ソフトウェア工学 32－1
（1983. 9. 28）

ソフトウェア開発過程の記録法の一事例

今泉 恵美子　　落水 浩一郎
（静岡大学　工学部　情報工学科）

1．はじめに　本考察では、プログラミング言語の独習システムを例にとり、要求定義からプログラム設計への変換過程を分析し、その記録法を論じる。

ここで、要求仕様は、W. Kent による情報の記述法[1]に準じて表現し、設計仕様は、C.A.R. Hoare による抽象プログラム記法[2]を採用するものとする。

2．研究の背景　ソフトウェア開発の各局面では、前局面における設計上の決定事項を入力仕様とし、その局面で生じる種々の制約に基づいて変換され、その結果出力仕様が形成される。この過程が何段階も繰り返されることにより、最終的にソースコードが生産される。このような作業形態では、多くの場合出力仕様のみが残される傾向となり、保守に際して次のような問題を生じる。保守とは、外界の変化に基づき、究極的にはソースコードを変更する作業である。このためには、外界の変化に対応するソースコードの部分を特定し、その部分に対する変更が与えるプログラム全体に対する影響を適切に判断する必要がある。この作業のための手掛りは開発時に作成される文書類であるが、前述のような形態で残される文書では、外界の変化をソースコードまで追跡する作業は困難である。

このような問題の解決のため、ソフトウェア開発においては、一貫して同種の決定（すなわち、どのような要請、制約、決定に基づいて、何を考慮した結果、何が決定されたか）が繰り返されているという視点に立ち、変更要求から対応するソースコード部分を特定する筋道を容易にとり出せるような開発過程の記録法（Decision－map法）の開発が本研究の最終的な目標である。

このような視点から開発過程の記録法を考える場合の要点は、変換過程である。開発過程における重要な変換には2つある。要求定義から抽象プログラムへの変換と、抽象プログラムから効率を考えた具体プログラムへの変換である。本考察では、その前者の場合を一つの事例に基づいて論じる。

3．事例研究　プログラミングにそれほど習熟していない時期に新しい言語を習得する場合、したいことを言語で表現できない、エラーの原因がわからない等の事柄がよくある。その主な理由は、言語の要点がつかめていないこと、マニュアルを引くカが充分ではないことの2点である。このような問題現象を分析し、以下のようにシステムの機能を定めた。

(1) システムが有する情報構造に関する要請　「言語を習得する過程でまとめられていく」そのような視点から編成された文法上の知識を保持すること。

(2) マン・マシンインタフェースに関する要請　システム利用のために特別な知識を必要としないこと。

ただし、プログラミング言語と

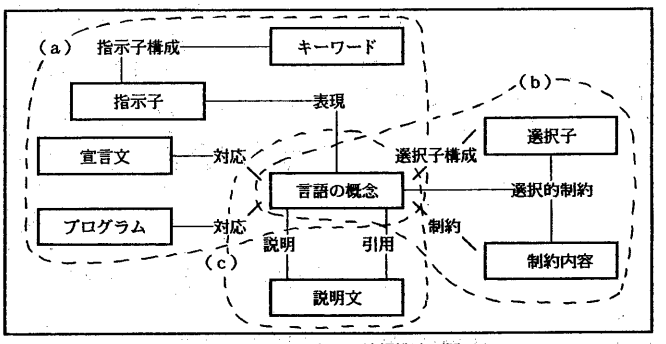

図1　システムの有する情報構造の概要

しては、FACOM 230-45S OSⅡ PL/I のファイル入出力関係に限定した。

3．1　要求定義　以下に要求定義の結果を示す。ただし、紙面の都合上、一部のみを示し、説明する。

3．1．1　システム機能要請(1)に関する部分　まず、図1に示す情報構造の作成過程を説明する。PL/I でのファイル入出力が理解しにくい点は、特にレコード入出力の場合、ファイル属性、ENVIRONMENT オプション（以下キーワードと呼ぶ）のうち、どのような入出力を行う場合にどれを指定すれば必要充分なのかということが簡単な説明では理解しにくい点にある。初心者に対するこのような情報の提示の仕方としては、

(1)指定されたキーワードに対応するプログラムまたはファイル宣言文を示す（図1(a)部）

(2)指定されたキーワードについて、意味および詳細な規約・省略規則等を示す（図1(b)，(c)部）

がある。

ここで次のような問題がある。(1)の場合、どのような例題プログラムおよびファイル宣言文を用意し、どのキーワードに対してどの例題プログラムまたは宣言文を示せばよいのか、(2)の場合は、各キーワードについてどの範囲のことを答える必要があるか。この問題を分析しつつ図1(a)部の詳細を定めたものが図2である。まず、キーワードの中で、省略時の解釈を適用せずに、何を指定すればそこまでの組合せは正しく、しかも入出力文との対応が正しいか否かを判断できるかを考えてみる。ストリーム入出力の場合は、図2の(A)部に示すように3つになる。レコード入出力の場合、この条件に含まれるキーワードの役割を分類してみると、図2の(B)部に示すように、ファイル機能、アクセス方法、データ転送方法、ファイル編成法の4種類に分かれる。この4種類に含まれる要素の組合せによりレコ

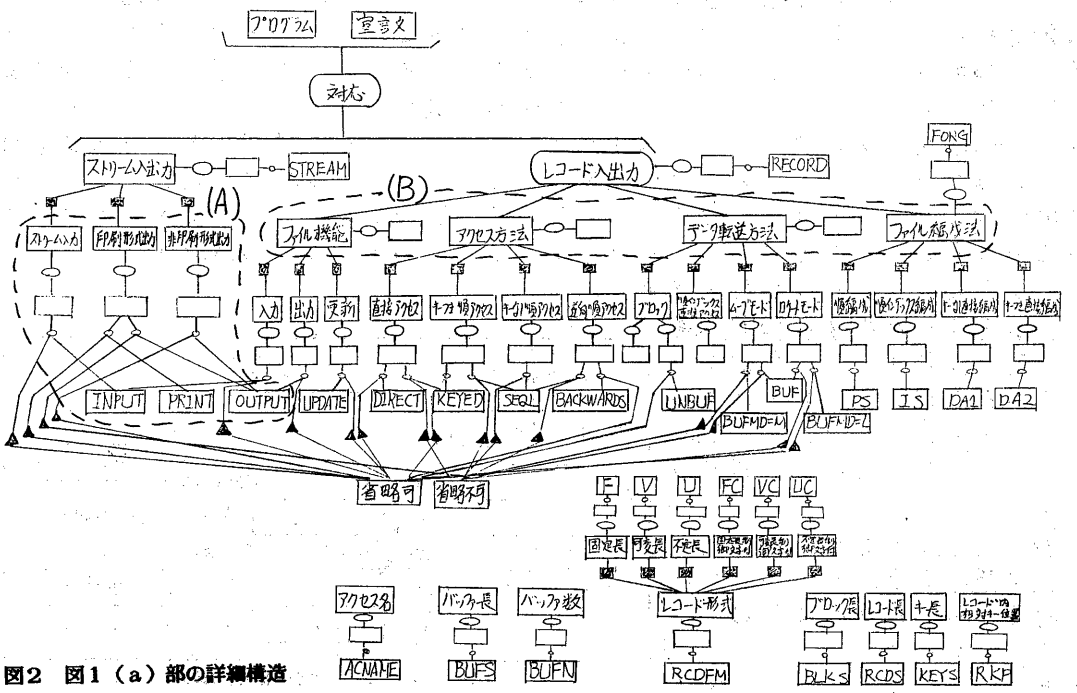

図2　図1（a）部の詳細構造

ード入出力の要素が定義される。そこで、この種類をレコード入出力の構成要素型、そこに含まれる要素をレコード入出力の構成要素とする。各キーワードが宣言文で果している役割をこのように考えると、それぞれが表わす（言語上の）概念を考え、キーワードがそれを表現する（図２記号○）と考えた方がよい。複数のキーワードを使って表現される概念があるために、指示子を導入してそれを表わすことにする。以上により、例題プログラムおよび宣言文は、ストリーム入出力とレコード入出力の要素対応に用意する。ここまでに含まれなかったキーワードについても同様に概念を与え、指示子との間に関連「表現」（図２記号○）を与え、キーワードが指示子を構成する（図２記号◦）ようにする。

編成の具体例を図３に基づいて示す。直接アクセスは、アクセス方法に所属（記号■）し、キーワードDIRECT、KEYEDで構成（記号◦）される指示子によって表現（記号○）される。DIRECTは省略できない（記号▲）が、KEYEDは省略できる（記号△）。

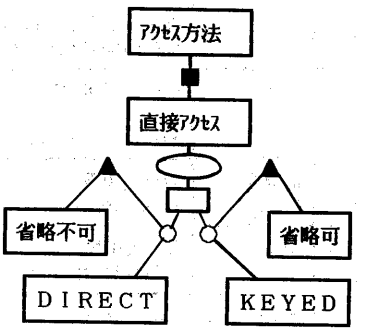

図３　関連「指示子構成」、「表現」、「所属」の役割

各概念は要素型と要素に分れ、レコード入出力は関連となる。要素型となる概念は、ストリーム入出力、ファイル機能、アクセス方法、データ転送方法、ファイル編成法、レコード形式、ブロック長、レコード長、キー長、レコード内相対キー位置、バッファ数、バッファ長、アクセス名である。

次に図１(b)部の詳細を定めた。図１(b)部の詳細を定めるということは、ストリーム入出力の要素およびレコード入出力の構成要素（決定要素）と、それらに含まれなかった概念（従属概念）との間に関連をつけることである。基本的には決定要素が従属概念を必要とする関連がある。このとき、従属概念は要素型である。また、ファイル編成法は、ストリーム入出力の要素に対しては必要とされることになる。各要素型は、それ自身が、または必要とされる環境によって、省略規則、制約、選択的制約を受ける。

図４にその一部を示し、説明する。順編成はブロック長を必要とする。このとき、ロケートモードならば、ブロック長は偶数でなければならない。ブロック長は、固定長のとき、nxRCDSでなければならない。ブロック長は省略できない。

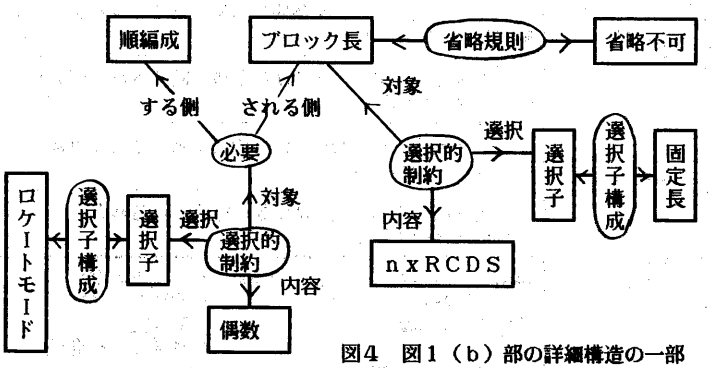

図４　図１（ｂ）部の詳細構造の一部

最後に各概念には、意味を説明するための説明文が必要となる。説明文で他の概念を引用して説明する場合には、さらにそれについて説明を行うために、説明文と引用されている概念の間に関連「引用」をつけておく必要がある（図１(c)部）。

３．１．２　考察（情報構造の視点を定めている関連群）　要求定義中に現れた関連をすべて列挙すると表１のようになる。要求定義の情報の中で最も重要な役

	関連名	role名（domain名）；…
1	省略規則	対象（要素型／必要link／指示子構成link）；内容（省略の内容） single,optional　multiple,mandatory
2	制約	対象（要素型／必要link）；内容（制約の内容） single,optional　multiple,mandatory
3	説明	対象（キーワード／指示子／概念）；内容（説明文） 指示子のみoptional 他はmandatory,single　single,mandatory
4	引用	出典（説明文）；対象（概念／キーワード） multiple,optional　multiple,optional
5	選択条件	対象（表現link）；内容（選択子） single,optional　single,optional
6	所属	型（要素型）；構成要素（要素） multiple,optional　single,mandatory
7	選択子構成	対象（選択子）；内容（要素） multiple,mandatory　multiple,optional
8	表現	内容（概念）；記号（指示子） multiple,mandatory　single,mandatory
9	必要	必要とする側（要素）；必要とされる側（要素型） multiple,optional　multiple,optional
10	選択的制約	対象（要素型／必要link）；選択（選択子）；内容（制約の内容） multiple,optional　multiple,mandatory　multiple, mandatory
11	指示子構成	対象（指示子）；内容（キーワード） multiple,optional　multiple,optional
12	レコード入出力	ファイル機能（ファイル機能に所属で結合される要素）；multiple,mandatory
		アクセス方法（アクセス方法に所属で結合される要素）；multiple,mandatory
		データ転送方法（データ転送方法に所属で結合される要素）；multiple,mandatory
		ファイル編成法（ファイル編成法に所属で結合される要素）multiple,mandatory
13	対応	索引（レコード入出力／ストリーム入出力）；内容（入出力文／例題プログラム／宣言文／宣言文の省略形） それぞれのdomainに対して single,mandatory　single,mandatory

表1　情報構造の視点を定めている関連群

割をになっているのは関連である。すなわち、「初心者が言語を習得するためには、どのような視点からデータがまとめられるべきか」に対する直接の解答が関連により与えられる。表1において、10は3項関連，12は4項関連、他は2項関連である。domain名が並記してある関連は、その関連の要素に、それらのうちのいずれか1つの要素が参加していることを意味する。

3．1．3　システム機能要請(2)に関する部分

問合せの内容は、3．1．1(1)，(2)に示したように、例題プログラムの提示、宣言文の提示、キーワードの意味・詳細な規約・省略規則等の提示であり、それぞれの対象を指定するためにキーワードを用いる。したがって、システムに対する入出力は、

Ⅰ．キーワードを入力し、対応する例題プログラムを選択し出力する。

Ⅱ．キーワードを入力し、対応する宣言文を選択し出力する。

Ⅲ．キーワードを入力し、キーワードからたどられる関連を出力する。

となる。Ⅰ，Ⅱの選択方法を表2のAに示し、Ⅲの関連の選択方法を表2のB，Cに示す。さらに、Ⅲにより出力された説明文の中で引用されている用語に対しても説明を与えるために、Ⅳ．Ⅲにより引用された概念に対して、表1のCに示される方法で関連を選択し、出力する

A．例題プログラムまたは、ファイル宣言文を示す場合
① 「ストリーム入出力」または「レコード入出力」に、関連「指示子構成」、「表現」で直接結合されるキーワードが指定された場合、関連「所属」で「ストリーム入出力」に結合される要素（以下、ストリーム入出力の要素）または関連「レコード入出力」のすべての要素がそれぞれ選択される。
② 「レコード入出力」の構成要素型に、関連「指示子構成」、「表現」で直接結合されるキーワードが指定された場合、関連「レコード入出力」のすべての要素を選択する。
③ 「ストリーム入出力」の要素または「レコード入出力」の構成要素に、関連「指示子構成」、「表現」で直接結合されるキーワードが指定された場合、前者はその要素、後者はそれを構成要素とする関連「レコード入出力」のすべての要素が選択される。
④ 関連「必要」の「必要とされる側」に参加している要素型である概念に、関連「指示子構成」、「表現」で直接結合されるキーワードが指定された場合、その関連要素の「必要とする側」である要素が「ストリーム入出力」の要素である場合はその要素、関連「レコード入出力」の構成要素となる場合は、それを構成要素としている関連「レコード入出力」のすべての要素が選択される。
⑤ 関連「必要」の「必要とされる側」に参加している要素型である概念（要素型ａ）に対して、関連「所属」で結合される要素である概念（要素ｂ）に、関連「指示子構成」、「表現」で直接結合されるキーワードが指定された場合、要素型が関連「所属」で結合されている要素である概念（要素ｃ）により、次のように選択する。要素ｃが「ストリーム入出力」の要素である時、要素ｃを選択する。　要素ｃが関連「レコード入出力」の構成要素である時、要素ｃを構成要素としている「レコード入出力」のすべての要素を選択する。　ただし、要素型ａと要素ｃの間の関連「必要」が関連「選択的制約」の「対象」となっている時、その「制約の内容」が要素ｂを指定していないときは要素ｃを選択しない
　１つのキーワードが①－④に同時に該当するものがある場合、それぞれで選択された要素の和集合とする。複数のキーワードが指定されている時は、それぞれのキーワードで選択された要素の積集合をとる。
　以上により選択された「ストリーム入出力」、「レコード入出力」の要素に関連「対応」で結合される例題プログラム、宣言文を出力の対象とする。
B．キーワードが表わす概念を示す場合
① キーワードから関連「説明」で結合される「説明文」と、キーワードから関連「指示子構成」、「表現」をたどって得られる概念を対象とする
C．概念に関する説明をする場合
① 概念から関連「表現」で結合される指示子に対しては、
ａ．「表現」が関連「選択」に参加していれば、その「選択内容」
ｂ．指示子が関連「構成」に参加していれば、そのキーワードとキーワードが関連「説明」で結合されている説明文および「構成」が関連「省略規則」に参加していればその省略内容
ｃ．指示子が関連「構成」に参加していなければ、指示子が関連「説明」で結合される説明文を対象とする。
② 概念に関連「説明」で結合される説明文と、概念が要素である場合は、関連「所属」で結合される要素型；要素型である集合は、関連「所属」に参加していれば、その要素に、概念が「レコード入出力」である時はその要素を対象とする。
③ 概念が関連「レコード入出力」の構成要素である時、それを構成要素とする関連「レコード入出力」の要素を対象とする。
④ 概念が関連「制約」に参加している時、その制約内容を対象とする。
⑤ 概念が関連「必要」に「必要とする側」で参加しているとき、その「必要とされる側」を対象とする。またその関連要素が、関連「制約」、「選択的制約」、「省略規則」に参加する時、それらの関連要素を対象とする。
⑥ 概念が関連「必要」に「必要とされる側」で参加しているとき、その「必要とする側」を対象とする。またその関連要素が、関連「制約」、「選択的制約」、「省略規則」に参加する時、それらの関連要素を対象とする。
⑦ 概念が関連「省略規則」に参加している時、省略の内容を対象とする。
⑧ 概念が関連「選択子構成」に参加している時、その選択子が参加している関連「選択的制約」の要素。

表２　情報（関連）操作の定義

ことが必要である。

ここで、Ⅰ，Ⅱの場合は用意された例題プログラムおよび宣言文を出力するのみである。Ⅲ、Ⅳについては、出力の方法を以下のように工夫した。各関連に対し、たどり方に従って関連を表現する文章を与え、関連をたどって得られる概念名、説明文等をそこにうめこむようにした。これにより、概念に伴う文法上の種々の知識をまとめた形で提示するとともに、関連する近い概念を整理した形で与え得る。

3．2　変換過程　3．1．1の結果得られた関連群（表１）と、それを構成する要素（図２にその一部を示す）がシステムに対するデータ設計であり、3．1．3の結果得られた操作（表２）およびⅠ，Ⅱ，Ⅲに示す入出力がシステムに対する機能設計である。以下にこれを抽象プログラムに変換する過程を述べる。

3．2．1　データ型の導出　3．1．1の結果得られた関連群およびその構成要素に対して、データ型を与えるものを定め、それから導出されるデータ型を定義する。本例の場合は、データを効率的に検索することがその目標となる。このような目標のもとに、関連を分類し、以下の方針でデータ型に変換した。

①一方向にたどる関連　「省略規則」、「制約」、「説明」、「引用」、「選択条

件」がこれに該当する。この場合は、たどるときの始点となる側に終点となる側を示すデータ型を与える。たとえば、「制約」は「対象」の側から「内容」の側へたどられる。「対象」になるものは、要素型である概念、関連「必要」の要素であり、この2つは省略内容を示すデータ型をもつことになる。

②双方向にたどられるが、他の関連に参加しない関連　「所属」、「選択子構成」がこれに該当する。この場合は、関連に参加している双方の要素にたどった結果を示すデータ型を与える。たとえば、「所属」は、「型」の側に「構成要素」の側である概念の集合を示すデータ型を、「構成要素」の側に「型」の側である概念を示すデータ型を与える。

③双方向にたどられるが、一方の側が1対1で参加している関連　「表現」がこの場合に該当する。「表現」の要素は関連「選択条件」に参加するが、これは①により「表現」が「選択条件」の「内容」をもつことになっており、さらに、これは②により概念の集合を示すデータ型になっている。指示子は「表現」に対して1対1で参加するので、このデータ型は指示子に対して与えることができる。概念に対し、指示子の集合を示すデータ型を与えることにより、「表現に対してはデータ型を与える必要はなくなる。

④双方向にたどられ、多対多対応である、または多項関連である　「必要」,「選択的制約」、「指示子構成」がこれに該当する。この場合は関連に対しデータ型を与え、関連に参加する要素に関連要素へのインデックスをつける。

⑤その他　「レコード入出力」と「対応」の場合、「対応」にはストリーム入出力の要素も参加することから、「対応」を残し、「レコード入出力の構成要素．ストリーム入出力の要素に「対応」へのインデックスをつけ、「対応」には、それぞれの要素を示すデータ型を与える。「対応」の「内容」は各要素とも1回必ず「対応」に参加し、「索引」の側も3種類の要素型のそれぞれに対して1対1となるので、「索引」を示すデータ型、「内容」の各要素型の要素を示すデータ型の直積として表わされる。

概念は、要素型、要素、「ストリーム入出力」・「レコード入出力」に対して別々のデータ型を与える。参加する関連と参加の仕方が異なることによる。これにより3つの型の概念が共に参加する関連を表現するデータ型は、概念の型を区別するようにする必要がある。以上により表3に示すデータ型が得られる。

```
type  キーワード情報 =( pkos: 構成情報ix; sets: 説明文ix);
type  構成情報 =( kw: キーワード; siji: 指示子; shor: 省略内容ix);
type  指示子情報 =( pkos: powerset 構成情報ix; gn: 概念子; sent: powerset概念;
                    sets: 説明文ix);
type  概念入出力形式情報 =( sets: 説明文ix; siji: 指示子);
type  概念型情報 =( sets: 説明文ix; siji: 指示子; yoso: powerset 概念;
                    ny: 入出力形式型; seig: 制限内容ix; shor: 省略内容ix;
                    ssei: powerset 選択的制約ix; hitk: powerset 必要ix);
type  概念要素情報 =( sets: 説明文ix; siji: 指示子; kata: 概念;
                      nyix: 入出力形式ix; seig: 制限内容ix; shor: 省略内容ix;
                      hity: powerset 必要ix; sent: powerset 選択子ix);
type  必要情報 =( yoso: 概念; kata: 概念; seiy: 制限内容ix; shor: 省略内容ix;
                  ssei: powerset 選択的制約ix);
type  選択的制約 =( tais: 対象; sent: 選択子ix; seiy: 制約内容ix;);
type  選択子情報 =( pgai: powerset 概念; ssei: powerset 選択的制約ix);
type  入出力形式情報 =( pgai: powerset 概念; seng: 宣言文ix;
                        shsn: 省略形宣言文ix; prog: プログラムix);
type  概念子 =( tp: 概念の型; id: 概念);
type  対象 =( gn:(ga: 概念),ht:(hi: 必要ix));
```

表3　導出されたデータ型

3．2．2　抽象プログラム

表3に示したデータ型をもとに、3．1．3節のキーワードを入力し、キーワードからたどられる関連を出力する部分の抽象プログラムを作成した。図5にそれを示す。アルゴリズムの概要は

以下の通り。キーワードに対応する概念名を出力したあと、その名概念に対し、概念を表わすキーワード、概念に対応する説明文を示し、関連「所属」、「レコード入出力」、「制約」、「必要」、「選択的制約」等の関連をたどり出力する。

```
type  キーワード情報 =( pkos: 構成情報ix; sets: 説明文ix);
type  構成情報 =( kw: キーワード; siji: 指示子; shor: 省略内容ix);
type  指示子情報 =( pkos:  powerset 構成情報ix; gn: 概念子; sent: powerset概念;
                    sets: 説明文ix);
type  概念入出力形式情報 =( sets: 説明文ix; siji: 指示子);
type  概念型情報 =( sets: 説明文ix; siji: 指示子; yoso:  powerset 概念;
                    ny: 入出力形式型; seig: 制限内容ix; shor: 省略内容ix;
                    ssei:  powerset 選択的制約ix; hitk:  powerset 必要ix);
type  概念要素情報 =( sets: 説明文ix; siji: 指示子; kata: 概念;
                    nyix: 入出力形式ix; seig: 制限内容ix; shor: 省略内容ix;
                    hity:  powerset 必要ix; sent:  powerset 選択子ix);
type  必要情報 =( yoso: 概念; kata: 概念; seiy: 制限内容ix; shor: 省略内容ix;
                  ssei:  powerset 選択的制約ix);
type  選択的制約 =( tais: 対象; sent: 選択子ix; seiy: 制約内容ix;);
type  選択子情報 =( pgai:  powerset 概念; ssei:  powerset 選択的制約ix);
type  入出力形式情報 =( pgai:  powerset 概念; seng: 宣言文ix;
                        shsn: 省略形宣言文ix; prog: プログラムix);
type  概念子 =( tp: 概念の型; id: 概念);
type  対象 =( gn:(ga: 概念),ht:(hi: 必要ix));
type  説明文ix        =0..Nsm
type  省略内容ix      =0..Nsn
type  制約内容ix      =0..Nsy
type  構成情報ix      =0..Nks
type  選択的制約ix    =0..Nst
type  必要ix          =0..Nhy
type  選択子ix        =0..Nsi
type  入出力形式ix    =0..Nny
type  宣言文ix        =0..Nsg
type  省略形宣言文ix =0..Nss
type  プログラムix    =0..Npg
type  制約内容 = sequence char;
type  省略内容 = sequence char;
type  説明文 =(in: powerset 概念子; bn: sequence char);
type  宣言文 = seqrence char;
type  プログラム = sequence char;
type  省略形宣言文 = sequence char;
  キーワードfile:     array キーワード    of キーワード情報;
  構成file:           array 構成情報ix    of 構成情報;
  指示子file:         array 指示子        of 指示子情報;
  概念入出力形式file: array 概念          of 概念入出力形式;
  概念型file:         array 概念          of 概念型;
  概念要素file:       array 概念          of 概念要素;
  必要file:           array 必要ix        of 必要情報;
  選択的制約file:     array 選択的制約ix  of 選択的制約;
  選択子file:         array 選択子ix      of 選択子情報;
  入出力形式file:     array 入出力形式ix  of 入出力形式情報;
  制限内容file:       array 制約内容ix    of 制約内容;
  省略内容file:       array 省略内容ix    of 省略内容;
  説明文file:         array 説明文ix      of 説明文;
  宣言文file:         array 宣言文ix      of 宣言文;
  プログラムfile:     array プログラムix  of プログラム;
  省略形宣言文file    array 省略宣言文ix of 省略形宣言文;
  kw: キーワード;
  kwinf:  キーワード情報;
  pgn: powerset 概念子;
  siji: 指示子;
  kosix:  構成情報ix;
  g:  概念子;
  sijinf: 指示子情報;
  pgl,pg2: powerset 概念子;
   gnyinf: 概念入出力形式情報;
   gkinf : 概念型情報;
   gyinf : 概念要素情報;
   phitk : powerset  必要ix;
   hitix : 必要ix;
   hitinf: 必要情報;
   phity : powerset  必要ix;
```

図5　抽象プログラム

プログラム中の㋑の部分は表2 B①に、㋺はC①に、㋩はC②に、㋥はC③に、㋭はC④に、㋬はC⑤に、㋣はC⑥に、㋠はC⑦に、㋡はC⑧に、それぞれ対応している。

```
kwの入力
kwinf:= キーワードfile[kw];
display kwinf.sets,frame F-0;
if kwinf.pkos ≠ {}  then begin
  pgl,pg2 := {}
  while kwinf.pkos≠ {} do begin
    kosix from kwinf.pkos;
    sijinf:=指示子file [構成file[kosix].siji];
    if size(sijiinf.pk)=1
      then pgl :∪sijinf.gn;
      else pg2 :∪sijinf.gn;
  end
  if pgl≠ {} then display kw, pgl,frame F-1
  if pg2≠ {} then display kw, pg2,frame F-2
  pgn:=pgl∪pg2;
  while pgn ≠ {} do begin
    g from pgn;
    psij:= {}
    case g.tp of
      ny: begin
          gnyinf:=概念入出力形式file[g.id];
          psij :∪ gnyinf.siji
          end
      kt: begin
          gkinf :=概念型file[g.id];
          psij :∪ gkinf.siji
          end
      ys: begin
          gyinf :=概念要素file[g.id];
          psij :∪ gkinf.siji
          end
    if size(psij)=1 then begin
      siji from  psij;
      sijinf:=指示子file[siji];
      if size(sijinf.pkos)≠1 then
        then display g,sijinf.pkos,frame F-11;
        else display g,sijinf.sets,frame F-12;
    end else begin
      display g,frame F-21
      while  psij ≠ {} do begin
        siji from  psij;
        sijiinf:= 指示子file[siji];
        display siziinf.sijk,frame F-22
        if sijinf.pkos≠ {}
          then display sijinf.pkos,frame F-23;
          else display sijiinf.pk,frame F-24;
      end
    end
    case g.tp of
      ny: display g.id,gnyinf.sets,frame A-4;
      kt: begin
          display g.id,gkinf,frame A-21;
          if gkinf.yoso ≠ {}
            then display g.id,gkinf.yoso,frameA-22;
          if gkinf.seiy ≠nil then display g.id,gkinf.seiy,frame B-1;
          if gkinf.ssei ≠nil then display g.id,gkinf.ssei,frame B-2;
          if gkinf.hitk ≠ {} then begin
            phitk:=gkinf.hitk
            display g.id,phitk,frame D-0;
            while gkinf.hitk≠ {} do begin
              hitix from gkinf.hitk;
              hitinf:=必要file[hitix];
              if hitinf.seiy≠nil or hitinf.ssei≠nil or hitinf.shor≠nil
                then begin
                display hitinf.yoso,frame D-2;
                if hitinf.seiy≠nil then display hitinf.seiy,frame D-3;
                if hitinf.ssei≠nil then display hitinf.ssei,frame D-4;
                if hitinf.shor≠nil then display hitinf.shor,frame D-5;
              end
            end
          end
          if  gkinf.shor≠nil then display g.id,gkind.shor,frame B-3;
          end
      ys: begin
          display g.id,gyinf.kata,frame A-11;
          display sets,frame A-12;
          if gyinf.ny ≠ {} then display g.id,gyinf.ny,frame
          if gyinf.hity ≠ {} then begin
            phity:=gyinf.hity;
            display g.id,phity,frame C-0;
            while gyinf.hity≠ {} do begin
              hitix from ginf. hity;
              hitinf:=必要file[hitix];
              if hitinf.seiy≠nil then display g.id,hitinf.seiy,frame C-1;
              if hitinf.ssei≠nil then display g.id,hitinf.ssei,frame C-2;
              if hitinf.shor≠nil then display g.id,hitinf.shor,frame C-3;
            end
          end
          if gyinf.set≠ {} then
            display g.id, gyinf.sent,,frame E;
          end
    end
  end
```

4．変換過程の記録法　本事例においては、要求定義中の関連と、抽象プログラム中のデータ型との対応関係が議論の焦点となる。前章までの議論を再整理してみよう。表4に対応関係の分類と、そのような対応の根拠を述べる。

対応関係	事例中の具体例	対応の根拠
（1） 関連とデータ型が1対1に対応する	関連「選択的制約」とデータ型「選択的制約」	関連が多項関連であるので1つのまとまったデータ型として残す方がよい、また他の関連に参加しないので、これ以上の情報を持つ必要がない
（2） 複数の関連が1つのデータ型に埋めこまれる	関連「必要」「制約」「省略規則」とデータ型「必要情報」	関連「必要」は多対多であり双方向にたどられることから、対応するデータ型を必要とする。「必要」は他の2つに参加しており、しかもその対応は1対1であり1方向にのみたどる。
	関連「選択子構成」「選択条件」とデータ型「指示子情報」	何段階かの関連において、たどり方が常に1方向である場合は、まとめて1つのデータ型に対応させうる
（3） 1つの関連が複数のデータ型に分解される	関連「所属」とデータ型「概念型情報」「概念要素情報」	「所属」は双方向にたどられるが他の関連には参加しないので分けた方がよい
	関連「制約」とデータ型「概念型情報」「必要情報」	たどり方は1方向であるが始点のdomainの要素が複数の型にわかれる

表4　関連とデータ型の対応関係の分類

事例中の対応関係を表現する関連の一般的性格をまとめる。(1)は変換の始点となる関連と終点となるデータ型の2項関連、(2)，(3)は、変換の始点となる関連と終点となるデータ型、およびデータ型において関連が表現されている位置となるフィールド名の3項関連。以上は、一般的性格が満たすべき最低限の条件である。(2)，(3)の場合は、さらにどのroleがどのデータ型のどのフィールドへ変換されたか、そのときのdomainは何か、という情報の記述も必要であろう。さらに、変換の理由との結合を考えると、上記のような対応を示す複数の関連と、複数の変換理由が結合し、1つの変換とならなければならない。そのためには、「変換単位」のようなobjectを導入し、「変換単位」と先の対応関係を示す関連の間に「変換の詳細」のような関連をつけ、「変換単位」と変換理由の間に「理由」のような関連が必要になるだろう。また、変換理由が他の変換である場合もあり、「変換理由」の詳細を分析する必要があるだろう。

変数、手続も含めた変換過程の把握は現在分析中である。すなわち、要求仕様中の機能要求が、データ型の決定に従い、変数と手続に編成される過程を明らかにする必要がある。

5．今後の課題　現在抽象プログラムから具体プログラムへの変換過程についても、本稿で述べた手法に基づいて、分析・定式化を試みている。問題の性格を変えたいくつかの事例研究を積重ねることにより、開発過程の「決定の構造」を記録するための関連を充実することも必要である。さらに、言語の独習システムは、自明な例題ではないが、比較的小規模なシステムである。保守の問題は大規模ソフトウェアの変更という問題に強く依存して発生することを考えれば、より大規模なシステムへの適用可否の問題も検討する必要がある。現在準備中である。

また、現状のデータベース技術の表現能力に関する制約は受けるが、feasibility確認のためのプロトタイプモデルの開発も計画している。

参考文献

(1) W. Kent，「Data and Reality」 North-Holland，1978.
(2) C. A. R. Hoare，"Notes On Data Structuring"，「Structured Programming」 ACADEMIC PRESS，1972