ソフトウェア工学 62－3
（１９８８．９．２８）

ソフトウェアの対象モデルと実現モデルの対応構造に関する一考察

小林要・木村高久・織田充
富士通㈱国際情報社会科学研究所

ソフトウェアには応用分野における用途の側面と，プログラム実体としての実現手段の側面の，２つの側面に関する情報が含まれている．ソフトウェアの効果的な再利用を図るために，これらの情報を，対象モデル，実現モデルとして明確に分離する方法について考察している．本研究では，応用分野における基本語彙と表現のパターン，プログラム分野における基本語彙と表現のパターン等をそれぞれ基盤言語から抽出し，それらを用いて対象モデル，および実現モデルを構築する方法を示し，さらに，役割パス等の概念を導入し，これらのモデルを相互に対応づける構造を提案している．例題を用いて，対象モデル，実現モデルおよび，相互の対応構造の具体的な実現方法について考察している．

A Consideration on The Structure of Corresponding An Application Model to An Implementation Model of Software

Kaname Kobayashi, Takahisa Kimura and Mitsuru Oda
International Institute for Advanced Study of Social Information Science,
FUJITSU LIMITED
Miyamoto 140, Numazu, Shizuoka, 〒410-03, JAPAN

Software is considered to have the two aspects, its application purpose in the application domain, and the implementation method in the program domain. For the purpose of effective reuse of software, these two aspects are decomposed by the models of the application and the implementation. These models are described by utilizing vocabulary and patterns of expression in each domain. The mapping in between these models is defined by introducing the new concept of the role paths which may identify the correspondences among patterns of expressions. An example is shown for the demonstration of this approach.

1．はじめに

近年になりソフトウェアの再利用技術への関心が高まってきた[1]．ソフトウェアの再利用による効果を上げるには，ソフトウェアの再利用率（全体行数に占める再利用した行数の割合）を飛躍的に向上させる必要がある．このためには，発見的な再利用ではなく，計画的な再利用を図ることが重要となってきた．

再利用技術の重要性が認識されるに従い，応用分野の用語や概念を分類整理して，その上での再利用を図る必要のあることも指摘されてきた[2]．一般に，何かを利用するという場合には，利用される実体と，利用する用途がある．利用される実体と，利用する用途とは，本来は独立な事柄である．本研究は，ソフトウェアに融合されている(a)用途目的の内容と，(b)計算機利用手段の実現の内容とを分離し，それぞれに部品化・再利用を図り，相互の対応関係をも扱おうとするものである．

特に，ここでは，ソフトウェアモデリング[3]の考え方を導入し，用途目的の側面からのソフトウェアの記述を“対象モデル”，また，計算機動作の実現内容の側面からのソフトウェアの記述を“実現モデル”と呼び，さらに，“対象モデルと実現モデルとの対応構造”について記述する方法を考察する．

2．ソフトウェアの対象モデルと実現モデル

2．1　基盤言語と解釈系の存在の仮定

ソフトウェアには，(1)応用対象分野の応用目的を果たす側面の情報，(2)計算機の装置挙動としての側面の情報との２つが反映している．それぞれの情報を表現する基盤言語B_1，B_2があると仮定し，それぞれ可能な記述の集合であると考える．またそれぞれに解釈系I_1,I_2があり，基盤言語B_1,B_2から，解釈領域D_1,D_2への写像であるとする．

$$I_1:B_1 \rightarrow D_1 \quad \text{および} \quad I_2:B_2 \rightarrow D_2 \qquad \cdots ①$$

例えば解釈系I_1を対象領域の専門家であるとし，解釈系I_2は特定の計算機システムと考えればよい．①は与えられるものとする．

2．2　モデル記述言語：パターンと基本語彙の抽出

基盤言語B_1,B_2の部分集合として，L_1,L_2を以下のように定め，これらをモデル記述言語と呼ぶ．モデル記述言語L_1,L_2の要素をモデル記述要素と呼ぶ．以下では対象モデル，実現モデルともに同様の構造をとることから，記号の節約のため，B_1,B_2を代表してBと書き，L_1 L_2を代表してL等とし，添字を省略する．

(1)　パターン(pattern)

Lの空でない部分集合E^j (j=1,2,...,n+1) があり，そのj=1 からn までの直積の部分集合E’を定義域としてE’から，値域E^{n+1}へのｎ引数関数ｐ（$x_1,x_2,\ldots,x_n$）が存在すると仮定する．そのような関数ｐの集合をＰと書き，パターン集合と呼ぶ．各々の関数ｐをパターンと呼ぶ．

パターン集合Ｐの要素であるｎ引数関数ｐの定義域を構成する集合E^j (j=1,2,...,n)を，Ｐの全てのｎ引数関数に関して和集合をとったものをE_{dn}とし，さらに，E_{dn}をすべてのｎについて和集合をとったものをE_dとし定義域集合と呼ぶ．またｐの値域であるE^{n+1}を，やはりＰの全てのｎ引数関数，全てのｎについて和集合をとったものをE_rとし値域集合と呼ぶことにする．このとき，以下が成立するものとする．

$$L = E_d \cup E_r \qquad \cdots ②$$

すなわち，パターンの集合Ｐとその定義域，値域の集合E_d，E_rとからモデル記述言語Ｌが規定されるものとする．

(2)　基本語彙（base word)

定義域集合E_dの要素の内で，値域集合E_rには登場しない要素のことを基本語彙と呼び，その集合全体をＷで表す．

$$W = E_d - E_r \qquad \cdots ③$$

2．3　基本役と視点

(1)　基本役（base role)

パターンｐのｎ個の引数には，それぞれに役割があると考え，各々の役割を識別するｒを以下のように，パターンｐと引数の位置ｋの組で与える．これを基本役と呼ぶ．

$$r = \langle p,\ k \rangle \qquad \cdots ④$$

基本役ｒによってパターンｐが指定されると，その引

数個数も決まるので，pの持つ他の基本役の集合も定まる．pの持つ基本役の集合をq（p）で表す．

$$q(p)=\{r_1, r_2, \ldots, r_n\} \quad \cdots ⑤$$

（ただしパターンpの引数個数はnで，$r_k=<p, k>$）

この場合，基本役r_kのパターンpを，z（r_k）＝pとし，基本役r_kからpを通じて得られるq（z（r_k））を特に，u（r_k）と書く．

(2) 視点（view）

q（p）の各々の要素r_k（k=1,2,...,n)に対して，結合可能なモデル記述要素e_k（k=1,2,...,n)を結合することを考える．この場合，パターンpの定義により，何らかのモデル記述要素e_{k+1}が得られる．従って，e_{k+1}をrとLの要素eとの組　r／e　の集合によって表すことができる．ただし，e_{k+1}，eそのものを用いるのではなく，以下のように，視点と基本役によって表現する．

（i）空集合ϕは視点である．
（ii）一つの基本語彙wだけからなる集合｛w｝は視点である．（w∈W）
（iii）基本役rと視点vとからできる組r／vは視点要素である．（視点そのものではない）
（iv）視点要素の集合v'があるとき，
v'＝｛$r_1/v_1, r_2/v_2, \cdots, r_n/v_n$｝，
このv'の全ての視点要素の基本役の集合をとる演算s（v'）が存在し，
s（v'）＝｛$r_1, r_2, \ldots, r_n$｝とすると，
v'の任意の視点要素r_k/v_kに対して，
s（v'）＝u（r_k）が成立し，かつ，v'のどの2つの視点要素r_k/v_k，r_j/v_jをとっても，$r_k \neq r_j$であり，さらに，v'の任意の視点要素r_k/v_kにおけるv_kには，その要素，さらにはその要素の要素を辿ったとしても自分自身v'が含まれることが無い場合には，v'は視点である．
（v）これら（i）(ii)(iii)(iv）によってのみ構成的に構成されたものが視点である．

(3) 視点グラフ(view graph)と役割パス(role path)

任意の視点vが与えられたとき，その視点要素r_i/v_iを視点vから視点v_iへの有向アークr_iがあるとし，各視点v_iをノードとすると，ラベル付有向グラフg(v)が得られる．これを視点グラフと呼ぶ．

視点グラフで，どこからもアークが来ていない（自分からはアークが出ていてよい）ノードをトップノードと呼び，どこへもアークが出ていない（自分へのアークが来ていてよい）ノードをターミナルノードと呼ぶと，視点vの視点グラフはトップノードがv自身で，ターミナルノードには，空集合ϕまたは，基本語彙（正確には基本語彙がただ一つから成る集合）が来る．以下では基本語彙一つだけからなる集合のことも，簡単のために混乱の無い限り，基本語彙と呼ぶ．

視点グラフでトップノードからターミナルノードへのパスを考える．これは，それぞれ辿るアークの順に，アークラベル，すなわち基本役をならべたもので表現できるので，これを以下のように表し，役割パスと呼ぶ．

$r_1/r_2/\cdots/r_k$

基本役が一段だけのパスの場合には，役割パスそのものが基本役になる．ターミナルノードが空集合かまたは基本語彙一つの集合であったことから，役割パスh_iを用いると，任意の視点vが次のようにも表現できる．

$$v=\{h_1/w_1, h_2/w_2, \cdots, h_n/w_n\} \quad \cdots ⑥$$

（ただし，w_iは基本語彙あるいは空集合）

2．4　モデル記述の生成

各々の基本役rはパターンとその中の引数位置とを，同時に識別しているため，rとLの要素eとの組r／eを指定することで，関数pの第k番目の引数に要素eを結合することに対応させることができる．また，視点と視点グラフとの関係は1対1であるので同一視できる．

任意の視点vが与えられた時，視点の定義より，視点が空集合でもなく，基本語彙一つの集合でもない場合には，必ず唯一のパターンpが対応している．そこで，任意の視点vからモデル記述mへの写像fを次のように設定する．

$$f: v \rightarrow m \quad \cdots ⑦$$

(1) vが空集合の場合には特定の$w_0 \in W$を与える．
(2) vが基本語彙｛w｝の場合には，wを与える．
(3) vが視点要素r／v'を持つ場合には，
（$r=<p, k>$とするとき），

視点ｖ’のモデル記述ｍ’（ｍ’＝ｆ（ｖ’））をパターンｐ（ｐ＝ｑ（ｒ））の第ｋ引数に結合し，その結果の記述ｍをｖのモデル記述とする．

$$m = p(m'_1, m'_2, \ldots, m'_n) \quad \cdots ⑧$$

ｆの定義により，モデル記述ｍの集合ＭはＬの部分集合になる．したがって，写像ｆの集合Ｆは，視点の集合Ｖからモデル記述の集合Ｍへの写像となる．

$$F : V \rightarrow M \quad (M \subseteq L) \quad \cdots ⑨$$

2．5　視点とモデル記述と解釈領域の関係

以上の定義により，基盤言語B_1，B_2からパターン集合P_1，P_2，および基本語彙集合W_1，W_2をそれぞれ抽出し，パターンと基本語彙を組み合わせることによって視点の集合V_1，V_2を構成し，かつ写像F_1，F_2とが定められると，図１のように，モデル記述の集合M_1，M_2を通じて，解釈領域D_1，D_2に対応づけられる．

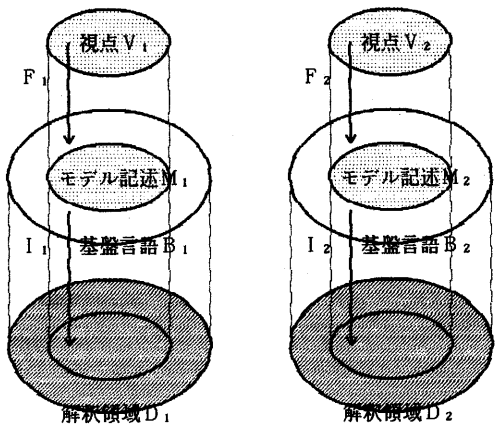
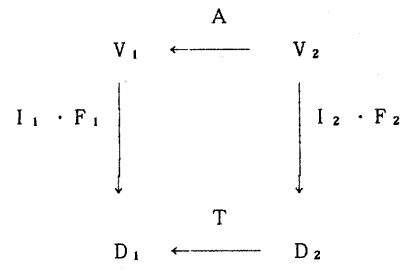


図１：視点とモデル記述と基盤言語

3．　対象モデルと実現モデルの対応構造

3．1　視点と解釈領域の間の可換性

対象モデルの記述はM_1の要素として得られるが，その源をたどると視点V_1に求められるので，ここでは対象モデルを視点V_1で代表させる．同様に，実現モデルも視点V_2で代表させる．いま，視点V_2(実現モデル)から視点V_1(対象モデル）への写像Ａを考える．実現モデルV_2は対象モデルV_1を用途・目的とするために実現されたものである，とするためには，解釈領域D_1においても解釈領域D_2からの写像Ｔがあり，Ａは図２を可換にするものでなければならない．



$$I_1 \cdot F_1 \cdot A(V_2) = T \cdot I_2 \cdot F_2(V_2) \quad \cdots ⑩$$

図２：対象モデルと実現モデルの対応構造

3．2　視点間の対応の構造

(1)　意味的対応

モデル記述言語L_1，L_2はB_1，B_2そのものではなく，その部分である．また，実際のモデル記述は，視点V_1，V_2から生成されるM_1，M_2であり，これらはパターン集合P_1，P_2のパターンの引数部分に，基本語彙集合W_1，W_2の語彙を結合したものである．

どんなL_1，L_2を選んでも，対象モデルと実現モデルの対応が付くというわけではなく，図１の可換図式を成立せしめる関係になければならない．このことから，ＡとＴにより対応関係をつけることの可能なL_1，L_2でなければならない．このためには，パターンおよび基本語彙自体が，すでに，ＡおよびＴにおいて対応されうる対象であることが必要になる．

すなわち，パターンおよび基本語彙の抽出を行う段階で，すでに，ＡとＴを前提して，応用分野の事柄と計算機動作の事柄とが意味的に対応のつけられる単位で切り出すことを前提とする．このことから，パターンや語彙には目的に応じた意味的なまとまりが要請される．

視点は基本語彙とパターンを組み合わせてできたものに相当することから，視点間の対応はパターンからパターンへの対応関係，および基本語彙から基本語彙への対応関係によって示される．以下でその構造について述べる．

(2) 基本語彙から基本語彙への対応

視点間の対応をとるための基本語彙から基本語彙への対応関係は，解釈領域D_1における個体と解釈領域D_2における個体との対応関係に基づく．個体の性質に関する条件は，同じ解釈領域内では，パターンの引数部分への適合条件や型の条件に関係する．パターンの引数部分に当てはまるべき個体は，前述のE^jの集合に帰属する基本語彙で指示されねばならない．

(3) パターン間の対応

パターンからパターンへの対応は，基本役から基本役への対応として表すことができる．ただし，常にパターン間の対応がとれるとは限らない．このため，いくつかのパターンや基本語彙を組み合わせた合成パターンを，対象モデル側でも作り，実現モデル側でも作ることによって，合成パターン間の役割パス同志の対応をとり，視点間の対応を図ることが必要となる．

合成パターンは，それに丁度対応するパターンがPにすでに用意されていたように組み入れることにすれば，対象モデル側におけるパターン集合内での，パターン間の対応構造が存在する．同様に実現モデル側におけるパターン集合内でも，パターン間の対応構造が存在する．それらのパターン間対応も，役割パス同志の対応関係として規定することができる．

4 例題

4．1 プログラムソースコード

図3はプログラム言語 Ada® を用いたプログラム例である．このプログラムの用途や目的については次節で述べることにし，ここではあえてその用途や目的を述べない．その理由は，プログラムソースコードが本質的に指示する意味内容は，その用途や目的ではなく，計算機の動作を指示するものであることを示したいからである．

```
procedure X(N:INTEGER;A,B,C:CHARACTER) is  -- (a)
    begin                                  --(b)
      if N>0 then                          --(c)
        X(N-1,A,C,B);                      --(d)
        PUT(A);PUT(B); NEW_LINE;           --(e)
        X(N-1,C,B,A);                      --(f)
      end if;                              --(g)
    end X;                                 --(h)
```

図3 ：プログラム実体例

この例では，わざと変数名称や手続き名称に応用目的分野の名称を用いていない．通常，プログラミングの際には，分かり易い名称を用いるべきとされるが，対象モデルと実現モデルとを区別した場合には，分かり易さにも2つの側面があることになる．実現モデルとして見れば，応用目的分野の用語を一切用いていなくとも，ソースプログラムから実現分野の解釈が可能である．

例えば，図3の(a)(b)(h)から，これは4つの引数を持つ手続きX であり，N が整数型,A,B,Cが文字型の変数であることが分かる．また，(c)から(g)までが手続きX の本体部分であり，それが丁度一つのif文で構成されていることも分かる．条件N>0 の時にのみ(d)(e)(f)の部分が実行され，成立しない場合には，何もせずに終わることが分かる．

また(d)(f)はX の再帰呼び出しに相当し，引数N-1 が用いられていることから，順次N の値が減じられていく過程でプログラムは停止することが分かる．(e)の部分では標準出力に文字変数A,B の値を書き出し，改行していることも分かる．

4．2 対象モデル

(1) 例題の問題記述と解決方法

図3のプログラムは実はハノイの塔の問題解決用のプログラムである．ここでは，問題を簡単に述べておく．

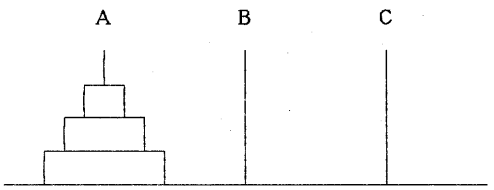


図4 ：ハノイの塔の初期状態

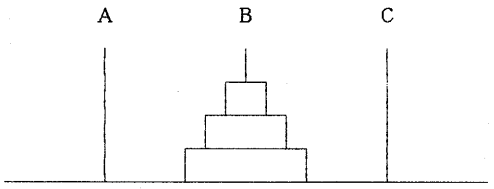


図5 ：ハノイの塔の最終状態

問題： 3つの棒A，B，Cが地面の上に立っている，棒Aだけには N枚の円盤が刺さっている．円盤は全て大きさが異なり，図4のように，下ほど大きいように順に

刺さっている．これを図５のように円盤を全て棒Ｂに移動したい．ただし，一回の移動では一枚の円盤の移動しかできないものとし，棒以外の所には刺せないし置けないものとする．また，小さな円盤の上に大きな円盤をのせることはできないとする．どのような手順で円盤を移動すればよいかを求める．

この問題に対する解決方法を，例えば，図６のように考えることにする．

> Ｎ枚の円盤を棒Ａから棒Ｂに，棒Ｃを用いて移動する手順を以下のように考える．ただしＮ＞０とする．
> ｉ．Ｎ－１枚の円盤を棒Ａから棒Ｃに，棒Ｂを用いて移動する．こうすれば，
> ⅱ．棒Ａの一番大きな円盤を棒Ｂに移動できる．
> そうした後は，
> ⅲ．Ｎ－１枚の円盤を棒Ｃから棒Ｂに，棒Ａを用いて移動すると目的が達成できる．

図６：対象の解法記述例

(2) 基本語彙と基本パターンの抽出

対象モデルは，実現モデルとの対応において捉えられるべきである．全く任意の対象モデル記述から，実現モデルへの対応問題を扱うのではなく，実現モデルから，$I_1 \cdot F_1 \cdot A$を経由した結果と，$T \cdot I_2 \cdot F_2$ を経由した結果とが等しくなるような対象モデルV_1 が考えられねばならない．そのためには，対象領域における解法記述をモデル化しなければならない．ここでは，以下のようなパターン集P_1 と基本語彙集W_1 の抽出を行う場合を例とする．

$P_1 = \{p_1, p_2, p_3, p_4\}$　ただし，

p_1="〔注目する円盤の枚数〕枚の円盤を棒〔複数円盤の移動元の棒〕から棒〔複数円盤の移動先の棒〕に，棒〔作業用の棒〕を用いて移動"
p_2="棒〔円盤一枚の移動元〕の一番大きな円盤を棒〔円盤一枚の移動先〕に移動"
p_3="〔目的の動作〕する手順を以下のように考える．ただし〔前提条件〕とする．〔その手順〕"
p_4="ｉ．〔第１動作〕する．こうすれば，
ⅱ．〔第２動作〕できる．
そうした後は，
ⅲ．〔第３動作〕すると目的が達成できる．"

ここで〔 〕部分は基本役を示す役割名称であり，パターンの引数部分を一意に指示するものとする．

これらのパターンの引数位置に期待される基本語彙集W_1 は次のようになる．

W_1 ＝ {"N","N-1","N>0","A","B","C" }

"N-1" や"N>0" はさらにパターンや基本語彙に分解することができるが，ここでは詳細を割愛し基本語彙とする．ただし注意すべきは，W_1 の要素には対象モデル内で定められるべき型がある点である．この例では，A,B,C が棒としての型を持ち，N-1 やN が円盤の枚数としての型，およびN>0 は前提条件としての型を持つ．

パターンの抽出に当たっては，実現の対象として，例えばp_1およびp_2を切り出した理由が存在し，円盤移動に関するまとまった動作単位として対応づけたい動機があったと考える．また，p_3は目的・前提条件・手順の相互の関係を示す意図があり，p_4 は第１動作によって第２動作への道が開け，かつ，第２動作が出来たあかつきには第３動作が可能となる，という３つの動作相互間の因果的関係を明確にする目的がある．

パターンの各引数にも期待される型がある．例えば，"注目する円盤の枚数" の位置には円盤の枚数の型を持つ基本語彙が結合されねばならない．このような条件はパターンがｎ引数関数であるとした時点で定義されるものと考える．

(3) 対象モデル

対象モデルとして視点v_1 を次のように考える．

v_1 ＝｛目的の動作／v_{11}，前提条件／N>0，
　その手順／v_{12}｝
v_{11}＝｛注目する円盤の枚数／N,複数円盤の移動元の棒／A,複数円盤の移動先の棒／B,作業用の棒／C｝
v_{12}＝｛第１動作／v_{13}，第２動作／v_{14}，
　第３動作／v_{15}｝
v_{13}＝｛注目する円盤の枚数／N-1,複数円盤の移動元の棒／A,複数円盤の移動先の棒／C,
　作業用の棒／B｝
v_{14}＝｛円盤一枚の移動元／A,円盤一枚の移動先／B｝
v_{15}＝｛注目する円盤の枚数／N-1,複数円盤の移動元の棒／C,複数円盤の移動先の棒／B,
　作業用の棒／A｝

ここで，対象モデルの全容を示すため，これらの全体を次のようなV_1とする．

$V_1 = \{v_1, v_{11}, v_{12}, v_{13}, v_{14}, v_{15}\}$

視点v_1におけるそれぞれの役割名称は，パターンとその引数位置とを規定することから，特にパターン名称を表に出す必要は無い．

視点v_1から，パターン集P_1基本語彙集W_1を用いて，比較的単純な引数結合により図6のモデル記述が得られる．ここでの基盤言語B_1は日本語である．

4．3　実現モデル

実現モデル側の基盤言語B_2として，特定の実行環境を想定した時の Ada®を採用したとする．この例に役立てられるパターンとしては以下のようなP_2を考える．

$P_2 = \{p_5, p_6, p_7, p_8, p_9\}$　　ただし，

```
p5="procedure 〔手続き入口〕 is
    begin
      if〔手続き条件〕then
        〔手続き内容〕;
      end if;
    end 〔手続き名〕; "
p6="〔4 引数1対3型入口動作名〕(
    〔4.1 入口引数〕:〔第1型〕;〔4.2 入口引数〕
     ,〔4.3 入口引数〕,〔4.4 入口引数〕:
      〔第2型〕)
p7="〔4 引数動作名〕(〔4.1 引数〕,〔4.2 引数〕,
    〔4.3 引数〕,〔4.4 引数〕)
p8="〔第1動作〕;〔第2動作〕;〔第3動作〕"
p9="PUT(〔第1印字文字〕);PUT(〔第2印字文字〕)
    ; NEW_LINE"
```

p_5は手続きの枠組みを示す一つのパターンであり，手続き条件成立時のみ手続き内容を実施し，不成立時には何も実行せずそのまま呼び出し元へ帰着するまとまりを示している．p_6とp_7は，ともに4引数の手続き入口の形式，および手続きの呼び出しを示すパターンであり，特にp_6は型宣言について4引数の内の後半の3個を同一の型として宣言するパターンである．p_8は3つの動作を一列に並べるパターンである．p_9は2つの文字を印字した後に改行するパターンに相当する．

ここでは詳細は略すが，言語上の特殊性から，役割パス間に制約条件が存在する．例えば，手続き入口における入口動作名は，同じ手続きの手続き名に相当しなければならない．このような制約条件は役割パスを用いて次のように述べられる．

「パターン集P，基本語彙集Wを用いて構成できる任意の視点vについて，次のような$w_1 \in W$，$w_2 \in W$，が存在すると仮定する．
(v∋　手続き入口／4 引数1対3型入口動作名／w_1)
∧(v∋　手続き名／w_2)
このときは常に，$w_1 = w_2$ でなければならない」．

同様に，それぞれの引数に入るべき対象の型に関する制約条件もあるが，ここではその扱いについての詳細を割愛する．注意すべきは，これらの制約条件が，役割パスを用いて表現できると同時に，これらの制約条件が対象モデル側の制約条件とは独立に決まっていることである．

次に，基本語彙集合W_2としては以下を考える．

$W_2 = \{$"N","N-1","N>0","A","B","C","INTEGER",
"CHARACTER","X" $\}$

この時，実現モデル記述としての図3のソースプログラムは以下の視点v_2から得られる．

$v_2 = \{$手続き入口／v_{21}，手続き条件／N>0，
手続き内容／v_{22}，手続き名／X $\}$
$v_{21} = \{$4 引数1対3型入口動作名／X，
4.1 入口引数／N,第1型／INTEGER，
4.2 入口引数／A,4.3 入口引数／B，
4.4 入口引数／C,第2型／CHARACTER $\}$
$v_{22} = \{$第1動作／v_{23}，第2動作／v_{24}，
第3動作／$v_{25}\}$
$v_{23} = \{$4 引数動作名／X,4.1 引数／N-1,4.2 引数／A，
4.3 引数／C,4.4 引数／B $\}$
$v_{24} = \{$第1印字文字／A,第2印字文字／B $\}$
$v_{25} = \{$4 引数動作名／X,4.1 引数／N-1,4.2 引数／C，
4.3 引数／B,4.4 引数／A $\}$

V_1を得たのと同様に，ここでもV_2として以下の集合を考える．

$V_2 = \{v_2, v_{21}, v_{22}, v_{23}, v_{24}, v_{25}\}$

4．4　対象モデルと実現モデルとの対応

対象モデルV_1と実現モデルV_2間の対応は役割パス同志の対応と基本語彙間の対応とによって行う．下記の←印の左辺では対象モデル側の役割パスを，右辺では実現モデル側の役割パスを示し，役割パスの対応全体の集合をArと表している．

Ar＝｛（目的の動作／注目する円盤の枚数
←手続き入口／4.1 入口引数），
（目的の動作／複数円盤の移動元の棒
←手続き入口／4.2 入口引数），
（目的の動作／複数円盤の移動先の棒
←手続き入口／4.3 入口引数），
（目的の動作／複数円盤の移動先の棒
←手続き入口／4.4 入口引数），
（前提条件←手続き条件），
（＊／第１動作←＊／第１動作），
（＊／第２動作←＊／第２動作），
（＊／第３動作←＊／第３動作），
（その手順／＊／注目する円盤の枚数
←手続き内容／4.1 引数），
（その手順／＊／複数円盤の移動元の棒
←手続き内容／4.2 引数），
（その手順／＊／複数円盤の移動先の棒
←手続き内容／4.3 引数），
（その手順／＊／作業用の棒
←手続き内容／4.4 引数），
（＊／円盤一枚の移動元←＊／第１印字文字），
（＊／円盤一枚の移動先←＊／第２印字文字）｝

基本語彙の対応については，この例では記号上ですでに対応をとっているので省略する．ただし，記号の指示する対象は解釈領域が違うので，正確にはやはりN←Nのように対応をつけなければならない．この場合にはたまたま同様な記号をあらかじめ付けてしまったので，紙面の節約のため，その対応関係を明示することを省略しただけである．その省略した対応関係をAwとする．

Arから分かるように，役割パス間の対応は，V_{13},V_{14},V_{15},V_{23},V_{24},V_{25}等の個々の視点毎の対応ではなく，パターン間の対応を示しており，少ない対応規則により，全体の対応関係をとっている．Arの中の＊印は，ワイルドカードのような働きをし，そこに任意の役割パスが挿入されても対応がとれることを示す．

実現モデルのパターンや基本語彙の中の，第１型，第２型，およびそれらに結合されたINTEGER,CHARACTER など，対象モデルには登場せず，実現モデルにのみ登場しているいくつかの基本語彙と基本役がある．これらは，実現モデルを完成するにあたり必須のものであるが，対象モデルとの対応構造においては登場しない．

この例では，第１印字文字や第２印字文字が文字型をとる都合上CHARACTER 型が選ばれており，また，N-1 などの演算が含まれる都合上INTEGER 型が選ばれている．これらには多少の任意性が残されており，基本的には実現モデルの完成にあたっての付属的な意思決定事項である．ただし，実現モデルの解釈領域にとっては本質的であるので，それらの意思決定についても記述対象とする必要がある．これらの付属的意思決定事項をAsとする．

以上により，対象モデル側については（P_1，W_1，V_1），実現モデル側については（P_2，W_2，V_2）によりそれぞれ特徴づけられ，（Ar，Aw）により相互に対応づけられ，さらに，実現モデル側には付属的意思決定事項Ａｓが加わって全体が構成されている．

5．おわりに

本研究では，ソフトウェアの再利用を考える場合に，ソフトウェアを対象モデルと実現モデルに分けて，それぞれを再利用することを提唱し，かつ，それらの間の対応関係について考察した．特に視点と解釈領域との間の可換性の具体的な意味について考察し，簡単な例題を示した．例そのものは再利用に程遠いが，モデル間の基本的な対応構造を示している．パターン，基本役，基本語彙，視点，役割パス対応，基本語彙対応，付属的意思決定対象，制約条件等について，それぞれを再利用容易なものとし，自動化することが，再利用を効果的に推進するものと考えられる．

参考文献

〔１〕Jones,T.C.:Reusability in Programming: A Survey of the State of the Art; IEEE Trans. on Software Engineering, SE-10, No,5, pp.488-493(1984)
〔２〕Prieto-Diaz,R., and Freeman,P.: Clasifying Software for Reusability, IEEE Software, Vol.4,No.1,pp.6-16.(1987)
〔３〕小林要，木村高久，織田充：ソフトウェアモデリングにおける役割付与の構造について，情報処理学会，情報学基礎研究会資料６－２（1987.9.21）