プログラミング言語　6－3
（1986. 5. 23）

COBOL85

COBOL言語の進化、標準化および認証

植村俊亮
（電子技術総合研究所ソフトウェア部）

COBOL言語の第3次規格制定の作業が進んでいる。言語仕様が最初に登場してから25年が経過したが、COBOLはいぜんとしてよく使われている。その理由の大半は、情報技術の進歩発展にあわせて、言語仕様の適切な保守改訂を怠らなかった関係者の不断の努力に帰せられるべきであろう。本稿では、四半世紀にわたるCOBOL言語の歴史と進化のあとを概観し、第3次規格を中心に、最新のCOBOL言語仕様を紹介して、その問題点を分析する。とくに整構造プログラミング機能を取り上げる。また関連して、COBOL言語仕様についての論争点の一つであった予約語の変遷を論じる。最後に、COBOLコンパイラの認証機関とそのシステムとについて報告する。

COBOL85 -- Evolution, Standardization and Validation of the Programming Language COBOL

Syunsuke UEMURA
Electrotechnical Laboratory, 1-1-4, Umezono, Sakura, Niihari, Ibaraki-ken 305

CODASYL COBOL Committee celebrated its silver jubilee in 1984, with the publication of the "COBOL Journal of Development 1984 25th Anniversary Issue". COBOL is one of the oldest and yet the most widely used programming languages even now. This is mainly because the language specification of COBOL has been updated evolutionaly to meet the state of the art of the information technology. This paper first describes the evolutional history of COBOL. It then discusses and analyses the features of the recent COBOL specifications, with special emphasis on its structured programming facility and the 3rd international COBOL Standard. Also introduced is the public validation organization and mechanism for COBOL compilers.

1. COBOLの現状

COBOL言語は，1959年にデータシステム言語協議会（CODASYL）が開発した．翌1960年に言語仕様書が始めて公刊されて，これが以後4半世紀を越えるCOBOLの歴史の出発点となった．1960年の言語仕様書の題名は，

“COBOL - A Report to the Conference on Data Systems Languages, including Initial Specifications for a Common Business Oriented Language (COBOL) for Programming Electronic Digital Computers”

といった．言語仕様書は，書名や体裁をすこしずつ変えながら，改訂版の刊行を続けている．データシステム言語協議会による最新のCOBOL言語仕様書は，1984年に刊行された次の報告書である[1]．

“CODASYL COBOL Committee Journal of Development 1984 25th Anniversary Issue”

これが，いわばCOBOL原典である．以下これをCODASYL COBOLとよぶ．

COBOL言語標準化の作業は，1962年にアメリカで開始された．データシステム言語協議会という開発組織があるので，言語の標準化をいかに進めるべきかがかしましく議論された．結局，COBOL言語仕様の開発はデータシステム言語協議会が，標準化はアメリカ規格協会が行なうという基本的な分担が明確になった．標準化の段階では，言語仕様の明確化はデータシステム言語協議会と連絡をとりながら行なうが，新しい機能の開発は行なわない．実際アメリカ規格協会からの言語仕様明確化の提案は，データシステム言語協議会のCOBOL委員会が慎重審議している．しかし最近では，開発と標準化との時間的なずれにともなう複雑な問題が発生して，単なる明確化では対応しきれなくなっており，原典にとらわれない標準化がまた論争をよんでいる．

1968年になって，ようやくアメリカ規格COBOLが制定され，これをもとに1972年に国際推薦規格とJIS規格とが制定された．以下これを才1次規格とよぶ．内容は1967年1月のCODASYL COBOLの部分集合であった．

1974年にアメリカ規格COBOLが改訂され，これをもとに1978年に国際規格を改訂，1980年にはJIS規格が改訂された[3]．以下これを才2次規格とよぶ．内容は1971年12月のCODASYL COBOLの部分集合であった．

1985年にアメリカ規格COBOLが再改訂され，国際規格COBOLも同年末に再改訂された[2]．これをもとにJIS規格を再改訂することをめざして，原案作成が1985年，1986年の2年計画で進行中である．以下これを才3次規格とよぶ．内容は，およそ1978年ごろのCODASYL COBOLに対応する．才3次規格作りはきわめて難行し，才2次との間に10年（わが国は7年？）の時間を要したことになる．

これが，いわばCOBOL規格の流れである．

以上の各種言語仕様の発展の概略と，相互の関連とを，図1に示す．

COBOLは，公的機関によるコンパイラ認証制度をいち早く確立したプログラム言語でもある．コンパイラが規格に合致していることを検証するシステムの必要性は，才1次規格当時から論じられ，アメリカ海軍を中心にそうしたシステムの開発が進められてきた．現在では，連邦政府の調達規格を満足していることを証明するために，FSMC（Federal Software Management Support Center）によるコンパイラ認定証が要求される．イギリスでは，National Computing Centreがこのサービスを提供し，アメリカと相互認証している．

CODASYL COBOL	特徴，主要な言語仕様改訂など	標準化など
COBOL-60	・1960年に公表された最初のCOBOL言語仕様．	
COBOL-61	・初期のコンパイラは，ほとんどこれによる． ・COBOLの基礎がこれで固まった．	
COBOL-61 Extended	・1963年 ・整列(SORT)文，報告書作成機能を追加．	[情報処理学会「和訳COBOL」(1963年)]
COBOL,1965年版	・磁気ディスクファイル機能を追加． ・表引き（SEARCH）文を追加．	[情報処理学会「COBOL,1965年版」(1968年)]
COBOL,1968	・プログラム間連絡機能を追加． ・NOTE，REMARKSを削除．注記文導入． ・この年から，JOD（開発報告）形式になる．	ANS X3.23-1968 COBOL
COBOL,1969	・通信機能を追加． ・文字列操作(STRING,UNSTRING)文を追加．	
COBOL,1970	・デバッグ機能，併合(MERGE)文を追加． ・報告書作成機能を全面改訂．	ISO R 1989-1972 COBOL JIS C 6205-1972 COBOL
COBOL,1973	・磁気ディスクファイル機能を全面改訂（相対ファイル，索引ファイルを導入． ・レベル番号77を書く位置を自由にした．	ANS X3.23-1974 COBOL
COBOL,1976	・データベース機能を追加． ・ビット操作機能を追加． ・レベル番号77，ALTERを削除．	
COBOL,1978	・整構造プログラミング機能を導入． ・プログラム間連絡機能を改訂，強化（プログラムの構造，ファイルの構造）．	IS 1989-1978 COBOL JIS C 6205-1980 COBOL
COBOL,1981	・浮動小数点形式，組込み関数を導入． ・正書法を改訂．自由形式にした． ・ENTER，CORRESPONDING，ラベル処理を削除．	
COBOL,1984	・検証(VALIDATE)機能を導入． ・SYNCHRONIZED，RERUNを削除． ・「25周年記念」と銘打った仕様書として刊行．	ANS X3.23-1985 COBOL IS 1989-1985 COBOL JIS C 6205-1987(?) COBOL

図1. COBOLの進化

2. 才3次規格COBOLの概要[4]

2.1 おもな特徴

才3次規格のための作業は，アメリカでは，1977年に始まった．しかし1970年代に入ってから，CODASYL COBOLは言語仕様をかなり激しく動かしていたので，標準化の立場でこれをどうとらえるか，なかなか意見が収束せず，原案作成に時間がかかった．ALTER文のように，一度は廃止が予定されながら，次期規格まで延期されたものがおおい．おもな特徴を以下に列挙する．

(1) 機能単位の水準の組合せを整理して，規格COBOLを三つの水準にまとめた（図2参照）．

(2) 廃要素（obsolete element）の考え方を導入し，言語仕様の激変をさけた．

(3) 整構造プログラミングの特能を導入し，プログラム間連絡特能を拡充した．

(4) データベース特能は，才3次規格の補遺とする計画で後廻しにした．

(5) 正書法の自由化など，いくつかのCODASYL COBOLの改訂を，規格には含めないで見送った．

2.2 規格COBOLの水準

才2次規格COBOLまでは，仕様が中核と11の特能単位とに分かれており，それぞれに水準が定められていて，水準の組合せは，処理系作成者の裁量にまかされていた．

才3次規格では，これを以下のように整理した（図2）．

(1) 中核と表操作とを一つの特能単位としてまとめた．

(2) 11の特能単位を，七つの必須特能単位と四つの任意特能単位とにまとめた．任意特能単位（報告書作成，通信，デバッグ，区分化）を作成するかどうかは，作成者の自由にまかせる．なおデバッグ特能（デバッグ行は除く）と区分化特能とは，次項にいう廃要素である．

(3) 規格COBOLを上位水準，中間水準，低位水準の三つの水準だけで構成する．特能単位ごとに水準が規定されているが，図2に示す組合せだけしか許さない．すなわち三つの水準は次のように規定されている．

(i) 上位水準 ― 七つの必須特能単位のすべての最高水準を作成．（整列併合は水準2がないことに注意．）

(ii) 中間水準 ― 七つの必須特能単位のすべてについて，水準1を作成．

(iii) 低位水準 ― 中核，順ファイル，プログラム間連絡の各水準1を作成．

前述のFSMCによる認定の統計をみても，たいへん妥当な水準の規定であると考えられる．

2.3 廃要素

廃要素は，規格COBOLの次回の改訂で削除される言語要素である．改訂時に突然削除するのではなくて，まず廃要素として，将来削除予定であること

		上位	中間	低位
必須特能単位	中核	水準2	水準1	水準1
	順ファイル	水準2	水準1	水準1
	相対ファイル	水準2	水準1	なし
	索引ファイル	水準2	水準1	なし
	プログラム間連絡	水準2	水準1	水準1
	整列併合	水準1††	水準1	なし
	原始文操作†	水準2	水準1	なし
任意特能単位	報告書作成	水準1		
	通信	水準1	水準2	
	デバッグ	水準1	水準2	
	区分化	水準1	水準2	

† 登録集特能の拡張・改称である．　†† 水準2はない．

図2 規格COBOLの水準

を明示，警告する．処理系は，プログラムが廃要素を使っていることを警告する枠構を用意しなければならない．

第3次規格で廃要素となった言語要素のおもなものを以下に列挙する．

(i) 区分化枠能．
(ii) デバッグ枠能（デバッグ行は残る）．
(iii) ALTER文．
(iv) ラベルレコード句，データレコード句．
(v) 見出し部の PROGRAM-ID 以外の段落．

廃要素は，プログラムの互換性，寿命などを勘案した上での，改革のための妥協であろう．

2.4 整構造プログラミング[5]

今回の改訂の中で，プログラム言語としてもっとも興味深く，実用的価値もありそうな部分は，これである．

データシステム言語協議会の COBOL 委員会が，1975年ごろにこの枠能を検討し始めたときの，いくつかの採決結果をまず紹介する．

(i) 整構造プログラミング枠能を導入するが，従来の言語仕様と混然一体となるように配慮する（賛成13，反対3，棄権1）．
(ii) IF文の分岐の終りを明示する書き方を予約語として導入する（13-0）．
(iii) GO TO文を削除する（0-12）．
(iv) CASEに相当する文を作る（11-0）．
(v)「一つの入口，一つの出口」の原則を守る（4-11-3）．
(vi) ブロック構造を明示する枠能（begin~end）を導入する（3-9）．

図1に示したように，1978年の CODASYL COBOL にこの枠能が導入され，その大半が第3次規格に採用された．以下に要点をまとめる．

2.4.1 制御構造

(1) END-IF など，制御の流れの終りを明示する語を導入する．“END-動詞”という形の予約語を範囲明示符という．範囲明示符で終る文を範囲明示文という．範囲明示文は無条件文である．

例．
```
IF SYOKUSYU = "EIGYOO"
    IF URIAGE > 1000000
        THEN COMPUTE KYUUYO =
                     KYUUYO * 1.5
    END-IF
    COMPUTE KYUUYO = KYUUYO * 1.2
ELSE COMPUTE ...
END-IF
```

従来どおり終止符空白で IF 文を完結文にしてまとめることもできる．しかしこの改訂で，完結文の存在意義は希薄になった．なお NEXT SENTENCE の部分には，CONTINUE 文を書くことができる．

(2) 条件文に対称な分岐ができた．たとえば READ 文では，範囲明示符 END-READ に加えて，NOT AT END 句を書くことができる．

例．
```
READ NYUURYOKU-FILE
    AT END PERFORM ATOSIMATU
    NOT AT END PERFORM SIGOTO
END-READ
```

こうした書き方が最適であるかどうか，検討の余地が残っている．

(3) PERFORM 文を次の2点で大拡張した．

(i) 従来の do while 型の PERFORM 文に加えて，repeat until 型の書き方が可能になった．すなわち繰返し条件の検査を，繰返しの前に行なう（WITH TEST BEFORE）か，後で行なう（WITH TEST AFTER）かを指定できるようになった．指定を省略すると，これまでどおり TEST BEFORE とみなされる．

(ii)「うち」PERFORM が書けるようになった．従来の PERFORM 文では，繰り返して実行する本体の手続き名だけを書いて，本体はよそに書くことしかできなかった．こんどは，本体の文を PERFORM 文自身のなかに直接書けるようになった．「うち」PERFORM 文の本体中には，手続き名を書くことができない．本体の繰返しを強制終了させる文

(EXIT PERFORM) は, CODASYL COBOL には存在するが, オ3次規格には向に合わなかった.

例. PERFORM UNTIL OWARI-SINGOO = 1
    READ NYUURYOKU-FILE
     AT END MOVE 1 TO OWARI-SINGOO
     NOT AT END
      PERFORM TEST-AFTER
       VARYING I FROM 1 BY 1
       UNTIL I = 80
        MOVE NYUURYOKU-KETA (I) TO ...
        PERFORM SYORI
      END-PERFORM
      WRITE SYUTURYOKU-RECORD
    END-READ
   END-PERFORM

(4) case構造に相当する EVALUATE文ができた. これで多枝分岐, 多枝結合が可能になった. たいへん強力な書き方が許される大きい文である.

例. EVALUATE SYOKUSYU SEISEKI TRUE
    NENREI
WHEN "E" "A" KAZOKU-SUU > 5
     25 THRU 30
  COMPUTE KYUUYO = KYUUYO * 1.2
WHEN "K" "A" THRU "C" ANY ANY
WHEN "S" ANY ANY ANY
  COMPUTE KYUUYO = KYUUYO * 1.5
WHEN OTHERS
  CONTINUE
END-EVALUATE

2.4.2 COBOLプログラムの構造

(1) COBOLプログラムの入れ子ができるようになった. 原始プログラムの手続き部の最後の部分に, ほかのCOBOL原始プログラムを入れ子にできる(図3). 入れ子プログラムの終りには, プログラム終り見出し (END PROGRAM プログラム名) を書いて, 終りを明示しなければならない.

(2) CALL文は, 別にコンパイルされるプログラムだけではなくて, 入れ子になったプログラムをも呼び出すことができる. ただしこれまでどおり, 再帰的な呼出しは禁止されている. 入れ子の外側を呼ぶこともできない. 一つのプログラムに含まれる複数のプログラムが共通して副プログラムを使いたいときは, プログラム属性 COMMONを指定する.

(3) データ名の有効範囲を指定する GLOBAL, EXTERNAL (およびCOMMON) 指定が導入された. GLOBAL属性をもつデータ名の有効範囲は, 自分自身と, そこに直接または間接に含まれるすべてのプログラムである. 作業場所節の01レベルのデータ項目にEXTERNALと書くと, そのレコードが外部レコードとなる. 複数のプログラムが同じ外部レコードを記述して, データを共用することができる. こうした指定をしないデータ名の有効範囲は, 原則としてそれを記述したプログラムのなかだけであって, 入れ子になっているプログラムのやまではおよばない.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. A.

  IDENTIFICATION DIVISION.
  PROGRAM-ID. B.

  END PROGRAM B.
  IDENTIFICATION DIVISION.
  PROGRAM-ID. C.

    IDENTIFICATION DIVISION.
    PROGRAM-ID. D COMMON.

    END PROGRAM D.
  END PROGRAM C.
END PROGRAM A.
```

図3. COBOLプログラムの入れ子

2.4.3 引数の引渡し方

呼ぶプログラム中のCALL文が，呼ばれるプログラムを呼び出す．このときCALL文自身と，呼ばれるプログラムの手続き部見出しとにあるUSING句で，データの引渡しを行なう．この引渡し方が大拡張された．両方のUSING句に書いた順番によって対応づけがきまるが，対応づけの方式が二つできた．

(i) 記憶場所共有（BY REFERENCE）と指定すると，CALL文で呼ばれたプログラムの実行中つねに，対応づけられたデータの値が等しいことが保証される．これは記憶場所を共有することと解釈される．

(ii) 値引渡し（BY CONTENT）と指定すると，呼ばれるプログラムの実行開始時には，両方のUSING句に書いた対応するデータ項目の値は一致するが，プログラムの実行中に，呼ばれたプログラムの側のデータ項目の値をいくら変更しても，CALL文の側のデータ項目の値は変わらない．

2.4.4 ファイルの共有

データに関するGLOBAL，EXTERNALと同じ属性をファイルに対しても指定できる．ただし実行時の共有（同時実行処理）については，データシステム言語協議会の審議待ちである．

データ部にFD（ファイル記述）項を書くと，ファイル結合子が定義されるという解釈となった．

2.5 その他の改訂

INITIALIZE（初期値設定）文，REPLACE（置換え）文ができた．表の次元数の上限を7とした．PERFORM文の繰返しも，VARYING句および6個までのAFTER句で，深さ7まで可能になった．データ項目の一部分を直接に参照できる．例．MOVE KETA (I:6) TO ...

FILLERは，書かなくてもよい．その他．

3. 予約語

COBOL言語が登場してからしばらくの間は，予約語の存在に対する否定的な議論が多かった．おぼえきれない，コンパイラ作成技術上不要であるといった論旨であった．しかし今日では，むしろ予約語の積極的な意味が認められるようになっている．予約されている不便よりも，プログラム全体をわかりやすくする効果が評価されているのであろう．

言語仕様の改訂を予約語に反映させる方向は二つ考えられる．

(1) 一度予約した語は墨守する．

(2) 予約語も連動させて改訂する．

CODASYL COBOLは，1965年版あたりまでは(1)であったが，やがて(2)に変化した．言語仕様の改訂にともなう予約語数の変化を図4に示す．個々の予約語の変遷については，参考文献6)を参照されたい．

予約語は，この25年間に7割（規格系列で5割）も増加している．この値は増減を含んでいるので，実際の変化はもっと大きい．COBOL-61に掲げられていた222の予約語のうち，この8種の版に共通して予約され続けていた語は，わずか134にすぎなかった．

	C61	C65	C67	C72	C84
CODASYL COBOL	222	284	273	328 (うち記号 8)	390 (別に記号21)
			J72	J80	A85
規格 COBOL			235	308 (うち記号 8)	358 (うち記号10)

記号の意味は次による．

C61	COBOL-61 （和訳COBOL）
C65	COBOL 1965年版
C67	COBOL 開発報告 1967 （1967年 9月版）
J72	日本工業規格COBOL C 6205-1972
C72	COBOL 開発報告 1970 （1972年 5月版）
J80	日本工業規格COBOL C 6205-1980
A85	アメリカ規格COBOL （第 3次改訂案）
C84	COBOL 開発報告 1984

図4 予約語数の変化

4. COBOL コンパイラの認証

1963年1月にアメリカ規格協会で開かれたX3.4.4の最初の会合で，COBOL標準検定問題集を作ることも検討された（X3.4.4の本来の目的は，アメリカ規格COBOLの起案であった）．実質的な作業は，アメリカ海軍や空軍が進めて，現実的な検定システムを開発した．1972年7月から，アメリカ連邦政府関係の村関が購入するCOBOLコンパイラは，すべて連邦規格COBOL（FIPS PUB 21-1）のいずれかの水準を満足しなければならないことになった．所定の水準を満足していることは，共通役務庁（General Services Administration）のFSMCによるコンパイラ検査を受けて，そこが発行する認定証によって，証明する．認定証といっても，実態は認定コンパイラ一覧（Certified Compiler List）に掲載されればよい．この一覧は季刊で，常時改訂が行なわれている．最新版（1986年4月）では，34社の101のコンパイラが「問題なし」と認証されている[8]．

コンパイラ検査を行なう認証システムの詳細および実施例については，参考文献7）を参照されたい．

FSMCは，1986年7月から，次期規格のための新認証システムによる検査を行なう．この新システムは，イギリスのNCC（National Computing Centre）が開発したと伝えられている．約800の検査プログラムから成るという．イギリスでも，Central Computer and Tele-Communication Agency（CCTA）が，この認証を奨励しており，アメリカとイギリスとは，相互認証を行なっている．

なおFSMCは，COBOL以外にも，FORTRAN，BASIC，Ada，Pascalの認証を行なっている．後2者の認証システムは，別途開発されたものである．いずれにせよ，わが国の対応はいちじるしく遅れている．

5. おわりに

（プログラム言語仕様書の法則）

プログラム言語仕様書のページ数は，改訂により増加する．

御教示いただいた情報処理学会JIS COBOL原案作成委員会の諸氏に感謝いたします．本稿では，同委員会が検討中の用語案と異なる用語をあえて使ったところがあります．

参考文献

1) CODASYL COBOL Committee Journal of Development 1984 25th Anniversary Issue, the Secretariat of the Canadian Government EDP Standards Committee (1984)

2) American National Standard Programming Language COBOL, X3.23-1985 (1985)

3) 日本工業規格電子計算機プログラム言語 COBOL JIS C 6205-1980 (1980)

4) 今城哲二：COBOLの標準化の動向，情報処理，Vol. 24, No. 9, pp. 1062-1069 (1983)

5) 植村俊亮，真野芳久：整構造COBOL (1)～(3)，bit，Vol. 10, Nos. 11, 12, 14, 共立出版 (1978)

6) 植村俊亮：COBOL言語における予約語，ETL-RM-85-31J，電子技術総合研究所 (1985)

7) 広瀬健編：ソフトウェア評価技法，bit臨時増刊1982年9月 (1982)

8) CERTIFIED COMPILER LIST, Report OIT/FSMC-86/004, April 1, 1986 (1986)

END-COBOL85