ソフトウェアの移換性から見た FORTRAN と COBOL の比較†

藤村 直美††

異なる計算機システム間で容易に移し換えられ，さらに移し換え先のいずれの計算機システムにおいても能率良く動作するソフトウェアを作成できれば，ソフトウェアの開発・利用の両面から有意義である．そのためには適切な記述言語を選択し，注意深くソフトウェアを記述する必要がある．ところがどのような記述言語を選べば，ソフトウェアの移換性（移し換えやすさ）を高めるために適当であるかという観点からの研究はなされていない．

ここでは主として文字や文字列を処理の対象とするある種のシステムプログラムを記述する際の言語選択の問題を，ソフトウェアの移換性という観点から，次の4項目について比較・検討を行う．

（1） 同一手続きを記述するために必要なソースプログラム行数
（2） 目的プログラムの実行効率
（3） 目的プログラムの大きさ
（4） 移し換えに伴うソースプログラム変更行数

その結果として，純粋な数値計算を除けば，COBOL で記述した方が実行速度は速く，目的プログラムの大きさは小さいこと，他方 FORTRAN で記述すると移し換えに際して必要なソースプログラムの修正は COBOL に比べてはるかに少ないことなどを明らかにしている．

1. はじめに

ある計算機システムで価値を認められたソフトウェアが別の計算機システムに困難なく移し換えられて能率良く働くことができれば，ソフトウェアの作成・使用の両面で有意義である．ソフトウェアの移し換えでは，問題のソフトウェアが移し換えを考慮して作成されているか否かが，移し換え作業の容易さ，作業の所要時間などに大きく影響する．あらかじめ移換性を考慮してソフトウェアが作成されていないと，移し換え作業に莫大な費用と時間を要し，移し換えが事実上不可能になることも起こり得る．ところがソフトウェアの作成に当って，移換性を高めるために具体的にどのような方策を採用すれば良いかという指針が必ずしも与えられているとは言い難い[1),2)]．

移換性という観点から見て望ましいソフトウェアとは，異なる計算機システム間で容易に移し換えられ，しかも移し換え先の計算機システムで充分実用になるだけの性能を有するものである．そのため新しく移換性の高いソフトウェアを作成するとき，記述言語の選択は重要な問題である．

ここでは主として文字や文字列を処理の対象とする，ある種のシステムプログラムを記述する際の言語選択の問題をソフトウェアの移換性という観点から FORTRAN と COBOL を中心に考察する．特にこの2つの言語を選ぶのは現在世界的に広く使われており，ほとんどの計算機システムにおいて使用できるためである．ここでは特に次の4項目を中心に比較・検討を行う．

（1） 同一手続きを記述するのに必要なソースプログラム行数
（2） 目的プログラムの実行効率
（3） 目的プログラムの大きさ
（4） 移し換えの難易度として，移し換えに伴うソースプログラム変更行数

計測の対象とする計算機システムおよび言語処理系は次の通りである．

（a） 九州大学工学部情報工学科の FACOM 230-45 S OS II/VS（以後 OS 2 と呼ぶ）の COBOL[3)]と FORTRAN[4)]．

（b） 九州大学大型計算機センターの FACOM M-190 OS IV/F 4（以後 OS 4 と呼ぶ）の COBOL[5)]と FORTRAN[6)]．

（c） 東京大学大型計算機センターの HITAC 8800/8700 OS 7（以後 OS 7 と呼ぶ）の COBOL[7)]と FO-

† A Comparison of FORTRAN and COBOL from the Viewpoint of Writing Portable Software by NAOMI FUJIMURA (Department of Computer Science and Communication Engineering, Kyushu University).

†† 九州大学工学部情報工学科

RTRAN[8].

なお OS 4 は IBM の OS/VS 2 と基本的に同等である.

2. テストプログラム

表 1 に示す 4 種類の手続きを FORTRAN と COBOL で記述してテストプログラムとする. COPY は入出力のみ(**図 1** 参照), STINGY は入出力に内部処理を少し加えたもの[9], COBOLDAP は COBOL プログラム輪郭作成システム[10]の前処理部で構文解析などの内部処理のほかにかなりの入出力処理を伴うもの, MATRIX は入出力処理なしである(**図 2** 参照). MATRIX は文字を処理するプログラムではないが, 数値計算の能力を比較するために特に加えた.

手続きの記述に際しては次のような点に注意した.

(1) FORTRAN 版と COBOL 版で同じアルゴリズムを採用する. ただし, そのために特に不自然な記述にならないようにする.

(2) FORTRAN 版の記述に際しては, 移換性を高めるために参考文献 1) や 11) で述べられている点に注意する. たとえば次のような諸点である.

- 標準規格から拡張された部分, 規格から逸脱した部分, 規格の定義が不明確な部分を使わない.
- 文字は一語に一文字とする.
- 文字のみを含む順編成ファイルを順アクセスで使う.
- OS に依存した機能は可能な限り使わない.

(3) COBOL 版の記述に際しては参考文献 2) で述べられている点に注意する. しかしながら移換性を高めるために手続きの記述を規格の低水準に限定してプログラムを書き難くするようなことはしない.

表 1 テストプログラム

Table 1 List of test programs.

プログラム名	機　　能
COPY	ソースプログラムファイルをコピーする
STINGY	ソースプログラムファイルを圧縮して印刷する
COBOLDAP	COBOL プログラム輪郭作成システムの前処理を行う
MATRIX	100×100 の行列の掛算を行う

```
C      COPY TEST PROGRAM
C
       INTEGER CR(80)
C
       NIN=1
       NOUT=2
C
  100  READ(NIN,10,END=200) CR
       WRITE(NOUT,10) CR
       GO TO 100
C
  200  STOP
C
   10  FORMAT(80A1)
C
       END
```

(a) FORTRAN 版
(a) FORTRAN version.

```
00010  IDENTIFICATION    DIVISION.
00020 *
00030  PROGRAM-ID.       CRCOPY.
00040  AUTHOR.           NAOMI FUJIMURA.
00050 *
00060  ENVIRONMENT       DIVISION.
00070  CONFIGURATION     SECTION.
00080  SOURCE-COMPUTER.  FACOM-M190.
00090  OBJECT-COMPUTER.  FACOM-M190.
00100 *
00110  INPUT-OUTPUT      SECTION.
00120  FILE-CONTROL.
00130       SELECT   INPUT-FILE      ASSIGN TO S-INPUT.
00140       SELECT   OUTPUT-FILE     ASSIGN TO S-OUTPUT.
00150 *
00160  DATA              DIVISION.
00170  FILE              SECTION.
00180 *
00190  FD  INPUT-FILE
00200       RECORDING MODE IS F
00210       BLOCK CONTAINS 0 RECORDS
00220       LABEL RECORD IS STANDARD.
00230  01  CR.
00240       10 FILLER      PIC    X(80).
00250 *
00260  FD  OUTPUT-FILE
00270       RECORDING MODE IS F
00280       BLOCK CONTAINS 13 RECORDS
00290       LABEL RECORD IS STANDARD.
00300  01  CP.
00310       10 FILLER      PIC    X(80).
00320 *
00330  PROCEDURE          DIVISION.
00340  HAJIME.
00350       OPEN INPUT    INPUT-FILE.
00360       OPEN OUTPUT   OUTPUT-FILE.
00370  CYCLE.
00380       READ INPUT-FILE AT END GO TO OWARI.
00390       WRITE CP FROM CR.
00400       GO TO CYCLE.
00410 *
00420  OWARI.
00430       CLOSE INPUT-FILE.
00440       CLOSE OUTPUT-FILE.
00450       STOP RUN.
```

(b) COBOL 版
(b) COBOL version.

図 1 COPY

Fig. 1 COPY.

3. 記述言語の相違による影響

3.1 ソースプログラム行数

表 2 に各プログラムの行数の内訳を示す. ここで, FORTRAN の非実行文とはソースプログラム全体から注釈行と実行文行を除いたもので, 宣言文, DATA 文, FORMAT 文そのほかの行を含む. COBOL に対

しては実行文と非実行文といった分類はこの場合適当でないので，各DIVISIONごとのソース行数を示している．なお議論の都合上，表2の数値には注釈行の行数は含まれていない．

表2から COBOLDAP を除いて，FORTRAN で記述した方が総所要行数が少なくて済んでいることがわかる．一般に COBOL プログラムでは見出し部 (Identification division)，環境部 (Environment division) などが常に必要なため，ソースプログラム行数は増大する傾向を有する．以下詳細に検討する．

最初に FORTRAN 版の非実行文（主として宣言文）と COBOL 版のデータ部に着目すると，全般的に COBOL 版の方がデータ部の行数が多い．これはデータ構造として配列しか持たない FORTRAN に対して，COBOL はもっと複雑なデータ構造を持ち，そのため宣言が複雑になるためである．一方FORTRAN版では移換性を考慮して，文字の格納をすべて一語に一文字としているため，COBOLDAP の FORTRAN 版では DATA 文で予約語表を初期設定する部分に 410 行とっている．COBOL では同じことが 169 行で簡潔に記述できる．

次に実行文を中心に検討する．COPY の COBOL 版はファイルのオープン・クローズおよび名札の行数だけ FORTRAN 版より手続きの行数が増加している(図1（b）参照)．FORTRAN では文字列の比較，配列の初期設定，配列の内容の転送などをすべてDO文を用いて詳細に記述しなければならない．COBOL では同じことが IF, MOVE 命令を用いて簡潔に記述できる．したがって STINGY では FORTRAN 版の方が実行文の行数が多くなる．

同様の現象は COBOLDAP においても観察される．それにもかかわらず，COBOL 版の方が手続き部の行数が多くなっている．それは主として手続き中に 72 個ある名札が，それ自身で一行を構成しているためで，これを考慮すると COBOL 版の方が FORTRAN 版に比べて手続きはほぼ 30 行少なくなる．

```
C      MATRIX MULTIPLY TEST PROGRAM
C
       REAL A(100,100),B(100,100),C(100,100)
C
       DO 100 I=1,100
       DO 100 J=1,100
         A(J,I)=1.0
         B(J,I)=1.0
  100  CONTINUE
C
       DO 200 I=1,100
       DO 200 J=1,100
         C(J,I)=0.0
         DO 200 K=1,100
           C(J,I)=C(J,I)+A(J,K)*B(K,I)
  200  CONTINUE
C
       STOP
C
       END
```

(a) FORTRAN 版
(a) FORTRAN version.

```
00010  IDENTIFICATION      DIVISION.
00020  PROGRAM-ID.         MATRIX.
00030  AUTHOR.             NAOMI FUJIMURA.
00040 *
00050  ENVIRONMENT         DIVISION.
00060  CONFIGURATION       SECTION.
00070  SOURCE-COMPUTER.    FACOM-M190.
00080  OBJECT-COMPUTER.    FACOM-M190.
00090 *
00100  DATA                DIVISION.
00110  WORKING-STORAGE     SECTION.
00120  77  I                PIC   S9(4)  USAGE IS COMP.
00130  77  J                PIC   S9(4)  USAGE IS COMP.
00140  77  K                PIC   S9(4)  USAGE IS COMP.
00150  01  A-DEF0           COMP-1.
00160      10  A-DEF1       OCCURS  100 TIMES.
00170          20  A        OCCURS  100 TIMES.
00180  01  B-DEF0           COMP-1.
00190      10  B-DEF1       OCCURS  100 TIMES.
00200          20  B        OCCURS  100 TIMES.
00210  01  C-DEF0           COMP-1.
00220      10  C-DEF1       OCCURS  100 TIMES.
00230          20  C        OCCURS  100 TIMES.
00240 *
00250  PROCEDURE           DIVISION.
00260  HAJIME.
00270      PERFORM CL  VARYING I FROM 1 BY 1 UNTIL I > 100
00280                   AFTER  J FROM 1 BY 1 UNTIL J > 100.
00290      PERFORM ML  VARYING I FROM 1 BY 1 UNTIL I > 100
00300                   AFTER  J FROM 1 BY 1 UNTIL J > 100
00310                   AFTER  K FROM 1 BY 1 UNTIL K > 100.
00320      STOP RUN.
00330 *
00340  CL. MOVE 1.0 TO A (I, J), B (I, J).
00350      MOVE 0.0 TO C (I, J).
00360 *
00370  ML. COMPUTE C (I, J) = C (I, J) + A (I, K) * B (K, J).
```

(b) COBOL 版
(b) COBOL version.

図 2 MATRIX
Fig. 2 MATRIX.

MATRIX では手続き部の行数はほぼ同じである．ただし COBOL 版のデータ部が FORTRAN 版の非実行文に比べて多くなっていることが目立っている．

3.2 目的プログラムの実行効率

移し換えられたソフトウェアは相手方の計算機システムでも可能な限り効率良く動くことが実用上望ましい．ここでは OS2 の上で最初に実現した4種のプロ

表 2　テストプログラムの静的構成

Table 2　Number of source lines of test programs.

program name	OS name	FORTRAN			COBOL				
		総行数	非実行文	実行文	総行数	見出し部	環境部	データ部	手続き部
COPY	OS 2	9	3	6	38	3	8	14	13
	OS 4	9	3	6	37	3	8	14	12
	OS 7	9	3	6	37	3	8	14	12
STINGY	OS 2	138	71	67	150	3	10	76	61
	OS 4	139	71	68	133	3	9	62	59
	OS 7	139	71	68	145	3	9	73	60
COBOLDAP	OS 2	1,110	629	481	864	4	12	322	526
	OS 4	1,111	629	482	857	4	10	318	525
	OS 7	1,112	629	483	859	4	10	320	525
MATRIX	OS 2	14	2	12	33	3	4	14	12
	OS 4	14	2	12	32	3	4	14	11
	OS 7	14	2	12	32	3	4	14	11

グラムを OS 2 から各処理系に移し換えて動かし，その実行効率を比較・検討する．計測のために各プログラムの入力データとして，COPY と STINGY はソースプログラムカードに換算して 24,884 枚の磁気テープファイル，COBOLDAP は COBOL で記述された前処理部自身（OS 2 で 967 行，OS 4 で 958 行，OS 7 で 960 行）を使用した．MATRIX はデータなしである．計測は各処理系とも最高度の最適化を指定して行った．ただし OS 2 と OS 7 の COBOL には最適化レベルがない．**表 3** に結果を示す．

表 3 から MATRIX を除いて，すべて COBOL 版の方が速いことがわかる．特に入出力のみの COPY では OS 2 で約 24 倍，OS 4 で約 46 倍，OS 7 で約 41 倍にも速くなっている．このように FORTRAN による入出力が COBOL のそれに比較して遅いのは，FORTRAN の入出力ルーチンが解釈実行型に作成されていること，書式にあわせてデータの編集が必要なことによると考えられる．すなわち COBOL による入出力の場合は連続したデータ領域と入出力バッファ間のデータ転送で済むのに対し，FORTRAN ではデータ転送に加えて，たとえば書式 80 A 1 にあわせて一字ごとに編集が必要となる．

表 3　目的プログラムの実行所要時間

Table 3　Execution time of object programs.

program name	OS 2			OS 4			OS 7		
	F	C	F/C	F	C	F/C	F	C	F/C
COPY	310,243	12,773	24.3	46,680	1,005	46.4	213,957	5,158	41.3
STINGY	450,021	40,463	11.12	63,290	2,420	26.2	237,406	17,329	13.70
COBOLDAP	22,744	13,300	1.710	3,085	905	3.41	13,050	3,999	3.26
MATRIX	23,121	*187,148	1/8.09	1,240	4,980	1/4.02	2,653	43,904	1/16.55

F: Fortran, C: Cobol, F/C: Fortran/Cobol, Unit: msec

* 50×50 の行列の実行所要時間から推定

表 4　目的プログラムの大きさ

Table 4　Size of object programs.

program name	OS 2			OS 4			OS 7 *		
	F	C	F-C	F	C	F-C	F	C	F-C
COPY	26	10	16	31	4	27	31	2	29
STINGY	42	18	24	62	14	48	65	12	53
COBOLDAP	44	28	16	51	19	32	62	17	45
MATRIX	138	**128	10	136	123	13	148	120	28

F: Fortran, C: Cobol

* アドレスバウンダリの調整等のためこれよりやや大きくなる

** 50×50 の行列の時の目的プログラムの大きさから推定

解釈実行型であることは書式を固定しアセンブリ語で記述したルーチンを使用することで改善できる．テストプログラム中で入出力部分の割合が最も高い COPY の入出力部のみをアセンブリ語で記述したものに置き換えると処理時間は約 1/4.7 (310,243→66,350 msec) になる．この状態でなお COBOL 版より 54 秒 (66,350-12,773 msec) も時間がかかっている．これは書式 80 A 1 に基づいて行うデータ (24,884×80 字) の編集に伴う負担と考えられる．個々の計算機の 1 語の大きさに合せて書式仕様をたとえば 40 A 2 とすれば，入出力速度を向上させることができる．しかし移換性の高いソフトウェアを FORTRAN で記述しようとすれば一語に一文字のみ格納するようにすることが必須である[11]．しかしながらそうすることは結果的に入出力を遅くすることになっていると考えられる．

3.3 目的プログラムの大きさ

目的プログラムの大きさは機能・速度が同じなら小さい方が良いことはいうまでもない．**表 4** に各処理系における目的プログラムの大きさを示す．各処理系ごとに機械命令，目的プログラムの構造，システムライブラリルーチンなどの設計が異なるため，目的プログラムの大きさが異なるのは当然としても，どのテストプログラムも FORTRAN 版の方が COBOL 版より大きくなっている．特に OS 7 において FORTRAN 版と COBOL 版の差が，COPY で 29 kB，STINGY で 53 kB，COBOLDAP で 45 kB と大きいのが目立っている．

一方，MATRIX はほとんどが配列のための領域 (120 kB 相当) で，プログラムの大きさの差がほかに比べて目立たないが，これを差引くとやはりかなり差がある

ことがわかる．ただしOS2 COBOLではアドレッシングの制約から100×100の単精度浮動小数点の配列が処理できない．ここでは50×50の配列としたときの目的プログラムの大きさから100×100のときの値を推定している．

なお表4からは明らかでないが，各処理系とも最適化が指定されると，実行速度とは別に目的プログラムの大きさを縮小する．

3.4 移し換えの難易度

ここではプログラムの移し換えの難易度として，移し換える時に必要であったソースプログラム変更行数を比較検討する．プログラムはOS2において実現され，後にほかの処理系へ移し換えられた．**表5**に各処理系におけるソースプログラムの総行数，注釈行数などおよびOS2から移し換えたときのソースプログラム変更行数を示す．

表5から明らかなことは，FORTRANで記述した方が，COBOLで記述した場合に比べて際立って修正行数が少ないことである．COPYやMATRIXのような簡単なプログラムではFORTRANで記述してあると無修正で移し換えられる．そのほかのFORTRANプログラムで修正を必要とする部分も，出力結果の見出しに含まれる計算機システム名や，プログラム中で現在時刻を参照する方法とその形式の相違による修正など，比較的単純なものである．

一方，COBOLプログラムでは，移し換えに伴う修正は計算機名やファイル定義名などの比較的機械的に修正できるものから，データの内部表現の変更など修正のために全く別のテストプログラムを実行して確認する必要があるような[2]面倒なものまである．また出力結果に付加する日付や時刻についても，プログラム中で参照する方法や表現の相違がデータの宣言や手続き部に大きく影響する．

一般にCOBOLは動作環境としてのOSとの整合部をソースプログラム中に環境部，データ部として含んでおり，この部分は計算機システムが変わると必ず影響を受ける．FORTRANでは事実上それに対応する部分がないので，環境の変化すなわち計算機システムの変化があってもその影響を受けにくいといえる．

表5 移し換えに伴う変更行数

Table 5 Number of source lines to be modified.

program name	種類	FORTRAN			COBOL		
		OS 2	OS 4	OS 7	OS 2	OS 4	OS 7
COPY	T	16	16	16	47	45	45
	C	7	7	7	9	8	8
	O	9	9	9	38	37	37
	U		0	0		5	5
	I	—	0	0	—	0	0
	D		0	0		2	2
STINGY	T	194	194	194	170	151	163
	C	56	55	55	20	18	18
	O	138	139	139	150	133	145
	U		3	4		16	10
	I	—	0	0	—	1	1
	D		0	0		20	8
COBOLDAP	T	1,362	1,363	1,364	967	958	960
	C	252	252	252	103	101	101
	O	1,110	1,111	1,112	864	857	859
	U		5	4		45	45
	I	—	1	2	—	0	1
	D		0	0		9	8
MATRIX	T	20	20	20	39	37	37
	C	6	6	6	6	5	5
	O	14	14	14	33	32	32
	U		0	0		16	16
	I	—	0	0	—	0	0
	D		0	0		2	2

T：総行数，C：注釈行，O：その他の行数，U：修正行数，
I：挿入行数，D：削除行数

4. おわりに

現在世界でも最も広く使われているFORTRANとCOBOLについて，ソフトウェアの移し換えという観点から4組のプログラムをもとに比較・検討した．その結果純粋な数値計算を除けば，COBOLで記述したプログラムの方が実行速度は速く，目的プログラムの大きさははるかに小さいことがわかった．近年システムプログラムをFORTRANで記述する例が見られるが[11),12)]，ここで示した結果からはCOBOLの方がそうした目的には適しているように考えられる．

本稿で述べたSTINGYは現在東大大型計算機センター[13)]，九大大型計算機センター[14)]そのほかで一般に公開されている．このプログラムは非常に頻繁に使用されることから，一般公開に際して主にその処理速度を重視し，COBOL版の方を提供している．これが“ともかく一度動けば良いプログラム”であれば，FORTRAN版を移し換えた方が有利である．

なお本研究にあたり，いろいろご指導頂いた九州大学牛島和夫教授に感謝の意を表する．

参考文献

1) 藤村，牛島：FORTRANプログラム動的解析システムの移し換えについて，情報処理，Vol. 17,

No. 11, pp. 1048-1055 (1976).
2) 藤村, 牛島: COBOL プログラム輪郭作成システムの移し換えについて, 情報処理, Vol. 19, No. 9, pp. 853-859 (1978).
3) 富士通: FACOM 230 OS II/VS COBOL 文法書, 87 SP-0240-1 (1974).
4) 富士通: FACOM 230 FORTRAN IV 文法書, 88 SP-0010-2 (1976).
5) 富士通: FACOM OSIV JIS COBOL 文法書, 64 SP-3010-3 (1977).
6) 富士通: FACOM OSIV FORTRAN 文法書, 64 SP-3030-2 (1976).
7) 日立: OS 7 COBOL プログラミング文法, 8700-3-006 (1973).
8) 日立: HITAC 8700 FORTRAN プログラミング文法, 8700-3-002 (1972).
9) 藤村, 牛島: 小さな COBOL プログラムの移換性について, 九大工学集報, Vol. 51, No. 3, pp. 313-317 (1978).
10) 藤村, 牛島: COBOL プログラム輪郭作成システムの実現と使用について, 九大工学集報, Vol. 50, No. 3, pp. 209-214 (1977).
11) Ryder, B.G.: The PFORT Verifier, Software-Practice and Experience, Vol. 4, No. 4, pp. 359-377 (1974).
12) Lowry, E. S. and Medlock, C.W.: Object Code Optimization, CACM, Vol. 12, No. 1, pp. 13-22 (1969).
13) 藤村: Stingy Printer プログラムの使用について, 東大大型計算機センターニュース, Vol. 10, No. 2, pp. 28-29 (1978).
14) 藤村: 圧縮印刷プログラムの使用について, 九大大型計算機センター広報, Vol. 11, No. 3, pp. 215-217 (1978).

(昭和55年3月14日受付)
(昭和55年9月18日採録)