることがすべての基本になるオブジェクト指向概念が，プログラミングを含めたシュミレーションに利用できることは，自然な方向である．Smalltalk-80を始めとし，これまでに実現されている様々なオブジェクト指向言語から，それらが，プロトタイプにも，アプリケーションの開発にも十分に利用できることが明かにされてきている[6,7].

確かにオブジェクト指向プログラミングはアプリケーション構築を容易にするための重要な要素を備えている．しかし同時に，オブジェクト指向プログラミングが持っている強力なパラダイムやメカニズムだけでなく，オブジェクトを設計するためのツールや応用領域を構築するために役立つオブジェクトを選んだり，オブジェクトの発展を管理することが重要となることも指摘されている[8].

オブジェクト指向プログラミング環境は，オブジェクトとして実現されているソフトウェアの発展が容易になるものでなければならない．これには，オブジェクト集合全体の正当性を維持するようなソフトウェア管理ツールも必要になる．特に，オブジェクト指向システムにおいてこれらのソフトウェアの発展を管理するために問題となるのは，オブジェクトの分類構造とそれに基づき継承される働きの理解度に関連する．オブジェクトの分類構造等をすべての問題領域を通じて共通に扱うことは，困難である．この点からも，各開発者の視点を組み込み，その上でこれらの管理や開発支援環境を実現する必要がある．

## 3．多視点に基づく表現システム

前述のようにオブジェクト指向プログラミングが注目されるのは，人が考える思考方法及び処理方法に準じた形で問題解決を進められるからである．これは，逆に実現されたオブジェクトは，その開発者や対象問題領域の思い込みが多分に含まれたものになることを意味している．このことから，これらのオブジェクト(特にその構造や関係表現)は，その開発者の視点を通じて利用されるべきであり，異なる視点で実現されたオブジェクトを利用するには，個々の視点に依存した分類や構造情報，関係表現を取り除いたレベルで，各オブジェクトが持っている本質的な機能を見いだす必要がある．

共有や再利用を考えている表現システムでは，このようなレベルの情報表現が提供できる必要がある．以下では，情報表現という言葉を用いて，計算機上の表現システムに記述されるすべての情報を示すこととする．これらのことからも，各開発者の視点を扱った表現システムが必要になる．視点は，開発者の違いあるいは問題領域の違いに

より，すべて異なり，これらはすべて実現されたオブジェクトに対する多視点となる．

このような視点の問題は，関係データベースにおいて，あらかじめ決められたデータ間の関係構造が利用者に強要されるという問題点[9]と類似した考え方でもある．データベース・システムにおいては，これらの問題に対処する方法として，仮想的スキーマ機能の実現[10]などのスキーマによるview操作が提案されている．

ここでは，この観点に注目し，多数の開発者がそれぞれ異なった視点で効果的に開発が進められることを目標にして，表現システム自身に開発者の視点を意識したViewsの考え方を導入する．

### 3.1 Viewsの考え方

開発者の視点を導入することのポイントは，表現システム上に記述された情報表現を新たな別の開発者または異なる問題領域においても理解できることである．ここでの理解とは，表現システム上に記述された情報表現に対して，開発者が対象としている問題領域のモデルと，表している内容の一致がとれることである．したがって，一般には各開発者によって，表現システムに記述された情報表現のすべてを全開発者が理解できるとは限らない．

いくつものオブジェクトが表現システム内に維持されていくオブジェクト指向表現システムでは，多数の対象問題領域において充実したプログラミングが進められるために，維持されている情報表現の理解を容易に行なえるような体系が必要である．

これらを目指した環境として，第一にHypertextやHyperCardなどで実現されているnavigationやbrows-ing等の機能に対応する検索機能を実現することが考えられる．これは，対象とするオブジェクトに的確に素早く到達するために効果がある．ただしこの場合にも，navigationやbrowsing等のコマンドやメニューなどが，開発者や利用者の視点に応じた柔軟な対応機能を持つことがよりその効果を上げることになる．

ここで提案する多視点の考え方では，表現システムが持つ情報表現を２つのレベルに分けている．これにより，視点に依存して操作する部分と共通に扱う部分の区別をし，視点の扱いを実現している．

表現システム内に実現される各オブジェクトの情報表現は，(1) 各開発者の視点に基づいた情報表現レベル(開発者のViewsと呼ぶ)の情報と(2) オブジェクトの共有化を進めるための表現システム内部の情報表現レベル(格納レベル)の情報に分けて実現している．オブジェクトの共有化は，各オブジェクトの理解の元に実現されるものである．したがって，表現システム内部の情報表現の格納レベルは，理解度の高い情報表現であり，各開発者の視点に依存した関係構造や情報表現を取り除いたオブジェクトとしての情報表現である．この格納レベルの情報表現に，開発者のViewsの情報である視点に依存した関係構造や情報表現を加えてアレンジしたものがその開発者の利用するオブジェクト環境になる．実際の開発段階で格納レベルの情報表現を，開発者のViewsの情報と結び付けることが先の理解を行なったことになる．つまり，開発者のViewsは，共有化を目指している情報表現である格納レベルの一つの理解方法を示すものであり，それらが結びついた結果が開発者の視点に基づいたオブジェクト環境となる．

このため，格納レベルにあるすべての情報表現が，開発者のViewsレベルの情報表現と結びつく必要はない．その開発者の問題領域に関係する情報表現のみがその視点に合わせて結びつけられる．この格納レベルと開発者のViewsレベルの区別は，個々のオブジェクト単位で行なえるものではなく，対象としている問題領域に依存して柔軟でなければならない．これはソフトウェア技術開発において，人間が関わっている部分に関する理解不足がソフトウェア開発を支援する諸技術開発のボトルネックになっている[11]のと同様で，複雑なものである．

この表現システムでは，次節の表現形式によって，これらの柔軟な機能を実現している．

### 3.2 表現形式

ここでは，多視点に基づく表現システムにおける情報表現として，格納レベルと開発者のViewsレベルの区別をつけるために基本となるオブジェクトの記述方法を述べる(図３)．図３において，斜体文字はこの表現システム側で決まったパターンであり，＜＞内に必要な情報を記述する．「」内の表現は，表現したい情報が複数個ある場合に，リスト構造による記述あるいは単にコロンでつなげて記述することのどちらでもよいことを示す．

オブジェクトが持つ内部状態の記述(**value**以降の

```
defclass <クラスオブジェクト名> : :
 relation
   supers  <Super-Class>, <Super-Class> ;
   parts
     pparts
     <Part-Name> - <Part-Class>=<Group-Name>, ... ;
     fparts
     <Part-Name> - <Part-Class>=<Group-Name>, ... ;
   group   <Group-Name>, <Group-Name>, ... ;
 value
        <Value-Name>(「<Default-Value>」),
        <Value-Name>, ... ;
 operation
    <Operation-Name>(「<Arguments>」) : <Body> ;
    <Operation-Name>(「<Arguments>」) ;
      . . .        .
```

図3　オブジェクトの記述方法

記述)や操作の記述(operation以降の記述)については，表現方法の違いがあるが，他のオブジェクト指向プログラミング言語と大きな違いはない．この表現システムに特徴的な記述は，オブジェクト間の関係表現におけるpartsの表現とgroupの表現である．これらの表現記述を明示的に行なうことで，開発者の視点の区別に役立てている．これに関連して，supersの処理が他のオブジェクト指向プログラミングとは若干異なっている．

groupでは，すべてのオブジェクトの分類関係を表し，そのオブジェクトが分類されるグループ名を列挙する．分類という観点では，Smalltalk-80のクラスカテゴリと同じであるが，一つのオブジェクトが複数のグループに含まれてよいこと，及びこれが開発者の視点の操作に利用されることが異なる．

partsの表現は，オブジェクト間の部分機能関係を表すための記述である．図3のように2種類の記述方法がある．ppartsが完全な部品としての機能を果たす関係記述であり，fpartsが機能継承の役目を果たす関係記述である．ここで提案する表現システムでは，オブジェクト間のメッセージ送信も，すべてこのparts関係を通じて行なうこととしている．したがって，原則的にはメッセージ送信先のオブジェクトはすべてここに登録される．これまでの階層構造によるオブジェクトの継承関係についても同様である．これらは，2節で述べた階層構造による継承機能の実現問題が根底にあることから導入した制約である．

オブジェクトの表面的な階層関係は，特にその対象の概念関係に依存する．

そこで，partsの表現の2種類からわかるように，この表現システムでのsupersの表現は，主に開発者の視点で対象としているオブジェクトの概念構造を理解する情報として用いられるものとなる．

一般のオブジェクト指向システムとここで提案する表現システムの関係表現に関する対応関係を，表1に示す．

表1　オブジェクトの関係表現の対応

| | 従来のobject指向システム | 提案する表現システム |
|---|---|---|
| 明示的な記述 | 階層構造 (super-sub) | オブジェクトの概念構造 (supers) |
| | | 継承関係(承諾機能) (parts内の fparts) |
| | | 利用関係(機能依託) (parts内の pparts) |
| 生め込まれた記述 | 操作の利用／受け渡し | (明示できない関係) |

この表からもわかるように，表現システムでは，オブジェクトの関係表現はすべて明示的に記述することを基本にしている．これによって関係表現としての曖昧性を厳密にさせる方向を取っている．

partsの表現の利用例として，オブジェクトAがWindow_Managerオブジェクトの機能を持つWindowオブジェクトにターム'test'を表示するためのメッセージ送信は，以下のようなppartsとoperation内の記述を行なうことで実現される．

```
defclass  A : :
 relation . . . ;
  parts
    pparts
      write_W - Window_Manager = WINDOW, ... ;
              . . . ;
 operation
    transfer(X,Y) : ..., (write_W <- print, 'test'),   ;
   ... . . .        .
```

このようにpartsは，Part-Name, Part-Class, Group-Nameの三つ組によって記述される．

Part-Name：そのオブジェクトの操作記述の中でこの部分機能を識別するための名前(write_W)

Part-Class：部分機能を実現しているオブジェクト名(Window_Manager)

Group-Name：部分機能を実現しているオブジェ

クトが含まれているグループ名(WINDOW)

ppartsの記述とfpartsの記述の違いは，表1に示したが，表現システムでの処理の違いとして見ると，前者は，Part-Classで指定されたクラスのインスタンスを個別に作成して処理を行なう．これは，内部状態の格納領域を個別に設けるということであり，機能依託である．後者は，新たに個別のインスタンスを作成することなく処理を行なう．これは，機能を承諾して処理する形である．

個々のオブジェクトの記述にオブジェクトの概念階層だけでなく，メッセージ送信関係となるpartsの表現及び，概念階層と機能の継承関係の区別等を加えることで，そのオブジェクトの振る舞いを実現するためのオブジェクトの関連構造を的確に把握できるようになる．さらに個々のオブジェクトは，その働きを反映したgroupの表現も持っているため，振る舞いにおける同類的な結びつきも把握できる．これらのことが，多視点での開発のための理解や改良を容易にさせる．

### 3.3 開発者のViewsと格納レベルの区別

異なる問題領域を対象にしている開発者が，それぞれ実現しようとしている情報表現を互いに理解させるために，理解度の高い情報表現と理解度の低い情報表現の区別を付けることを3.1で示した．前者が格納レベルの情報表現であり，後者が開発者のViewsレベルの情報表現である．これは，各開発者が実現された情報表現を利用するために理解しなければならない情報と，理解する必要のない(自由度のある)情報との区別でもある．表現システムでは，これらの区別を，関係表現の事細かな明示化により実現している．

各オブジェクトの表現形式とこの区別との対応関係を，表2に示す．オブジェクトの主な働きを規定する情報表現が格納レベルにあり，開発者の視点によって変更できない(変更する必要のない)情報表現である．内部状態と操作の記述はオブジェクトの基本及び外部とのインターフェースとなる部分であるから視点による変更操作の対象外になる．

開発者の視点との密接な関わりは，関係表現であり，これにも（1）そのオブジェクトの基本的振る舞いを詳細に決める関係表現と，（2）オブジェクトを概念的に理解し柔軟な振る舞いを決める関係表現がある．

表2　表現形式と情報表現レベルの対応

| 表現形式 | | 情報表現レベル | 多視点での変化 |
|---|---|---|---|
| 関係表現 | オブジェクトの概念構造 (supers) | 開発者のVlews | 変更可 |
| | 承諾機能の記述 (parts内の fparts) | 格納レベル | 変更不可 |
| | 機能依託の記述 (parts内の pparts) | 開発者のVlews | group内で変更可 |
| | グループの記述 (group) | 格納レベル | 拡張可 |
| 内部状態の記述 (value) | | 格納レベル | 変更不可 |
| 操作の記述 (operation) | | 格納レベル | 変更不可 |

（1）は，parts内のfpartsであり，承諾機能を記述している関係表現である．

承諾機能は，そのオブジェクト自体が持つ操作の記述に付随したものであり，その記述から抽出した機能である．これを実現するための関係表現がfpartsの記述である．したがって，これは視点によって変更されず，オブジェクトを利用する際に理解しなければならない情報表現である．

グループ(group)の記述はオブジェクトの操作内容に依存するため，変更することはできない．ただし，操作記述の拡張や視点の変化により，新たなグループとしての見方を追加することができる．

（2）は，すべて開発者の視点に依存した関係表現であるから，開発者のViewsレベルで，開発者の責任において自由に変更／改良できる．この変更／改良が視点の違いである．これはオブジェクトの概念構造であるsupersと機能依託の記述関係であるparts内のppartsである．

機能依託の変更は，オブジェクトの振る舞いを完全に取り替えないように，また，変更可能オブジェクトの検索を簡単化するため，あらかじめ指定されたgroup内での変更に制限している．これが，groupを視点として利用している所である．

これらの自由度のある関係表現は，各オブジェクトが作成された時点での情報がデフォルトとして格納レベルに与えられる．したがって，各開発者がその視点に合わせて変更しない場合は，開発者のViewsレベルでも常にデフォルトの関係表現が用いられる．

変更不可の部分の関係表現は，個々のオブジェ

クトが持つ機能を正しく実現できる状態を保持するために必要な，変更できない最小の関係を表すものである．

### 3.4 表現システムの効果

オブジェクト指向概念の基本は，オブジェクトである．したがって，この概念に基づく表現システムでは，情報表現としてオブジェクト(内部状態，操作,関係表現)が適切に格納され，登録されなければならない．各オブジェクトの付随的情報(一部の関係表現)は，開発や利用，発展の妨げになる．このためオブジェクトの情報表現としてはこのシステムのような格納レベルの情報表現を維持することが必要である．

格納レベルにあるオブジェクトを理解し，必要とするオブジェクトを選択し，自由度のある関係表現を開発者のViewsで決めることで開発者の視点を実現している．開発者のViewsに基づくオブジェクト環境では，問題としている対象領域に無関係なオブジェクトは，省かれ，問題領域に合うようにオブジェクトの関係表現が変更される．このように開発者のViewsは，その視点により理解するオブジェクトの集まりを示している．開発者のViewsを明確にすることが格納レベルに明示されたオブジェクトの限定された関係表現である．

従来，他のオブジェクトが持つ操作を利用するためのメッセージ送信は，そのオブジェクトの操作の中に埋め込まれて記述されていた．これを明示的な関係表現(機能依託関係)として記述することで，そのオブジェクトが実現している機能およびオブジェクト間の機能的関係を理解し易くさせた．これは，これまで操作の実現内容の辿り読みや実際にシミュレートするしかなかった点が，あらかじめ機能依託関係を理解できた上でこれらの情報を利用できるようになったことによる．

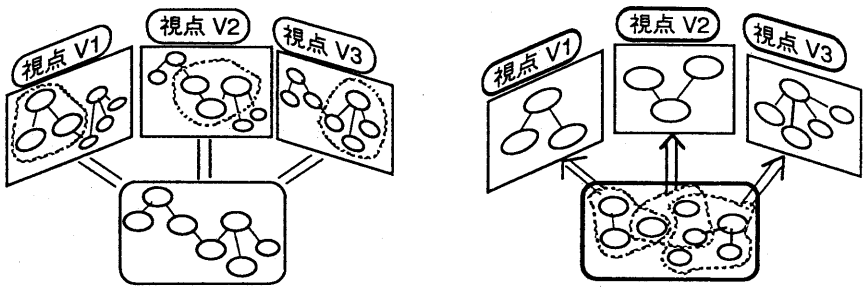
また，これまでのオブジェクト間の階層構造表現に対して，概念的体系構造を表すためのオブジェクトの概念構造の記述と，実質的な継承関係を表す承諾機能の記述に分離したことで，概念体系として理解すべき情報と操作の実現を理解するための情報が明確に区別され，必要とする情報を確実に得られるようになった．

これまでは図4 (a)のように各開発者の視点を，システム内部の既に実現されたオブジェクト構造等に合わせながら開発が進められていた．これがこの表現システムの方法を実現することで，図4 (b)のように，オブジェクトを理解するための必要最小限の情報表現(変更不可のもの)だけを持ったオブジェクト集合が，表現システムの格納レベルに登録され，各開発者のViewsにより，その中の必要なオブジェクトを開発者の視点に合った形でオブジェクト環境として投影する．このようにして共同利用しながら開発を進められようになる．

また，他の開発者の視点での関係表現を通して見ることで，視点ごとに注目しているオブジェクトの集まりやそれらの関係を見ることができ，この情報が各開発者の立場を理解するための指針としても利用できる．

## 4．まとめ

知識表現の分野で，これまでに提案された知識表現方式は数多い．それらの中でも基本的な知識




システムのオブジェクト表現　　表現システムの格納レベル

(a) これまでの利用方法　　(b) 提案する利用方法

図4　開発者のViewsと格納レベル

の関係記述は，この表現システムの関係表現として実現されている．
さらに特殊な記述形態や機能を付加することは十分考えられる．しかし，これまでのいくつかの知識表現方式のように，記述方法の制約や必要以上の機能付加による利用可能性を十分考慮しなければならない．

データベースの分野では，質問言語を実行することによって得られた新たな関係表現もすべて同レベルで扱えるデータベースというフィールドが用意される．この拡張としてデータベースでView[12]の研究が進められている．ここでの表現システムはこれを，プログラミングの各開発者の視点に対応させ，開発者のViewsとして発展させ，自由度の高いフィールドを用意したことに対応する．

さらには，関係表現だけでなく，オブジェクト自身のとらえ方にも注目し，各開発者の視点により，個々のオブジェクト自身のとらえ方も開発者のViewsレベルの情報により意向に合うように改良できることが必要になる．これは，表現システムの格納レベルにあるオブジェクトに対して，結合や分割操作を施して開発者の視点として適当な機能を持ったオブジェクトを，その視点に合うようにアレンジして開発者の視点に合ったオブジェクト環境を実現することである．これについても，開発者のViewsを発展させることで実現できる．

オブジェクト指向概念のもとで問題解決を進めることが，他のパラダイムに比べて，効果的である部分は，問題を段階的に，かつ部分的に解決していけることや問題に現われる事柄をオブジェクトとしてとらえて操作，処理できることである．しかし，現実には，オブジェクトとしてのとらえ方における柔軟性が，実現されたオブジェクトの理解を困難にさせることにもつながっている．ここで，提案した多視点を導入した表現システムは，この点に対する対処方法の一つを提案したことになる．

最近では，グループウェアやＣＳＣＷ[13]などの提案が活発になって来ている．これらの考え方も統合したプログラミングパラダイムの検討や，意思決定方法を導入したプログラミングなどの検討も今後さらに発展するものと思われる．

これらの本質を突き詰めて，プログラミングのレベルに反映させれば，新たなパラダイムや本質的な概念の発展を期待できると考えている．

## 参考文献


[1] M.L.Chan & B.Henderson-Sellers "Corporate Object-oriented Development Environment(CODE)" ACM SIGSOFT, Vol.15, No.1, 1990

[2] Meyer Bertrand "Object-oriented Software Construction" Prentice Hall, 1988

[3] 片山佳則 "多目的意思決定に基づくオブジェクトの組み立て法" 日本ソフトウェア科学会第６回大会論文集，pp169-172, 1989

[4] Lewis J. Pinson & Richard S. Wiener "An Introduction to Object-Oriented Programming and Smalltalk" Addison-Wesley, 1988

[5] Won Kim & Frederick H. Lochovsky " Object-Oriented Concepts, Databases, and Applications" ACM Press, 1989

[6] Proc. 1989 ACM Conf. on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA '89), Special Issue of SIGPLAN Notices, Vol.24, No.10, 1989

[7] Proc. 1988 ACM Conf. OOPSLA '88, Special Issue of SIGPLAN Notices, Vol.23,No.11, 1988

[8] Oscar Nierstrasz "A Survey of Object-Oriented Concepts" Part 1 in [5]

[9] DOOD89 Related Events "Tutorial on Deductive and Object-oriented DataBases" Advanced Software Technology and Mechatronics Research Institute of Kyoto (ASTEM RI), 1989

[10] Katsumi Tanaka, Masatoshi Yoshikawa & Kozo Ishihara "Schema Design, Views and Incomplete Information in Object-Oriented Databases" Journal of Information Processing, Vol.12, No.3, 1989

[11] 落水浩一郎 "データベース技術とソフトウェア技術" 情報処理学会データベースシステム研究会，90-DBS-76-3, 1990

[12] Sandra Heiler & Stanley Zdonik " Object Views : Extending the Vision" Proc. Sixth International Conference on Data Engineering, IEEE,Los Angeles, 1990

[13] 神田陽治 "コラボレーション技術の研究(解説A)", 電子情報通信学会誌，掲載予定