U.D.C. 681.32.06.002:519.683.2:003.63

図面を用いたプログラム開発方式と支援ツール“SDL/PAD”

Visual Programming Technology and Support System “SDL/PAD”

前澤裕行*　*Hiroyuki Maezawa*
小林正和*　*Masakazu Kobayashi*
小滝房枝*　*Fusae Odaki*

ソフトウェアの生産性向上には，大多数を占める中級，初級プログラマを対象とした開発技法，支援ツールの整備が不可欠である。複雑なソフトウェアの構造のビジュアル化はそのための有効な手段である。ビジュアルプログラミングは，図形を用いた新しいプログラム開発方式であり，プログラム自動生成システムはその支援を行なう対話形の設計，プログラミングシステムである。文章形式に対する図形式表現での理解のしやすさの優位性を実験的に検証し，これに基づき図形式の対話エディタ及び図形式記述されたプログラム仕様とソースプログラムとの双方向自動変換機能をもつ統合化システムである。構造化設計，構造化プログラミング技法を誘導する機能を備え，プログラムの開発，保守に適用できる。

1 緒　　言

プログラムは，設計者の意図を計算機に伝えるためのコミュニケーション手段である。データが計算機に入力されたとき，それをどのように加工して新しいデータとして出力するかを計算機に指示したもの，あるいは人間の知識自体を計算機に伝承させるため記述したもの，それがプログラムである。

人と計算機とのコミュニケーション手段としてプログラムを定義するとき，その表現手段は，計算機に都合のよいものでなく，人間が考える能力を助けるものであることを基本とすべきである。かつて，プログラムの表現手段としては数値を用いた。いわゆる機械語である。この表現形式は，その名の示すとおり極めて機械寄りのものであったので，これを少し人間寄りにしたものとして，COBOL，FORTRANなどの高級言語が開発された。以降，プログラムをより人間の扱いやすい表現にするための改良が続けられ，最近では，関数形言語やオブジェクト指向言語，述語論理形言語が注目を集めている。

これらの表現形式が，いずれも文章形式であるのは奇妙な事実である。人と人とのコミュニケーションでは，文章以外に，表，図といった表現手段が多く用いられる。表，図が人の考えを整理し，伝達する手段として文章に優る点があるのはよく知られた事実である。ハードウェアの開発では古くから図面が主なコミュニケーション手段であり，プログラムの開発でも，最近では計画，システム設計といった上流工程では仕様の図表現が主流となりつつある。

現在まで，プログラム言語の表現が文章形式に限定されていたのは，プログラマと計算機との対話環境が，多くの人々が対話というとキーボードを思い浮かべるように，文字情報に限定されていたことが主な原因となっている。もちろん，図形処理手段も存在したが，プログラミングでの対話手段として採用するには，応答性が遅いこと，CRT(Cathode Ray Tube)画面の密度が低く表示できる画面の大きさが制限されていたこと，などがあい路となっていた。しかし，近年，マイクロコンピュータによる端末のインテリジェント化が急速に進んで応答性の問題が解消され，CRTの表示密度も1,000ドット×1,000ドット程度の高密度のものが普及し，マウスなどの新しい高操作性をもつ入力手段が開発された，など図形表現での計算機との対話環境が急速に整ってきた。

このようなハードウェアの発達を背景として，従来の文章形式に代わる図形形式を用いたプログラミング方式(ビジュアルプログラミング)が生まれ，これを支援するツールも開発されている[1]。本稿では，ビジュアルプログラミングの有効性及びツール化の要件，実現方式について述べる。

2 図形表現法と文章表現法

ソフトウェアの開発に用いる図形表現法としては，古くはフローチャートに始まり，データフロー図，状態遷移図など多くの記法が提案されている[2),3)]。J. Martinは，これらの記法を1冊の著書[4)]にまとめ，ソフトウェアの開発には図形を用いるべきであると主張している。

はたして，プログラムの記述形式として，図形は文章よりも優れているのであろうか。この問題については，(1)考えるための手助けとなる(考えやすさ)，(2)理解のしやすさ(読みやすさ)，(3)記述のしやすさ(書きやすさ)など種々の評価尺度があり，明確な結論は出ていない[5),6)]。ここでは，(1)「図形表現と文章表現での理解のしやすさ」の差の有無，(2)「理解のしやすさ」の差はどの程度のプログラムの複雑さで表われるか，について実験した結果[7)]の概要を紹介する。

(1)　実験方法

図形表現と文章表現の特徴が最も顕著に表われているものとして，プログラムの処理手順構造記述を対象として，理解のしやすさの側面から比較評価した。実験に当たっては，理解のしやすさを，「論理が追いやすい」という側面からとらえ，プログラムの複雑さは条件文の入れ子の数としてとらえた。被験者は47人，全員が男子で平均年齢24.5歳，入社後1～2年の人々である。これを2グループ，すなわち文章形式で記述した処理手順を読み，内容を理解して質問に答えるグループと，それと同一内容の処理手順を図形形式で記述したものをもとに同様な質問に答えるグループに分けた。プログラミング経験期間の平均値は，文章グループが32箇月，図グルー

* 日立製作所システム開発研究所

```
READ  USERID
CHECK  USERID
IF    ERROR
   THEN      OUTPUT  ERROR  MESSAGE
ENDIF
DO  UNTIL    DATA='EOF'
   READ  NEXT  DATA
   DO  CASE  DATA=
      KANJI : STORE  DATA  INTO  KANJI  AREA
      KANA : STORE  DATA  INTO  KANA  AREA
      OTHERS : OUTPUT  ERROR  MESSAGE
   ENDDO
ENDDO
RETURN
```

図1　SDLによるプログラム仕様の記述例　プログラムの処理手順をIF, THEN, ELSE, DOなどのキーワードで構造化して記述してある。

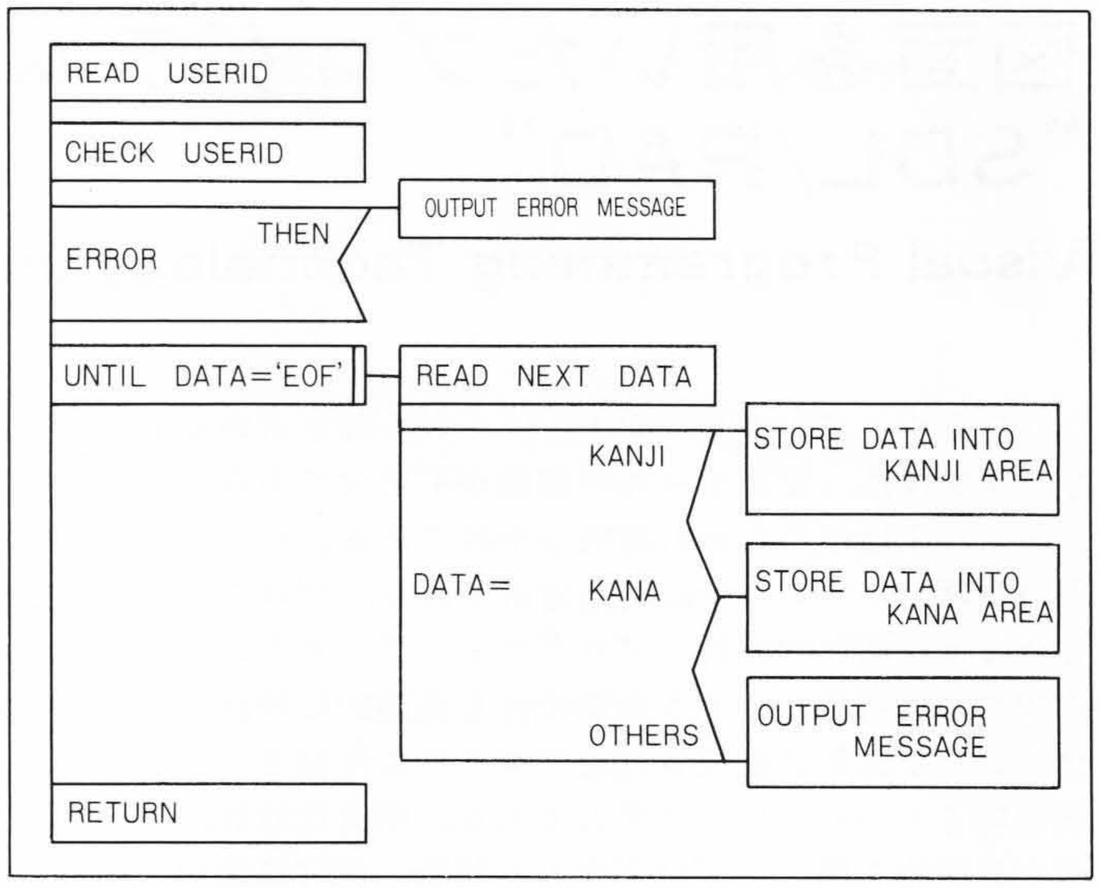

図2　PADによるプログラム仕様の記述例　図1と同一内容を連接(□□)，反復(□□)，選択(□<)などの記号を用いて記述してある。

プが17箇月であり，過去作成したプログラムのステップ数平均値は，それぞれ3.7kステップ，2.5kステップである。

文章形式の記述法としては，IF，THEN，ELSE，DOなどのキーワードを用いるSDL(Software Design Language)を用い，図形式としてはPAD(Problem Analysis Diagram)[8]を用いた。SDL及びPADの記述例を**図1**及び**図2**に示す。実験のために作成した六つのテスト問題の特性は**表1**に示すとおりである。同表で，分岐数とは条件文の分岐のネスト数を示す。キーワード種類数はSDL文内で用いられるキーワードの種類の数を示す。

比較は，制限時間内により多くの質問に答えるスピードテスト形式で行なった。質問は両グループに共通とし，テスト結果をx^2検定を用いて検討した。検定した帰無仮説は『文章形式(SDL)と図形形式(PAD)とは「理解しやすさ」に差異がない。』である。

(2)　実験結果及び考察

スピードテストの結果は**表2**に示すとおりである。質問1～6の全問で図グループの平均値は文章グループの平均値よりも高く，質問1，4，6で5～10%の有意差が検出された。更に，総合得点(質問1～6の総和)でも図グループのほうが文章グループよりも有意(有意水準2%)に平均値が高かった。このことは，理解のしやすさを「論理を追いやすいこと」ととらえるならば，文章形式(SDL)よりも図形形式(PAD)のほうが理解しやすいことが確認できたことを意味する。

表3は，プログラムの複雑さ(分岐数)とx^2値との関係をまとめたものである。同表によれば，文章形式(SDL)と図形式(PAD)との理解しやすさの差は，プログラムが単純なとき(ネスト数が少ないとき)には現われず，複雑(ネスト数が多くなる)になるほど顕著になると予想できる。差が現われ始める境界は，分岐数が3程度であると推察できる。

3　ビジュアルプログラミング支援ツール

プログラム自動生成システムSDL/PADは，ビジュアルプログラミングを可能とする対話形のプログラミング支援ツールである。SDL/PADは，過去，大形計算機とグラフィック端末を用いた旧タイプのものを開発し，その後インテリジェント端末を用いた新しいタイプを開発した。旧タイプのものについ

表1　実験に用いたテスト問題の特性　6種類の処理手順をSDLとPADで二とおりに記述し，制限時間内にできるだけ多くの質問に答えるテストを行なった。

特性＼問題No.		1	2	3	4	5	6
分岐数		5	5	2	3	1	5
キーワード種類数		1	1	2	5	4	4
ステップ数(ステップ)		35	57	46	38	48	46
ページ数	SDL(ページ)	1	1	1	1	1	1
	PAD(ページ)	1	2	1	1	1	1
制限時間(min)		8	12	7	12	10	10

表2　スピードテストの結果　PADを用いたグループのほうが平均値が高い。問題1，4，6及び総合得点でPADのほうが理解しやすいこと(有意差)が検証された。

問題No.	1	2	3	4	5	6	総合得点
分岐数	3	5	2	3	1	5	—
キーワード種類数	1	1	2	5	4	4	—
ステップ数	35	57	46	38	48	46	—
PAD群平均値	6.9	8.8	2.6	7.0	4.2	6.4	35.8
SDL群平均値	5.9	8.3	2.5	5.9	3.4	4.9	31.0
x^2 値	3.105	0.095	0.012	4.763	0.000	3.253	5.576
有意水準(%)	10	—	—	5	—	10	2

表3　プログラムの複雑さが理解しやすさに及ぼす影響　プログラムの複雑さを分岐の数で代表させると，分岐数が3以上になるとPADとSDLとの間で読みやすさに差が生じていることが分かる。

問題No.	5	3	1	4	2	6
分岐数	1	2	3	3	5	5
x^2 値	0.000	0.012	3.105	4.763	0.095	3.253
有意水準(%)	—	—	10	5	—	10

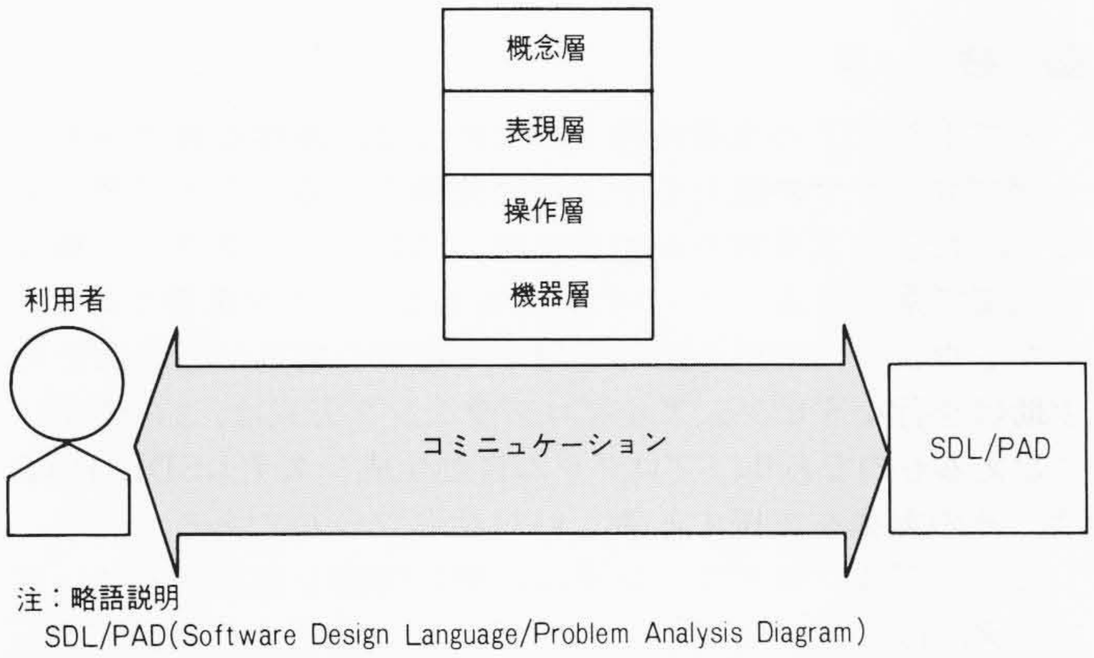

図3　SDL/PADのコミュニケーションモデル　設計者と支援ツールのコミュニケーションを四つの層に分けて，各々の層について規約を設けている。

ては，文献8)に詳しく紹介したので，本稿では新タイプのものについて，基本となる考え方，機能及び構成を述べる。

3.1　基本となる考え方

SDL/PADの対話インタフェース設計に当たっては，**図3**に示すコミュニケーションモデルを用いて，概念層，表現層，操作層及び機器層に分けて設計した。

(1)　概　念　層

プログラムの構造をプログラマが認識するときのとらえ方を規定する層である。SDL/PADでは構造化の概念を基本とした。構造化設計の概念を導入し，プログラムの構造をモジュール間の関係，モジュールのインタフェース(入出力データ)，モジュール内の処理手順，モジュール内のデータに分離して記述する方式を採用した。更に，構造化プログラミングの概念を導入し，処理手順を，連接，反復，分岐の3要素だけを用いて記述する方式を採用した。また，処理手順を概要から詳細へと段階的に記述していく方式とした(**図4**)。

(2)　表　現　層

概念層を人間の目に見えるように表現するための形式を規定する層である。SDL/PADでは，図形式を標準とした。モジュール間関係は樹木図を，インタフェース及び内部データは表形式で，処理手順はPADで記述する方式を採用した。

(3)　操　作　層

概念層，表現層で規定された記述形式を用いて，プログラムを対話的に設計する際の操作の形式を規定する層である。SDL/PADでは，操作層に以下の方式を用いた。

(a)　オブジェクト指向直接操作方式：対話での情報のすべて，すなわちプログラム構造だけでなく，挿入，削除などの命令をも，「もの(オブジェクト)」として画面上に表示し，「もの」を画面上で選択指定することにより対話を行なえるようにした直接操作方式を採用した(**図5**)。

(b)　文法誘導形編集方式：SDL/PADで用いる図面は，画素の形状，結合，配置に一定の規則(文法)をもつ。この規則をSDL/PADに内蔵化させることにより，画素の大きさ，結合，配置を自動決定し，入力を誘導する文法誘導形編集方式を採用した。

(4)　機　器　層

ハードウェアを規定する層である。SDL/PADではワークステーションを用いることにより高応答性を実現し，仮想画面の概念を導入することにより，スクロール，ズーミングなどの画面制御を容易とした(**図6**)。

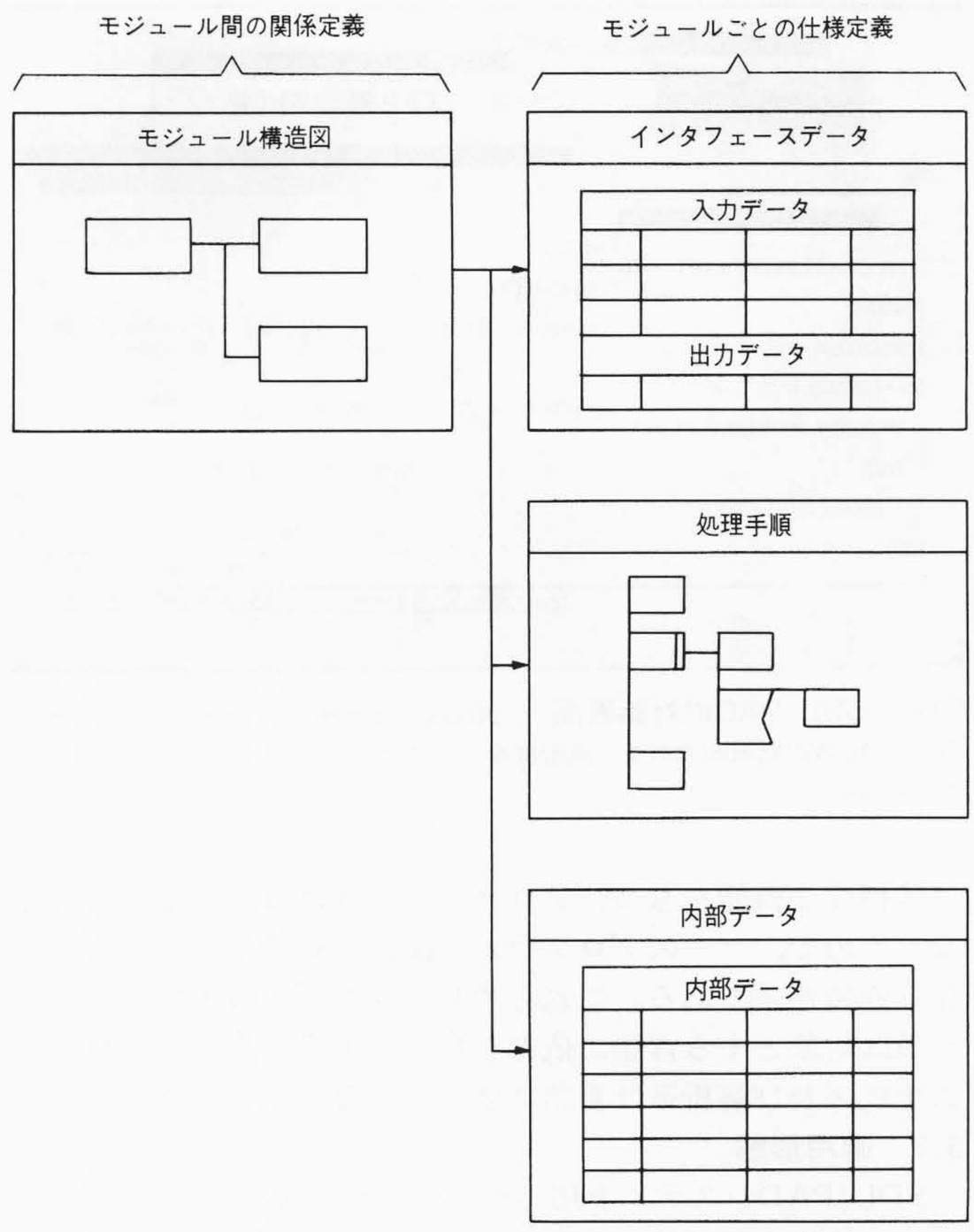

図4　SDL/PADの仕様記述モデル　概念層を規定するモデルで，構造化設計，構造化プログラミングなどの概念を導入している。

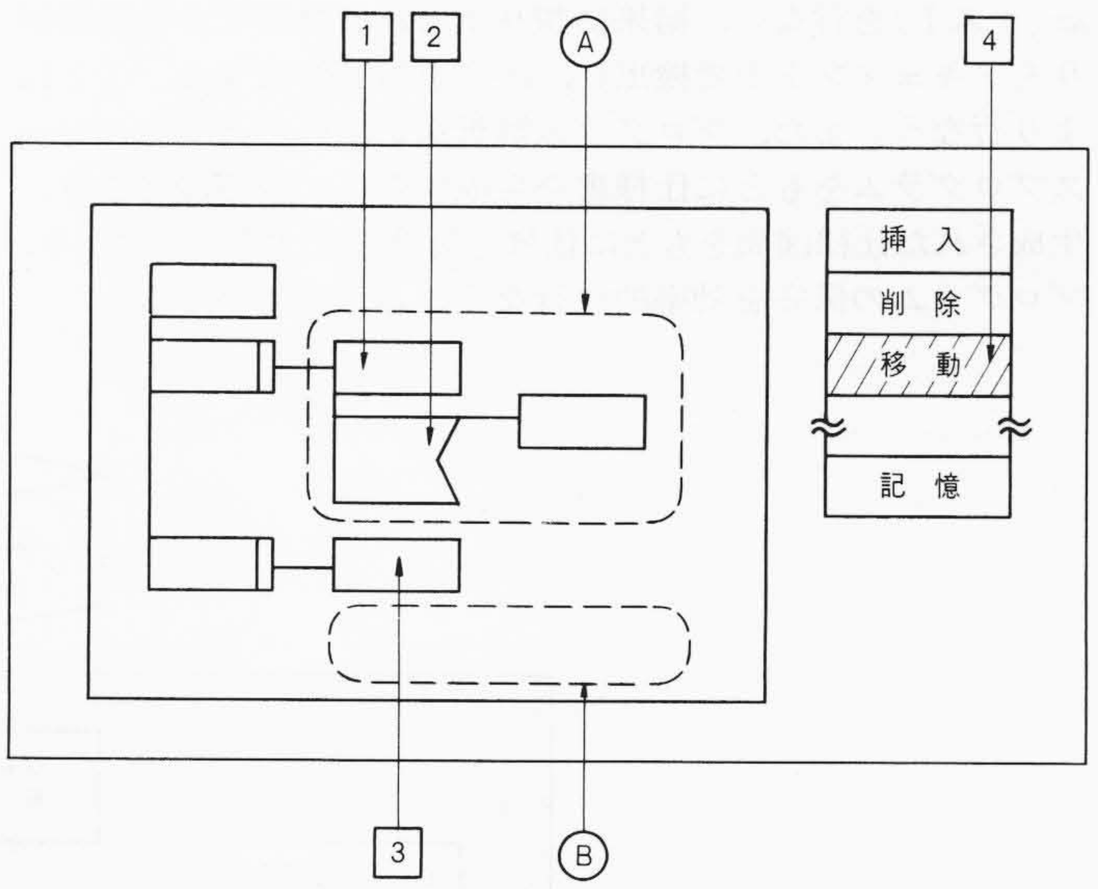

図5　オブジェクト指向直接操作方式　操作層を規定する方式である。画面上で“もの”を選択するだけで容易に操作が行なえる。例えば，Ⓐの部分をⒷの位置に移動する場合，1，2，3，4を画面上で選択するだけでよい。

3.2　機能及び構成

SDL/PADは，六つのサブシステムから構成される(**図7**)。

その中で，四つのサブシステムは，前節に述べた概念層，表現層の規定に基づき，モジュール間関係，インタフェース，処理手順，内部データをそれぞれ対話設計するための図表エディタである。プログラム生成系は，設計仕様を特定言語(COBOL，FORTRAN，Cなど)で記述したソースプログラム

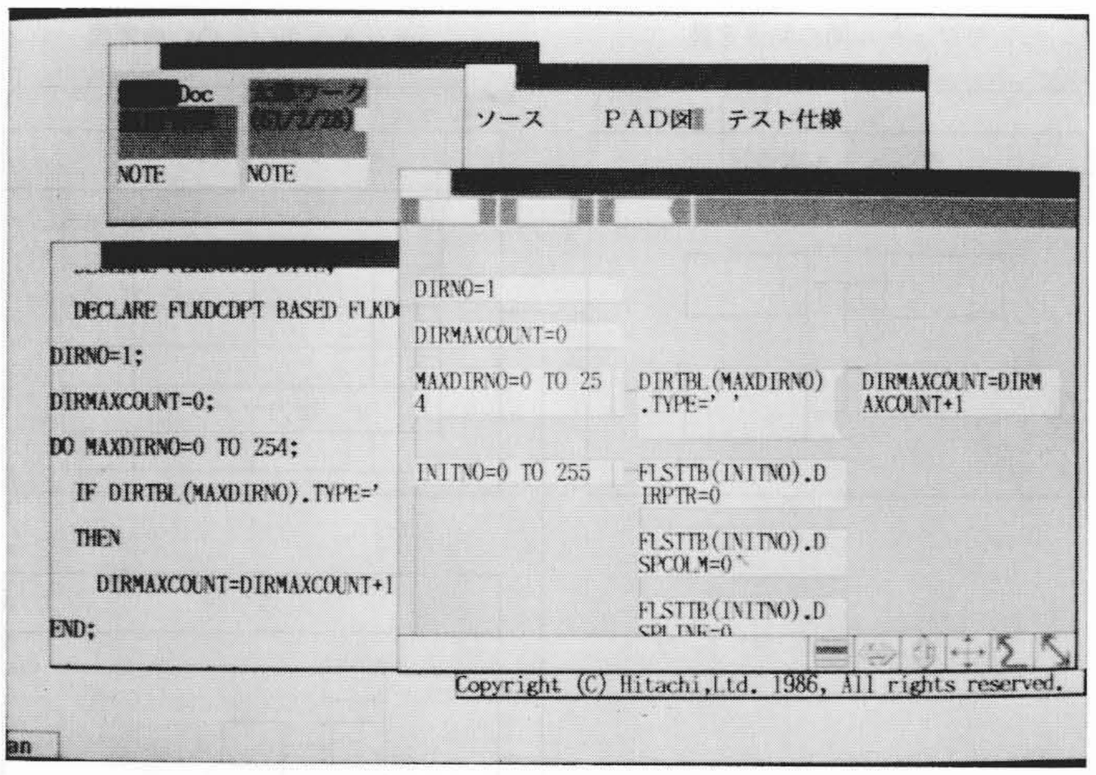

図6 SDL/PADの対話画面 2050ワークステーション上で稼動させたSDL/PADの対話画面である。高密度なCRTを用いマルチウインドウが実現されている。

に変換する機能をもつ。プログラム解析系は，生成系と対をなすもので，ソースプログラムの構造を解析して設計仕様を作る逆変換系である。これらのサブシステムの中で，エディタ系は対象とする言語に依存しない汎用系であり，プログラム生成系及び解析系は言語に依存する専用系である。

3.3 適用形態

SDL/PADシステムを用いたプログラム開発は，(1) プログラムの仕様を対話的に定義，(2) 定義した仕様をドキュメント出力し，内容をレビュー，(3) レビュー結果を反映して仕様を対話修正，(4) 仕様からソースプログラムへ自動変換，(5) ソースプログラムをコンパイルし，実行させる，(6) 実行結果の検証(テスト)を行ない，結果に誤りがあれば原因となる設計誤りをドキュメント上で検出し，誤りを対話修正する，ことにより行なう。また，プログラム解析系を用いると既存のソースプログラムをもとに仕様書を生成することが可能となり，生成された仕様図面をもとに仕様を対話修正することにより，プログラムの保守を効率的に行なうことが可能となる。

4 結 言

ソフトウェアの生産性向上のためには，高度な能力をもつ上級プログラマの能力を十二分に発揮させることも必要であるが，むしろ大多数を占める中級，初級のプログラマに焦点を当てて開発技法，ツールの整備を行なうことが重要である。ソフトウェアの構造を図形を用いて理解を容易にし，設計の手助けを行なうビジュアルプログラミング方式は，この要求にこたえるものであり，プログラム自動生成システムSDL/PADは，その実施を支援する新しい対話形ツールである。

ビジュアルプログラミングは，更に図面を実行するインタプリタ，図形式の表現でテスト デバッガを行なう図面デバッガへと発展させることにより，その基本理念を完成させることができる。SDL/PADを中心として，このような統合的環境の構築を進めている。

参考文献

1) Visual Programming, IEEE Computer, Vol. 18, No. 8 (1985-8)
2) 青山：ソフトウェア開発に用いられる図形表現法，情報処理，Vol. 25, No. 5, pp. 451～461(1984-5)
3) 佐藤：プログラミング用ドキュメンテーション，情報処理，Vol. 22, No. 5, pp. 383～389(1981-5)
4) J. Martin : Diagramming Techniques for Analyst and Programmers, Prentice-Hall(1985)
5) H. R. Ramsey, et al. : Flowcharts vs Programming Design Language : An Experimental Comparison, Proc. of the Human Factors Society 22nd Annual Meeting(1978)
6) S. B. Sheppard, et al. : The Effects of Symbology and Spatial Arrangement on the Comprehension of Software Specification, Proc. of 5th ICSE(1981)
7) 小滝，外：プログラム論理仕様記述法PDLとPADの「読みやすさ」における比較評価，ソフトウェア研究会，28-19(1983-2)
8) 前澤，外：プログラム自動生成システム“SDL/PAD”，日立評論，**66**，6，463～468(昭59-6)

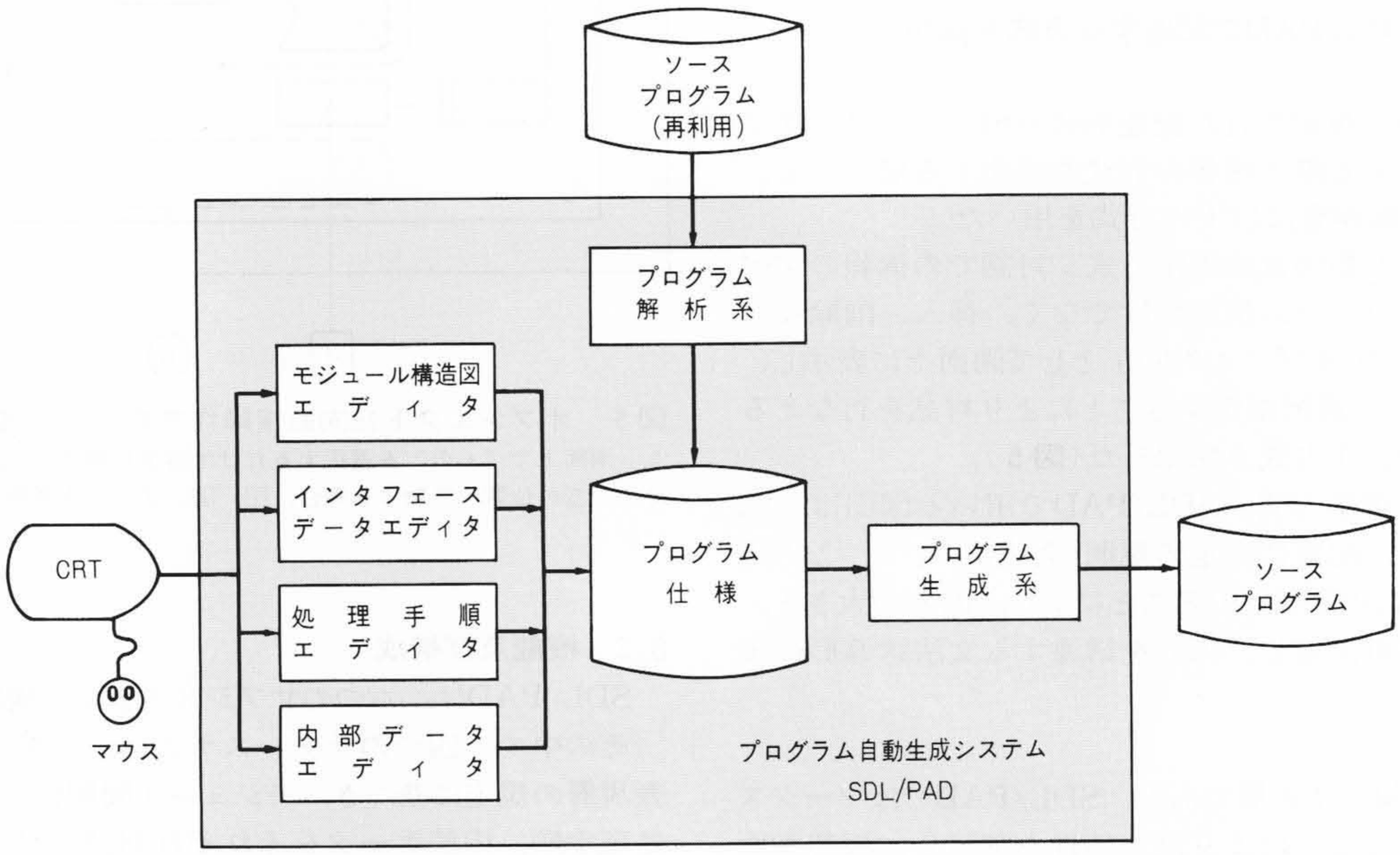

図7 SDL/PADのシステム構成 大形計算機上で稼動したものがワークステーション(T560/20マルチワークステーション)上に移殖されている。2050，2020への移殖も進められている。