15. 要求仕様システムISDOSの試用

慶応大学工学部　斎藤信男, 村井　純, 相場　亮
Nobuo Saito　Jun Murai　Makoto Aiba
日本IBM東京サイエンティフィックセンター　藤崎 哲之助, 戸沢義夫, 諸橋正幸
Tetsunosuke Fujisaki　Yoshio Tozawa　Masayuki　Morohashi

1. はじめに

ソフトウェア工学における最近の話題の一つとして，要求仕様記述技法とそのシステムがある。筆者らは，要求仕様記述システムの一つであるISDOSを使用する機会があったので，このシステムがどの位使い良いか，又，ソフトウェアの作成に対してどのように役に立ちそうかを実際に使用しながら評価してみた。

システムのあらゆる側面を試してみる事はできなかったが，その長所及び追加した方が望ましい点などを現在までの経験に基づいて報告する。

1. 1　ISDOSとは

ISDOS（Information System Development and Optimization System）[1]は，ミシガン大学のD. Teichroewによって提案，開発された要求仕様記述システムである。現段階のこのシステムは，仕様を記述する言語であるPSL（Problem Statement Language）と，記述された仕様を解析するPSA（Problem Statement Analyzer）とから構成され，既に多くの計算機上に実現されている。

PSLでは，記述の対象となるシステムの構成要素であるいくつかの型のオブジェクトと，そのオブジェクト間のいろいろの相互関係とを表わす言語要素が用意されている。要求仕様を充分に記述するためには，自然言語に近いことが望ましく，一方では，計算機による処理を能率良く行なうために形式化した言語が望ましい。PSLは，自然言語に近づけるという姿勢はとっているものの，非常に制限された記述形式を用いている。

PSAは，PSLの記述を入力として処理し，一定の形式のデータベースに変換する。以後，ユーザは，そのデータベースからいろいろの形式のレポートを作成したり，データベースの内容を変更したりする。これらの機能を果たすために，PSA コマンドが用意され，会話形式で要求仕様の記述を完成することができる。

1. 2　PSL/PSA概説

1) PSL[2]

PSLでは，記述の対象となるシステムの構成要素として図1に示す型のオブジェクトが用意されている。処理の実体を占めるPROCESS，外部環境との接点であるINTERFACE，システムの動的状況を表わすためのEVENT，CONDITION，データの構造を表わすためのENTITY，GROUP，ELEMENT，SETなどがその代表的なものである。一方，これらのオブジェクトの相互関係により，対象システムの性質，機能，動作などを表わす。オブジェクト間の相互関係を表わすために，各型ごとに，いくつかの動詞が用意され，自然言語風に表わす。これらの関係は，いろいろの視点から規定されるが，たとえば，PROCESS間の構造を示すSystem Structure，データの構造を示すData Structure，PROCESSとデータ間の関係を示すData Derivation，PROCESS, EVENT, CONDITIONの相互関係を示すSystem Dynamicsなどの視点がある。これらの関係は，top down（downward）

的，およびbottom up（upward）的の両方の立場から記述できるが、一方は他方の記述に従って、PSA内で自動的に生成する。こうすると、全体の記述には冗長性が入り込むけれども、それだけ理解し易くなっている。

PSLでは、PROCESSでの処理の仕方を詳細に記述することはせず、専ら、オブジェクト間の相互関係や、インターフェースの記述を中心に置いている。処理の手続きの内容を書くために、いくつかの種類のComment Entryが用意されている。

2) PSA[3]

PSAはPSLの記述を処理して記述のデータベースを作ること、そのデータベースに基づいて種々のレポートを作ったり、そのデータベースを変更することなどを行なう。これらの機能は、すべて、PSAコマンドを介してユーザが指示する。PSAコマンドを図2に示す。PSLの記述を入力してデータベースを作り出すINPUT-PSL，データベースの内容を更新するCHANGE-NAME，DELETE-NAME，DELETE-PSL，REPLACE-PSL，いろいろのレポートを作り出すFORMATTED-PROBLEM-STATEMENT，PICTURE，PROCESS-CHAIN，NAME-LIST，PSA自体を制御するSET，HELP，STOPなどがある。

PSAシステム自体は、FORTRANで作られているので、いろいろの機械に移植し易い。

2．記述例

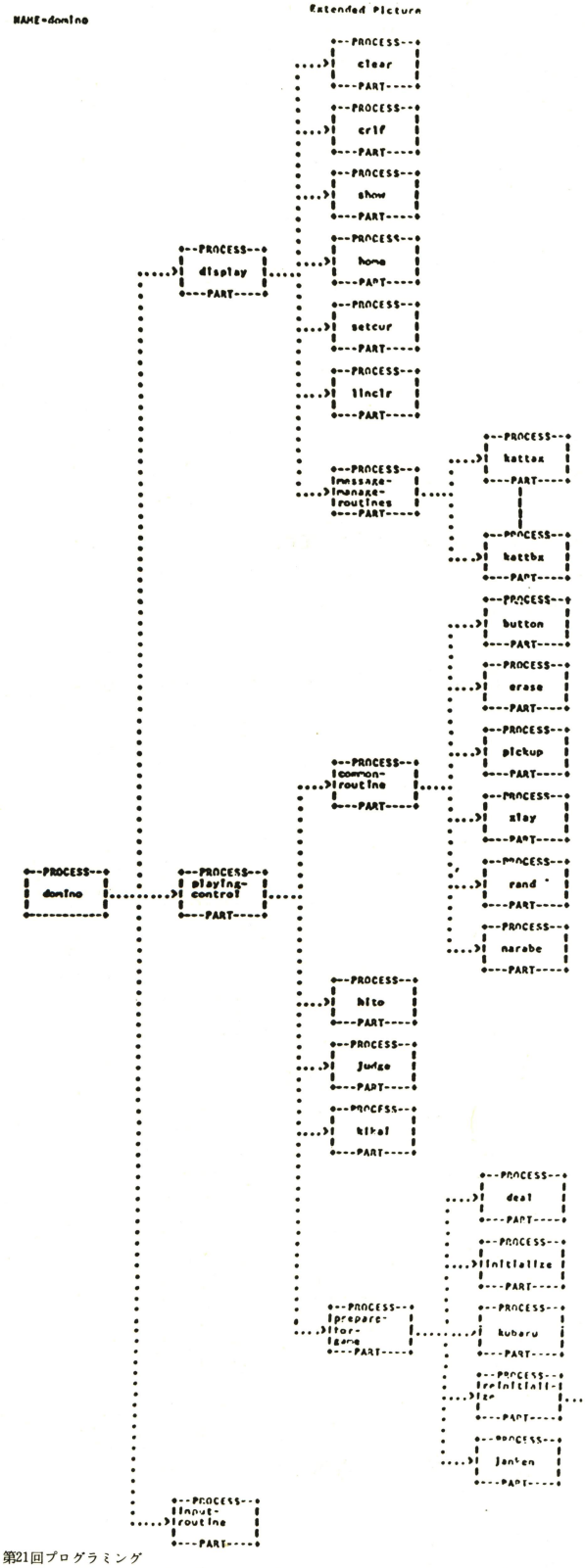
2．1　対象とした問題

ISDOSを使用して、その評価をするために、対象システムを設定した。ここでは、筆者の一人が以前に作成したことのあるプログラムを例として採り上げた。これは、CRTディスプレイを使ったドミノ・ゲームをするプログラムである。これを以前プログラムを作成した通りにその仕様をPSLで記述した。又，PSL/PSAが設計ツールとしてどの位役に立つのかを知るために、同じドミノ・ゲームをするプログラムを、以前に作った方法や考え方を改良して新たに作成するための仕様記述を行なった。以前に作られたプログラム（以下旧プログラムと称する）に対する仕様記述を図3，それを改良したプログラム（以下新プログラムと称する）の仕様記述を図4に示す。

2．2　記述例の目的

この例題では，既に作成したプログラムの仕様記述をPSLで先ず行ない，次にその仕様記述を参考にしながら別の視点に立ってプログラムを改良して，それに対する仕様記述を行なっている。旧プログラムは、入力モジュール，出力モジュール，ゲームの制御モジュールに分割されている。又，人間の手，機械の手，場，山のドミノパイは、すべて28枚のパイをあらわすリストの操作で表現してあるので、そのリスト処理のモジュールは重要な働きをする。これに対し，新プログラムは、ゲームを行なう人間，機械，およびその間にいる審判の三つのモジュールに分割されている。

旧プログラムでは、bottom-up的にプログラムを作っているのに対し、新プログラムでは、top-down的にプログラムを作っている。旧プログラムを新プログラ



第21回プログラミング
シンポジウム 1980. 1

SUMMARY OF OBJECT TYPES

OBJECT TYPE	SYNONYMS
ATTRIBUTE	ATTR, ATTRIBUTES
CONDITION	COND, CONDITIONS
ELEMENT	ELE, ELEMENTS
ENTITY	ENT, ENTITIES
EVENT	EV, EVENTS, EVT
GROUP	GR, GROUPS
INPUT	INP, INPUTS
INTERFACE	INTF, ORGANIZATIONAL-UNITS, ORGU, ORGANIZATIONAL-UNIT, REAL-WORLD-ENTITY, INTERFACES
MEMO	MEMOS
OUTPUT	OUT, OUTPUTS
PROCESS	PROC, PROCESSES, PRC
PROCESSOR	PROCR, PROCESSORS, PRCR
RELATION	RLN, RELATIONS
RESOURCE	RSC, RESOURCES
SET	SETS
SYSTEM-PARAMETER	SYSP, SYSTEM-PARAMETERS, SYSPAR
UNIT	UNITS
UNDEFINED	

図1 PSLのオブジェクトの型

Command	Abbreviation
ASSERTION-CONSISTENCY	ASCO
ATTRIBUTES	ATTR
CHANGE-NAME	CNAM
CHANGE-NAME-TYPE	CNT
COMBINE	COMB
CONTENTS	CONT
CONTENTS-ANALYSIS	CA
CONTENTS-COMPARISON	CC
DATA-ACTIVITY-INTERACTION	DAI
DATA-BASE-SUMMARY	DBS
DELETE-COMMENT-ENTRY	DCOM
DELETE-NAME	DNAM
DELETE-PSL	DPSL
DICTIONARY	DICT
DISPLAY	DIS
DYNAMIC-INTERACTION	DI
ELEMENT-PROCESS-ANALYSIS	EPA
ELEMENT-PROCESS-USAGE	EPU
EXTENDED-PICTURE	EP
FORMATTED-PROBLEM-STATEMENT	FPS
FREQUENCY	FREQ
FUNCTION-FLOW-DATA-DIAGRAM	FFDD
HELP	HELP
IDENTIFIER-ANALYSIS	IA
INPUT-LAYOUT	ILO
INPUT-PSL	IPSL
KEYWORD-IN-CONTEXT	KWIC
LAYOUT	LO
LIST-CHANGES	LC
NAME-LIST	NL
NAME-SELECTION	NS
PICTURE	PICT
PROBLEM-NAME	PNAM
PROCESS-CHAIN	PC
PROCESS-SUMMARY	PSUM
PUNCH-COMMENT-ENTRY	PCOM
RELATION-STRUCTURE	RSTR
REPLACE-COMMENT-ENTRY	RCOM
REPLACE-PSL	RPSL
RESOURCE-CONSUMPTION-ANALYSIS	RCA
SECURITY-ANALYSIS	SECA
SET	SET
STOP	STOP
STRUCTURE	STR
SUBSET-ANALYSIS	SSA
SYSTEM	SYS
UNIT-STRUCTURE	US
UTILIZES-ANALYSIS	UTLA

図2 PSAコマンド

図3-(1) 旧プログラムのExtended Picture
(System Structure)

```
                                        Formatted Problem Statement
  Parameters:   DB=DOMI  FILE=FPSNAM  NOINDEX  NOPUNCHED-NAMES  PRINT  NOPUNCH  SMARG=5  MMARG=20  AMARG=10  RMARG=25
RMARG=70
    CMARG=1  HMARG=40  SEVERAL-PER-LINE  COMMENT  NONEW-PAGE  NONEW-LINE  NOAIL-STATEMENTS  COMPLEMENTARY-STATEMENTS  L
INE-NUMBERS
    PRINTEOF  DLC-COMMENT  NOSORT-NAME-LIST
     1 DEFINE PROCESS                              domino;
     2      /*   DATE OF LAST CHANGE = Nov 19, 1979, 11:19:41 */
     3      SYNONYMS ARE:  domi;
     4      DESCRIPTION;
     5      DOMINO
     6        This system presents all environment to play DOMINO.
     7        One player could play DOMINO whith this system.
     8 ;
     9      SUBPARTS ARE:  display, playing-control, input-routine;
    10
    11 DEFINE PROCESS                              playing-control;
    12      /*   DATE OF LAST CHANGE = Nov 10, 1979, 13:15:30 */
    13      SYNONYMS ARE:  pcon;
    14      SUBPARTS ARE:  common-routine, hito, judge, kikai,
    15                     prepare-for-game;
    16      PART OF:       domino;
              .
              .
              .
              .
    22
    23 DEFINE ENTITY                              flags-and-counters;
    24      /*   DATE OF LAST CHANGE = Nov 10, 1979, 13:15:30 */
    25      SYNONYMS ARE:  check;
    26      CONSISTS OF:
    27          fcheck, passc, time, registers, passm, number-of-stock;
    28
              .
              .
              .
              .
   350
   351 EOF EOF EOF EOF EOF
```

図3 - (2)　旧プログラムのFormatted Problem Statement

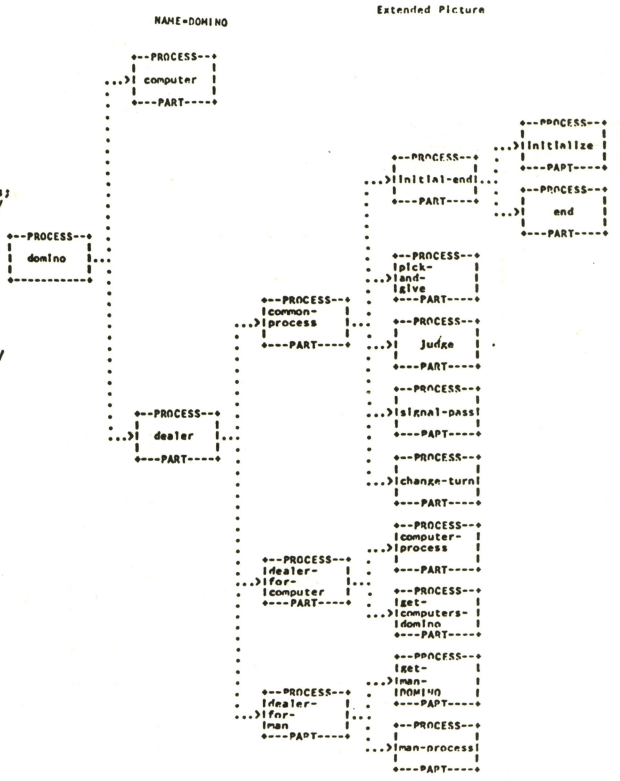
```
                                        Formatted Problem Statement
Parameters:   DB=DOMINDEX  FILE=FPSDEX  NOINDEX  NOPUNCHED-NAMES  PRINT  NOPUNCH

  RMARG=70  CMARG=1  HMARG=40  SEVERAL-PER-LINE  COMMENT  NONEW-PAGE  NONEW-LINE

  LINE-NUMBERS  PRINTEOF  DLC-COMMENT  NOSORT-NAME-LIST
  1 DEFINE PROCESS                              domino;
  2      /*   DATE OF LAST CHANGE = Nov 13, 1979, 13:33:48 */
  3      SUBPARTS ARE:  computer, dealer;
  4
  5 DEFINE PROCESS                              man-process;
  6      /*   DATE OF LAST CHANGE = Nov 16, 1979, 21:25:37 */
  7      SYNONYMS ARE:  manproc;
  8      PART OF:       dealer-for-man;
  9      UPDATES:       playing-status
 10        USING:       playing-status, man-domino;
 11      TRIGGERED WHEN:
 12                     lay-or-pick BECOMES FALSE;
 13      MAKES:         empty-hand TRUE;
 14      TERMINATION-CAUSES:
 15                     end-of-man-process;
 16
              .
              .
              .
              .
 30
 31 DEFINE EVENT                               end-of-man-process;
 32      /*   DATE OF LAST CHANGE = Nov 13, 1979, 13:33:48 */
 33      SYNONYMS ARE:  mend;
 34      MAKES:         turn, pass FALSE;
 35      ON TERMINATION OF:
 36                     man-process;
 37
              .
              .
              .
              .
 44
 45 DEFINE CONDITION                           lay-or-pick;
 46      /*   DATE OF LAST CHANGE = Nov 15, 1979, 15:08:15 */
 47      CHANGED BY:    get-man-domino;
 48      BECOMING FALSE TRIGGERS:
 49                     man-process;
 50      MADE TRUE BY:  pick-and-give;
 51      BECOMING TRUE TRIGGERS:
 52                     pick-and-give;
 53      TRUE-WHILE;
 54          input is pick up request;
 55      FALSE-WHILE;
 56          input is domino-pai;
 57
              .
              .
              .
              .
363
364 EOF EOF EOF EOF EOF
```

図4-(2)　新プログラムの
Formatted Problem Statement

図4 - (1) 新プログラムのExtended Picture
(System Structure)



ムに書き直すとき，その仕様をPSLで記述してあれば，どれだけ直し易いかを経験的に評価することが，この例題を記述した目的である。

3．試用による評価

3．1　人間機械系として見た評価

ISDOSは，仕様記述を機械に支援させて行なうシステムであり，システム設計者，あるいはプログラマが便利に使える道具であるべきである。このためには，人間と機械とのインターフェースが良く設計されていて，手ざわりの良いシステムとなっていなければならない。このような観点からISDOSを評価すると，次のような特徴や問題点がある。

(1) 入力

PSLで書かれた仕様は，Input-PSL（IPSL）コマンドを介して入力され，PSAはこれをデータベースに変換する。仕様記述は，それを表わすデータベースが重要であるから，入力ソースというものは保存しておかない。個々のPSLの文は，入力時に一つづつその構文をチェックされ，構文的に正しければ，処理される。PSLはstepwiseに記述を追加してゆくことを前提としているので，たとえば未定義のオブジェクトを参照していても，入力時には誤りとはせず，入力文字列通りの処理をしてしまう。

そこで，入力時にオブジェクトの綴りを誤った場合，その修正が面倒である。入力ソースが保存してあるならば，エディタで綴りを修正すれば良いが，既に誤った綴りに基づいてデータベースができている。そこで，次の修正手続きをとる。

i) Formatted-Problem-Statement（FPS）コマンドで，誤ったオブジェクトを出力

ii) Delete-Name（DN）コマンドで，そのオブジェクトを削除。（同時に，この誤ったオブジェクトを参照している箇所は，すべて削除される。

iii) オブジェクト名を修正し，i)で得られた，誤ったオブジェクトに関する記述を回復するために，改めて入力する。

PSAには，Change-Name（CNAM）コマンドでオブジェクト名の全般的な変更ができるが，既に登録されているオブジェクト名，又はそのsynonymを変更先の名前に指定すると，拒否される。従って，CNAMコマンドを綴り誤りの修正には使えない。誤り修正の例を，図5に示す。

PSAは，Input-PSL，Delete-PSL，Replace-PSLで入力した文の構文が誤りであると，それを無視して，その次の文から処理を続行する。オブジェクトの定義文が誤っていた場合，以下に書いた関係を示す文はすべて意図に反したオブジェクトに対する記述と解釈され，データベースが作られてしまう。従って，IPSLでPSLの文を入力するのは会話型で行ない，入力した文が誤っていれば，その場で正しく修正することが望ましい。ISDOSでは，FORTRANをベースにしているので，一括処理の影響が大きい。しかし，仕様記述はstepwiseに作成してゆくものであり，会話型処理を基盤とすべきである。

(2) 出力

PSLの記述に対応するデータベースが一旦できると，いくつかのPSAレポート

コマンドを使って、たくさんの形式のドキュメントが得られる。システム設計の過程，多数の人の参加によるプロジェクト管理などでは，人間にとってわかり易いドキュメントを簡単に出力できることが望ましい。この点では、PSAはかなりの努力がなされ，文章によるレポートだけでなく，視覚も利用した理解し易さを求めるために，図形レポートが何種類か用意されている。

図1や図3は，プロセス間の関係を示すSystem-Structureの図形レポートである。これは，Extended-Picture（EP）コマンドで一つのプロセスを指定すると，そこからデータベースをたどり，指定した属性の関係を図式的に出力する。図6には，データ構造についての図式レポート，図7は，プロセスと事象や条件との関係を指定したオブジェクトからたどって作成した図式レポート（Process-Chain）である。又，指定したオブジェクトに対して，記述されている関係をすべて表示したのが図8（Picture-Report）である。

オブジェクト間の関係，たとえばプロセスとそれが処理するデータとの関係を行列として出力したのが図9（Element-Process-Usage-Report）である。データの仕様変更をした場合など，どこに影響が及ぶか，一目瞭然となる。

このような工夫をして，仕様記述の理解し易さを高めることをねらっており，ある程度の成果は上がっている。

（3）レポートの量

ISDOSでは，いろいろの形式のレポートの出力が可能であり，いろいろの視点から仕様記述を理解することができる。出力の指定はPSAのレポートコマンドで簡単に行なえるから，大量のレポートが出力される可能性もある。たとえば，仕様記述全体を表わすFormatted-Problem-Statementでは，各オブジェクトに関し，指定されているすべての関係を出力している。IPSLで入力したPSLのソースではオブジェクトの関係を一つの動詞だけを使って記述していても，PSAが自動的にその動詞のcomplement（たとえば，能動形に対する受動形）を使った記述を生成してくれる。従って，FPSの出力は，PSLソースの2～3倍になっている。

仕様記述の読み易さを高めるためには，むやみに大量のレポートを出すことは望ましくない。全く初めて対象システムの仕様を読む人にとって，たとえば，Formatted Problem Statement，オブジェクト名の一覧表であるName List，System Structureや，Data Derivationに関する全体のExtended Picture，動的な様態を示すいくつかのProcess Chainなどが揃っていれば，対象システムのある程度の理解が可能となろう。これらの量は，PSLソース入力の数倍となることは確かである。

3．2　設計の道具としての評価

新しいソフトウェアシステムの開発に際して，ISDOSを設計の道具として使った場合，どの位使い易く，効果があるのかという観点から評価する。

（1）仕様記述の詳細度

設計の道具として，対象システムの機能，構造，性質を充分細かく記述できなければならない。PSLでは，Processの行なう手続きの内容は，Comment Entryとして書くので，その内容に関する処理は行なわれない。一方，システム内にあ

```
                                        Name List
      NAME                      TYPE                SYNONYM
    1 fcheck                    *** Undefined ***     -                    Nov 12, 1979 10:13:21
    2 input-domino              *** Undefined ***     -                    Nov 12, 1979 10:13:21
        .                           .                 .                        .
        .                           .                 .                        .
        .                           .                 .                        .
        .                           .                 .                        .
   47 initiallze                PROCESS               -                    Nov 10, 1979 13:15:30
   48 inpr                      PROCESS               -                    Nov 15, 1979 10:47:26
   49 input-routine             PROCESS             inputr                 Nov 12, 1979 10:13:21
   50 janken                    PROCESS               -                    Nv 10, 1979 13:15:30
 Enter command (and any parameters)
FPS N=inpr
                                        Formatted Problem Statement
    1 DEFINE PROCESS                        inpr;
    2      /*   DATE OF LAST CHANGE - Nov 15, 1979, 10:47:26 */
    3     RECEIVES:
    4          domino-or-pass-or-pickup;
    5
    6 EOF EOF EOF EOF EOF
 Enter command (and any parameters)
DNAM N=inpr
                                        Delete Name Report

  Deleted - inpr
 1 names input. 1 names deleted.
 Enter command (and any parameters)
IPSL U
                                        IPSL Input Source Listing
DEF PROCESS inputr;
      1 >DEF PROCESS inputr;
RECEIVES  dorp;
      2 >RECEIVES  dorp;
EOF
      3 >EOF
 3 lines processed with 3 statements.
```

図5　綴り誤りの修正の例

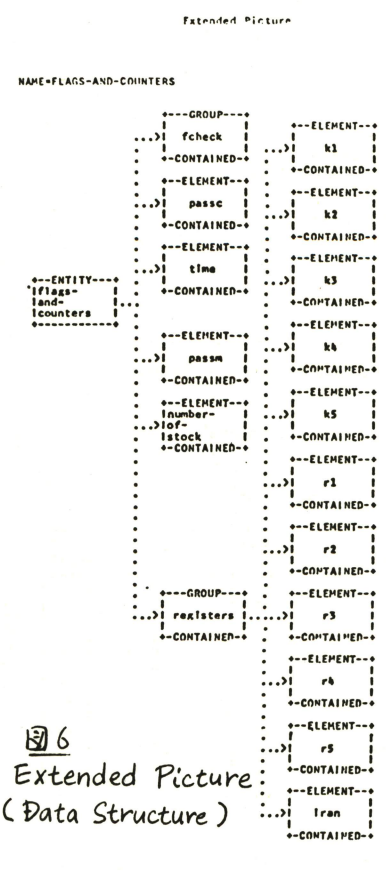


図6
Extended Picture
(Data Structure)

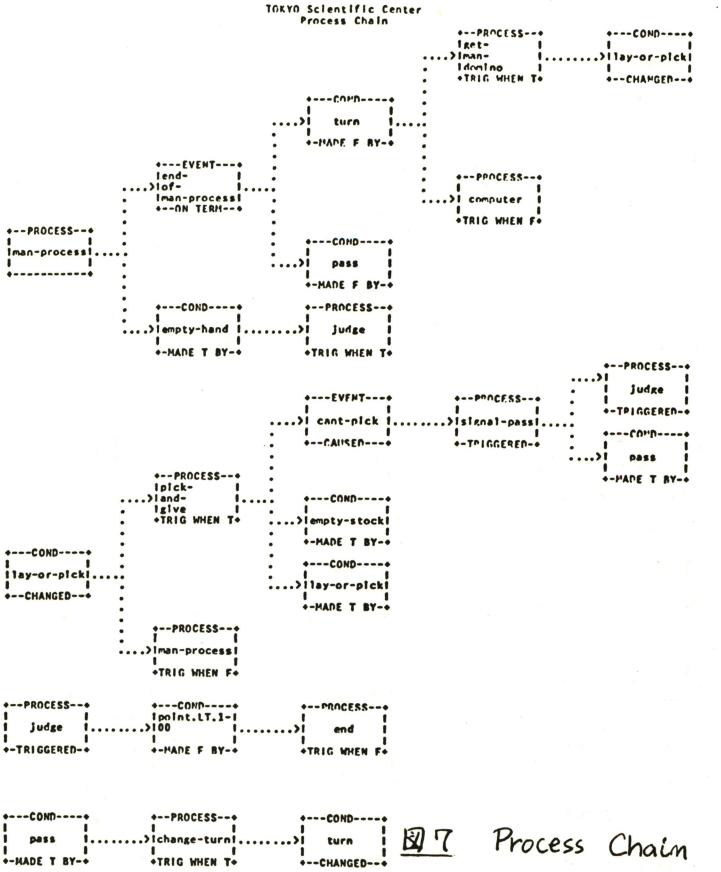


図7　Process Chain

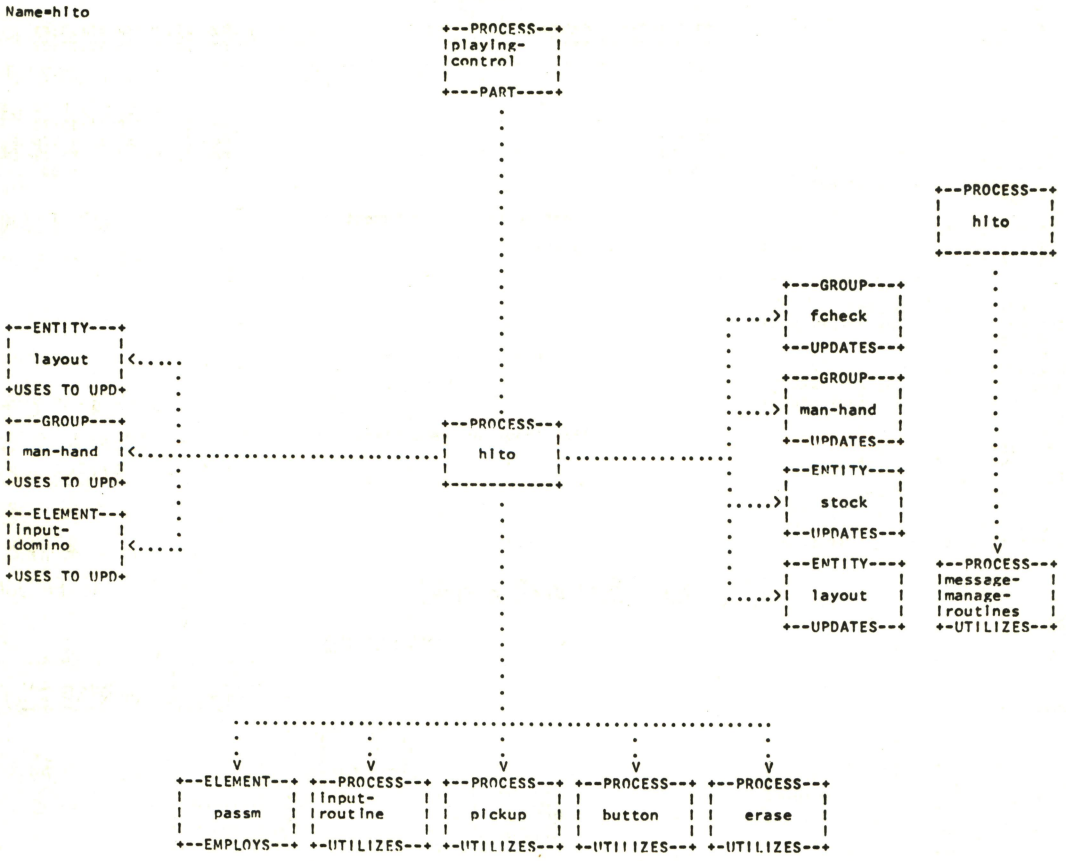


図8 Picture Report

Element Process Usage Report

ELEMENT PROCESS INTERACTION MATRIX

Row Names

1 change-turn PROCESS
2 initialize PROCESS
3 end PROCESS
4 pick-and-give PROCESS
5 judge PROCESS
6 signal-pass PROCESS
7 computer PROCESS
8 computer-process PROCESS
9 get-computers-domino PROCESS
10 get-man-domino PROCESS
11 get-computer-domino PROCESS
12 man-process PROCESS

Column Names

1 domino-pai ELEMENT
2 this-turn ELEMENT
3 mans-point ELEMENT
4 computers-point ELEMENT
5 number-of-man-hand ELEMENT
6 number-of-stock ELEMENT
7 number-of-computer-hand ELEMENT
8 pick-up-request ELEMENT

	12345678
1	
2	*******
3	
4	
5	
6	
7	*
8	*******
9	
10	
11	
12	********

DATA PROCESS INTERACTION MATRIX

Row Names

1 change-turn PROCESS
2 initialize PROCESS
3 end PROCESS
4 pick-and-give PROCESS
5 judge PROCESS
6 signal-pass PROCESS
7 computer PROCESS
8 computer-process PROCESS
9 get-computers-domino PROCESS
10 get-man-domino PROCESS
11 get-computer-domino PROCESS
12 man-process PROCESS

Column Names

1 playing-status ENTITY
2 man-hand ENTITY
3 computer-hand ENTITY
4 point GROUP
5 number-of-computer-hand ELEMENT
6 number-of-man-hand ELEMENT
7 computer-domino ENTITY
8 last-pai GROUP
9 man-domino INPUT

	123456789
1	
2	***
3	
4	
5	
6	
7	* **
8	* *
9	
10	
11	
12	* *

図9 データとプロセスの相互関係

られるデータの構造は、ENTITY, GROUP, ELEMENT, SETの形のオブジェクトを使って詳細に記述でき、又、それがとる値もVALUEを使って与えることができる。図6には、その一例が示されている。システムのモジュール化を設計する時には、処理の対象となるデータが重要な点になるので、データ構造をきちんと記述できる機能は、効果的である。

又、システムの動的な状況をあらわすために、PROCESS, EVENT, CONDITIONの三つの形のオブジェクトを使い、細かな記述が可能である。図7には、その一例が示されている。

(2) チェック機構

与えられた仕様記述に矛盾がないこと(Consistency)、必要な記述がすべて与えられていること(Completeness)をチェックする検証機能は、ISDOSのようなシステムでは必要な側面である。ISDOSの最初のVersionには、COMPLETENESSというPSAコマンドが用意されていたが、現在のVersion(Version A5.1)では、それは削除されている。一方、PSLにASSERT文を設け、そこで、指定された条件とオブジェクトがもつAttributeや、System Parameterとの一貫性を調べるASSERTION-CONSISTENCYというPSAコマンドが追加されている。

与えられた記述のcompletenessのCheckは、記述をstepwiseに完成してゆくことを基本にすれば、困難となる。従って、陽に与えられたASSERT文の処理だけをすることに後退している。

自動的にPSAがチェックするのは、IPSLなどのコマンドで入力したPSLの構文の正しさだけである。結局、コンパイラが持つ程度の能力をもったシステムといえよう。

(3) Stepwiseな記述

対象システムを設計する過程においては、topdown的に仕様を作ってゆくのが普通である。勿論、一度に仕様が完成するわけではなく、stepwiseに記述してゆく。この思考過程の各段階ごとに、PSLを使って仕様を書き、その解析レポートから次の段階に対する示唆が得られれば、設計が容易になる。

ISDOSでは、このような使い方を想定して、PSLの記述が完全でない、すなわち参照しているオブジェクトがまだ定義されていなくても、正しい入力として受け付ける。むしろそのようなオブジェクトは、未定義という型をもつものとして登録されるので、次の段階でName-Listをとり、未定義のオブジェクトがどれだけあるかをチェックできる。

前述したように、オブジェクト間の関係をする動詞で記述すると、その相手となったオブジェクトの側には、その関係のcomplementを用いた同一の関係の記述が自動的に与えられる。したがって、未定義のオブジェクトも、参照されたときどんな積りの型のオブジェクトとして使うかという記述はあるはずなので、定義する段階でそれと矛盾するような記述をすると、誤りとなる。(図10)

PSLの記述量が多くなると、それを正しく理解して、次の段階に進むのが困難となる。あるいは、記述の欠落したとえば、あるオブジェクトが他の何の関係も持たずに孤立していることなども気がつきにくい。このような場合, Extended Pictureあるいは Picture Reportを見れば、視覚的に判断がつく。このような解析し

ポートを利用すれば、前段階での書き忘れの部分を見つけ易い。これを、機械の支援によって行なえれば、人間の能力を本質的な部分に向けることができる。

(4) 記述の変更

既に処理されてデータベースとなっているPSLの記述を変更する場合、いくつかのPSAコマンドが用意されている。

- Delete-Name ・・・ オブジェクトと、それが持っていた関係をデータベースから消去。
- Change-Name ・・・ オブジェクトの名前を変更。それを参照していたところでも名前を変更。
- Delete-PSL ・・・ オブジェクトのもとに記述してある指定した関係を消す。
- Replace-PSL ・・・ オブジェクトのもとに記述してある関係の一部を変更する。
- Replace-Comment-Entry ・・・ オブジェクトに記述してあるComment Entry を変更する。

このような変更において、矛盾の生じないようなチェックが行なわれるので、これらの変更用コマンドを適当な順序で使わないと、意図した変更が行なわれないことがある。

(5) 設計用道具としての総合的評価

ISDOSは、ソフトウェア生産のライフサイクルにおいて、いくつか機能を果すことを狙っている。たとえば、多数の人間が参加する大規模プロジェクトにおいて、ドキュメンテーションが機械の支援で生成できることは、非常に効果的であろう。

設計用道具としては、仕様の検証が自動的に行なわれるならば、威力を発揮するであろう。現在のPSAは、機械に自動的に仕様の一貫性や、完全性をチェックするのではなく、その作業は人間にまかせ、その代り、人間の判断の資料を豊富に提供しようとしている。この方針は、現実的なものであり、PSL/PSAに習熟した人にとっては、このシステムは設計用道具として充分有効に働くであろう。

4. 拡張したい機能

PSL/PSAを使用した経験から判断すると、現在のPSL/PSAに対して次のような機能を追加すると、更に使い易いものとなる。

4.1 チェック機能

stepwiseに仕様の記述を行なえるために、PSLの記述の完全性をチェックすることは不可能である。しかし、人間が完全に仕様を記述してしまったと考えた後では、PSLの記述が完結しているかどうかを自動的にチェックしてくれる方がよい。

4.2 入力の工夫

PSLの入力は、会話型を主体としなければならない。このとき、あるオブジェクトに対して、どんな関係が使えるか、などの情報が、システム側から示されると便利である。たとえば、図11に示すように、FPSのオプションALL STATEMENTを指定すると、オブジェクトの使えるすべての関係がコメントとして出力される。

```
FPS N=fcheck
                                                    Formatted Problem Statement
     1  /* UNDEFINED NAME -                    fcheck
     2   DATE OF LAST CHANGE - Nov 10, 1979, 13:15:30
     3     CONTAINED IN:  flags-and-counters;
     4     UPDATED BY:    rename;
     5      */
     6
     7 EOF EOF EOF EOF EOF
 Enter command (and any parameters)
NS S='UNDEFINED'
 79 names(s) in data base.
                                                          Name Selection
     1    computer-hand                      *** Undefined ***
     2    computer-point                     *** Undefined ***
     3    cpoint                             *** Undefined ***
     4    fcheck                             *** Undefined ***
            .                                      .
            .                                      .
            .                                      .
            .                                      .
    24    time                               *** Undefined ***
 24 names output.
 Enter command (and any parameters)
IPSL U
                                                  IPSL Input Source Listing
DEF PROCESS fcheck;
     1 >DEF PROCESS fcheck;
 *** ADEFST(6) Error - "fcheck" cannot be typed PROCESS because of its existing relations. It will be ignored.
 *** ADEFST(4) Warning - No valid names in section header. Statements in this section are checked for consistency, but n
o actions
     will be taken.
 1 lines processed with 1 statements.
 1 Errors.
 1 Warnings.
```

図10　未定義オブジェクトの扱いと入力時のチェック

```
                          Formatted Problem Statement
 1 DEFINE PROCESS                              man-process;
 2      /*   DATE OF LAST CHANGE - Nov 16, 1979, 21:23:37 */
 3     SYNONYMS ARE:  manproc;
 4   /* DESCRIPTION                                                    */
 5   /* ASSERT                                                         */
 6   /* ASSERTED BY       TO BE     FOR                                */
 7   /* ATTRIBUTES ARE                                                 */
 8   /* KEYWORDS ARE                                                   */
 9   /* SEE-MEMO                                                       */
10   /* RESPONSIBLE-PROBLEM-DEFINER IS                                 */
11   /* SOURCE IS                                                      */
12   /* TRACE-KEY IS                                                   */
13   /* GENERATES                                                      */
14   /* RECEIVES                                                       */
15   /* SUBPARTS ARE                                                   */
16      PART OF:       dealer-for-man;
17   /* ACCESS-RIGHT IS                                                */
18   /* SECURITY IS                                                    */
19   /* ADDS         TO                                                */
20   /* CREATES                                                        */
21   /* DERIVES          USING                                         */
22   /* DESTROYS                                                       */
23   /* MAINTAINS         USING                                        */
24   /* MODIFIES         IN                                            */
25   /* REFERENCES         IN                                         */
26   /* REMOVES          FROM                                          */
27      UPDATES:        playing-status
28        USING:        playing-status, man-domino;
29   /* EMPLOYS                                                        */
30   /* HAPPENS          TIMES-PER                                     */
31   /* CAUSES                                                         */
32   /* CHANGES                                                        */
33
            .
            .
            .
            .
59
60 EOF EOF EOF EOF EOF
```

図11　Formatted Problem Statement の
All Statement オプションの利用

このような示唆があれば、正しい入力をすることが容易になる。

4．3　出力の工夫

PSAの出力は、いろいろの工夫がなされており、随分見やすいレポートになっている。図式レポートなどは、その一つである。実際に対象システムを理解しようとすると，システムの全体像を何らかの形式でつかみたいと思う。現在のExtended Pictureや，Picture Reportでは，出力装置の機能による制限などで、全体像の出力は不可能である。たとえば、プリンタの出力を何枚かつなぐと、全体の構成が一目でわかるような工夫はできないか。

4．4　プログラムとのつながり

PSLは、あくまでも仕様記述であり、それを実現するプログラムとの関係は，何も記述していない。プロセスの具体的な動きは、Comment Entry のPROCEDUREを使って記述している。PSAは、この内容に関しては何の処理もしない。ここに具体的な実現のプログラムを記述し，仕様記述で指定されたインターフェースとの整合性をチェックしたり，実現のプログラムのソースコードだけを取り出すレポートコマンドを設けたりして、仕様記述と実現との間にPSAが介入することが考えられる。

4．5　記述の階層化

PSLの記述自体には構造はなく，オブジェクトの定義が羅列しているだけである。これに対し、記述の上でのブロック構造や，モジュール構造を導入し，名前の重ねあわせなどを許すことが考えられる。又，この構造と，記述している対象システムの構造とを関連づければ、前述の一貫性や完全性のチェックを，システムの一部分に限定して行なうことができる。

4．6　日本語化

ソフトウェアシステムの仕様記述の言語として一番強力なものは自然言語である。詳細な意味を記述するのは、それ以外にはない。PSLも、自然言語の機能を少しでも取り入れようとしている。

この場合，自然言語風に記述した仕様がどの位理解し易いのかを判断するためには、日本語風に記述したもので実験した方が良い。

PSLは，自然言語風といっても、非常に制限した文法を持っているので、PSAの前後にサブプロセサを実現すれば、比較的容易に日本語風のPSLに変換することができよう。

5．おわりに

ISDOSシステムのPSL/PSAを使って例題を記述し，その使い勝手や，有効性を試してみた。完全性のチェックなどに期待はずれの点も見られたが、むしろ、機械は人間の判断に寄与する資料を豊富に生成するという方針でシステムが作られていると考えられる。PSL/PSAの実際の使用状況を想定すると，この方が現実的であり，効果も上がるであろう。

PSL/PSA の機能は、我々が今まで経験したもの以外にもいくつか用意されて

いる。たとえば、プロジェクト管理の機能がその一つであり、ある場面においては非常に有効に働くであろう。

この研究は、日本IBMのTSCフェローシップ・プログラムにより行なったものである。

御討論頂いた、東京大学　米田信夫、慶應義塾大学　土居範久、東京工業大学　米沢明憲の諸氏、ならびに関係者各位に感謝する次第である。

参考文献

1) ソフトウェア工学 ── 要求仕様技術, bit　臨時増刊, Aug 1978
2) Problem Statement Language Language Reference Summary (A5.1)
ISDOS Project, Sep 1978
3) Problem Statement Analyzer Command Reference Summary (A5.1)
ISDOS Project, Jan 1979