ソフトウェア工学 43－2
（１９８５．７．３１）

構造化プログラム図ＳＰＤとその応用

東　基衞、田端　哲、沖　佳広
（日本電気株式会社）

１．はじめに

ソフトウェア工学の発展、とりわけ構造化プログラム技法の提案と普及とともに、それまでの中心的プログラム図法であった流れ図に対する欠点が指摘されだした[1]。いわく、「基本的な制御構造が明確に表現できない。」等である。以来、流れ図の欠点を解消するため数多くのプログラム図法が提案されており、最近ではそれらに対する分類や評価も試みられている[2-6]。ＳＰＤ（Structured Programming Diagram）は、そういった図法の一つに位置付けられる（図１－１）。

ＳＰＤの特徴には、次の三項目が挙げられる。

①構造化プログラミング技法に対応している。

ＳＰＤは簡単な記号を用いて、順次、選択、繰り返しの三種の制御構造を表現できる。

②利用者にとって使い易い。

単純な縦横の線からなるＳＰＤは、特殊なテンプレートを必要とせず、作成や修正を容易に行える。

③プログラムの部品化・再利用を促す。

特にモジュール名とその実現方式を明示できるようモジュール記号を導入し、モジュールの利用状況を把握し易くしている。

これらは、実用的なプログラム図法であるために必要な要件ととらえることができる。もっとも、これらの特徴全てが提案時からＳＰＤに備えられていたわけではなく、十年以上に及ぶ現場部門での使用と改善の間に徐々に加えられて現在の姿となってきている。すでに歴史もあり、社内外に相当数の利用者を有するＳＰＤであるが、こういった図法の変遷過程を書きとどめるのも意義があると考えられたため、本稿では特にＳＰＤの発展経緯も含め、以下に記法及び応用法について解説する。

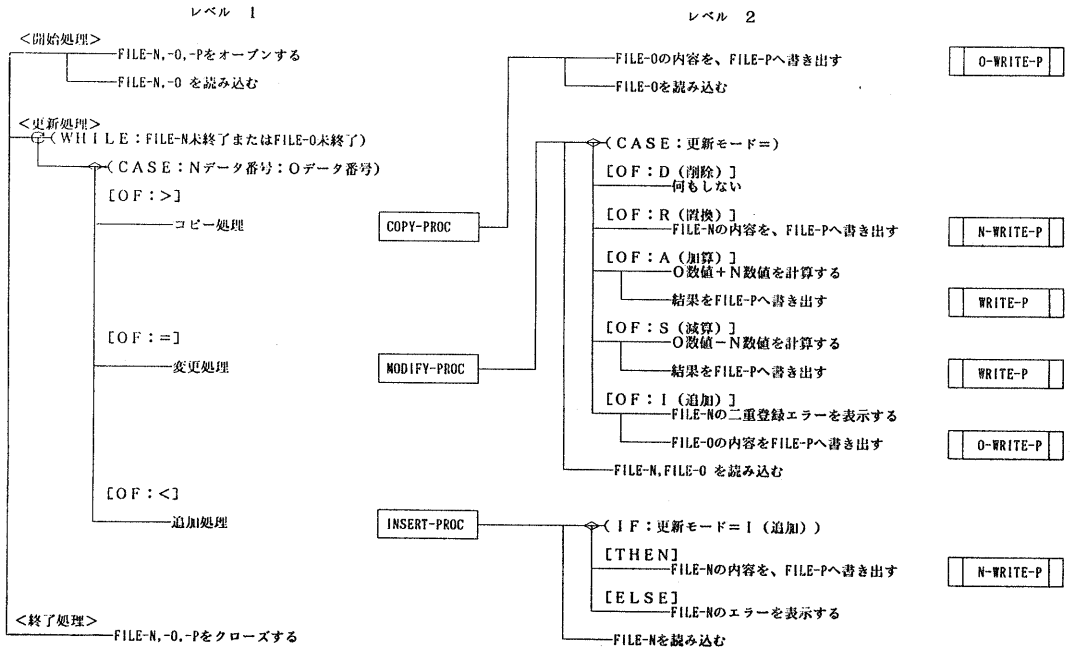

図１－１　ＳＰＤの例

2．記法

（1） 記号

ＳＰＤは次の三種類のプログラム構造を同時に一枚の紙に表現できる。

①機能の階層構造

プログラム内で個別の機能を実現している各モジュールの呼び出し関係を左から右へのレベルにより表現する。

②制御構造

構造化プログラム技法で用いる、順次、選択、繰り返しの三種の制御構造を簡単な記号で表現する。

③物理構造

各モジュールが、物理的にどのような方式で実現されているかを、モジュール記号を用いて表現する。

これら各構造に対するＳＰＤ記号を表２－１に示す。

ＳＰＤでは基本的な制御構造を表す際、ＩＦ、ＷＨＩＬＥ等のキーワードを用いている。これは構造を直感的に理解できること、および条件の記述に制約を加え標準的な記述が得られるようにすることを目的としており、その設定はなるべく一般的なものとなるよう考慮されている。しかしながら、ＳＰＤを用いる環境、とりわけ使用言語の違いやＳＰＤに要求される記述の詳細度（コーディングとどの程度近いものが要求されるか）により標準キーワードセットでは不十分な場合、変更が可能である。また逆に設計の初期段階でおおまかな構造を検討する際のメモ書きや、人間の作業手順を示す場合等では、キーワードを用いずに全て日本語で記述しても良い。

表２－１　ＳＰＤ基本記号

名称		ＳＰＤ記号	説明
階層構造記号	レベル	レベル　Ｎ	機能（モジュール）の呼び出し関係の階層番号を示す。
	ブロック	<ブロック名>	機能（モジュール）内の大まかなまとまりをブロック名で示す。
制御構造記号	順次	手続き１ 手続き２	手続きを上から下へ１回づつ実行する。
	前検査繰り返し	(WHILE:条件) 手続き	条件が真の間、０回以上手続きを実行する。
	後検査繰り返し	(UNTIL:条件) 手続き	条件が真の間、１回以上手続きを実行する。
	無限繰り返し	(LOOP) 手続き	手続きを無限回実行する。
	２分岐選択	(IF: 条件) [THEN] 手続き１ [ELSE] 手続き２	条件が真であれば手続き１を偽であれば手続き２を実行する。ＴＨＥＮ／ＥＬＳＥ節のどちらか一方は、無くてもよい。
	多分岐選択	(CASE: 条件) [OF:値１] 手続き１ [OF:値２] 手続き２ [OTHER] 手続きＮ	上から下へと条件と値の組み合わせを検査し、真のものに対する手続きをただ一つ実行する。真となる値がない場合はＯＴＨＥＲ節の手続きを実行する。ＯＴＨＥＲ節は必要に応じて用いる。
	終了	→終了指示	指定した構造（ブロック）又はモジュールの実行を終了する。
物理構造記号	モジュール	Ｘ　モジュール名	指定したモジュールを呼び出す。Ｘには、モジュールの実現方式を記号で示す。 例）省略：内部モジュール Ｅ　：外部モジュール Ｃ　：マクロ 等
	標準モジュール	Ｘ　モジュール名	指定した標準モジュールを呼び出す。Ｘは同上。

（2） 解説

①階層構造の表現

定義したモジュールの内容をそのモジュールが呼ばれている場所の右側に展開していくことにより、ＳＰＤは複数のモジュールの内容とそれらの関係を同時に一枚の紙に表すことができる。この場合のモジュール呼び出しの階層をレベルと呼ぶ。すなわち、最上位のモジュールがレベル１であり、以降レベル２、レベル３と続く。

小さなプログラムであれば、そこで使用されている全てのモジュールを一枚の紙に書くことができる。どのレベルまでを一枚の紙に書くかは、モジュールの大きさ、数、使用する紙の大きさ、見易さ等により一概には言い難いが、一般的にはＡ４であれば１から３レベル、Ａ３であれば２から４レベルが適当であろう（図２－１）。

このように、最終的なプログラム仕様としてモジュールの呼び出し関係を示すのがレベルの記法であるが、これを設計の初期段階から積極的に利用するとさらに効果的であ

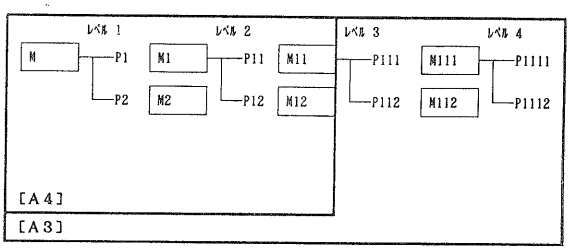

図２－１　レベル数と紙の大きさ

る。つまり、機能の抽出、分割／統合、配置に関する検討時からＳＰＤ風の記法を採用しておくことにより、設計の開始から終了まで統一的な表現が可能となり、無用な混乱を避けることができる。データ構造の表現についても同様である。これらのことについては、後ほど詳述する。

また、ある階層の一つのモジュール内での大まかな機能のまとまりを明示したい場合、ブロック記号を用いてブロック名を与えることができる（図２－２）。必要に応じて注釈的に用いると良い。

②制御構造の表現

ＳＰＤでは、制御構造の入れ子はある構造の手続きの部分が別の構造に置き換わることにより表現される。例えば、繰り返し構造の手続き部に順次構造が含まれている場合図２－３のようになる。さらに、その中に選択構造が含まれている場合は図２－４のようになる。

なお、手続き部を伴わない特殊な構造として終了がある。これは、階層化された入れ子状の構造の中で、どの構造を終了させるかを示すための記号である。ＳＰＤでは一番外側の構造が終了した時点をモジュールの終了としているため、通常の処理では必ずしも終了記号を用いる必要はない。異常終了や無限繰り返しからの脱出等止むを得ない場合のみに使用するのが望ましい。終了させる構造を指定する表現は「繰り返し構造をＮ個抜ける。」、「次のＸＸ処理に移る。」等自由であるが、前述のブロック記号を併用すると終了させる構造がより明確となる（図２－５）。

③物理構造の表現

作成したＳＰＤに対し、モジュール記号によりモジュール名を与えることができる。モジュール名は、最終的にはコーディング上の名前を用いるが、設計の中間段階では日本語等を用いてもかまわない。ＳＰＤの手続き記述部や条件記述部の右側にモジュール記号を書くと、その場所でモジュールを呼び出していることを意味する。もし、モジュールが共通化（標準化）されていてその内容が既知のものであれば共通モジュール記号を用いる。

モジュール記号の先頭には、そのモジュールの実現方式を記号で示す。記号は使用言語に応じて利用者が自由に設定できる。例えばＣＯＢＯＬであれば、段落や節の呼び出しは省略、副プログラムの呼び出しは’Ｅ’、複写展開は’Ｃ’等を用いると良い。また、モジュール呼び出しに伴いパラメータを必要とする場合は、手続き部に二重括弧でパラメータ列を記述できる。以上を図２－６にまとめて示す。

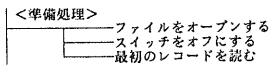

図２－２　ブロック記号

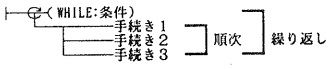

図２－３　繰り返し（順次）

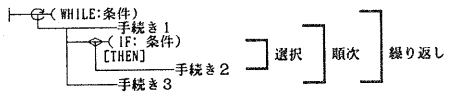

図２－４　繰り返し（順次（選択））

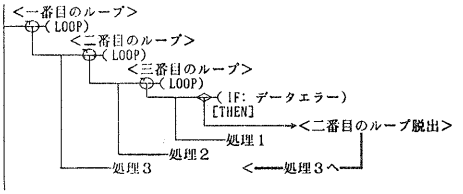

図２－５　終了記号の用法

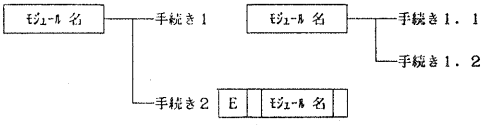

図２－６　モジュール記号

3．発展経緯

ＳＰＤの原形は、事務処理システム開発での作業手順、ドキュメント技法、処理パターン等の標準化を目的として1975年から社外提供され始めた体系的標準集ＳＴＥＰＳにおけるプログラムパターン表現用図法「ＳＰ仕様」に見ることができる[7-10]。

当時のＳＰ仕様は、ワーニエ図[11]を基に縦横の線を組み入れた単純な物であり、主として機能の階層構造を表現するものととらえることができる（図３－１）[12]。

その後利用者から、機能の制御構造、つまり各構造が順次的に実行されるのか、繰り返し実行されるのか、選択的に実行されるのかが表現されると良いという意見に応えて、1977年に丸矢印や菱形の制御構造記号が導入された（図３－２）[13]。

さらに1982年には、プログラムの部品化・再利用を行い易くするために、プログラムで使用されているモジュール名と、それまでは手続き記述の一部として左端に英数字で表していたモジュールの実現方式とを併せて明示できるモジュール記号が導入され、ほぼ現在の形となっている（図１－１）。

このように、ＳＰＤは利用者の意見を基に改良されてきた実践的な技法であるが、最近ではＩＳＯ（国際標準化機構）におけるプログラム基本構造の検討過程で定義された構造全てを表現するため、新たな制御構造記号の導入も試みている[14]。

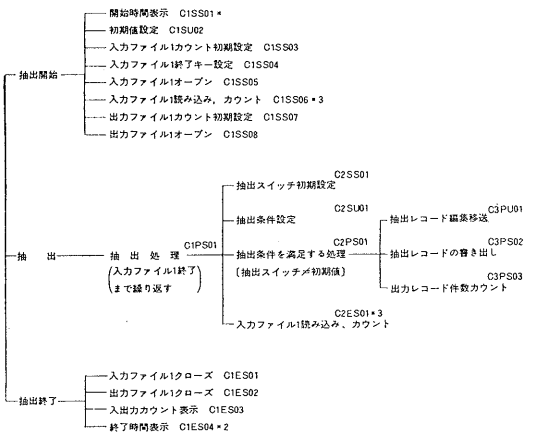

図３－１　ＳＰ仕様７５

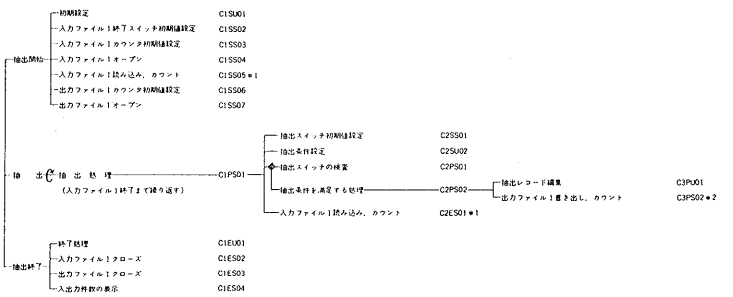

図３－２　ＳＰ仕様７７

4．　応用

（１）　設計

ＳＰＤは単なる図法であり、ある特定の設計方法論を支援するものではない。しかしながらＳＰＤを用いた設計法として何らかの指針を望む声が高いため、筆者らは次のような手順を推奨している。

①機能設計段階

与えられた問題を理解し、機能の抽出、分割／統合を検討し、機能の階層構造を定める（図４－１）。

②制御設計段階

各機能の実行順序を検討し、制御構造を定める（図４－２）。

③生産設計段階

物理的なモジュールの配置、モジュールの部品化／再利用について検討し、物理構造を定める（図１－１）。

一口に設計作業といってもいくつかの段階があるが、ＳＰＤを用いるのはシステムからプログラムへの分割が終わ

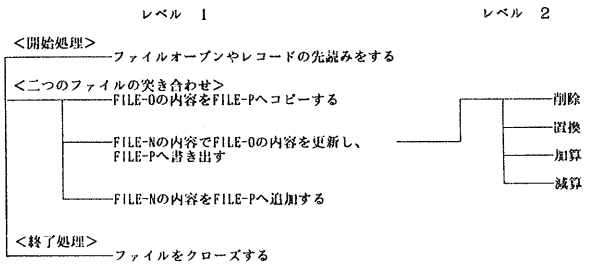

図4－1 機能設計段階

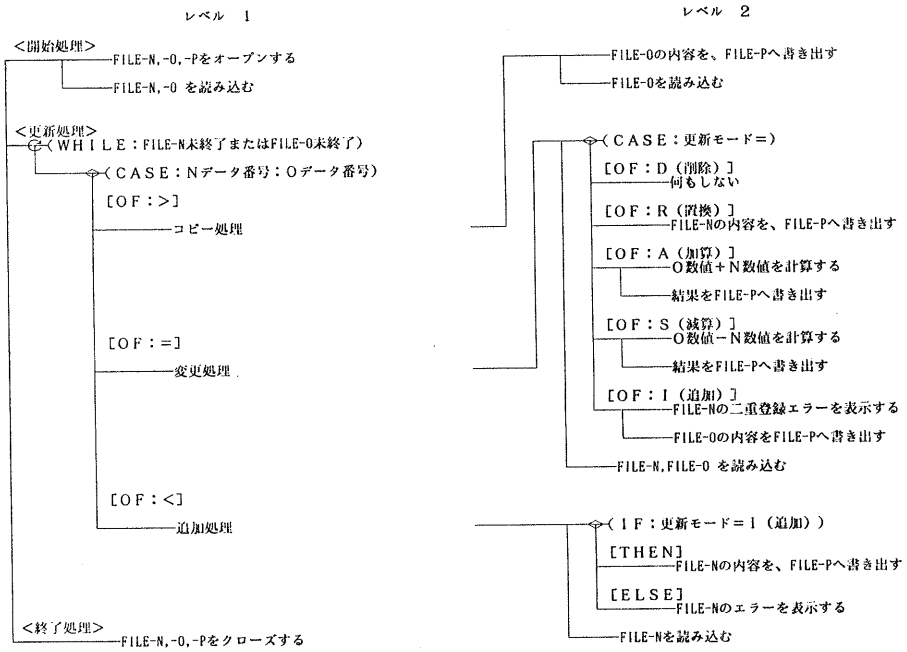

図4－2 制御設計段階

った後のプログラム設計段階が適当と考えられる。この段階での設計法は、制御構造主導型とデータ構造主導型の二種に分類されることが知られている[15]。上記の手順は前者に近い。データ構造をＳＰＤで表すのは容易なため、ＳＰＤはデータ構造主導型設計法にも無理無く対応できる（図4－3）。

（2） 製造・検査

ＳＰＤにおける図要素の配置（条件や手続きの記述位置）は、疑似言語表現に近い。特にキーワードを用いている場合、ＳＰＤから縦横の線を排除するとそのまま疑似言語表現となる（ただし、各構造の終わりを示すＥＮＤは無い）。従って、Ｐａｓｃａｌやｃのような構造化文を持つ言語を用いてコーディングを行う場合は、図上に連番を与える等他の図法の一部に見られるような特別な作業を必要とせず、そのまま上から下へと進めていくことができる。構造化文が弱いＦｏｒｔｒａｎ、ＣＯＢＯＬ、アセンブラ等については、各基本構造に対し簡単なコーディング規則を設けてやると良い。

図上になんらかの連番が要求される場合、ＳＰＤではある一つのレベルにおける条件や手続きの記述が必ず行単位

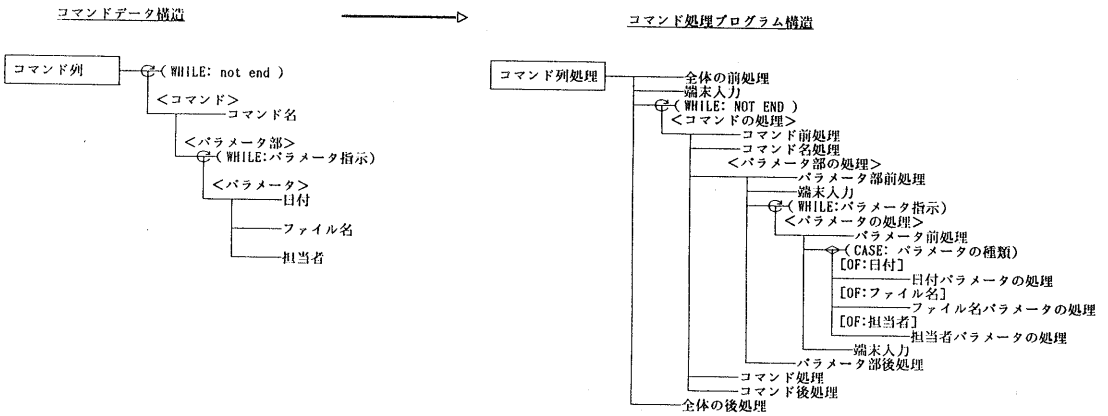

図４－３　データ構造主導型設計方法

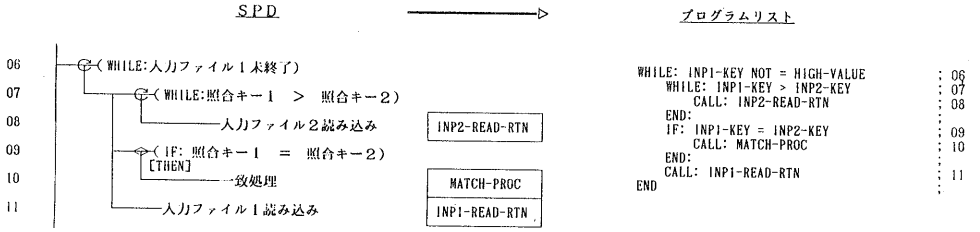

図４－４　ＳＰＤ行番号とプログラムコメント

になるため、それぞれの条件記述や手続き記述の左側あるいは右側に対応して行番号を付与することができる。同じ番号をソースプログラム上にコメントとして表すことにより、さらに対応性が良くなる（図４－４）。逆に、ＳＰＤを基に作成したプログラムの検査や保守を行う場合等では、プログラム上の行番号をＳＰＤ上の対応する行に記入することにより、これら作業に有効なドキュメントを得ることも可能である。

５．まとめ

ＳＰＤの記法、発展経緯および応用について示した。ＳＰＤと同種の数々のプログラム構造図を概観すると、見易さを向上させるために大がかりな図形を多用したり、処理や条件の記述を箱で囲むことにより、書き易さやスペース効率を損ねてると感じられるものが散見される。逆に恐らく最も書き易いであろう疑似言語は、基本構造に対する視覚性が弱いように感じられる（図５－１）。両者のバランスが評価を左右するものと思われる。

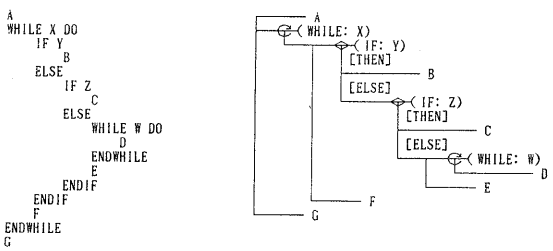

図５－１　ＳＰＤと疑似言語の比較

ＳＰＤは手書きを前提としており、他の図法の一部に見られるような、機械による支援が無いと作成／修正に多大の労力を要するといった性格のものではない。しかしながら、作成過程であるいは作成後に多くの修正が予想される場合、一部を変更して様々な場面で再利用する可能性が高い場合、活字での清書が必要な場合等には、機械を用いたほうが都合がよい。

ＳＰＤは単純な縦横の線と若干の記号から構成されているため、容易に機械処理を行うことができる。例えば、英数文字集合[16]のみでもＳＰＤに近い図を作成することが

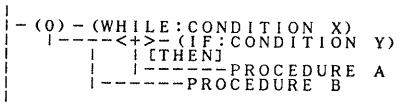

図5－2　英数文字集合によるＳＰＤの表現

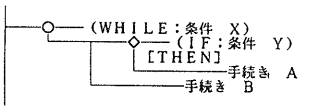

図5－3　日本語文字集合によるＳＰＤの表現

できる（図５－２）。さらに、罫線素片を含む日本語図形文字集合[17]であればほぼ完全なＳＰＤが作成できる（図５－３）。このように機械処理に際し高度（高価）な図形処理装置を必要としないことも、ＳＰＤの特徴の一つに挙げられる。

ＳＰＤを用いたプログラム設計・製造を支援するソフトウェアツールとしては、以下の三種が開発され、現在評価中である。

①ＳＰＤ編集系

②ＳＰＤ→ソース変換系

③ソース→ＳＰＤ生成系

とかくこのような図法を比較する際、「条件１」、「手続き１」といった形式的な表現を用いて、あるいは最大値を求めるような簡単な問題に対する手順を書き比べて語られることが多い。実作業で使う感覚は、幾分異なると予想される。ＳＰＤの導入を検討する際は、まず実作業に即したサンプルを書いてみることをお勧めする。そしてその時、ＳＰＤが現場主導で発展してきたものであることが理解されると期待する。

参考文献

[1] Shneiderman,B. et al: Experimental investigations of the utility of detailed flowcharts in programming, Comm. ACM, Vol.20, No.6 (1977) pp.373-381.

[2] 花田他：コンパクトチャートを用いたプログラム設計法，情報処理学会論文誌，Vol.22, No.1 (1981), pp.44-50.

[3] 二村他：ＰＡＤ(Problem Analysis Diagram)によるプログラムの設計および作成，情報処理学会論文誌 Vol.21, No.4 (1980), PP.259-267.

[4] Azuma,M. et al: SPD: A humanized documentation technology, Proc. of IEEE COMPSAC'83, Chicago 1983.

[5] 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.6 (1981), pp.383-389.

[6] 二村：構造化プログラム図式，コンピュータソフトウェア，Vol.1, No.1 (1984), PP.64-77.

[7] 水野他：ソフトウェアの標準化，日本経済新聞社，1977.

[8] 水野他：管理者のためのソフトウェアの標準化，コンピュータレポート，Vol.18, No.8 (1978.7)-Vol.19, No.1 (1979.1).

[9] Azuma,M. et al: STEPS: Integrated software standards and its productivity impact, Proc. of IEEE COMPCON'81 (Fall), Washington, 1981.

[10] 東他：ＳＴＥＰＳ，bit, Vol.13, No.5 (1981),pp.651-686.

[11] Ｊ．Ｄ．ワーニエ（鈴木訳）：ワーニエプログラミング法則集，日本能率協会，1975.

[12] ＳＴＥＰＳ－２　プログラミング標準説明書，日本電気,1975.

[13] ＳＴＥＰＳ／Ｃ　プログラミング標準説明書，日本電気,1977.

[14] ISO/TC 97/SC 7 N 307, 1983.

[15] 紫合：ソフトウェア設計法について，コンピュータソフトウェア，Vol.1, No.2, (1984), pp.55-68.

[16] 情報交換用符号，日本工業規格(JIS-C 6220-1976).

[17] 情報交換用漢字符号系，日本工業規格(JIS-C 6226-1983).