Vol. 27 No. 4 情報処理学会論文誌 Apr. 1986

部品合成によるプログラム自動生成システム ARIES/I†

原田 実†† 篠原靖志††

ファイル処理プログラム開発の生産性および信頼性を飛躍的に向上するために，エンドユーザでも記述可能な日本語文章による処理要求からプログラムを自動生成するシステム ARIES/I (Automatic Resource Integration Engineering System/one) を開発し，運用評価した．自動化方式という観点からみると，ARIES/I は，あらかじめソフトウェアを部品化して管理し，要求ごとにそれにあった部品を検索し，要求に合わせて編集し，最終的に一本の独立した実行可能なプログラムに統合する一連の過程をすべて自動化するアプローチを採っている．

1. はじめに

企業における業務処理を計算機を使って機械化するようになって 20〜30 年たつが，この間常にプログラム開発の生産性を上げなければならないという問題が情報システム部門を悩ましてきた．このため各企業ではプログラミング技法や設計技法の普及や標準化，プログラムの再利用，プログラム部品ライブラリーの整備など地道な努力を重ねてきた．

しかし，今日大量のバックログを抱え，さらに社会の OA 化や高度情報化に伴う新規要求に答えるためには，プログラム開発作業を徹底して自動化する以外に方法はないように思える[1]．

このため我々は，部品の検索・編集・合成の自動化による自動プログラミング・システムの実験版として ARIES/I (Automatic Resource Integration Engineering System/one)* を開発した[2),3)]．

自動プログラミングの研究は，1960 年代後半の定理証明法を応用した理論的アプローチに端を発すると考えられる．特に対象を事務処理プログラムに限ると，米国では MIT の Protosystem 1[4)]やペンシルベニア大学の MODEL[5)] などがあるがまだ開発中の域を出ていない．

むしろ国内において，DSL[6)]，KIPS[7)]，ARIES[2),3)] などが実験的ではあるが運用・評価の段階まで進んでいる．このうち，後者二つは仕様入力に自然言語を用い，出力がデータフロー型の COBOL の超高水準言語である点が共通している．KIPS ではデータやプログラムが知識を内蔵したオブジェクトとして定義され，それらが有機的に機能してプログラムを生成する．一方 ARIES は設計者やプログラマのルール化された知識を用いて必要なプログラム部品を検索し編集・組み合わせることによってプログラムを生成する．

以下ではこの ARIES について，要求の表現モデル，システム構成とプログラム生成過程，プログラム部品，評価，今後の課題などについて解説する．

2. 要求の表現モデル

企業では業務の主管部からデータ処理要求が出されるが，通常これは業務用語を用いた日本語文で記述されることが多い．当所でも同様であり，例えば**図1**に示したような要求が企画部から出された．

ARIES はこのような自然言語による要求を入力すると**図2**に示すような過程を経てプログラムを生成する．この間，要求文Aは図2に示すような表現モデルに変換されていく．以下に各表現モデルの内容と形式について説明する．

2.1 要求文

ARIES ではユーザは業務要求を特定の「言い回し」を使って表現する．例えば，要求文Aは「…に対して…をレポートせよ」という言い回しを用いている．一般に，言い回しは特定の処理（以後，要求オペレーションと呼ぶ）を表す動詞と助詞から構成されており，各オペランド（…で表した部分）がどのような実体クラスのものであるかも定義している．

2.2 データモデル

データモデルは企業を構成している実体と実体間の関係を，どのような意味や属性を持つデータによって表現しているかを定義している．ARIES では，実体と関係を統一的にクラスとして扱いフレームで表現す

† A Program Generator ARIES/I—By Automatic Fabrication of Reusable Program Components—by MINORU HARADA and YASUSI SINOHARA (Knowledge Engineering Section, Information System Department, Economic Research Center, Central Research Institute of Electric Power Industry).

†† (財)電力中央研究所経済研究所情報システム部知識処理研究室

* この論文では，すでに公表してきた ARIES/I にさらに現在改良中の内容も付加して説明する．

要求文 A:
『テーマ台帳に登録されたテーマを担当する研究者に対して，氏名コード，氏名，所属する研究室名をレポートせよ.』

図 1 業務処理要求の一例
Fig. 1 A date processing requirement sample.

る.

図3(a)では，“人”という(実体を表す)クラスが“氏名コード”と“氏名”という(属性)スロットを持ち，さらに物理ファイルとの対応を取るために各“人”のこれらの属性値がそれぞれ氏名テーブル・ファイルの氏名コード項目および氏名項目に入っていることなどが表されている．一方，“担当する”という(関係を表す)クラスは，object と actor という(格)スロットを持ち，テーマと担当者という役割の下で，“テーマ”と“研究者”という実体クラスを関係づけ，さらに，この個々の関係が担当簿ファイルのテーマコード項目および氏名コード項目によって表されていることを示している.

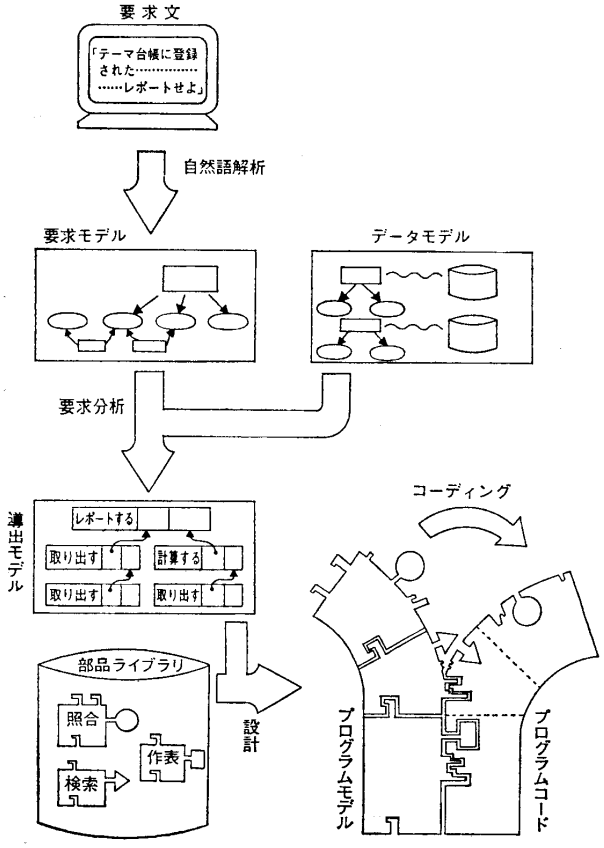

図 2 部品合成による自動プログラミング
Fig. 2 Automatic programming by program parts fabrication.

2.3 要求モデル

要求モデルは，入力された要求文に現れる(概念)項目とそれらの間の関係を，データモデルと同様に，関係と実体のフレームのネットワークとして表現する.

図3(b)に示した要求モデルでは，四つの関係(以下～で示す)が六つの実体(＿で示す)を，『“テーマ台帳”というファイル%1に登録されているテーマ%3を担当している研究者%5に対して，その氏名コード%10と氏名%11および彼が所属する研究室名%13をレポートする.』という意味で結びつけている.

2.4 導出モデル

導出モデルは要求モデルをデータモデルに突き合わせて解釈した結果を，オペレーション名やファイル名やデータ項目名などのデータ処理の言葉で表現した，プログラムを生成するのに十分な情報を持った要求文の表現モデルである.

導出モデルはデータの導出関係を表す導出フレームのネットワークであり，個々の導出フレームはオペレーション(データに対する操作)を表現している.

導出フレームの例としては，要求オペレーションを表すもの(例：図3(c)の“レポートする”)と，要求オペレーションのオペランドとなるデータを，外部ファイルからのレコード検索によって導出するもの(例：図3(c)の“取り出す”)および他のデータ項目や定数から計算によって導出するもの(例にはないが“計算する”というオペレーション名を持つ)などがある.

2.5 プログラムモデル

プログラムモデルは導出モデルに表されている導出関係をどのようにして表現するかを，(プログラム)部品の列で表現したモデルである.

例えば図3(d)では，Xappend 部品はテーマコードを入力し，担当簿からこのテーマコードを持つレコードの氏名コードを取り出し，テーマコードと対にして出力することを表している.

2.6 プログラムコード

プログラムコードは要求文に表された機能を実現する HYPER COBOL で記述されたプログラムである.

HYPER COBOL は富士通から提供されているデータフロー型の超高水準プログラミング言語であり，標準ユニットと呼ばれる並行に動作する機能モジュールとその間を流れるデータストリームによってプログラムを記述する.

標準ユニットには順ファイルとの照合用の MATCH ユニットや乱ファイル検索用の CHAIN ユニ

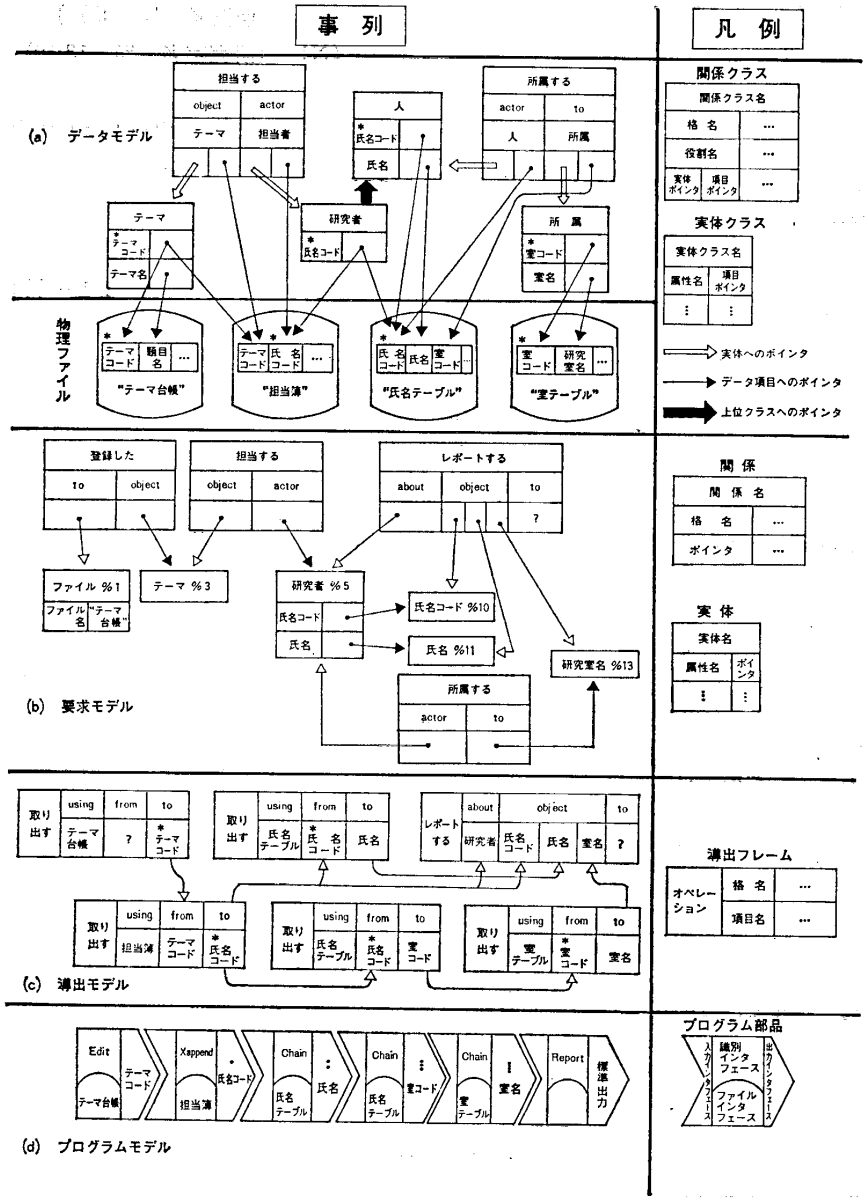

図 3　データモデルと要求の表現モデル
Fig. 3　Data model and requirement representation models.

```
((レポート
  (TE (対する
       (NI (研究者 % 5
            (担当する
             (GA (研究者 % 5))
             (WO (テーマ % 3
                  (登録する (NI ("テーマ台帳" % 1)
                                (WO (テーマ % 3)))))))))))
(1)
(WO (& (氏名コード % 10)
       (氏名 % 11)
       (研究室名 % 13 (所属する (GA *X_1342) (NI (研究室名 % 13)))))))
```

図 4　要求文Aに対する要求モデル
Fig. 4　The requirement model for the requirement statement A.

ットなど12個が用意されているが，設計規則を単純にしかつ生成するコードを最適にするために，n:m 照合用の % APPEND と % XAPPEND およびプログラム中で何回も使える % SORT などのユニットを追加し，HYPER COBOL を拡充した．

要求文Aに対するプログラムコードの一部を図8に示す．ここで，100 行から 11,600 行までがストリームエリア (SA) などのデータの宣言であり 11,700 行から 17,800 行までが手続きの記述である．例えば，STEP1 では，EDIT ユニットと SUMMARIZE ユニットを使ってテーマ台帳 (TDAIMSA1) に登録されているテーマ (THEME-14) を取り出している．最後の STEP6 では，REPORT ユニットを使って標準形式で氏名コード，氏名，所属を出力する．

3. プログラム生成過程

図2に示した自動生成の各フェーズを説明する．

3.1 自然言語解析

自然言語解析では，要求文を読み込んで，要求モデルを生成する．例えば，要求文Aに対しては，図4に示す要求モデルを作成する．

要求モデル中の表現：

(関係名 (格1　項目1)(格2　項目2))

は，項目1と項目2が，関係名で表される関係にあることを表している．

要求モデル中の表現：

(項目名1　%　id) と

(項目名1　%　id
　(関係名　(格1 (項目名1　%　id))
　　(格2　項目2)))

は，各々，項目を表現する．特に後者の表現は，

```
文法の構文規則：
        (記号1  修飾情報1  メッセージ操作1 opt)
        (記号2  修飾情報2  メッセージ操作2 opt)
                .....
    ⇒   (記号N  修飾情報N  意味合成規則)
ここで,
記号          :  文法カテゴリー名
修飾情報      :  (修飾され得る修飾形  修飾し得る修飾形)
メッセージ操作:  メッセージを操作するPROLOG述語
意味合成規則  :  記号1, …の意味から記号Nの意味を合成するPROLOG述語

日本語の文法例：
 (名詞句 ((連体)(格助詞)))(格助詞 ((格助詞)*))
              ⇒ (名詞句+格助詞 ((-) *)(名詞句+格助詞))
```

図 5 ARIES/I のパーザでの文法
Fig. 5 The grammar rule of the ARIES/I parser.

項目2と関係名で表される関係にある項目（項目名1 % id）を表している．

この解析には Prolog によるボトムアップ・パーザ (BUP) を，日本語の修飾・被修飾の関係が自然に表現できるようにしたものを使っている．

文法は基本的には図5を示すような形式で与える．句に対しては，文法カテゴリー名と修飾・被修飾についての情報を与える．ただし，修飾・被修飾情報についての部分は変数を許す．これによって，修飾形についてのあいまいさを残したまま解析を進めていき必要情報が得られた時に，あいまいさを解消する．例えば，「決め」という句は「させる」などの未然形を受けるものにも「たり」のような連用形を受けるものにもかかり得る．ARIES では，これを未然形か連用形を受ける句として解釈しておき，次の句を読んで，その句が受け得る '形' を調べてから，「決め」の '形' を絞り込むという操作を行う．このため，構文解析中のバックトラックを減少させることができた．

3.2 要求分析

このフェーズでは，自然言語解析フェーズで得た要求モデルから導出モデルを作成する．

まず，要求文において省略された項目間の関係を補完した要求モデルを作る．例えば，「"項目1に対して，項目2をレポートする." という表現では，項目2は項目1に関係している」という知識を使って，図4中の(1)で示した項目は，

```
(& (氏名コード  %  10
     (の (研究者  %  5)
         氏名コード  %  10)))
   (氏名  %  11
     (の (研究者  %  5)
         (氏名  %  11)))
   (研究室名  %  13
         (所属する (GA  (研究者  %  5))
                   (NI  (研究室名  %  13)))))
```

のように補完される．

次に，この補完された要求モデルを導出モデルに変換する．例えば，要求モデル中の表現：

```
(項目名0  %  n0
   (関係名  (格 (項目名0  %  n0))
            (格1 (項目名1  %  n1…))))
```

に対する導出モデルMを作成するには，

```
(項目名1  %  n1…)
```

に対する導出モデルM0に，

```
(項目名0  %  n0
   (関係名  (格  (項目名0  %  n0))
            (格1 (項目名0  %  n1))))
```

についての導出フレームFを追加すればよい．

現在，このような項目の導出はファイル検索によって行えると仮定している．したがって，まずデータモデルと突きあわせて，関係名で表される関係を実現しているファイルの名前 FL，そのレコード中で（項目名1 % n1）を表すフィールド名 FD1，(項目名0 % n0) を表すフィールド名 FD0 を決定し，その結果，導出フレームFとして，

```
((operation  取り出す)
 (using      FL    )
 (from       DF1   )
 (to         DF0   ))
```

を M0 に追加して M を得る．

一方，要求オペレーション“レポートする”に対しては，

```
『レポートする
     (TE (対する (NI  *項目1))
     (WO *項目2))
```

という要求表現が，

```
((operation  レポートする)
 (object     *項目2)
 (to         0))
```

という導出フレームに対応する』

という解釈知識を使って，このフレームを項目1と項目2についての導出モデルに追加する．

以上の一連の操作により，要求モデルに対する導出モデルを得る．

3.3 設　計

設計では，導出モデルに設計規則を適用して，プログラムモデルを作成する．

```
(ASSERT (MODULE ((OPERATION レポート)                    ┐
                 (OBJECT    *DETAIL)                    │
                 (ABOUT     *ABOUT)                     │ (a)見出し部
                 (TO        *OUT))                      │
                *INSA                                   │
                *OUTSA                                  │
                *PART)                                  ┘
        (=*PART                                         ┐
          (FUNIT #REPORT    *STEP                       │
                 *INSA                                  │ (b)部品選択部
                 *OUTSA     *ABOUTFLD                   │
                 *RECFORM   *DETAIL))                   ┘
        (GET-STEP-NO  *STEP)                            ┐
        (GET-RECFORM *INSA *RECFORM)                    │
        (IF (= *OUT NIL) (= *OUTFILE "OUT")             │
                 (= *OUTFILE *OUT))                     │
        (DCL-AREA *OUTSA SA (FILE *OUTFILE)             │ (c)部品適合化部
                            (PATTERN FILE)              │
                            (USE PRINT))                │
        (MAP-CONCEPT *ABOUT *ABOUTO)                    │
        (GET-FLD-OF *ABOUTO *INSA *ABOUTFLD ?))         ┘
```

図 6 “レポート”導出フレームに対する部品選択適合化規則
Fig. 6 The program parts selection and customization rule for the “report” frame.

設計規則には，部品結合規則と部品選択適合化規則の２種類がある．

部品結合規則は，プログラムモデルにおける部品の並び順序を決めるもので，フロー設計についての知識である．この知識を使って，導出モデルにおける項目間の導出関係に従って各導出フレームに対応する部品をトポロジカル・ソートする．この時，部品間を流れるデータストリーム中の重複する同一レコードの個数をできるだけ少なくなるように最適化する．

部品選択適合化規則は導出モデルにおける各導出フレームに対応する部品を選択し，その部品を環境に応じて適合化するために必要な情報（適合化インタフェース）を決定する．

図 6 に“レポートする”というオペレーションを表す導出フレーム用の部品選択適合化規則を示す．この例に示したように，部品選択適合化規則は以下の三つの部分からなる．

（a） 見出し部：導出モデル中の各導出フレームに適合する部品選択適合化規則を検索するための見出しであり，導出フレームが表すオペレーションと一致する．

（b） 部品選択部：導出フレームに表された関係を実際に実現する部品を決定する部分である．例えば，“取り出す”関係の導出フレームに対する規則の部品選択部は，図 7 に示すように，using スロット内のファイル属性や from スロット内の導出元項目がキーであるかないかに応じて最適な部品を決定する．

（c） 部品適合化部：選択された部品の適合化インタフェースを決定する．この時，入力領域など環境* にすでに登録されている情報を使うと共に，出力領

外部ファイルの			選択する部品
ソートキー	同一キー値を持つレコードの出現回数	編成	
導出元項目	1回	順編成	#MATCH
導出元項目	1回	VSAM編成	#CHAIN
導出元項目	複数回	—	#APPEND
導出元項目以外	—	—	#XAPPEND

図 7 “取り出し”導出フレームに対する部品選択規則
Fig. 7 The program parts selection rule for a “take-out” frame.

* プログラムコードを生成するのに必要なデータ資源に関する情報を環境と呼ぶ．

```
//HYPERCOB  EXEC PROC=HCOBCG,
//   PARM=(DSCHART,INSOURCE,LIB,RECCNT,UUNIT,
//     'UNITCNT,NONUM')
//HCOB.SYSLIB DD DSN=COOAA.COPYLIB.COBOL,DISP=SHR
//HCOB.SYSIN  DD *
000100 ID GENKA.
000200*
000300 SA    TDAIMSA1 FILE(TDAIMSA1)
000400               RECORD(TDAIMSR1)
000500               KEY(THEME)
000600               ORG(VSAM IND).
000700 RECORD.
000800 01    TDAIMSR1.
000900       INCLUDE TDAIM.
001000*
                      ≀
011700 SA    REPORT10 FILE(REPORT10).
011800*STEP1
011900 EDIT IN(TDAIMSA1) OUT(SA2).
012000           MOVE THEME OF TDAIMSR1 TO THEME-14 OF REC2.
012100 SUMMARIZE IN(SA2) OUT(SA1) KEY(THEME-14).
012200     WHEN INIT.
012300           MOVE CORR REC2 TO REC1.
                      ≀
013700*STEP3
013800 CHAIN IN(SA3,SIMETKA) OUT(SA4) KEY(SIMEI-15 OF REC3).
013900     WHEN VALID.
014000           MOVE NSIMEI OF SIMETKR TO NSIMEI-11 OF REC4.
014100     WHEN INVALID.
014200           DISPLAY "INVALID KEY OCCURRED AT STEP3".
014300           DISPLAY "SIMEI= " SIMEI-15 OF REC3
014400             " NOT IN  SIMETKA".
                      ≀
017000*STEP6
017100 REPORT IN(SA9) OUT(REPORT10) RD(RD10).
017200 RD    RD10 PAGE(LINES(63) COLUMNS(136))
017300                  CONTROL(SIMEI-15) STANDARD.
017400 SOURCE.
017500 DE           SIMEI-15 OF REC9 HD(NC"氏名コード"),
017600             NSIMEI-11 OF REC9 HD(NC"氏名"),
017700             NSHOZOKU-13 OF REC9 HD(NC"研究室名").
017800*
/*
//GO.SYSOUT  DD SYSOUT=*
//GO.TDAIMSA1 DD DSN=COOAA.DAIMOKU.TEST.NOADJ.KSDS,DISP=SHR
//GO.TANTMSA3 DD DSN=COOAA.TANTOBO.DATA,DISP=SHR
//GO.WKA3 DD DSN=COOAA.WKA3.KSDS,DISP=SHR
//GO.SIMETKA DD DSN=COOAA.TABLE02.EXT.NOADJ.KSDS,DISP=SHR
//GO.SHOZTKA DD DSN=COOAA.TABLE01.NOADJ.KSDS,DISP=SHR
//GO.WKA7 DD DSN=COOAA.WKA7.KSDS,DISP=SHR
//GO.REPORT10 DD DSN=&&SOUT,DISP=(NEW,PASS,DELETE),UNIT=SYSDA,
//       VOL=SER=TEMP01,SPACE=(TRK,(60,20)),
//       DCB=(BLKSIZE=3030,LRECL=303,RECFM=VBA)
```

図 8 要求文Aに対して ARIES が生成したプログラムコード
Fig. 8 The program code generated by ARIES for the statement A.

域など未登録の情報についてはこれを決定し環境に登録する．

このように導出フレームに対応する部品を選択する過程を，見出し部が表すオペレーション機能による絞り込みと対象となるファイルやキー項目の状態による最終決定の2段構えにすることによって，限られた部品を多様な要求に柔軟に対応させることができる．

3.4 コーディング

このフェーズでは，プログラムモデルから実際のプログラムコードを，JCL と共に出力する．

プログラムモデル中の各部品は，要求される機能を表すひな型プログラムコードをその適合化インタフェースに合わせて編集し出力するコーディング知識を内蔵している．したがって，各部品をプログラムモデル中の順序に従って評価すれば，このコーディング知識がプログラムの手続き部を構成するコードを出力する．

一方，データ部のファイル節や作業場所節中のデータ宣言や JCL 中のデータセットの割当命令などは，設計時に主として生成された環境を調べることによって生成される．

このようにして，要求文Aに対するプログラムが**図8**に示すように生成される．

4. 部　　品

部品は，ある一定の機能を果たすプログラム・コードのひな型である．部品は，それが利用される環境の具体的詳細を与える記述（適合化インタフェース）が与えられると，要求される機能を行うプログラム・コードを出力する．

ARIES は Prolog で記述されており，部品の定義も同様に Prolog で行う．このため，例えば #EDIT 部品は**図9**に示すように記述できる．

```
(ASSERT (FUNIT #EDIT……(1)識別インタフェース
        *STEP
        *INSA *INREC *MCF                    } (3)適合化インタフェース
        *OUTSA *OUTREC *MCFO)
        (FORMAT "*/S/N" *STEP)
        (FORMAT "   EDIT IN(/S) OUT(/S)./N"
                    *INSA *OUTSA)
        (FORMAT "          MOVE /F OF /S
                           TO /F OF /S./N"
                *MCF    *INREC
                *MCFO   *OUTREC))            } (2)部品本体
```

図 9 #EDIT 部品の定義
Fig. 9 The definition of "#EDIT".

```
(ASSERT (FUNIT #EDIT-SUMMARIZE……(1)識別インタフェース
        *STEP *INSA   *INREC   *MCF          } (3)適合化インタフェース
              *MIDSA  *MIDREC
              *OUTSA  *OUTREC  *MCFO)
(FONIT #EDIT *STEPS
              *INSA   *INREC   *MCF          } (2)部品本体
              *MIDSA  *MIDREC  *MCFO)
(FUNIT #SUMMARIZE *STEP
              *MIDSA  *MIDREC  *MCFO)        } (2)部品本体
              *OUTSA  *OUTREC))
```

図 10 #EDIT-SUMMARIZE 部品の定義
Fig. 10 The definition of "#EDIT-SUMMARIZE".

部品は一般にこの図に示すように，(1)自己を他から識別するための識別インタフェース，(2)要求されている機能を実現するプログラムのひな型コードである部品本体，(3)部品が利用するデータ資源などを記述する適合化インタフェースから成る．

一方，複合機能を果たす部品は，複数の部分部品から成ることが多いが，このような部品（例：#EDIT-SUMMARIZE）を定義するには，**図10**に示すように，その部分部品（#EDIT と #SUMMARIZE）を書き並べるだけでよく，作成や理解が容易である．

このようにして定義された #EDIT-SUMMARIZE 部品は，プログラムモデル中では，(FUNIT #EDIT-SUMMARIZE STEP1 TDAIMSA1 TDAIMSR1 (THEME) SA2 REC2 SA1 REC1 (THEME 14)) のように具体的な適合化インタフェースを伴って記述され，これが評価されると，図8に示したプログラムの(a)の部分を出力する．

5. おわりに

5.1 効　　果

ARIES による効果をまとめると**図11**のようになる．これらについて簡単に項目ごとに説明する．

(1)〔プログラミングの生産性〕

ARIES は図1に示した3行からなる要求文Aに対して，300行近い HYPER COBOL のコードを生成した．他の要求例に対しても同様で，結局10倍程度のプログラミング生産性の向上が達成された．

さらに，利用者は意図する処理を日常の業務で使っ

- プログラミング生産性
  - 生成プログラムコード数 / 入力要求文行数 比……数十倍
  - 工数比……約 100 倍
- 生成プログラムの信頼性……飛躍的に向上
- 生成プログラムの性能……かなり向上

図 11 ARIES/I の効果
Fig. 11 Programming improvement by ARIES/I.

ている言い回しやファイル名やデータ項目名などの業務用語を使って記述するだけで，プログラミングする必要がない．したがって，工数比で見ても，一数時間/2～3分—100倍近い生産性向上が期待できる．

（2）〔生成プログラムの信頼性〕

部品および部品の合成規則は，通常の市販のパッケージやコンパイラと同様に非常に高い信頼性の下で作成できるので，手作業によるプログラミングに比べれば，かなり高い信頼性のプログラムを生成できる．

（3）〔生成プログラムの性能〕

人手による場合，最適化はプログラマの技量に依存することが多い．また優秀なプログラマでも納期に迫られて，綿密な最適設計を怠りがちである．

一方，ARIES は，データストリーム中の重複する同一レコードの出現頻度を少なくするように常に最適化を行う．したがって平均的に見てかなり高い性能を持つプログラムを生成できる．

5.2 今後の研究の方向性

（1） 適 用 性

適用範囲を拡大するには，ARIES が受容する要求文の「言い回し」とデータモデルに登録されているデータ資源を増やす必要がある．

特定の言い回しを追加するには，これをシステムが理解するための用語辞書と解釈規則の追加およびプログラムを生成するための部品選択適合化規則と部品そのものの追加が必要である．部品の自動合成が ARIES の主要テーマである以上この「言い回し」すなわち「部品」の追加を簡易に行うインタフェースの作成は今後の大きな課題である．

一方，データモデルの拡充については，ファイルのコピーライブラリに意味情報を追加したものから自動的に作成するプログラムを開発中である．

（2） ユーザ・インタフェースの多様化・階層化

ARIES の本質的な問題点は要求文の記述をすべて自然言語で与えるというユーザインタフェースである．一般的にデータ処理情報をすべて日本語文章で完全に記述するのは，非常に骨の折れる退屈な仕事である．特に給与計算など計算条件が入り組んだ業務処理要求については特にそうである．このことは ARIES の理論的・技術的な問題点ではないが，実用上は大きな問題点である．

これを解決するには，従来の Waterfall モデルによるプログラム開発過程がそうであるように，ユーザ・インタフェースとなる要求表現の抽象レベルをいくつか用意する必要がある．高レベルのインタフェースには，ストラクチャード・アナリシスで用いられているような図表現による処理フローと ARIES で用いたような自然言語による機能表現が，一方，詳細レベルのインタフェースには，デシジョン・テーブルのような厳密な表現が適していると考えられる．

ユーザの思考過程に合わせて，この両方のインタフェースを使い分け，自動設計と自動プログラミングの両方を適宜行う，真に一貫したソフトウェア・オートメーション・システムの構築が我々の最終目標である．

このため，現在当所ではパターン指向型プログラム開発技法：PDM[8] に基づく，デシジョン・テーブルを中心とした設計仕様から COBOL プログラムを自動生成するシステム SPACE：(Specification Acquisition and Compiling Engine) を開発中である．

謝辞 本研究をご支援くださった，若林剛部長と鈴木道夫室長を始めとする当所情報システム部の皆様に感謝いたします．

参 考 文 献

1) 原田 実：ソフトウェア生産性向上ツール(下)，日経コンピュータ，No. 65, pp. 175-199 (1984).
2) 原田 実：日本語要求仕様からのプログラム自動生成—ARIES/I—，要求定義および仕様化技術ワークショップ報告書，情報処理振興事業協会技術センター，pp. 254-260 (1984).
3) 原田 実，篠原靖志，鈴木道夫：プログラム自動生成システム ARIES/I の開発，情報処理研究，(財)電力中央研究所，No. 13, pp. 59-88 (1985).
4) Hammer, M. and Ruth, G.: Automating the Software Development Process, *Research Directions in Software Technique*, pp. 767-792, MIT Press, Cambridge (1979).
5) Prywes, N. and Pnueli, A.: Compilation of Nonprocedural Specifications into Computer Programs, *IEEE Trans. Softw. Eng.*, Vol. SE-9, No. 3, pp. 267-279 (1983).
6) 河野央男，ほか：データに着目した仕様を入力とする COBOL プログラム生成システム"DSL"の開発，日立評論，Vol. 65, No. 7, pp. 53-58 (1983).
7) 杉山，秋山，亀田，牧之内：対話型自然言語プログラミングシステムの試作，電子通信学会論文誌，Vol. J67-D, No. 3, pp. 297-304 (1984).
8) 原田 実：パターン指向型プログラム開発技法：PDM，(財)電力中央研究所，p. 113 (1985).

(昭和60年6月25日受付)
(昭和60年12月19日採録)

原田　実（正会員）

昭和26年生．昭和50年東京大学理学部物理学科卒業．昭和55年東京大学理学系大学院博士課程修了．理学博士．昭和55年（財）電力中央研究所に入所．以来，同経済研究所情報システム部にて，OAやプログラム設計技法や自動プログラミングの研究に従事．著書「ソフトウェアの構造化設計法」（共訳：日本コンピュータ協会）．現在，同部知識処理研究室主査研究員．

篠原　靖志

1960年生．1982年東京大学理学部情報科学科卒業．1984年同大学院修士課程情報科学専攻修了．同年（財）電力中央研究所入所．現在，同所経済研究所情報システム部知識処理研究室研究員．その間，プログラム自動生成，エキスパートシステムの研究に従事．ソフトウェア科学会会員．