U.D.C. 681.32.06.002:519.683.2

PADの開発

Development of PAD (Problem Analysis Diagram)

二村良彦*　*Yoshihiko Futamura*

PADは，日立製作所で発明され，1979年に公表された。それは流れ図に代わるプログラム図式を中心とした構造化プログラム開発技術であり，その特長は次のとおりである。

(1)　プログラムの構造を見えるようにする。
(2)　プログラムの製造(コーディング)と検査を系統的にできるようにする。
(3)　プログラムで処理するデータの構造を，プログラムの構造と同じ記法で表現できるようにする。

この技術は，日立製作所をはじめとする多数の国内企業で定着し，米国など海外にも広まりつつある。本稿では，PADの開発思想を中心に，PADとは何かについて概説する。

1　緒　　言

PAD(Problem Analysis Diagram)は，「構造化プログラム技法」と呼ばれる計算機科学での最も顕著な実用的成果に基づいたプログラム開発技術である。構造化プログラム技法は，ダイクストラ(Dijkstra)[1]，ヴィルト(Wirth)[2]など二人のチューリング賞受賞者を含む多数の研究者により，1960年代の終わりごろから提唱され，かつ普及活動が行なわれた。そして，プログラム開発に関するプログラマの意識を変革した。PADは構造化プログラム技法を実践しやすくするための「思考の道具」として，1980年に日立製作所によって提案された。そして，日立製作所での試用を通じてその有効性が立証され，現在では世界中にPADユーザーが広まりつつある。

本稿では，PADとは何か，PADの効果，普及状況，PADと他の類似の技法との関係などについて概説する。PAD自身の詳細については文献3)を，PAD開発経緯の詳細は文献4)を，またPADと類似の他技法との比較については文献5)を参照していただきたい。PADに関連したソフトウェアツールについては本特集でも取り上げられているが，その他のツールについては文献6)に多数報告されている。

2　構造化プログラム技法とPAD

ダイクストラは1960年代の終わりごろ，構造化プログラムの読みやすさ(理解しやすさ)について述べた[1]。ただし，構造化プログラムとは主として連接，反復及び選択の三つの基本形(**図1**)を用いて作成されたプログラムのことである。どのような計算でも，この三つの基本形だけを用いて記述できることはベーム(Böhm)[7]などにより証明され，広く知られていた。構造化プログラムを書くことの利点は，「頭の中で考えた計算手順と紙の上に書いた計算手順を対応しやすくする。それによってプログラムの構造を見やすくする」ということである。例えば，頭の中の計算手順が反復構造で表わされているのに，それを紙の上に書くときに名札付きの文とGOTO文の組合せにしたのでは，紙の上に書いたものが理解しにくいものになってしまう。そのようなことから，「反復構造を書く際にはGOTO文を使わないで，WHILE Q DO H;のように書くほうがよい。」というのがダイクストラの主張であった。

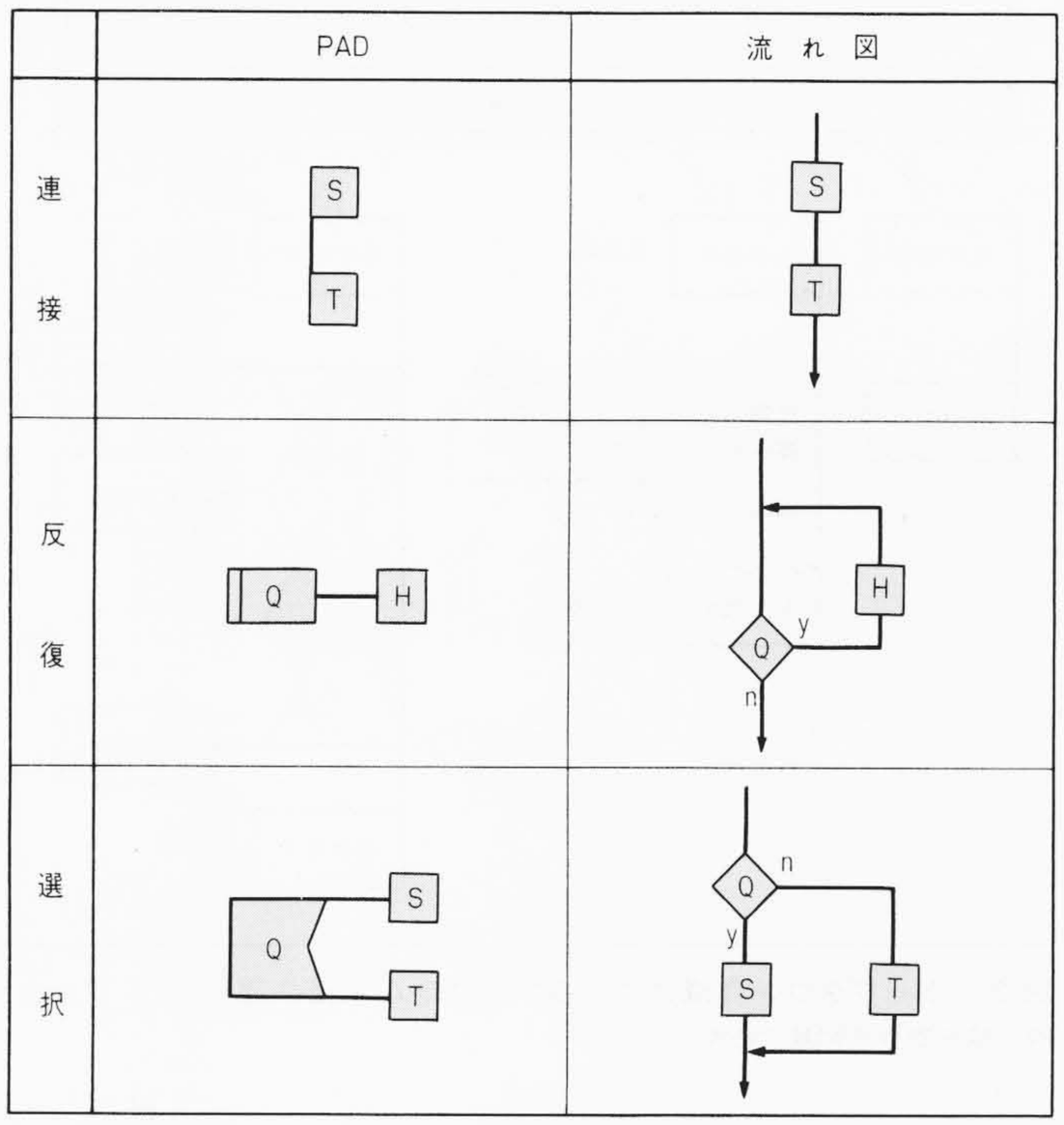

図1　プログラムの3基本形　Q は While Q の意味である(文献3)から引用)。

選択構造についても同様である。

更にダイクストラは，構造化プログラムの作成法としてトップダウン作成法(又は段階的詳細化)を示した。それは次のようなガイドラインである。「与えられた仕様(大仕様)に基づいてプログラムを作る際に，一つの大きなプログラムを一度に作ろうとしてはならない。まず，トップダウン的に考えて大仕様を幾つかの小さな仕様(小仕様)に分割する。そして，各小仕様に対するプログラム(サブプログラム)を作成する。そして，サブプログラムを統合して大仕様を満たすプログラムを完成させる」。このガイドラインをPADで記述すると**図2**

* 日立製作所基礎研究所　工学博士

のようになる。

ただし，トップダウン作成法は詳しい作業手順を与えていない。例えば，この方法に慣れていないプログラマが「プログラムをトップダウン的に作成せよ」と命じられても，「どのようにしてトップダウン的に作ればよいか分からない」と思うのが普通である。トップダウン作成法を実行するためには，プログラマは経験と勘に基づく必要がある。そこで「もっと系統的な方法はないものか」という声にこたえて登場したものがワーニエ(Warnier)などが提唱したデータ分析法である。

ワーニエは1970年代の初めごろ，プログラム作成の際にプログラムが処理する入出力データの構造を利用することを強く主張した[8](データ構造の基本形を**図 3** に示す)。

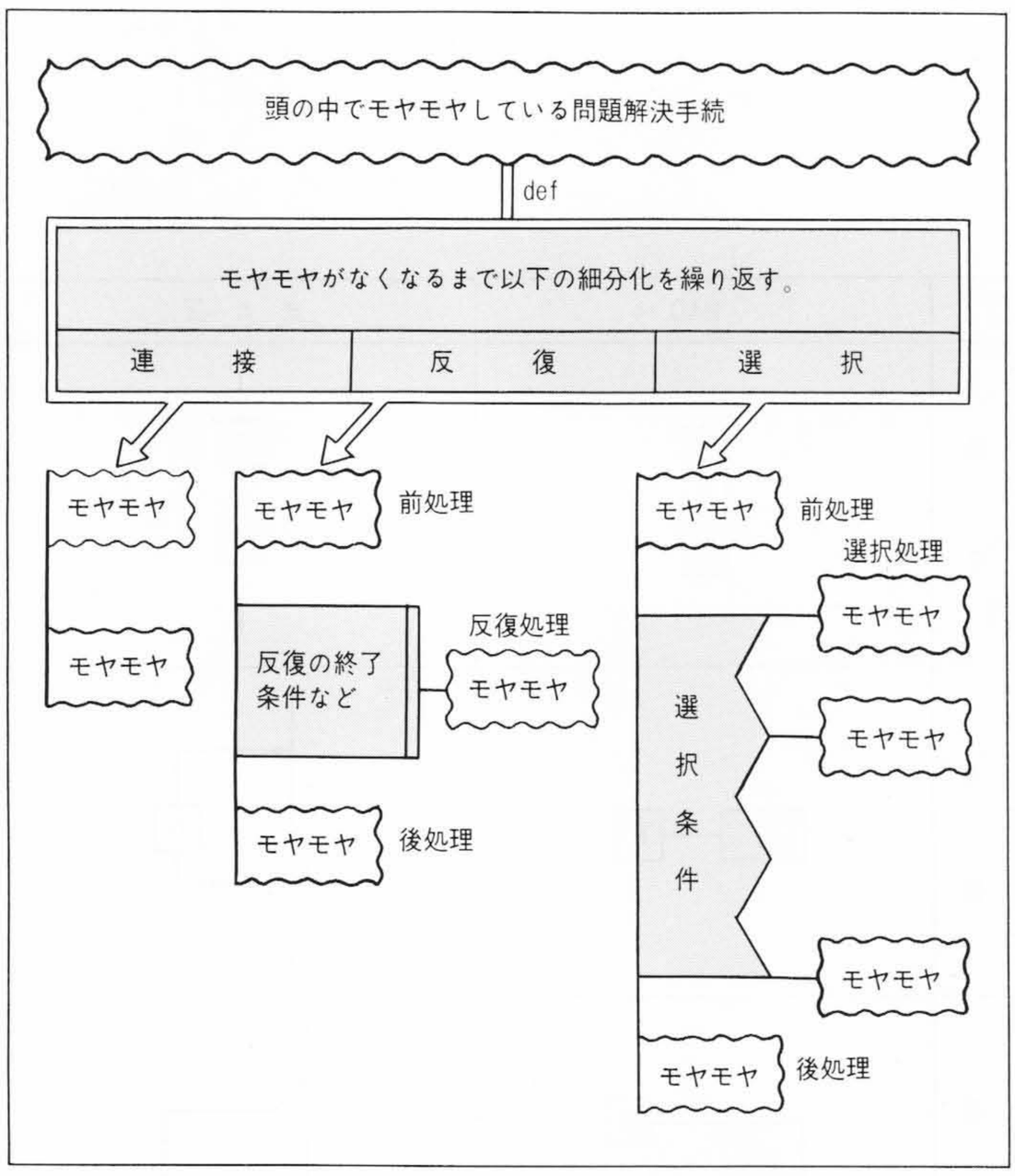

図 2　トップダウン作成法によるプログラム開発の原理　□は反復一般を表わす記号である。

基本形	PAD	意　味
連　接	A B	データAの後にデータBがある。
反　復	N　A	データAがN回繰り返して現われる。Nが不定の場合にはこれを省略してよい。また，I=1，N のような繰り返し条件を書いてもよい。
選　択	A B	データA又はデータBがある。P のように選択の条件を書いてもよい。更に I=1，N のような記法も許す。

図 3　データ構造の基本形　本図は，文献3)から引用したものである。

「プログラムの構造は，それが処理するデータの構造と似る場合が多い」ということは，古くから知られており，このことはコンパイラなどの言語処理系の作成の際に利用された[9]。しかし，プログラムの構造と類似の表記法(ワーニエ図)を用いて入出力データの構造を記述することにより，データ構造とプログラム構造の類似性をより明確にしたのは，ワーニエが最初であった。ただし，ワーニエ図では，プログラム及びデータの大まかな構造しか記述できなかった(例えば，反復や選択の条件を記述する方法を規定していない)。その点を改良し，完全なプログラム構造及びデータ構造を記述できるようにした図式がPADの原型となった[10),11)]。

入出力データの構造がプログラムの仕様書の中で明確になっている場合には，ワーニエ流のデータ分析法は有効な場合が多い。しかし，「与えられた数列をソート(整列化)する」という問題のように，仕様書の中で与えられた入出力データの構造がプログラムの構造を決める上であまり役に立たないことも多々ある。したがって，データ分析法で簡単にプログラムが作成できるのは幸運な場合である。それ以外の場合には，トップダウン作成法などに頼る必要がある。

以上で説明したトップダウン作成法やデータ分析法のように，構造化プログラムを作成するための考え方及び技術を構造化プログラム技法と呼ぶ。構造化プログラム技法の有効性は1975年のベイカー(Baker)の論文[12)]などによって証明されたと考えられる。それは現在最も広く普及しているプログラム技法である。ダイクストラなどが言葉によって示した構造化プログラム技法を，2次元的な図式により表わし，より実践しやすくしたものがPADである。「人間が計算手順を考えるときには，1次元的な言葉でよりも2次元的な図式に近い形で考えている。そして，プログラムを2次元的な図式で表現するほうが人間のもとの考えにより近い。」という仮説がPADの背景にある。

PADでは，上で述べた構造化プログラム技法に適した図式と同時に，次の二つの規則も確立した。

(1)　図式から各種プログラム言語へのコーディング規則

(2)　図式の正しさを確認(全部分の動作確認)するための検査規則

したがって，PADは「データ構造の記述，プログラムの製造(コーディング)及び検査を，同一の図式に基づいて行なうことを可能にした最初のプログラム図式である。」ということができる(コーディング及び検査規則の詳細は文献 3)に述べられている)。

3　PAD，流れ図及びNSチャート

流れ図は，1940年代にゴールドスタインとノイマンなどにより使用されて以来，プログラムの開発の際に広く使用されてきた。しかし，構造化プログラム技法を実践する上では，プログラム構造が見にくいなどの理由で，流れ図が不向きなことは1970年代の初めごろから明らかであった。例えば**図 4** に示した二つの流れ図を見ていただきたい。二つの流れ図は全く同じ処理を表わしている。その相違点は判定3と4から出ているNの矢印だけである。同図の上の図ではそれらの矢印が左に戻っているために，それらは反復形を表わすように見える。一方，下の図では判定3と4から出ているNの矢印は右にいっているためにそれらは選択形のように見える。すなわち，一見しただけではどの基本形が使われているのかが分からない。また，同じ構造を表わすにもかかわらず，図形としては異なるように書けている。

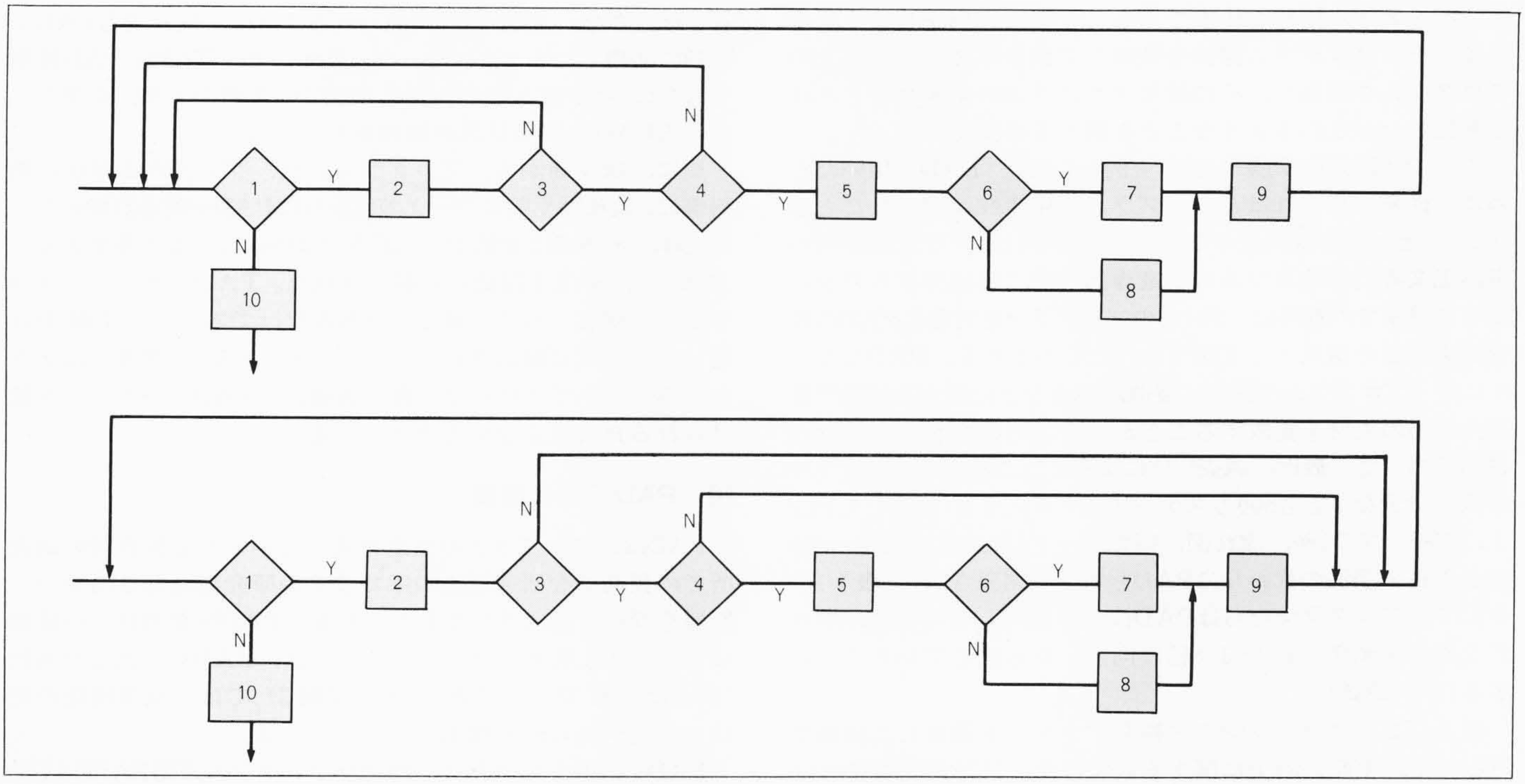

図4　同じ手順を表わす二つの流れ図　流れ図では基本形の結合の仕方が見にくい(文献3)から引用)。

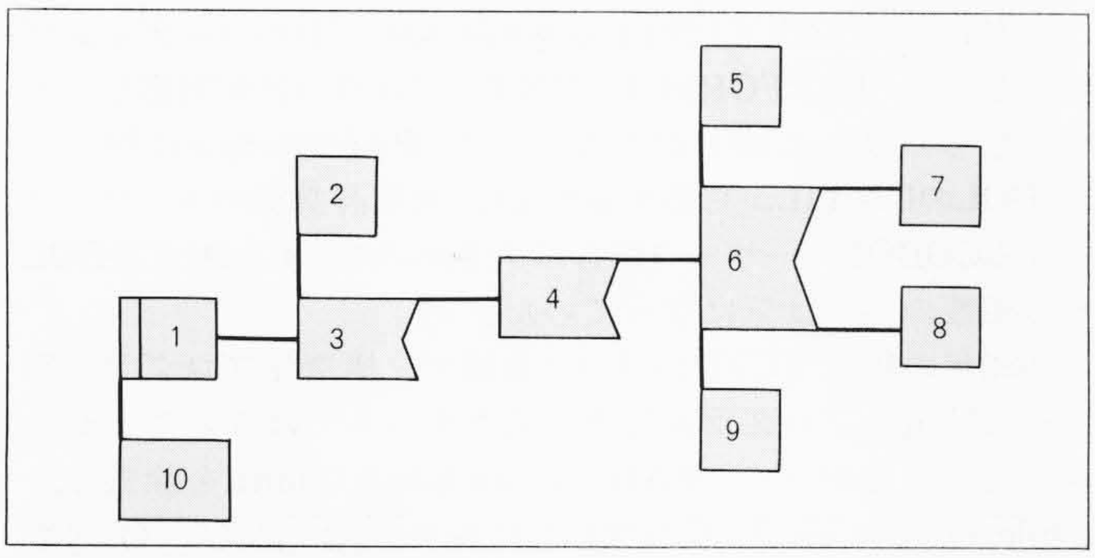

図5　図4の流れ図と同じ手順を表わすPAD　PADでは結合の仕方がひと目で分かる(文献3)から引用)。

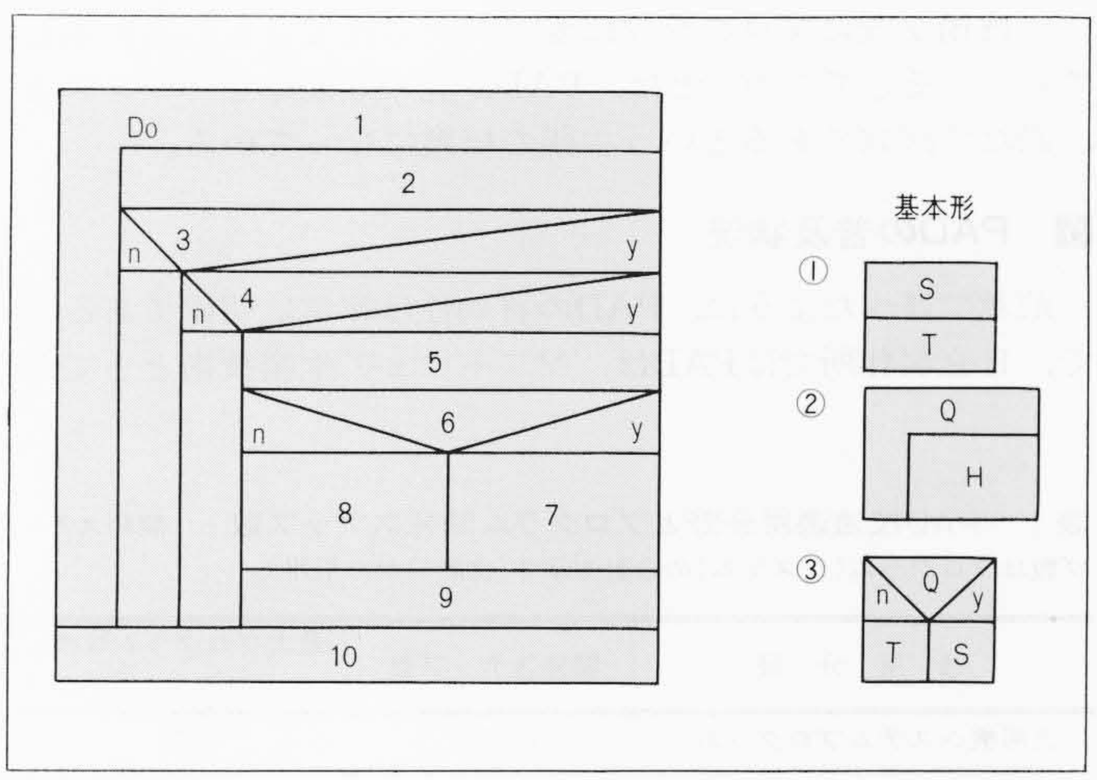

図6　図4の流れ図と同じ手順を表わすNSチャート　NSチャートは，図式の変更・修正などがめんどうである(文献3)から引用)。

この例により流れ図がもつ大きな欠点は，

(1)　プログラムの構造が見えない。

(2)　同じ構造でも違う図式で書けてしまう。すなわち，「構造は同じでも流れ図にすると個人差がでやすい。」

ということが理解できよう。

一方，PADでは基本形の結合の仕方はひと目で分かる。また，**図4**の手順ならばだれが書いても**図5**に示したとおりになる。すなわち，流れ図と比べるとPADには次の特徴がある。

(1)　プログラムの構造が見やすい。

(2)　図の書き方に関する個人差が少ない。

図6には，**図4**と同じ手順をNSチャートと呼ばれる図式で示した。この図も流れ図と比べれば，PADと同様，前の2点を特徴として挙げることができる。しかし，NSチャートは四角形が入れ子になっているために，修正が非常にめんどうである。プログラムには修正がつきものであるので，修正がめんどうということは大きな欠点である。

PADのようなプログラム図式の比較評価は文献5)に詳しく述べられているので参照していただきたい。

4　PADの効果

PADのようなプログラム図式の実際的目的は，プログラムの設計，製造，検査，保守などの各段階で常に作業のよりどころとなることである。したがって，プログラム図式の評価基準の中では次の5点が重要である。

(1)　プログラムの構造が見やすいこと。

(2)　プログラム設計法と相性がよいこと。

(3)　図式からのコーディングが容易なこと。

(4)　図式の正しさの検査がしやすいこと。

(5)　図式が書きやすく，また修正もしやすいこと。

しかし，評価基準の中にはプログラマの主観によるものもあり，客観的に評価することは容易ではない。更にPADのような新しいプログラム図式を使用したために，プログラムの生産性がどれほど向上したかを定量的に評価することは，長

い時間を要する困難な仕事である。実際に評価をしようとすれば，そのプログラム図式を使用して数メガステップ以上のプログラムを開発し，その結果プログラムの生産性がどれほど変化したかというようなことを調べる必要がある。

ここではPADの効果を示す一例として，「PADを用いたために，従来よりどれほどプログラムの生産性が向上したと思うか。」という主観調査の結果を示す。それは，日立製作所の京浜工業専門学院及び茨城工業専門学院で昭和55年6月から58年6月までの期間に，304人のプログラマを対象にPADの教育を実施した結果として調べられたものである。PAD及びそれに伴うプログラムの設計，製造，検査などに関する技術[13]普及のための人材を養成することが，そこで行なわれた教育の目的であった。最近の調査[5),14)]によると，これらのプログラマの近辺で少なくとも500万ステップのプログラム(主としてPL/I，アセンブラ)が，流れ図，擬似コード(構造化言語の一種)などの在来手法の代わりにPADを用いて開発された(**表1**)。そして，プログラマたちはPAD技法を使用したために，プログラムの生産性が1.3～1.8倍に向上したと感じていることが示された(**表2**)。

表1，**2**はプログラマの主観をアンケート調査した結果であるので，生産性向上に関する数値自身には絶対的な意味はないと思われる。しかし，この数字はPADのプログラマに対する受けのよさを示す指標と考えられる。また，PADが500万ステップ以上のプログラム開発に用いられたという事実は，この技術が既にプログラマに受け入れられていることを示している。そしてこのことは，PADが先の5基準をバランスよく満たす図式であるという主張の根拠になっている。

5 PADの普及状況

前章で述べたように，PADの有効性は非常に明確であるので，日立製作所ではPADは，ソフトウェア生産技術として定着した。また，文献[15)~17)]に見られるように，他社でもPADが有効に活用されるようになった。Computer Today PAD特集号[6)]には，我が国と中華人民共和国での多数の大学，企業などによるPADの利用状況が報告されている。

更に，数年前から，プログラム言語，プログラム技法の教科書に，流れ図や擬似コードの代わりにPADが使われ始めた。例えば，米国では文献18)，国内では19)，20)など多数ある。文献20)，3)は中国語にも翻訳された。PADの英文での詳細な解説は現在一つの文献21)があるだけであるが，文献3)は近い将来英文に翻訳される予定である。PADの普及には大きな弾みがついてきたので，書店店頭に「PADコーナー」が設けられる日も遠くないと考えている。

表1　PAD技法適用分野とプログラム開発ステップ数　開発ステップ数はプログラム(システム)の合計を示す(文献5)から引用)。

適用分野	開発ステップ数	最大プログラム規模(ステップ)
汎用機システムプログラム	1,676k	200k
プロセス制御	1,504k	250k
汎用機オンラインプログラム	1,044k	505k
技術計算	588k	50k
OA機器制御	166k	70k
事務計算	150k	50k
計	5,128k	—

注：略語説明　OA(オフィスオートメーション)

表2　PAD技法による生産性向上率　表中，機能設計では，例えばHIPOの処理の部にPADを利用する(文献5)から引用)。

評価項目	生産性向上率
機能設計	1.4倍
論理設計	1.8倍
検査データ設定	1.5倍
コーディング	1.8倍
検査	1.5倍
変更	1.3倍

6 PAD開発の経緯

PADは，プログラムの生産性向上という日立製作所の研究所での長期的な研究活動の中で多くの研究者たちと出会い，影響を受けて完成されたものである。PAD開発の詳しい経緯については文献4)で述べた。ここでは，PADを公表した当初(例えば文献22))と最近(例えば文献3))では，反復構造の記法に変化が見られる理由について述べる。

PADを設計した当初は，構造化プログラム言語PASCALを念頭に置いていた。そのため，反復構造はPASCALのwhile，until及びforの三つを考え，各々に対して[while Q]—[H]，[until Q]—[H]及び[i:=1 to n]—[H]の図式を定めた。しかし，この方式ではCOBOLに対する[while Q]—[H]のコーディング規則として，PERFORM H UNTIL(NOT Q)が対応し，どうもしっくりした感じがしなかった(PADのwhileに対してCOBOLのUNTILが対応するから)。大形計算機のユーザーに占めるCOBOLユーザーの割合は大きいので，PADとCOBOLとの相性の悪さは気になっていた。

1982年のISO TC97/SC7パリ会議で，構造化プログラム用図式は特定のプログラム言語に依存すべきではないことが決定した。したがって，PADからはwhile及びuntilを除去し，[while Q]及び[until Q]の代わりに各々[Q]及び[Q]を使うことが決定した。これにより，先ほどのCOBOLでの種類の問題は解決する。そこで文献3)に述べたように，[Q]をプログラム言語などに依存しない基本形とし，プログラム言語などに依存する問題向き反復全般を表わす記号としては[ ]を使用することにした。そして，[Q]は問題向き反復の特殊な場合とすることにより，当初のPADとISOの決定(ISO DIS8631)との間で矛盾が生じないようにPADの文法を拡張した。したがって，現在，当初のPADを書いてもPADの文法違反にはならない。また，「問題向き反復」という概念により，PADの反復構造はどのような言語の反復構造にも対処できるようになった(詳しい議論は文献3)を調べていただきたい)。

7 PADと新しいプログラム技法

PADは，LISPなどの再帰形プログラム言語と非常に相性が良い。例えば，Nの階乗を求める再帰的関数F(N)は，PADでは**図7**のように書ける。すなわち，現在の文法のままでPADは，再帰形プログラム技法に対処できる。今後しばらくのうちには，パーソナルコンピュータ上でもLISPが現在のBASIC並みに広く使われると予想される。そのときにもPADは，現在のように効果的に使うことができると考えられる。

PROLOGなどによる逐次形論理形プログラムも，LISPに対すると同様にして再帰的定義，連接及び選択構造を用いて記

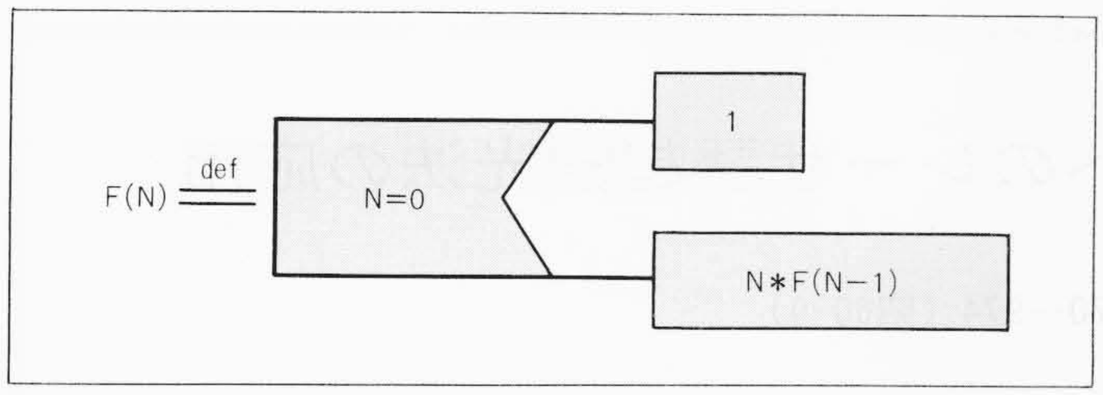

図7　PADによる再帰関数の記述例　F(N)はN*(N-1)*…*2*1の計算を示す(文献5)から引用)。

述することができる。また，抽象データ形や目的指向プログラム技法に対しても，各オブジェクトやメソッドの詳細手順を記述する際にPADを利用することができる。更にPADは並列処理やガード付きコマンドの記法も含んでいる(文献3))ので，OCCAM，CSP，並列PROLOGなどの並列処理言語にも原理的には対処可能である。

しかし，PADで原理的に記述できるということと，PADが真にプログラムの生産性を向上させるということとは別問題である。新しいプログラム技法は，それ自身が有効なものとして社会に受け入れられるか否かの試練を受けなければならない。そして，構造化プログラム技法のように，多数のプログラマによって実践されるようになって初めて，PADを利用するほうが良いか悪いかという議論を取り上げればよいと考える。

建築や機械の分野には良い図面があるのに，プログラムに関しては良い図面がないと長い間叫ばれてきた。その理由は，プログラム作成については，社会の大部分のプログラマに共通して使われる作成法(いわゆるパラダイム)が確立されなかったからである。しかし，ダイクストラなど，多数の計算機科学者の努力によって構造化プログラム技法が確立され，この技法を知らないとプログラマとしての資格を問われる時代が1970年代の後半に到来した。このような歴史的背景のもとで，構造化プログラム技法を実践しやすくすることが時代の要請となった。そしてPADのような図面が登場し，多数のプログラマに受け入れられる機会が存在したのである。

日立製作所は，産業界の中で長期にわたって計算機科学での先端的研究を継続してきた。そして，構造化プログラム技法や更に新しいプログラム技術の成熟度と，社会のニーズの両面をにらんできた。その結果，タイミングよくPADを提案することができた。将来も，真に有効なプログラム技法を提案する機会をうかがってゆく。

8 結　言

構造化プログラム技法は万能ではない[23]。しかし，それは有効性が立証され，かつ現在最も普及しているプログラム技法である。これを修得していないと，プログラマとして非常に不利である。

PADは，構造化プログラム技法を修得し，かつ実践する上での便利な記法として世界中に広く普及しつつある。12世紀にアラビア経由でヨーロッパに伝わったインドの数字(アラビア数字)が，当時ヨーロッパで使われていたローマ数字に置き換わり，それ以後の数学の進歩に絶大な貢献をしたと言われている[24]。PADもアラビア数字のように，国境を越えて後世に生き残ってほしいと願っている。

参考文献

1) Dijkstra, E.W.: Notes on Structured Programming, In Dahl, O.-J., Hoare, C.A.R. and Dijkstra, E.W.: Structured Programming, Academic Press, 1972. 邦訳(野下ほか)，構造化プログラミング，サイエンス社
2) Wirth, N.: Systematic Programming; An Introduction, Prentice-Hall, 1973. 邦訳(野下ほか)，系統的プログラミング/入門，近代科学社
3) 二村：プログラム技法-PADによる構造化プログラミング，オーム社(1984)
4) 二村：PAD誕生，Computer Today 5月号，pp. 4～10，サイエンス社(1985年)
5) 二村：構造化プログラム図式，日本ソフトウェア科学会誌1巻1号，岩波書店(1984年4月)
6) PAD特集号：Computer Today 5月号，サイエンス社(1985年)
7) Böhm, C. et al.: Flow Diagrams, Turing Machines and Languages with Only Two Formation Rules, C.ACM, Vol. 9, No. 5 (May 1966)
8) Warnier, J.D., et al.: Entainement à la programmation, Les Edition d'Organisation, 1971. 邦訳(鈴木)，ワーニエ方式によるプログラミング学習(上，下)，日本能率協会
9) Irons E.T.: A syntax directed compiler for ALGOL 60, C. ACM, Vol. 4, No. 1, January, 1961
10) 二村：構造的プログラム作成のための一方法，電子通信学会総合全国大会予稿S8－9(1978年3月)
11) 二村：FORTRAN-PASCALによる構造化プログラム技法，日本情報処理開発協会情報処理研修センタ(1978年3月)
12) Baker F.T.: Structured programming in a production programming environment, IEEE Trans. of Software Engineering, June 1975
13) 二村：プログラム設計法PAD/PAM，情報処理学会会誌，25巻，11号(1984年11月)
14) 二村：プログラム設計法PAD/PAMとその効果，プログラム設計技法の実用化と発展—シンポジウム，情報処理学会(1984年4月)
15) 古谷：事務処理システムへのPADの導入(相模鉄道株式会社)，HITACユーザー研究会，1985年論文集
16) 松本：PAD/HSPMを利用した構造化設計の採用について(株式会社北洋相互銀行)，HITACユーザー研究会，1983年論文集
17) 奥山：PADによるオンラインシステム開発の効率化(中央信託銀行株式会社)，HITACユーザー研究会，1983年論文集
18) Peters L.: Software Design: Methods & Techniques, Yourdon Press, New York, 1981
19) 木下：マイクロソフトBASIC詳細，サイエンス社
20) 青木：BASIC数値計算法，コロナ社，1984年
21) Futamura, Y. et al.: Problem Analysis Diagram (PAD), JARECT, Vol.12, オーム社—North-Holland, 1984
22) 二村，外：PAD(Problem Analysis Diagram)によるプログラムの設計と作成，情報処理学会論文誌，21巻7号(1980年7月)
23) 二村：一流プログラマの条件，数学セミナー別冊，日本評論社(1985年9月)
24) プライス(岡本，藤本訳)：ホワイトヘッドとの対話，みすず書房(1980年)

論文抄録

半導体プロセスへのレーザ誘起蛍光法の応用

日立製作所　鈴木敬三

電子通信学会誌　68—9，970～974（昭60-9）

半導体素子製造プロセスは，約10年前から，ふっ酸などの水溶液を用いるウエットプロセスからプラズマやイオンビームを用いるドライプロセスに移行してきた。現在では，工程の大部分は既にドライ化され，更に微細化・高性能化を目指した技術改良が激しく行なわれている。従来，ドライプロセス技術の改良は，操作パラメータと処理結果との因果関係に関する経験的認識に基づいて行なわれており，そこに内在する気相(プラズマ)と固体(試料)表面との相互作用に関する物理・化学的認識は十分でなかった。しかし，今後，プロセス技術を更に改良するためには，このような相互作用に関する知見，特に気相状態を正しく把握することが不可欠である。このような状況の中で，ドライプロセスでの気相分析法として最近注目を集めているのが，LIF(Laser Induced Fluorescence：レーザ誘起蛍光)法である。本論文は，LIF法に関する解説をプロセス技術開発者向けに行なったものである。

LIF法とは，被測定粒子(原子・分子)をその電子準位間隔に同調したレーザ光で励起し，励起粒子からの蛍光を測定する方法である。これによって，被測定粒子の種類，濃度を知ることができる。また，被測定粒子が分子である場合は，振動・回転エネルギー準位に関する(したがって，分子構造に関する)知見を得ることができる。励起光は，通常エキシマレーザで色素レーザをポンピングして得られる。場合によって周波数2倍化素子を用いることによって，220～1,000 nmの波長の励起光が得られる。

気相測定技術としてのLIF法の長所は以下のようである。(1) 極めて高感度である。通常，10^8～10^{10}分子/cm^3以上の濃度測定が可能である。ただし，絶対濃度測定にはシステム全体の校正が必要である。(2) 準位選択的(State Selective)な測定法である。したがって，振動や回転自由度の温度を測定できる。このことは，非平衡プラズマの測定手段として有効である。(3) 時間分解可能な測定法である。時間分解能は10～100nsである。(4) 空間分解可能な測定法である。空間分解能は0.1～1 mmである。(5) 被測定系を汚染することのないクリーンな測定法である。一方，LIF法の短所は測定可能な粒子種が限定されていることである。論文中には，これまでにLIF測定された半導体プロセス関連粒子種の一覧表が掲載されている。

半導体プロセスに用いられるプラズマは，非平衡であり，かつ限られた空間内に非一様に分布している。また，その中には化学的に活性なラジカルが多種存在している。高感度，準位選択，空間・時間分解能をもつLIF法は，このようなプラズマ診断技術として，他の気相測定法にない特徴をもっていると言える。今後，被測定粒子種の拡大が行なわれれば，LIF法は半導体プロセスの基礎・応用両面の研究でその有用性を発揮すると考える。

日立における機械翻訳システム

日立製作所　梶　博行・岡島　惇

情報処理学会誌　26—10，1214～1216（昭60-10）

近年，機械翻訳に対するニーズが急速に高まってきている。その背景として，経済や技術の国際化に伴って翻訳が必要な文書が増加し，人手では処理しきれなくなったことがある。一方，シーズ面でも，言語理論の進歩とともに，計算機の性能が飛躍的に向上した。この結果，人間と機械の適切な機能分担を前提とすれば，機械翻訳システムの実用化が可能であると考えられるようになった。日立製作所では，長年にわたる自然言語処理の研究成果を結集して，日英・英日機械翻訳システムの実用化を推進している。本論文では，日英翻訳システムATHENE(Automatic Translator of Hitachi for English and Nihongo with Editing support)/N，英日翻訳システムATHENE/Eについて，それぞれの翻訳方式を解説した。更に，ATHENE/Nの文法記述言語，ATHENE/Eのユーザーインタフェースについても述べた。

ATHENE/Nは，語句の担う概念の文中での意味的役割に着目した概念依存構造を中間表現として用いたシステムである。これは，省略が多く語順の自由度が高いという日本語の特質を考慮したものである。日本語文は格文法に基づいて解析され，概念依存構造に写像される。日本語と英語の発想の違いに起因するギャップは，概念依存構造の構造変換により吸収し，英語らしい表現の文を生成することができる。

機械翻訳のための文法は大規模複雑であり，システム開発の観点からは文法記述用の問題向き言語が必要である。ATHENE/Nの文法記述言語では，個々の規則を属性付き有向グラフの書換え規則として記述するとともに，規則の適用制御方法をパラメータで指定することができる。強力で柔軟な記述力とともに，記述や理解の容易性にも優れ，拡張性・保守性の高い翻訳システムを実現している。

ATHENE/Eは構文主体のトランスファ方式のシステムである。英語側の中間表現としては句構造を重視したQPS(Quasi Phrase Structure)を，日本語側の中間表現としては格構造を重視したCPS(Case Phrase Structure)を用い，これらの間に変換を行なう方式をとっている。構文解析では種々の多義性を解消することが重要であるが，品詞列の禁止パターンを利用して多品詞を解消する方法，述部構造とその意味制約を利用して前置詞の深層格を決定する方法などを用いて，解析精度の向上を図っている。

実用的な翻訳システムでは，ユーザーのインタフェースも重要な課題である。ATHENE/Eは，句単位修正や多義選択などの後編集機能，慣用的表現をユーザーが辞書登録できる機能を提供している。

ATHENE/N，ATHENE/Eとも，多様な言語現象に対して文法規則を積み上げ，翻訳率のいっそうの向上を図る必要がある。また，実用化という面から，編集機能などの翻訳支援環境の整備も重要な課題である。