ICOT Technical Report: TR-218

TR-218

オブジェクト表現開発のための クラス構成支援について

片山佳則
(富士通)

November, 1986

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191~5
Telex ICOT J32964

Institute for New Generation Computer Technology

オブジェクト表現開発のためのクラス構成支援について

片山 佳則

富士通㈱ 国際情報社会科学研究所

オブジェクト指向概念を導入した表現システム(言語)を用いて、プログラムや知識を表現する場合、必要な知識をどのようなクラスオブジェクトの単位に決めるか、更にそれらのオブジェクトの関係をどう構成するかが重要な問題となる。また、オブジェクトの構成は、表現されたオブジェクトの再利用可能性に対しても影響を与える。現在のところ、これらはすべて開発者に任されており、結果として、自由に表現されている。

本稿では、このような問題を解決するために、表現する対象に最適なオブジェクト構成を与え、オブジェクト表現の開発を支援する方法とそのシステムについて提案し、適用事例を述べる。

A CLASS STRUCTURE SUPPORTING SYSTEM FOR OBJECT-ORIENTED REPRESENTATIONS

Yoshinori KATAYAMA

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

When we express programs and knowledges by using an object oriented representation system (language), it is very important to divide knowledge into modules and to express them by class objects. It is equally important to structure the objects by relating them in an appropriate manner. Such properly structured modular objects are reusable for other purposes. Currently, these tasks are totally left to programmers' choice. As a result, objects and structures are freely written on an object-oriented representation system.

This paper proposes methods and a supporting system for properly structuring objects of a representation system. It also presents examples how a programmer can obtain supports by the system.

1. はじめに

オブジェクト指向プログラミングの始まりである Smalltalk[1] やFlavors[2] ,Loops[3] に続き，オブジェクト指向概念を導入したシステム・表現言語[4]-[9] は多数作成されている．特にこれらの表現システムは，オブジェクトの基本的考えに基づき，表現自体は体系化され効率よく行えるようになっている．しかし，これらの表現システムを用いて知識やプログラムをオブジェクトとして適切に表現／開発する場合には，その技法に精通した者「エキスパート」が行う(間に入る)必要がある．このようなエキスパートも含め，オブジェクト概念を導入した表現システム上で知識の表現を行う際には，その設計に基づく段階的開発支援[10] が考えられる．さらに，エキスパートに頼らず，オブジェクトの本質を把握した表現を実現するための支援方法も必要である．特に，新しい分野の知識を表現する場合には，その知識に対して適切なオブジェクトの構成を考えるのは大変困難である．このため，これらの点を包括して処理するような支援システムとして，対象に応じた最適なオブジェクトの構成表現を導く機能を持つシステムが必要になる．これにより表現を行う場合にエキスパートのような仲介者の必要もなくなり，自由に表現システムを利用できるようになる．さらに，各開発者は，表現する内容のみを深く検討することができ，オブジェクト構成についてはすべてシステムに任せることができる．

本稿で提案するシステムは，これらの機能を実現し，表現したい内容に合ったオブジェクト構成の構築を適切に支援するものである．これによって，オブジェクト構成のための関係表現はシステムが持つ概念ですべて統一して実現され，その意味でもオブジェクトの関係が明確になり，各オブジェクトの働きを容易に把握できるようになる．

オブジェクト構成の構築支援は，オブジェクト間のメッセージ交換(自分自身へのメッセージ交換も含む)とスロットへのアクセス情報を分析することで行われる．構築支援の機能については，システム側で完全に構成を変更させてしまうことも可能であるが，現段階では，システムが自動的に構成を変更することまでは行わず，ユーザに提示するのみに止めている．

2. オブジェクトの関係表現

これまでに開発された表現システムは，オブジェクトとして様々な知識を表現するためにオブジェクト間の各種の関係を実現している．それらは，大別すると次のようになる．ここでは，各クラスは，概念を表すものとする．

(1)IS-A関係： クラスの包含関係を表す．

(2)PART-OF関係：オブジェクトの機能・構造的関係を表す．

(3)MANAGER-OF関係：実行を管理する関係を表す．

(4)INSTANCE-OF関係：表現されたクラスを利用するための実体(インスタンス)との関係を表す．

知識を表現することに関しては，特に(1)と(2)が基本的な関係となる．その他の関係は，管理やプログラムの実行を考慮する際に必要となるものである．従って，表現のためのオブジェクト構成の構築支援としては，第一に(1)と(2)の関係についての支援機能を考慮しなければならない．この２つの関係が各システムでどのように利用されているかは，次のようにまとめることができる．

＜IS-A関係＞： 概念の包含関係を表し，集合間の関係としては，部分集合関係(Subset/Superset) を表現する．述語間の関係としては，一般化と条件づきによる特殊化(Generalization/Specialization)を表現する．特に，種類についての関係と考え，AKO(A Kind Of)関係とも呼ばれる．さらに細かい分析[11] も可能である．この関係では属性の継承が行われる．

＜PART-OF 関係＞： 部分集合としての関係ではなく，機能・内部構造に関して部分関係を表すものである．基本的には，構成要素としての関係が強い[12]．IS-A関係と異なり，ほとんどの表現システムではこの関係による属性の継承はない．この関係は，概念(クラス)間に結ばれたものがそれぞれの概念のインスタンスに対する関係としてもそのまま受け継がれ，インスタンス間の結びつきをも表す．

これらの関係構成を正しく構築するための支援が本研究の目的である．これらの関係は概念間の関係であり，オブジェクト表現に対するこのような関係構成の支援は，クラスオブジェクトの構成を支援する方法に対応する．この構成を正しく構築するためには，関係表現に合ったクラスの分割を規定する方法が重要になる．

IS-A関係におけるクラスの分割方法は，部分集合関係の場合，属性の継承を有効に利用できることに視点を置き，記述量と適切な記述位置を定め，全体として整った構成を考えれば良い．他方，一般化のためのクラス分割では，記述したものの意味などの複雑な要因を考慮する必要がある．

PART-OF 関係としてのクラスの分割方法は，IS-A関係のように継承関係を持たないため，機能・内部構造としての部分関係を的確に表すための分割規則を設定する必要がある．この部分関係では，各部分はその部分特有の属性(性質・特徴)と機能を持つ必要がある．オブジェクト表現においてこのような特有の属性と機能を表現するためのオブジェクトは，属性をスロットで表現し，そのスロットの入出力としての操作だけでなく，内部的な操作も含めて，そのオブジェクトの機能を表すメソッドを持つ必要がある．さらに部分関係として表現することから，部分に対応するオブジェクトは全体としてのオブジェクトから利用される形(部品としての関係)になっている必要がある．これらの関係を用いたオブジェクト構成を，Fig. 1 に示す．

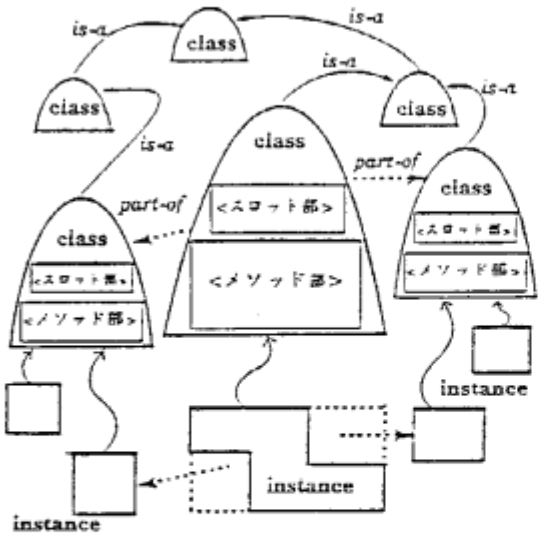


Fig.1 オブジェクト構成

オブジェクトの構成を検討する場合には，その対象を一つのオブジェクトに限定して進める場合と複数のオブジェクトを対象として全体の構成を考える場合がある．後者の場合は，複数のオブジェクト間の関係(メッセージ交換)に注目しなければならない．

あるオブジェクトが処理を分担させるために，他のオブジェクトにメッセージを送る場合を考える．オブジェクトが他のオブジェクトとメッセージを交換するということは，その基本的レベルを考えれば情報を受け渡して計算を進めることである．概念としてこれを表現する場合には，それらのメッセージ交換はIS-A関係による継承の流れをメッセージとして動的に探索するか，PART-OF 関係により内部構造の一部分として表されているオブジェクトに対して行われるかのどちらかである．この後者の考え方は，Actor 理論における acquaintances[3] という通信情報の局所性に対応する．このように機能分担の処理を考えた場合，オブジェクトが他のオブジェクトと直接メッセージ交換をする場合は，これまでに述べたIS-A関係か PART-OF関係を必ず持つことになる．これらの関係は動的な動作関係も含むことから，今後はsuperset関係(IS-Aに対応) subparts関係(PART-OFに対応)と呼ぶことにする．

一つのオブジェクトがどのようなsuperset関係とsubparts関係を持つかを正確に把握することが，複数のオブジェクト間の関係を把握し，最終的には全体の構成を明らかにすることになる．ただし，各オブジェクトの関係を個々に分析する場合には，全体構成の中で，オブジェクトの同一性の検討を行う必要がある．この同一性に関しても，個々のオブジェクトの機能が明確にされ，他のオブジェクトとの関係が明らかであれば，それらの関係を用いることで同一性検討処理は容易になる．

3. クラス（オブジェクト）構成のための分割概念

本節では，一つのオブジェクトを分割して，その構成を与えるための基本的考え方を提示し，それに従った分割方法をまとめる．さらにオブジェクトから得られる情報の詳細な分類を行い，それらがこの分割方法においてどのように利用されるかをまとめる．

3.1 分割の基本的考え方

クラス分割において，一般化を目的とするためには，意味などの複雑な要因を考慮しなければならない．そこで，ここでは，シンタックス情報に基づく機能・構造としての部分関係による分割を目的とし，subparts関係を重視する．subparts関係で表せないものに対して，superset関係の使用を検討する．

オブジェクトを分割するには，オブジェクト内のスロットやメソッドの関係状態から独立性の高いいくつかの setを見つけ出す必要がある．複数の setが見つけられない場合は，現時点でそのオブジェクトを分割することは不可能である．さらに，subpartsとなることを前提にしてsetを見つけることから，2節で述べたように，各setはスロット(属性)を含んでいなければならない．従ってこれらのsetは，オブジェクト内のスロットを中心にし，それと結合すべきスロットやメソッドを取り込む形式で作成することができる．スロット，メソッドの関係から，取り込むための結合規則は次の二つにまとめることができる．

(a) スロットの更新(put)操作を持つメソッドは，そのスロットと結合する．

(b) メッセージ交換関係をグラフに対応させた場合，グラフとして強連結(strongly connected)[4]となるメソッドはすべて結合する．

この結合規則だけで作成されるsetを，最小setとする．これらの最小setがpartとなるかどうかによって，クラスの分割を行うことができる．

あるsetがpartとなるかどうかを分析するためには，全体とpartが，スロット・メソッドを通じて，どのような関係にあるべきかを明確にさせておかなければならない．これに従ってpartとなる setを見つけ出すことができ，subpartsに関するクラス分割が行われる．

＜partと全体の関係＞

(1) 全体側からpartが持つスロットやメソッドに対する操作の関係は次の通り．

○ メソッドは自由に呼びだせる．

○ スロットを更新する際には，表現システムが用意している特別なPREDICATE "putvalue"で直接更新することはできない(メッセージ交換による更新のみ)．

○ スロットの参照は自由にできる．

基本的に，全体側からpart側の操作に対する制約は，スロ

ットの更新についてのみで，その他に関しての制約はない．これらは，partが部分機能を分担していること及び更新操作は各オブジェクト内で閉じた内部の働きであることから明らかである．

(2) part側から全体が持つスロットやメソッドに対する操作の関係は次の通り．

○ メソッドは呼び出せない．

partは全体側から利用される形であり，その逆にはなれない．［メッセージ交換については，返答のためだけの場合を考えられるが，ここでは，情報伝達のためだけにメッセージ交換を行うことを考慮していない］

○ スロットの更新・参照はできない．

部分機能がその処理において全体の持つスロット（属性）に関する操作を必要とする場合，それは部分機能であるpartとして閉じていないことになる．

part側から全体に対しては，すべての操作が制約を受ける．これは部分側が主になって全体に対して影響を与えるような操作を行わないことが基本であるからである．

最小setがpartとなるかどうかは，各最小setについてここで示したpartと全体の関係を分析することで判断できる．この最小setによるクラス分割の実際例については，4節で示す．

一般に複雑なオブジェクトを分割したい場合には，最小set(結合規則(a)(b)だけでの取り込みによるsetの作成)だけでは不十分である．そこで複雑なオブジェクトに対しても適用できるように，分割の考え方をさらに発展させる．

最小setは，スロット中心に考えていることから，最小setに含まれないメソッドが多数存在する．これらのメソッドだけでもさらに新たなset(メソッドset)を構成できる．このメソッドsetは，スロットを持たないためこれだけではpartの対象にならないが，最小setの拡張や，オブジェクト自身の働きを理解する上でも役立つことになる．

メソッドset：最小setに含まれないメソッド群のうち，
呼び出し関係のあるメソッドのset．

このメソッドsetと最小setにより，再度オブジェクト内のセットの構成を検討する．このレベルで分割単位となるセット(基本set)の作成方法自体は，最小set作成の場合と同様の手順であるが，基本set作成の検討対象はスロットやメソッドではなく，それぞれ最小setとメソッドsetとなる．結合を決めるための結合規則は，次の規則だけである．

(c) 強連結な関係を意識せず，最小setと関係のあるメソッドsetは結合する．

規則(a)に対応して，スロットの参照関係から，複数の最小setを結合する規則も可能であるが，この参照関係は，オブジェクト間のメッセージ交換方法をユーザが再検討することによって変更できる．現在は，支援機能として考えているのでこの参照関係による結合を規則としていない．

規則(c)で作成された新しい基本setについて，partとなるかどうかを分析することで，より複雑なオブジェクトのクラス構成支援が行える．この実際例は，5節で示す．

オブジェクト内部のセットの構成は，オブジェクトの分割以外にも利用できることを，実行例と結びつけて議論する．

3.2 セットの分割方法に関連する関係情報

結合規則(a)(b)によって分離される関係から，最小setがオブジェクトに対してpartとなれるかどうかの判定（クラス構成につながる）を同時に行うことができる．これは3.1節で示したpartと全体の関係に基づいて決められる．

＜スロットとメソッドの分離に関して＞

① メソッドが参照(get)しているだけのスロット
［スロット側（そのスロットが含まれているset）はpartとなり得る］

「partとなり得る」という表現は，そのスロットを含むsetが他のメソッドとの関係でpartになり得ない条件がなければ「partにする」ことを意味する(以下同様)．

＜メソッド間の分離に関して＞

② メソッド間が強連結ではなく，単に呼び出される関係だけの場合
［呼び出されるメソッドを含む各setはpartとなり得る］

⑤ メソッド間が強連結ではなく，単に呼び出す関係だけの場合
［呼び出す側のメソッドを含むsetはpartとなり得ない］

次に，一つのオブジェクトから取り出す情報を詳細に分類し，それがどう利用されるかをまとめる．

［A スロットとメソッドの関係をスロット側から集める］

スロットが受ける操作の内容が，更新(上書き)「put」のみ及び更新・参照「put/get」の両方である場合，それらすべてのメソッドは分離できない(結合規則(a))．

操作内容が参照「get」のみであるメソッドはすべて分離する(分離条件①)．

<関係表現との対応>

分離する／しないに関わらず，スロット側はすべてpartとなり得る．

［B スロットとメソッドの関係をメソッド側で集める］

メソッドが行う操作の内容が，参照「get」のみの場合は分離し，更新（上書き）「put」のみ及び更新・参照「put/get」の両方の場合は分離できない．

<関係表現との対応>

分離した場合のみ，メソッド側はpartになり得ない．その他は，partとなり得る．

[C　メソッド間のメッセージ交換関係(オブジェクト内部)]

メッセージ交換関係は次のように分類する.

メソッドが行う操作の対象：

- C1：複数のメソッド
- C2：一つのメソッド
- C3：他のメソッドとの関係はない

操作の内容：

c1：他のメソッドからメッセージを受け取る(receive)

c2：他のメソッドへメッセージを送る (send)

c3：c1とc2の両方の操作を持つ (send/receive)

これらの分類された情報は，次のように利用できる.

操作対象に関わらず，操作内容がc1のみ及びc2のみの場合は分離できる (分離条件②③)．操作内容がc3の場合は分離できない (結合規則(b))．

<関係表現との対応>

C1及びC2の場合：　操作内容c2による分離を行わない限りpartになり得る(分離条件②)．c2による分離を行った場合は，partにはなり得ない(分離条件③)．

C3の場合：　partになり得る.

これらA～Cの情報を用い，前節の規則・条件を当てはめることで，クラス構成のための適切な分割が行える.

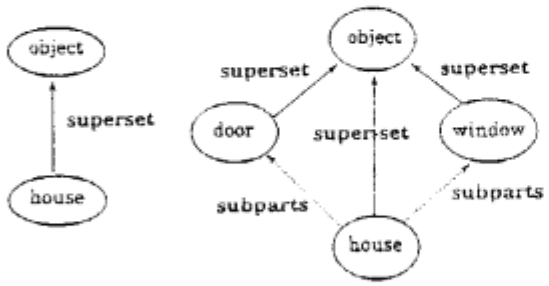
4. クラス構成のための支援

本節では，システムがどのようにクラス構築支援を行うかを，簡単な事例を取り上げて説明する．ここで示すオブジェクトは，表現システムKORE/KR[15]上で実現したものである.

```
defclass house ::
 superset   object ;
 value_set    door-close,   window-shut ;
  method
     go       :=: (@door=open, ...; ...);
    enter     :=: (@window=open, ...; ...);
  door_open   :=: (@door=open, ...;
                   #(door,open), ...);
  door_close :=:  ... ;
  window_open :=:  ...  ;
  window_shut :=: @(window,State),
    (State=shut, ...; #(window,shut), ...).
```

@,#は，すべて表現システムが持つマクロ表現である．意味は @:getvalue　#:putvalue

Fig.2 事例オブジェクト(クラスオブジェクトhouse)



(a)初期のクラス構成　(b)支援後のクラス構成

Fig.3 事例オブジェクトのクラス構成

```
defclass house ::
 superset   object ;
 subparts  door1-door, window1-window ;
  method
    go :=: (door1 (- get,[state,SI),
            (S=open, ...; ...) ;
   enter :=: (window1 (- get,[state,SI),
              (S=open, ...; ...) ;
   door_open   :=: (door1 (- open) ;
   door_close  :=: (door1 (- close) ;
   window_open :=: (window1 (- open) ;
   window_shut :=: (window1 (- shut) .

defclass door ::
 superset   object;
 value_set  state-close ;
  method
    open  :=: (@state=open, ...;
               #(state,open), ...) ;
    close :=:  ...  .

defclass window ::
 superset   object;
 value_set  state-close ;
  method
    open :=:   ...  ;
    shut :=: @(state,State),
      (State=shut, ...; ...) .
```

Fig.4 支援後の事例オブジェクト
(クラスオブジェクト house,door,window)

基本となるクラスはFig. 2であり，本稿で提案するシステムにこのオブジェクトを入力してクラス構築の支援を受けると，クラス構成はFig. 3(a)からFig. 3(b)になり，それぞれのクラスはFig. 4に示したオブジェクトとして実現される.

この支援システムは，はじめに対象オブジェクトのシンタックスを分析して3.2節で示した必要な情報を取り出し，3.1

```
**********  structure_object1   **********
THE STRUCTURE-DATA OF  house  CLASS **

*****   CLASS STRUCTURE INFORMATION  *****
        ----------------------------
        *****  part
          **SLOTS**
part(a)   door
          **METHODS**
         door_close/0 door_open/0
        ----------------------------
        *****  part
          **SLOTS**
part(b)   window
          **METHODS**
         window_shut/0 window_open/0
**********  end of data   **********

*****************************************
**** INFORMATION ABOUT OTHER METHODS ****
*****************************************
 ------------------------------
    **METHOD SET**
  go/0 enter/0
     GETVALUE
       window-get door-get
 ------------------------------
**********  end of data   **********
```

システムが持つ組み込みのメソッドはすべて情報からカットされる．各メソッドの記述には引数Typeが付加されdoor-close/0は引数0のメソッドを意味する.

Fig.5 クラス構成のための支援情報(その1)

節で述べた規則・条件により，Fig. 5に示すような新しいクラス構成のための情報が導かれる．Fig. 5の情報では，houseクラスを二つの最小setであるpartオブジェクト(part(a), part(b))を持つオブジェクト構成に変更すべきことを示している．

さらにpartオブジェクトをsubpartsとして表現した場合にhouse自身が持つメソッド集合も同時に示されている(INFORMATION ABOUT OTHER METHODS)．この場合go及びenterメソッドは，このオブジェクトの中でそれぞれ独立なものではなくメソッドsetとしてまとめられている．houseオブジェクトはこの他にメソッドを持っていないことがわかる．

このようにして，Fig. 3(b)のクラス構成が実現され，Fig. 4の各オブジェクトが作成される．

5. 複雑なオブジェクトに対するクラス構成のための支援

4節で示した事例では，最小setがpartとして完全に分離可能なものであった．しかし，3.1 節で述べたように，すべてのクラスオブジェクトがこの最小setでpartとして完全に分割されるわけではない．表現する知識(プログラム)が複雑な場合はpartとなり得ない最小setが多数出てくる．

システムがpartと判断できないのは，他の(その最小set以外の)スロットやメソッドとの関係において，3.1節での規則や条件によりpartとなりえない以下(1)(2)の状態になっているためである．

(1) そのsetが計算(メッセージを受け取って処理をすること)を行うために，set以外のスロットの参照を必要としている．

[Fig. 6ではKEY slotsとして提示されている]

(2) そのsetが計算を行うために，set以外のメソッドを呼び出す必要がある．

[Fig. 6ではKEY methodsとして提示されている]

これらの状態においても，次の処理を行うことでpart形式にすることができる．(1)に対して，少ないスロット参照は，メッセージ交換の時のargumentとして処理するように記述を変更する．(2)に対して，呼びだすメソッドが他のどの最小setにも含まれていない(メソッドsetにある)場合には，3.1節の結合規則(c)を用いてそれらを統合したものを分割単位の基本setとしなおす．これらの処理によって，さらに進んだ構成支援を行うことができる．この事例をFig. 6に示す．

```
**********  structure_object2  **********
THE STRUCTURE-DATA OF  structure  CLASS **

*****   CLASS STRUCTURE INFORMATION  *****
 ----------------------------
 *****  main
      KEY slots
        structure  class
SET A
      **SLOTS**
      method_pattern3 method_pattern2
      methods slot_pattern s_list m_list
      **METHODS**
    search_slot_pattern/5 set_slot_pattern/0
    set_method_pattern/0 ...
   ----------------
                 ------------
 *****  main
      KEY slots
        m_list s_list
SET B
      **SLOTS**
      structure class
      **METHODS**
    rename1/8 rename/0 arrange_all/2
    collect_structure/50
   ----------------------------
 *****  main
      KEY slots
        method_pattern3 methods
        slot_pattern method_pattern2
      KEY methods
        part_analysis21/11 part_anal1/2
SET C   part_ana411/3 part_ana412/5

      **SLOTS**
      main_set part_method part_rec_method
      part_slot next_part_slot part_set
      **METHODS**
    part_analysis2/0 part_ana4/5
    part_ana2/6 part_anal/4 ...
**********  end of data  **********

****************************************
**** INFORMATION ABOUT OTHER METHODS ****
****************************************
 ----------------------------
      **METHOD SET**
   part_analysis21/11
      RECEIVE
        part_analysis2/2
      GETVALUE
        method_pattern3 method_pattern2
 ----------------------------
      **METHOD SET**
   rename1/6
 ----------------------------
      **METHOD SET**
   search_slot_p1/6
      SEND
        search_slot_p2/4
      RECEIVE
        search_slot_pattern/5
 ----------------------------

        - - -

**********  end of data  **********
```

Fig.6 クラス構成のための支援情報(その2)

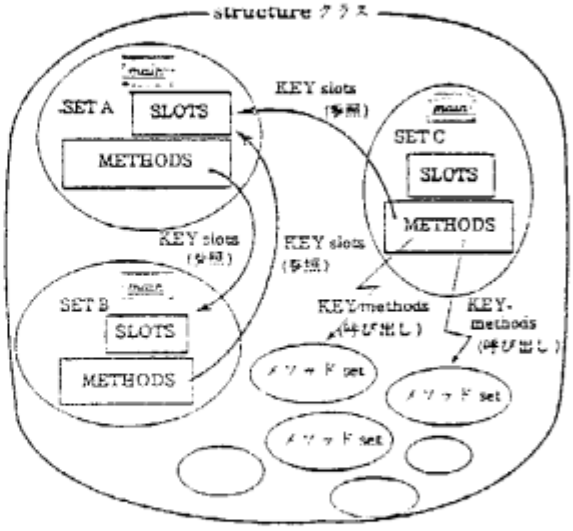


Fig.7 クラス構成のための支援情報(その2)の関係図

Fig.6は4節と同じアルゴリズムだけで処理をした場合の結果である．この事例の対象オブジェクトとして，本論で取り上げている支援システム自身のクラスstructureを用いた(この支援システム自体は，subparts関係を用いて作成されている)．Fig.6のセット間の関係をFig.7に図示する．

Fig.7により，そのオブジェクトを構成するset群とその間の関係を確認することができる．Fig.6ではpartとなる最小setがなくすべてmainと表示されている．最小setに含まれないメソッドsetの情報もINFORMATION ABOUT OTHER METHODSとして同時に示されている．これらのメソッドsetの情報からも，そのメソッドが必要であるかどうかを検討することができる．例えば，二つ目のメソッドsetとして出力されているrename1/6はこのオブジェクト内部では全く利用されていない(SENDもRECEIVEもない)ことが分かり，そのオブジェクトの機能を再検討することにより，不必要であることがわかる(この形式で表されるものが全て不必要ではない)．このように一度作成した後では見つけにくいようなことも的確に指摘してくれることがわかる．このFig.6の情報に対し，本節で述べた，(2)の変更手続きを行うことでFig.8の結果が得られる．

```
**********  structure_object2   **********
THE STRUCTURE-DATA OF  structure  CLASS **

*****   CLASS STRUCTURE INFORMATION   *****
 ----------------------------
 SET A,SET B については Fig.6 と同様
 ----------------------------
 *****  main
      KEY slots
        method_pattern3 methods
        slot_pattern method_pattern2
SET C
    **SLOTS**
    main_set part_method part_rec_method
    part_slot next_part_slot part_set
    **METHODS**
  part_ana412/5 part_ana411/3 part_anal1/2
  part_analysis21/11 part_analysis2/0
  part_ana2/6 part_anal/4 part_ana4/5
     ...
**********  end of data   **********

****************************************
**** INFORMATION ABOUT OTHER METHODS ****
****************************************
 ----------------------------
    **METHOD SET**
  rename1/6
 ----------------------------
    **METHOD SET**
  search_slot_p1/6
    SEND
      search_slot_p2/4
    RECEIVE
      search_slot_pattern/5
 ----------------------------

       . . .

**********  end of data   **********
```

Fig.8 クラス構成のための支援情報(その3)

変化するのは3つ目のset(SET C)であり，Fig.6で示されたKEY methodsがすべてメソッドset内にあることから，set内に取り込まれている(SET Cの中のMETHODSの項に含まれている)．set内に取り込まれたものは，メソッドsetから除かれている(Fig.6のINFORMATION ABOUT OTHER METHODS にあった3つ目のpart-analysis21/11がFig.8ではない)．

また，Fig.6の結果に対して，mainの原因となっているKEY slots について(1)の変更を行うことでpartとすることができる．これらの変更結果の概略はFig.9のようになる．

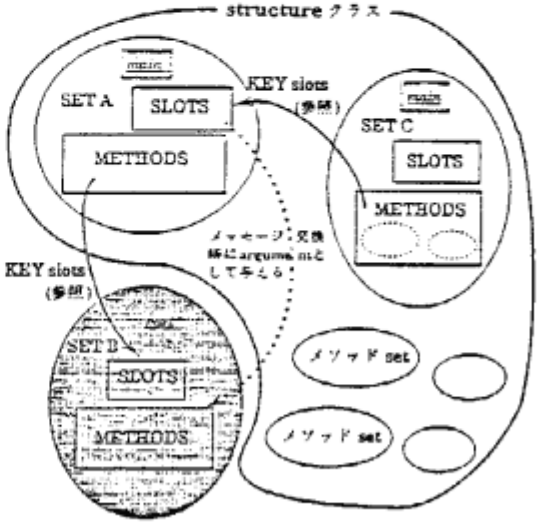


Fig.9 クラス構成のための支援情報(その3)の関係図

このように，(1)や(2)の変更を適宜行うことで，mainからpartへの移行が行われる．mainとして示されたsetをすべてpartで表現する必要はない．1節において，本論で提案するシステムは，クラス構成を自動的に変換しないことを述べた．これは，本節で示したような適切な変更(特に(1)について)がすべてユーザに依存したものであり，表現の細かい変更までをシステムが単純に行わないためである．このシステムではオブジェクトの動作を構造的な面だけで把握しているため，主な目的は適切なクラス構成を構築するための支援の手助けとなる情報を示すこととしている．これらの情報だけでも，支援機能としては非常に効果のあることがわかる．

6. 考察

これまで実行例とともに，クラス構成支援システムの動作及び支援機能を述べたが，この支援システムの問題点と利点について考察する．

この構築支援システムでは，現在概念レベルでメッセージの受け取り側が明確でない場合の処理が行われていない．一般に，他のオブジェクトとのメッセージ交換は，ほとんどがsubparts関係を通して行われることを2節で述べたが，メッセージの引数としてオブジェクト名を受け取り，それに対してメッセージを送る場合などもある．この形式での互いのオブジェクト間のメッセージ交換処理についても検討する必要がある．また，スロットの利用に関しての分析を進めること

で同種の形で用いているものの判断やメソッドが持つ argumentの分析によるスロット化などの検討も考えられる．さらに，対象とするオブジェクトが大規模であると，関係情報に対する検索を効率的に行うことが不可欠となる．現段階では，オブジェクトの開発と並行して同時にこのシステムを用いることで4節，5節で示したように様々な効果が得られている．

このクラス構成支援システムの利点は，次のようにまとめられる．

○ クラスオブジェクト構成のための関係表現（特にsub-parts 関係）を適切に扱うことができる．

○ オブジェクトの関係表現は，システムで統一した意味を持って実現される．

○ 作成されたクラスオブジェクト内部の構造（スロットを中心にしたメソッドのまとまり方及びメソッドだけでのまとまり方等）を正確に把握することができる．

これらの利点は，各オブジェクトの再利用のためにも重要な働きを担い，さらにオブジェクト開発のデバッグにも貢献し，作成したオブジェクトを容易に理解させるための変換としても重要な役割を持つ．また，構成の支援を受けるために各オブジェクトから得られた情報は，オブジェクトの管理などに対する情報としての利用も考えられる．

また，本論文ではクラス構成のための関係表現としてsub-parts関係に重点を置いたが，視点を変えることによって最小setや基本setは，superset関係に対しての利用も考えることができる．これらの点については，再度検討する必要がある．

ここで提案したようなクラスオブジェクトの構成支援を，表現システムが持つことによって，ユーザはオブジェクト指向にそれほどこだわらず，一つのクラスオブジェクトの世界ですべての知識を表現するだけでよくなる．システム側でそれらの世界を機能的に見て適切に分離し，それらの関係をシステムが持つ意味で統一して設定してくれる．さらに，各クラスオブジェクトの世界の中でもその内部的なsetを示してくれる．このset は以後の開発(機能付加)に役立ち，ユーザ自身がさらに構成を考える際にも役立つものである．

7. 結び

オブジェクト指向概念を導入した表現システムのオブジェクト（クラス）構成問題に対処するための構築支援システムについて述べた．これは，オブジェクトが持つスロットやメソッド，及びそれらの利用状況を用いて，最適なオブジェクト構成への支援を行うものである．今後は，6節で述べた問題点等の解決を行う必要がある．

最後に，本研究に関して貴重な議論，助言をして頂いた戸田部長，情報社会学室の新谷，平石研究員に感謝します．

尚，本研究は，第五世代コンピュータプロジェクトの一環として行ったものである．

［参考文献］

1) Goldberg,A.and Robson,D. : Smalltaik-80 The Language and its Implementation, Reading,Massachsetts, Addison-Wesley (1983)

2) Weinreb,D.and Moon,D. : Lisp Machine Manual (1981)

3) Bobrow,G.and Stefik,M. : The LOOPS Manual,Xerox PARC Knowledge-based VLSI Design Group memo KB-VLSI-81-13 (1983)

4) Chikayama,T. : Unique Features of ESP, Proc. International Conference on Fifth Genaration Computer Systems, pp.292 ～298 (1984)

5) Mizoguchi,F.,Ohwada,H.and Katayama,Y. : LOOKS:Knowledge Representation System for Designing Expert Systems in a Logic Programming Framework,Proc. International Conference on Fifth Genaration Computer Systems, pp.606 ～612 (1984)

6) Furukawa,K.,Takeuchi,A.,Kunifuji,S.,Yasukawa,H., Ohki,M.and Ueda,K. : Mandala:A Logic Based Knowledge Programming System, Proc. International Conference on Fifth Genaration Computer Systems,pp.613～622 (1984)

7) 石川，所：並列オブジェクト指向知識表現言語Orient84/K, コンピュータソフトウェア,Vol.3 No.3 July (1986)

8) Bobrow,G.,Kahn,K.,Kiczales,G.,Masinter,L.,Stefik,M. and Zdybel,F. : COMMONLOOPS Merging Common Lisp and Object-Oriented Programming,ISL-85-8, Xerox Palo Alto Research Center, August (1985)

9) 米澤，柴山,Briot,J.,本田，高田：オブジェクト指向に基づく並列情報処理モデルABCM/1とその記述言語ABCL/1, コンピュータソフトウェア,Vol.3 No.3 July (1986)

10) 片山：オブジェクト表現を用いたプログラム開発の支援環境－プログラム開発における計画・管理システムの実現－,ICOT Technical Report TR-169 (1986)

11) Brachman,R.J. : What IS-A Is and Isn't:An Analysis of Taxonomic Links in Semantic Networks, IEEE Computer 16(10), pp.30 ～36 October (1983)

12) 片山，新谷，平石：問題解決支援環境KORE（その3）－知識表現サブシステムKORE/KRとその概要－，第32回情報処理全国大会論文集，pp.1147 ～1148 (1986)

13) Hewitt,C. : Viewing Control Structures as Patterns of Passing Messages, Journal of Artificial Intelligence, Vol.8 pp.323 ～364 (1977)

14) Harary,F. : GRARPH THEORY, ADDISON-WESLEY, October (1972)

15) 片山：表現システムKORE/KR の概要，富士通国際情報社会科学研究所，Brainware (1985)