ソフトウエア工学 17－11
（1981. 2. 13）

構造化COBOLの実用化

関 栄四郎　二階堂 徳也
（鉄道技術研究所）

1 はじめに

国鉄には多くの情報システムがありソフトウェア管理上それぞれいろいろな工夫をしており、ソフトウェア工学上の最近の諸手法を既にとり入れ，あるいはとり入れつつあることは、第12回の本研究会において報告した[1]。ここではその一環として行なわれた構造化プログラミング用COBOLの実用化に至るまでの経過と，その問題点および解決策について述べる。開発したプリプロセッサは，CODASYL COBOL '78[2]の構造化プログラミング機能に酷似しているが，CODASYLにない勝れた機能も持っているので，それらについてはささやかな提案のつもりである。大方の御批判をいただければ幸いである。

2 COBOLによる構造化プログラミングとその問題点

構造化プログラミング（Structured Programming ―以下SPと略す）という言葉がとりざたされてから久しく，その言葉の意味の広さにより，実に多くの議論がなされ，またいろいろなSPが実現されている。COBOLによるSPというと通常は狭い意味，すなわちコーディングレベルで用いられる。そしてGOTO文の使用を禁止すると必然的に生ずるような構造のものをCOBOLによるSPと称して，実業務で使われているようである。これらは多く，GOTO文の禁止条項の他にニミの構造上の約束をつけ加えたもののようであるが，それなりに効果のある方法であり，実際上の効果があったと報告されている。

しかしCOBOL言語（現在一般に使用されているもの）のみによるSPはCOBOLの制御構造表現力の貧弱さにより制約が生ずるのはやむをえない。すなわち，ある点で「そうしない方がよい，そうしない方が分り易い」とか，「適当な予約語がないので複数の文である機能を代用する」とかするところがでてくる。例えばGOTO文を禁止するとPERFORM文を多用したTree状のプログラムができ上るが，このTreeの段数があまり多くなると分り易さを損なう。PERFORM文は機能的にまとまったものを呼ぶ（サブルーチンコールと考えてよい）場合にはまことに都合がよいが，繰り返し構造を実現するために用いると繰り返しを制御する文と実際の繰り返し部分が離れたところに書かれるので分りにくい構造となるのである。またIF文も分岐構造の表現に不便な場合がある。すなわち，分岐構造の中に分岐構造を入れ子にする場合，現行COBOLでは複数に分岐した制御が合流するのは最終点だけという制限があるので，それ以外の構造を表現するときには工夫を要する。以上述べたことは考えようによっては何でもないともいえるが，ある種の理想を追うものにとっては改善すべき欠点となっているのである。

3 プリコンパイラESCOBOLとCODASYL COBOL 78

上で述べた点を問題点として認識し，それを改善するために各社のコンピュータにかかるプリコンパイラESCOBOL[3]（Elementary Structured COBOL）を開発した。ESCOBOLは上述の繰り返し構造および分岐構造を表現し易くす

るとともに、字下げ（インデンテーション）機能をもっている。字下げは簡単なようではあるが、それを実用化する場合には多くの問題を含んでおり後述するように最も苦労したところである。

ＥＳＣＯＢＯＬ開発の基本的考え方は次の通りであった。

(1) 従来のＣＯＢＯＬに慣れているプログラマが違和感を抱かないように、新設の予約語はＣＯＢＯＬ言語の印象と似たようなものとし必要最低限にとどめた。

(2) 各社の計算機に適用できるようにプリコンパイラ自体をＣＯＢＯＬで作成した。

ＥＳＣＯＢＯＬの基本機能は昭和52年に決定したが、その後発表されたＣＯＤＡＳＹＬ　ＣＯＢＯＬ'78のＳＰの部分と酷似しており一部予約語は全く一致している。

表　1　ESCOBOLの機能とCODASYL COBOL'78の比較

No	文名称	ESCOBOL予約語	構　文	記　事	CODASYL COBOL'78
1	選択文	IF END-IF	IF {THEN　命令－1 　　NEXT SENTENCE} {ELSE　命令－2 OTERWISE NEXT SENTENCE} END-IF	任意の多重IF文(IFのネスト)が構成できる。 NEXT SENTENCEは対応するEND-IFの直後の意味である。	ESCOBOLに同じ。 ただしNEXT SENTENCEを禁止している。
2	繰り返し文	PERFORM-NEXT (終了条件) END-PERFORM	PERFORM-NEXT △ ～ TIMES MANY 〃 UNTIL 条件 VARYING 〃 END-PERFORM	繰り返すべき手続きをPERFORM-NEXTの直後に書ける。 MANY TIMES 指定はESCOBOLの特長である。	インラインPERFORM文として実現されている。 PERFORM-NEXTの代りにPERFORMだけでよい。 終了判定箇所指定(WITH TEST AFTER)ができる。
3	脱出文	LEAVE △ 名前 FINAL		繰り返し構造の任意の位置から脱出できる。 直近、名前付、または最外側のEND-PERFORMの外に出る。	対応する命令はない。
4	オプション文	*OPTION*		ESCOBOLプログラムの先頭でIndentation、CIF出力、およびRenumberの要、不要を指定できる。	

CODASYL COBOL'78のEVALUATE文（CACE文）相当のものはESCOBOLにはない

ＥＳＣＯＢＯＬとＣＯＤＡＳＹＬの相違点は次の通りである。

(1) ＩＦ～ＥＮＤ－ＩＦを導入し構文が長くなってくるとSentenceの意味があいまいになってくるので、ＣＯＤＡＳＹＬではＮＥＸＴ　ＳＥＮＴＥＮＣＥの使用を禁止しているが、ＥＳＣＯＢＯＬでは英語の意味にこだわらず、NEXT ＳＥＮＴＥＮＣＥを「対応するＥＮＤ－ＩＦの直後へコントロールを移す」と明確に定めている。この方が有効の場合が多い。

(2) ＥＳＣＯＢＯＬでは繰り返し終了条件の一つとしてＭＡＮＹ　ＴＩＭＥＳを設けた。これは際限ない繰り返しを意味するが、ＬＥＡＶＥ文と対応してスマートな構造のプログラムを実現する。

(例)
```
PERFORM-NEXT  MANY  TIMES
READ  INPUT-FILE  AT  END  LEAVE

[繰り返し手続き]

END-PERFORM
```

(3) CODASYLにはESCOBOLにない．WITH TEST AFTERの終了判定箇所指定がある。ESCOBOLはその代り任意の場所に挿入できるLEAVE文がある。LEAVEはIFと組み合わせることにより任意の場所で終了判定ができるのでより一般的といえる。Switch等の導入によって無理にループの始あるいは終で終了判定をする必要がない。なおLEAVEにはループが入れ子になっている場合に多段飛びこし脱出の機能もある。

(4) ESCOBOLにはCODASYLで提案されているevaluate文（case文）に相当するものがない。これは機会があったら追加していきたい。

4 実用化にあたっての問題点

ESCOBOLは昭和53年春にその第1版を完成し、その後ほぼ3年間にわたって国鉄内の5機種（メーカーは3社。研究所のものを除く）で実務に使用しながら改善を重ね，最近ESCOBOL／81として改定版をリリースした。以下主な改良点を述べる。

(1) レベル表示の導入

字下げをする場合，レベルが深くなると対応が分りにくくなるのでそれを表示する縦線（：コロン）を導入した。

```
PERFORM-NEXT
  UNTIL KEY-OLD = SYURYO AND
        KEY-TRN = SYURYO.
  IF KEY-OLD < KEY-TRN
  PERFORM OLD-TO-NEW ELSE
  IF KEY-OLD > KEY-TRN
  PERFORM TRN-TO-NEW ELSE
  IF T-TEISEI = SYUSEI
  MOVE T-DATA TO N-DATA
  PERFORM TRN-TO-NEW ELSE
  IF T-TEISEI = SAKUZYO
  PERFORM READ-FILE  ELSE
  PERFORM ERR-INZI
  END-IF
  END-IF
  END-IF
  END-IF.
END-PERFORM.
```

（入力プログラム）

```
PROCEDURE DIVISION.
    PERFORM-NEXT
    :   UNTIL KEY-OLD = SYURYO
    :   AND KEY-TRN = SYURYO.
    :   IF KEY-OLD < KEY-TRN
    :   :   PERFORM OLD-TO-NEW
    :   ELSE
    :   :   IF KEY-OLD > KEY-TRN
    :   :   :   PERFORM TRN-TO-NEW
    :   :   ELSE
    :   :   :   IF T-TEISEI = SYUSEI
    :   :   :   :   MOVE T-DATA TO N-DATA
    :   :   :   :   PERFORM TRN-TO-NEW
    :   :   :   ELSE
    :   :   :   :   IF T-TEISEI = SAKUZYO
    :   :   :   :   :   PERFORM READ-FILE
    :   :   :   :   ELSE
    :   :   :   :   :   PERFORM ERR-INZI
    :   :   :   :   END-IF
    :   :   :   END-IF
    :   :   END-IF
    :   END-IF.
END-PERFORM.
```

（プリコンパイルリスト）

図 1 インデンテイションの例

(2) 字下げリスト保存ファイル（CIF）の導入

字下げをして見やすいプリコンパイルリストを出すのはよいが，印刷されたリストはプログラマが書いたものとは行数および桁位置が異なってくる。そのためプリコンパイルリストには「入力行番号」として、もとの行を表示し、プログラム保守の便をはかっている。これで十分に保守はできる筈である。

しかしもとと違うのは不便であることを否めないので、プリコンパイルリストと同じものを次回以降のプログラムファイルとするよう，プリコンパイル時に出力する機能を設けた．（出力はオプション）この場合同一行が複数に分れることがあるので、リナンバ機能も設けてある．（リナンバもオプション）

(3) エラーのときのCIF出力

前述のCIFはプリコンパイルエラーの時は出力不要と考えたが(新たにやり直せばよい)ある現場からエラーの時も出力してほしいと言われ(元のファイルを必らず置きかえるようにしているため)そのようにした。これなどは考え方の問題であり、またプログラムライブラリ管理方式の問題である。これに類する細かい問題がたくさんあった。

(4) 連続プリコンパイル

プリコンパイルはプログラム単位に行なうものと考えそのように作っておいたが、ある機種ではプログラムライブラリ管理方式の制約から、その時間帯に処理するものはすべて連続してプリコンパイルする必要があり、当該機種用の連続プリコンパイルバージョンを開発した。この問題をお互いに理解し合うのに相当な時間がかかったことを付記しておく。

以上の改良点のうち (1) (2) (3) の3つまで字下げを採用したことにもとづくもので、字下げが実用上いかに多くの問題を含んでいるかがよく分った。またこれらの改良に際し、各メーカーの間の差をあらためて認識させられた。磁気テープは互換性があってどのメーカーのものでも読め、COBOL言語も一応互換性がある。しかしその間のところ、例えばプログラムライブラリの管理方式が各社(極端な場合同一メーカーでも機種により考え方がちがう)ばらばらで、いろいろと苦労するのである。

5 あとがき

ESCOBOLを最初に開発してから改良に約3年もかかったのは、本文でも述べたように各メーカーのプログラム管理方式等に予想外のばらつきがあったことにもよるが、それ以上にこの種の技法の普及の速度が遅いことに原因がある。最初のうちはプリコンパイルの意味の理解不足から、プリコンパイルリストとコンパイルリストの二重管理が必要だとか、COBOL文への変換規則が分らないと使えないとかの意見が出て、普及のむずかしさを痛感した。コンパイラ言語を初めて使った人がコンパイル後のオブジェクトプログラムを追わないと気がすまなかったことに対比できようか。

最近各種の勝れたソフトウェアツールが多く開発されているが、ESCOBOLのようにどの機種でも使えるものは少ないようである。ソフトウェア工学の戦国時代が早く乗りこえられ、統一仕様のツール類が多く出現することを強く望むものである。

参考文献

(1) 関, 二階堂, 大規模システムにおけるソフトウェア管理, 情報処理学会ソフトウェア工学研究会(12-5) 1979
(2) 植村, 真野 整構造COBOL(1)~(3) bit Vol.10, No.11~14 1979
(3) 関 二階堂 プリコンパイラESCOBOL 昭和54年度情報処理学会第20回全国大会 P145~146