AUTOFLOW ENHANCEMENTS FOR DOCUMENTATION AND MAINTENANCE OF SCIENTIFIC APPLICATIONS

Martin A. Goetz
Applied Data Research, Inc.

Most documentation of computer programs can be summed up in the phrase, "Even when it's good, it's bad." Management may occasionally give documentation token priority, but programmers seem to give it no priority at all, perhaps because of their training. Programmer training is either formal or informal. In formal training courses, documentation is usually not a standard part of the curriculum; in informal or on-the-job training, it is usually not even mentioned. This lack of training is a basic reason for the problem of documentation, a problem that is compounded whenever management deemphasizes program documentation simply because past experience has shown that what had been produced was generally ineffective.

The chief reason that documentation is so poor may be that it has been considered a manual process when it should have been considered a computer problem. Certainly, no one considers compiling a manual process today, although, years ago, compiler functions were performed manually.

The need for documentation seems to be obvious. The primary concerns of both managers and programmers are program productivity, debugging, flexibility, integration, and reliability. Good documentation helps to fulfill these purposes; poor documentation, on the other hand, does not. Any organization can obtain good documentation, either manual or automatic, if it concentrates on program organization rules; programming standards, including the naming of tagged lines, proper commentary, modular programming, and restrictions in the use of certain programming techniques; program monitoring and security, including systematic recording of changes in programs, systematic recording of reasons for changes, and protection of programs; technical overviews of programs (using tape recordings, if preferred); and parallel development of programs and documentation.

Program organization rules are important because, although good programmers have an organized approach to writing programs, they, unfortunately, usually develop styles of their own. Rarely will two programmers use the same organization. Because a programmer does not work on a program forever, it is obvious that organization should not be permitted to suffer from the idiosyncrasies of the individual programmer. The same can be said for programming standards, which, by definition, can be effective only if they are both universally published and observed.

If programmers followed consistent program organization rules and programming standards, much of today's documentation problem would not have arisen. The computer industry is almost 20 years old; it should stop philosophizing about what ought to be and resolve this unsatisfactory situation.

Only automated documentation of programs offers any hope for realizing what may be called "accurate" program documentation. This paper will discuss how to improve automated documentation and, specifically, how the AUTOFLOW system can be enhanced to provide acceptable levels of documentation.

Given that programmers may cooperate only to a limited extent in documenting their programs and that computer programs can be developed to generate information that could not be produced manually, the following three elements are essential for an integrated documentation system within the framework of today's data processing environment:

(1) Logical analysis or graphic dissection of a program
(2) History and control of programs
(3) An understanding of the program

A flowchart produced by AUTOFLOW is much more meaningful than one that has been produced manually. These logical flowcharts are accurate, present complete references between all transfer points, and graphically portray the logical flow by automatic rearrangement of those segments of the program that interact. Figure 1 is an example of a two-dimensional AUTOFLOW flowchart from a FORTRAN program.

The number and type of cross-referenced reports produced by AUTOFLOW depend on the source language being used. For COBOL, AUTOFLOW can produce four special reports: procedure division summary, data name cross-reference listing, data division index, and data record map. For PL/I, four special reports are produced: on-unit action blocks, label-assignment cross-reference, duplicate declaration map, and condition prefix map. For FORTRAN, the one special report is the nonprocedural statements listing. Other special reports for FORTRAN could be produced by AUTOFLOW and would be of great value. Figures 2 through 10 are hypothetical reports that could be produced from a FORTRAN program by systems such as AUTOFLOW.

Figure 2 illustrates the header information that is common to all reports. The information includes the general title, FORTRAN analysis report; the user name, e.g., Goddard Space Flight Center; and the system. The run time for the analysis and the data are also presented. The report itself is essentially a listing of the local variables used by the program. The information presented is the mnemonic label, the type of variable, the definition of the variable, the line number where it is defined, the type and value of the definition, and then the references made by other statements in the FORTRAN source program to the local variable.

References in all reports consist of the source line number and, in parentheses, the AUTOFLOW page and box number. The variable labels in the first column are sorted alphanumerically. The label types are standard for IBM FORTRAN (integer 2, integer 4, real 4, real 8, logical, etc.). The DECLARATIONS column specifies where and how the variable is defined (i.e., through a data statement or an equivalence statement). If the variable is defined by a data statement, the value of the definition will be shown. Doubly-defined variables would be indicated by the notation DD in the definition area.

Figure 3, a cross-reference of statement numbers, lists only those statements that can be referenced by other statements within a program, i.e., statements with statement numbers. The appropriate line number, flowchart location, and type of statement (e.g., format,

```
AUTOFLOW CHART SET              DEMON                         10/29/70
 CARD NO        ****                    CONTENTS                   ****

    1           C          A SET OF ROUTINES ILLUSTRATING THE USE AND MISUSE OF VARIOUS
    2           C          FORTRAN STATEMENTS.  IT IS NOT INTENDED TO BE AN EXAMPLE OF
    3           C          GOOD, SENSIBLE OR EVEN REASONABLE PROGRAMMING.
    4           C
    5           C
    6           C
    7                      COMMON BCOM1(1000),BCOM2,BCOM3(1000),BCOM4(1000)
    8                      COMMON/LABCM1/LCOM1A
    9                      NAMELIST/NMLIST/N1,N2
   10                      DIMENSION BCOM2(1000)
   11                      DATA N1/1001/, N2/3/
   12                      INTEGER BCOM1, ROUTE1, ROUTE2
   13                      INTEGER*2 ROUTE3, ROUTE4
   14                      LOGICAL LGL1
   15                      REAL*8 BCOM2
   16           C
   17                      F1(A,B,N)=(A/2+B/2)**N
   18                      F2(X,Y)=(X-.01)/2+(Y-.01)/2
   19           C
   20                      READ(NMLIST)
   21                      DO 300L=1,N1
   22                      LCOM1A=L
   23                      CALL READER (&500)
   24                      L=LCOM1A
   25                      IF(BCOM1(L)) 100,120,140
   26          100         ASSIGN 320 TO ROUTE1
   27                      GO TO 160
   28          120         ASSIGN 340 TO ROUTE1
   29                      GO TO 160.
   30          140         ASSIGN 360 TO ROUTE1
   31          160         GO TO ROUTE1,(320,340,360)
   32          180         R=.01
   33          200         S=R
   34                      A=F1(R,S,N2)
   35          220         IF(A.GT.BCOM1(L)) GO TO 260
   36                      R=R+.01
   37                      GO TO 200
   38          260         S=R
   39                      H = 33 - Z
   40                      H=F2(R,S)
   41                      CALL WRITER (B,L)
   42                      IF(L.EQ.1) GO TO 300
   43                      DO 280 LL=2,L
   44                      BCOM3(LL)=BCOM3(LL)+BCOM3(LL-1)
   45                      DO 280 LLL=2,LL
   46          280         BCOM4(LLL)=BCOM4(LLL)+F1(BCOM4(LLL-1),BCOM4(LLL-1),1)
   47          300         CONTINUE
   48          C
   49          320         ROUTE2=1
   50                      LGL1=.FALSE.
   51                      GO TO 380
   52          340         ROUTE2=2
   53                      LGL1=.FALSE.
   54                      GO TO 380
   55          360         ROUTE2=3
   56                      LGL1=.TRUE.
   57          380         GO TO (400,420,440),ROUTE2
   58          400         WRITE(6,9000)BCOM1(L),L
   59                      GO TO 440
   60          420         WRITE(6,9001)L
   61          440         IF(LGL1) GO TO 180
   62          460         BCOM2(L)=0
   63                      GO TO 300
   64          C
   65          500         WRITE(6,9002)
   66                      STOP
   67          9000        FORMAT(1X,'<0',2I10)
   68          9001        FORMAT(1X,'=0',I10)
   69          9002        FORMAT(1X,'EOF')
   70                      END
```

Figure 1.–AUTOFLOW flowchart for FORTRAN program.

```
10/29/70         INPUT LISTING                               AUTOFLOW CHART SET - DEMON

FORTRAN MODULE        (NAMSO,LIST)

  CARD NO          ****                                  CONTENTS                                ****

        1                        SUBROUTINE READER (*)
        2          C
        3                        COMMON BCOM1(1000),BCOM2(1000),BCOM3(1000),BCOM4
        4                        DIMENSION BCOM4(1000),EQ1(1000),EQ2(1000),XQ1(10)
        5                        COMMON/LABCM1/LCOM1A
        6                        COMMON/LABCM2/LCOM2A
        7                        INTEGER EQ1,XQ2
        8                        EQUIVALENCE (EQ1,BCOM1)
        9                        REAL*8BCOM2,XQ2
       10          C
       11                        READ(5,9000,END=500)EQ1(LCOM1A)
       12                        X=EQ1(L1COMA)
       13                        CALL WRITER (X,LCOM1A)
       14                        LCOM1A=LCOM2A
       15                        RETURN
       16                        F=14
       17          500           RETURN1
       18          9000          FORMAT(58X,I5)
       19                        END
```

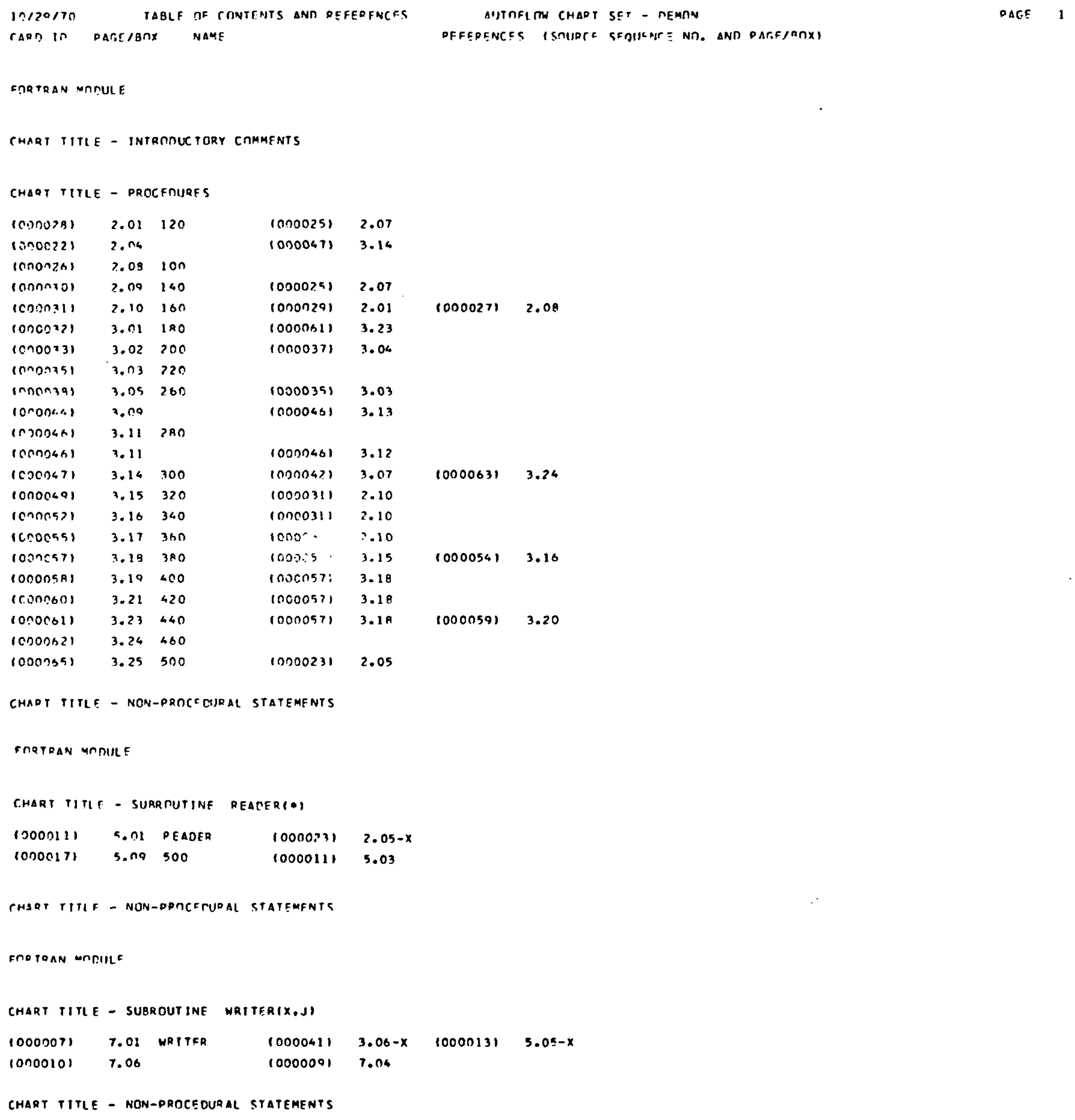
```
10/29/70         INPUT LISTING                               AUTOFLOW CHART SET - DEMON

FORTRAN MODULE        (NAMSO,LIST)

  CARD NO          ****                                  CONTENTS                                ****

        1                        SUBROUTINE WRITER (X,J)
        2          C
        3                        COMMON BCOM1(1000),BCOM2(1000),BCOM3(1000),BCOM4(1000)
        4                        COMMON/LABCM2/LCOM2A,LCOM2B
        5                        REAL*8 X
        6          C
        7                        WRITE(6,9000)X,J
        8                        G = 23 + YY
        9                        IF(LCOM2A.GT.2)LCOM2A=LCOM2A-SQRT(3.)
       10                        RETURN
       11                        F=21
       12          9000          FORMAT('0',F20.4,I10)
       13                        END
```

```
10/29/70         PROCEDURAL STATEMENT LABEL INDEX        AUTOFLOW CHART SET - DEMON                          PAGE   1

            PG.BX   NAME           PG.BX   NAME           PG.BX   NAME           PG.BX   NAME           PG.BX   NAME

             2.08  100              3.01  180              3.11  280              3.17  360              3.23  440
             2.01  120              3.02  200              3.14  300              3.18  380              3.24  460
             2.09  140              3.03  220              3.15  320              3.19  400              3.25  500
             2.10  160              3.05  260              3.16  340              3.21  420
```

```
10/29/70         PROCEDURAL STATEMENT LABEL INDEX        AUTOFLOW CHART SET - DEMON                          PAGE   2

            PG.BX   NAME           PG.BX   NAME           PG.BX   NAME           PG.BX   NAME           PG.BX   NAME

             5.01  READER           5.09  500
```

```
10/29/70         PROCEDURAL STATEMENT LABEL INDEX        AUTOFLOW CHART SET - DEMON                          PAGE   3

            PG.BX   NAME           PG.BX   NAME           PG.BX   NAME           PG.BX   NAME           PG.BX   NAME

             7.01  WRITER
```

Figure 1 (continued).–AUTOFLOW flowchart for FORTRAN program.



```
10/29/70         TABLE OF CONTENTS AND REFERENCES          AUTOFLOW CHART SET - DEMON                                 PAGE    1
CARD ID    PAGE/BOX     NAME                               REFERENCES  (SOURCE SEQUENCE NO. AND PAGE/BOX)


FORTRAN MODULE


CHART TITLE - INTRODUCTORY COMMENTS


CHART TITLE - PROCEDURES

(000028)     2.01  120           (000025)   2.07
(000022)     2.04                (000047)   3.14
(000026)     2.08  100
(000030)     2.09  140           (000025)   2.07
(000031)     2.10  160           (000029)   2.01      (000027)   2.08
(000032)     3.01  180           (000061)   3.23
(000033)     3.02  200           (000037)   3.04
(000035)     3.03  220
(000039)     3.05  260           (000035)   3.03
(000044)     3.09                (000046)   3.13
(000046)     3.11  280
(000046)     3.11                (000046)   3.12
(000047)     3.14  300           (000042)   3.07      (000063)   3.24
(000049)     3.15  320           (000031)   2.10
(000052)     3.16  340           (000031)   2.10
(000055)     3.17  360           (000[illegible]   [illegible].10
(000057)     3.18  380           (000[illegible]   3.15      (000054)   3.16
(000058)     3.19  400           (000057)   3.18
(000060)     3.21  420           (000057)   3.18
(000061)     3.23  440           (000057)   3.18      (000059)   3.20
(000062)     3.24  460
(000065)     3.25  500           (000023)   2.05

CHART TITLE - NON-PROCEDURAL STATEMENTS


FORTRAN MODULE


CHART TITLE - SUBROUTINE  READER(*)

(000011)     5.01  READER        (000023)   2.05-X
(000017)     5.09  500           (000011)   5.03

CHART TITLE - NON-PROCEDURAL STATEMENTS


FORTRAN MODULE


CHART TITLE - SUBROUTINE  WRITER(X,J)

(000007)     7.01  WRITER        (000041)   3.06-X   (000013)   5.05-X
(000010)     7.06                (000009)   7.04

CHART TITLE - NON-PROCEDURAL STATEMENTS
```

Figure 1 (continued).–AUTOFLOW flowchart for FORTRAN program.

```
10/29/70                         AUTOFLOW CHART SET - DEMON                         PAGE 01

CHART TITLE - INTRODUCTORY COMMENTS


                  A SET OF ROUTINES ILLUSTRATING THE USE AND MISUSE OF VARIOUS
                  FORTRAN STATEMENTS.  IT IS NOT INTENDED TO BE AN EXAMPLE OF
                  GOOD, SENSIBLE OR EVEN REASONABLE PROGRAMMING.
```

```
10/29/70                         AUTOFLOW CHART SET - DEMON                         PAGE 02

CHART TITLE - PROCEDURES
```

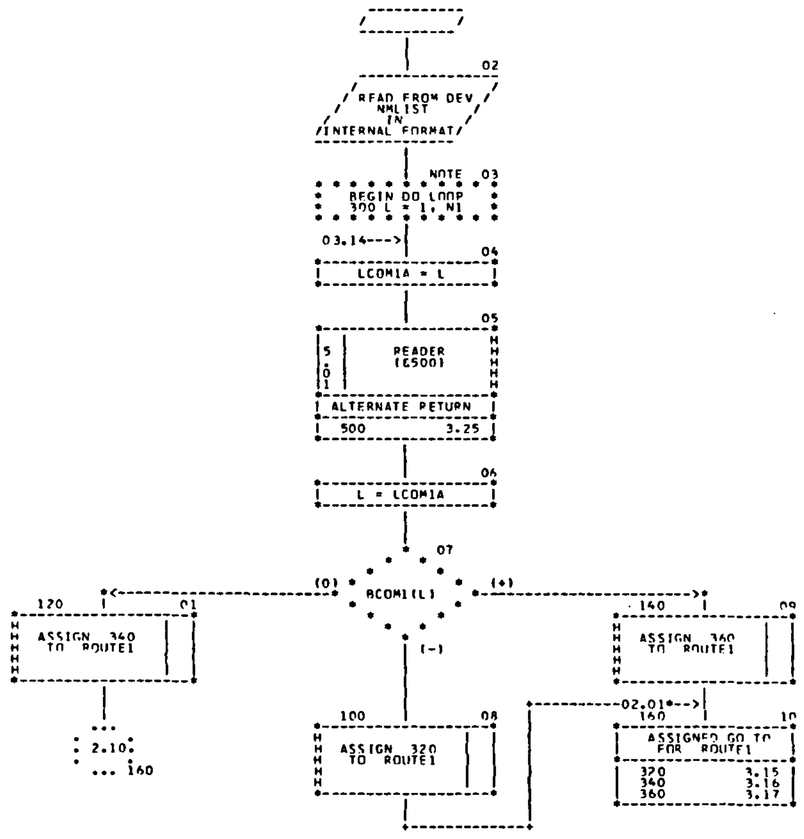


Figure 1 (continued).–AUTOFLOW flowchart for FORTRAN program.

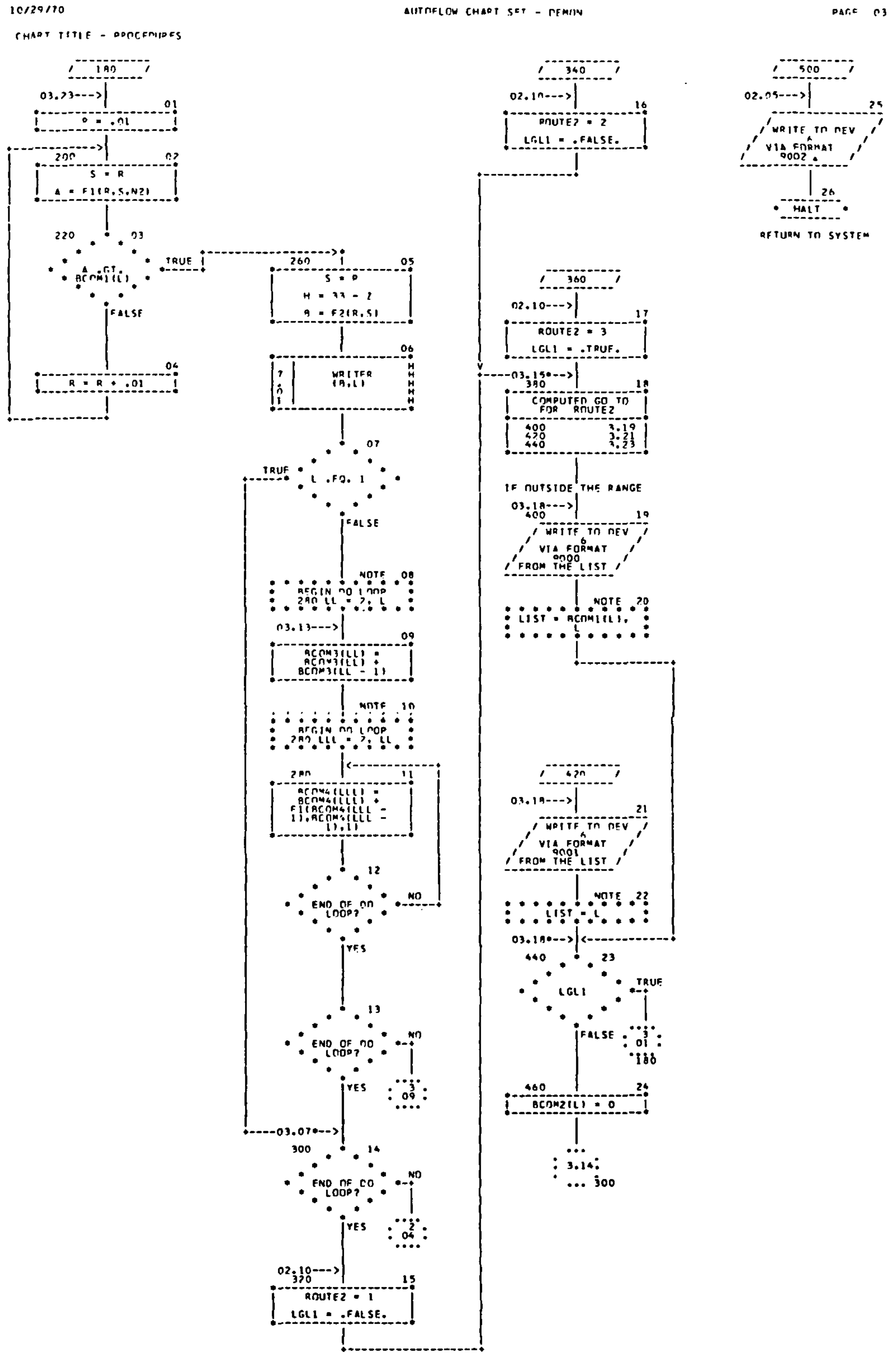


Figure 1 (continued).–AUTOFLOW flowchart for FORTRAN program.

```
10/29/70                              AUTOFLOW CHART SET - DEMON                              PAGE . 04

 CHART TITLE - NON-PROCEDURAL STATEMENTS


                     COMMON BCOM1(1000),BCOM2,BCOM3(1000),BCOM4(1000)
                     COMMON/LABCM1/LCOM1A
                     NAMELIST/NMLIST/N1,N2
                     DIMENSION BCOM2(1000)
                     DATA N1/1001/, N2/3/
                     INTEGER BCOM1, ROUTE1, ROUTE2
                     INTEGER*2 ROUTE3, ROUTE4
                     LOGICAL LGL1
                     REAL*8 BCOM2
                   STATEMENT FUNCTION DEFINITION:     F1(A,B,N)=(A/2+B/2)**N
                   STATEMENT FUNCTION DEFINITION:     F2(X,Y)=(X-.01)/2+(Y-.01)/2
              9000   FORMAT(1X,'<0',2I10)
              9001   FORMAT(1X,'=0',I10)
              9002   FORMAT(1X,'EOF')


  10/29/70                             AUTOFLOW CHART SET - DEMON                             PAGE . 05

  CHART TITLE - SUBROUTINE  READER(*)
```

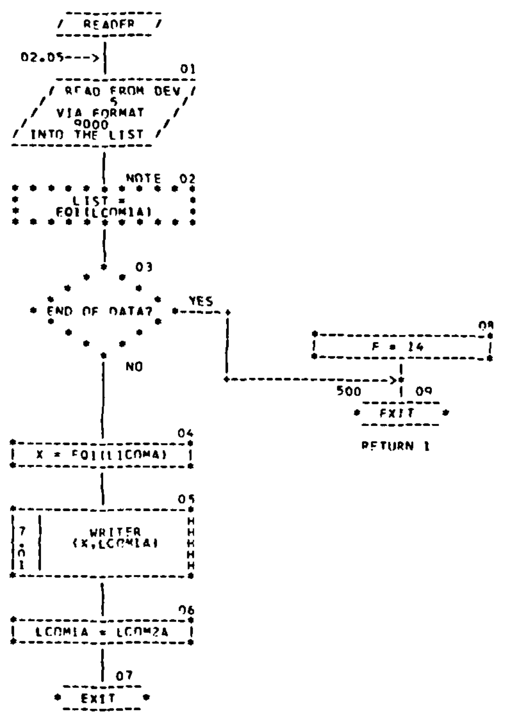
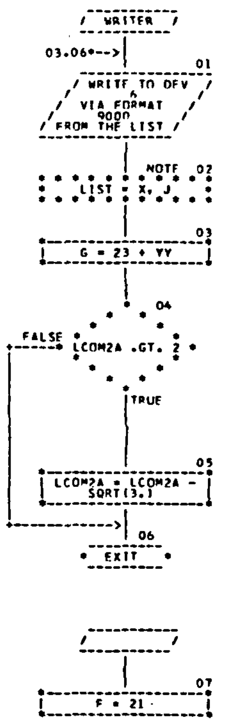


Figure 1 (continued).–AUTOFLOW flowchart for FORTRAN program.

```
10/29/70                              AUTOFLOW CHART SET - DEMON                         PAGE  06

 CHART TITLE - NON-PROCEDURAL STATEMENTS


                        COMMON BCOM1(1000),BCOM2(1000),BCOM3(1000),BCOM4
                        DIMENSION BCOM4(1000),EQ1(1000),EQ2(1000),XQ1(10)
                        COMMON/LABCM1/LCOM1A
                        COMMON/LABCM2/LCOM2A
                        INTEGER EQ1,XQ2
                        EQUIVALENCE (EQ1,BCOM1)
                        REAL*8BCOM2,XQ2
               9000     FORMAT(58X,I5)
```

```
10/29/70                              AUTOFLOW CHART SET - DEMON                         PAGE  07

 CHART TITLE - SUBROUTINE  WRITER(X,J)
```



```
10/29/70                              AUTOFLOW CHART SET - DEMON                         PAGE  08

 CHART TITLE - NON-PROCEDURAL STATEMENTS


                        COMMON BCOM1(1000),BCOM2(1000),BCOM3(1000),BCOM4(1000)
                        COMMON/LABCM2/LCOM2A,LCOM2B
                        REAL*8 X
               9000     FORMAT('0',F20.4,I10)
```

Figure 1 (concluded).–AUTOFLOW flowchart for FORTRAN program.

```
REPORT NO. 1                                   FORTRAN ANALYSIS REPORT                                         PAGE   1
                                         NASA, GODDARD SPACE FLIGHT CENTER
        TIME  16.20.31                                 SYSTEM NAME                                     DATE  OCT 15 1970

                                               PROGRAM: MAIN
                                          LOCAL VARIABLE REPORT BY PROGRAM
************************************************************************************************************************
  LABEL   TYPE                                         APPEARANCES: LINE#(PG.BX)
                *    DECLARATIONS    *             ASSIGNMENTS             *              REFERENCES
************************************************************************************************************************
A       REAL*4  *                  * 34(03.02)                          * 35(03.03)
B       REAL*4  *                  * 40(03.05)                          * 41(03.06)
H       REAL*4  *                  * 39(03.05)                          * NONE
L       INT*4   *                  * 21(02.03) 24(02.06)                * 22(02.04) 25(02.07) 35(03.03) 41(03.06) 42(03.07)
                *                  *                                    * 43(03.08) 58(03.19) 60(03.21) 62(03.24)
LGL1    LOGIC*4 * 14               * 50(03.15) 53(03.16) 56(03.17)      * 61(03.23)
LL      INT*4   *                  * 43(03.08)                          * 44(03.09) 45(03.10)
LLL     INT*4   *                  * 45(03.10)                          * 46(03.11)
NMLIST  NLIST   * 09               * 20(02.02)                          * NONE
N1      INT*4   * 11  DATA    1001 * 11                                 * 21(02.03)
N2      INT*4   * 11  DATA       3 * 11 36                              * 34(03.02)
R       REAL*4  *                  * 32(03.01) 36(03.04)                * 33(03.02) 34(03.02) 36(03.04) 38(03.05) 40(03.05)
ROUTE1  INT*4   * 12               * 26(02.08) 28(02.01) 30(02.09)      * 31(02.10)
ROUTE2  INT*4   * 11               * 49(03.15) 52(03.16) 55(03.17)      * 57(03.18)
ROUTE3  INT*2   * 13               *                                    * NONE
ROUTE4  INT*2   * 13               *                                    * NONE
S       REAL*4  *                  * 33(03.02) 38(03.05)                * 34(03.02) 40(03.05)
Z       REAL*4  * UNDEFINED        * UNDEFINED                          * 39(03.05)
************************************************************************************************************************
```

Figure 2.–Header information.

```
REPORT NO. 2                                    FORTRAN ANALYSIS REPORT                                          PAGE   1
                                           NASA, GODDARD SPACE FLIGHT CENTER
         TIME 16.20.31                               SYSTEM NAME                                      DATE  OCT 15 1970

                                                    PROGRAM: MAIN
                                           CROSS REFERENCE OF STATEMENT NUMBERS
****************************************************************************************************************************
LINE STMT  TYPE                   REFERENCES: LINE#(PG.BX)
****************************************************************************************************************************
 26   100  ASSIGN                25(02.07)
 28   120  ASSIGN                25(02.07)
 30   140  ASSIGN                25(02.07)
 31   160  ASSIGNED GO TO        27 29
 32   190  COMPUTATION           61(03.23)
 33   200  COMPUTATION           37
 35   220  LOGICAL IF            NONE
 38   260  COMPUTATION           35(03.03)
 46   280  COMPUTATION           43(03.08) 45(03.10)
 47   300  CONTINUE              21(02.03) 42(03.07) 63
 49   320  COMPUTATION           26(02.08) 31(02.10)
 52   340  COMPUTATION           28(02.01) 31(02.10)
 55   360  COMPUTATION           30(02.09) 31(02.10)
 57   380  COMPUTED GO TO        51 54
 58   400  WRITE                 57(03.18)
 60   420  WRITE                 57(03.18)
 61   440  LOGICAL IF            57(03.18) 59
 62   460  COMPUTATION           NONE
 65   500  WRITE                 23(02.05)
 67  9000  FORMAT                58(03.19)
 68  9001  FORMAT                60(03.21)
 69  9002  FORMAT                65(03.25)
****************************************************************************************************************************
```

Figure 3.–Cross-reference of statement numbers.

computational, or assignment), are specified. Again, all references to each statement number are listed by line number and AUTOFLOW page and box references.

Figure 4 is a cross-referenced listing of global variables used by the specific program that is being analyzed. This report is very similar to the local variable report, except that it lists only those variables that reside in blank or labeled common data areas. The information presented in the report includes the label mnemonic, the type of label, its definition, data used in the label, and all references to the label by other statements in the FORTRAN source program. The label type is broken down not only by data type (integer, real, logical, etc.) but also by the type of common area (whether it is blank common or label common and, if label common, by the mnemonic name of the label common area).

Figure 5 is a summary of all of the variables used in all of the programs input to a single AUTOFLOW run and is similar to the local variable report for a specific FORTRAN program. It contains essentially the same kind of information presented in the local variable report, mnemonic label for a variable, the type of variable, the definition of the data for the variable, and all references to that variable. The unique aspect of this report is that it does not reference only those local program variables that are accessible within a specific program but rather those variables that can be passed between programs through a common data area. In the references column, program identification, line number, and AUTOFLOW page and box number are indicated.

Figure 6 is the program subroutine usage report. This presents the names of subroutines within an individual program, the call parameters that are used by or passed to the subroutine, and any references (by line number and AUTOFLOW page and box number) to that subroutine in the specific FORTRAN program being analyzed. In the call parameter area, the variable name that is being passed to the subroutine and some additional information are found. If a global variable is being passed to a subroutine for its own use, an ampersand is appended to the mnemonic label in the CALL statement. A second type of variable that may be passed is a dummy variable, one that is not directly used by the program. This is a variable that has been passed to the present subroutine by a calling subroutine. A dummy variable is indicated by the pound sign appended to it. A third parameter is a return address, indicated by an asterisk. The call parameter portion of the listing also specifies the levels of all variables that are local to the program.

Figure 7, the system subroutine usage report, is very similar to the program subroutine usage report. The name of the program containing the call, the subroutine name, and the local, global, and dummy parameters passed to the called subroutines are specified. The report summarizes all subroutine usage within all program modules processed in a single AUTOFLOW run. Briefly, this listing establishes the hierarchy of subroutine calls among the modules for a given execution.

Figure 8 is the DO loop analysis report for a specific program. This listing indicates the complexity of the DO loop control within the program. The body of the report presents the source and flowchart locations of the start of the loop, the variables used for starting and ending values, and the increment used for the variable counter.

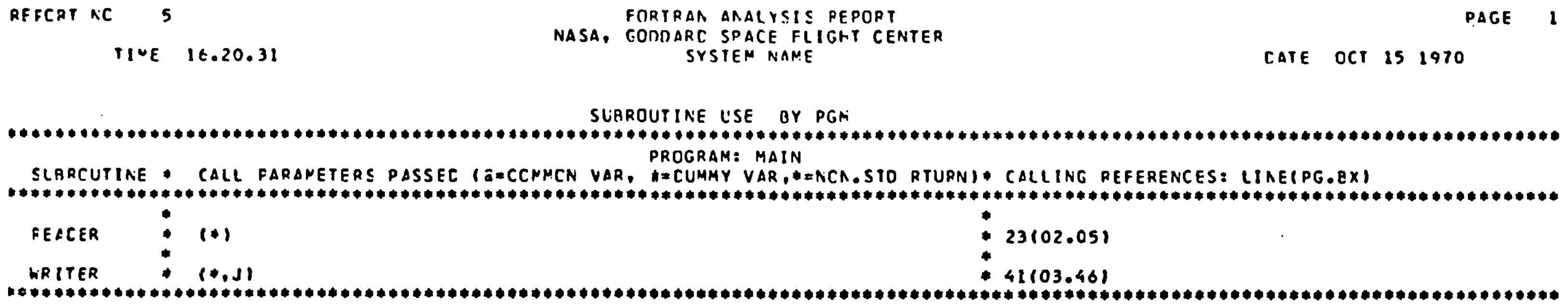
The complexity map, a bar diagram constructed of X's, depicts the logical structure of DO loops in a histogram format. This histogram graphically portrays the nesting effect.

```
REPORT NO.   3                                FORTRAN ANALYSIS REPORT                                    PAGE   1
                                          NASA, GODDARD SPACE FLIGHT CENTER
         TIME  16.20.31                              SYSTEM NAME                                DATE  OCT 15 1970

                                   CROSS REFERENCE OF GLOBAL VARIABLE USE BY PROGRAM
*****************************************************************************************************************
                                                     PROGRAM: MAIN
  LABEL     TYPE                                            APPEARANCES: LINE#(PG.BX)
                 *    DECLARATIONS   *               ASSIGNMENTS               *                REFERENCES
*****************************************************************************************************************
BLANK COMMON     *                   *                                         *
 BCOM1   INT*4   * 07 12             *                                         * 25(02.07) 35(03.03) 58(03.19)
 BCOM2   REAL*8  * 07 10 15          * 62(03.24)                               * NONE
 BCOM3   REAL*4  * 07                * 44(03.09)                               * 44(03.09)
 BCOM4   REAL*4  * 07                * 46(03.11)                               * 46(03.11)
                 *                   *                                         *
COMMON/LABCM1/   *                   *                                         *
 LCOM1A  INT*4   * 08                * 22(02.04)                               * 24(02.06)
*****************************************************************************************************************


*****************************************************************************************************************
                                                     PROGRAM: READER
  LABEL     TYPE                                            APPEARANCES: LINE#(PG.BX)
                 *    DECLARATIONS   *               ASSIGNMENTS               *                REFERENCES
*****************************************************************************************************************
BLANK COMMON     *                   *                                         *
 BCOM1           * 03 04 07 08       * 11(05.01)                               * 11(05.01) 12(05.04)
 BCOM2           * 03 09             *                                         * NONE
 BCOM3           * 03                *                                         * NONE
 BCOM4           * 03 04             *                                         * NONE
                 *                   *                                         *
COMMON/LABCM1/   *                   *                                         *
 LCOM1A          * 05                * 14(05.06)                               * 11(05.01) 12(05.04) 13(05.05)
                 *                   *                                         *
COMMON/LABCM1/   *                   *                                         *
 LCOM2A          * 06                *                                         * 14(05.06)
*****************************************************************************************************************


*****************************************************************************************************************
                                                     PROGRAM: WRITER
  LABEL     TYPE                                            APPEARANCES: LINE#(PG.BX)
                 *    DECLARATIONS   *               ASSIGNMENTS               *                REFERENCES
*****************************************************************************************************************
BLANK COMMON     *                   *                                         *
 BCOM1           * 03                *                                         * NONE
 BCOM2           * 03                *                                         * NONE
 BCOM3           * 03                *                                         * NONE
 BCOM4           * 03                *                                         * NONE
                 *                   *                                         *
COMMON/LABCM1/   *                   *                                         *
 LCOM2A          * 04                * 09(07.05)                               * 09(07.04) 09(07.05)
 LCOM12          * 04                *                                         *
*****************************************************************************************************************
```

Figure 4.–Cross-reference of global variables used.

```
REPORT NO.   4                                      FORTRAN ANAYLSIS REPORT                                       PAGE   1
                                            NASA, GODDARD SPACE FLIGHT CENTER
        TIME  16.20.31                                   SYSTEM NAME                                  DATE  OCT 15 1970

                                                SYSTEM USE OF GLOBAL VARIABLES
**********************************************************************************************************************************
 LABEL    TYPE                                           APPEARANCES: PGM-LINE#(PG.BX)
                  *    DECLARATIONS    *            ASSIGNMENTS                                         REFERENCES
**********************************************************************************************************************************
BLANK COMMON      *                    *                                         *
 BCCM1   INT*4    * MAIN- 07 12        * READER- 11(05.01)                       * MAIN- 25(02.07) 35(03.03) 58(03.19)
                  * READER- 03 04 07   *                                         * READER- 11(05.01) 12(05.04)
                  *         08         *                                         *
                  * WRITER- 03         *                                         *
 BCCM2   REAL*8   * MAIN- 07 10 15     * MAIN- 62(03.24)                         *
                  * READER- 03 09      *                                         *
                  * WRITER- 03         *                                         *
 BCCM3   REAL*4   * MAIN- 07           * MAIN- 44(03.09)                         * MAIN- 44(03.09)
                  * READER- 03         *                                         *
                  * WRITER- 03         *                                         *
 BCCM4   REAL*4   * MAIN-07            * MAIN- 46(03.11)                         * MAIN- 46(03.11)
                  * READER- 03 04      *                                         *
                  * WRITER- 03         *                                         *
                  *                    *                                         *
COMMON/LABCM1/    *                    *                                         *
 LCCM1A  INT*4    * MAIN- 08           * MAIN- 22(02.04)                         * MAIN- 22(02.04)
                  * READER- 05         * READER- 14(05.06)                       * READER- 11(05.01) 12(05.04) 13(05.05)
                  *                    *                                         *
COMMON/LABCM2/    *                    *                                         *
 LCCM2A  INT*4    * READER- 06         *                                         * READER- 14(05.06)
                  * WRITER- 04         * WRITER- 09(07.05)                       * WRITER- 09(07.04) 09(07.05)
 LCCM2B  INT*4    * WRITER- 04         *
**********************************************************************************************************************************
```

Figure 5.–System use of global variables.



```
REPORT NO    5                                  FORTRAN ANALYSIS REPORT                                             PAGE    1
                                             NASA, GODDARD SPACE FLIGHT CENTER
        TIME  16.20.31                                 SYSTEM NAME                                      DATE  OCT 15 1970

                                                  SUBROUTINE USE  BY PGM
*****************************************************************************************************************************
                                                     PROGRAM: MAIN
  SUBROUTINE *  CALL PARAMETERS PASSED (@=COMMON VAR, #=DUMMY VAR,*=NON.STD RTURN)* CALLING REFERENCES: LINE(PG.BX)
*****************************************************************************************************************************
             *                                                                      *
  READER     *  (*)                                                                 * 23(02.05)
             *                                                                      *
  WRITER     *  (*,J)                                                               * 41(03.46)
*****************************************************************************************************************************
```

Figure 6.–Program subroutine usage report.

```
REPORT NO.    6                                    FORTRAN ANALYSIS REPORT                                                   PAGE   1
                                              NASA, GODDARD SPACE FLIGHT CENTER
         TIME  16.20.31                                  SYSTEM NAME                                                DATE OCT 15 1970

                                                SYSTEM SUBROUTINE ANALYSIS

************************************************************************************************************************************
                                                   SYSTEM SUB CALL ANALYSIS
PGM NAME SUB CALLED PARAMETERS PASSED (@=GLOBAL VAR, #=DUMMY VAR,  *=NON-STD RETURN)
************************************************************************************************************************************
MAIN     READER      *
         WRITER       B, L

READER   WRITER       X, @LCOMIA
************************************************************************************************************************************


************************************************************************************************************************************
                                                SYSTEM SUBROUTINE USAGE
 NAME    *  CALL PARAMETERS  (@=COMMON VAR, #=DUMMY VAR, *=NON-STD RTRN)* CALLING REFERENCES: PGM-LINE#(PG.BX)
************************************************************************************************************************************
READER   *  (*)                                                      * MAIN- 23(02.05)
         *                                                           *
WRITER   *  (X,J)                                                    * READER- 13(05.05) MAIN- 41(03.06)
************************************************************************************************************************************
```

Figure 7.–System subroutine usage report.

```
REPORT NO  7                                      FORTRAN ANALYSIS REPORT                                                     PAGE   1
                                              NASA, GODDARD SPACE FLIGHT CENTER
        TIME  14.20.31                                   SYSTEM NAME                                          DATE  OCT 15 1970

                                                  DETAIL OF LOOP  ANALYSIS BY PROGRAM
*******************************************************************************************************************************
                                                          PROGRAM: MAIN
 START           END            LOOP CONTROL          COMPLEXITY MAP
LINE  STMT   LINE  STMT   VARIABLE INIT TEST INCR          LINE
*******************************************************************************************************************************

21            47    300   L          1    N1    1      X
31    160                                              X                            EXIT TO 320,340,360
43            46    280   LL         2    L     1      X X
45            46    280   LLL        2    LL    1      X X X
46    280                                              X X X
47    300                                              X



                                                      SUMMARY
                                               LEVEL      NO OF LOOPS

                                                 1            1

                                                 2            1

                                                 3            1

*******************************************************************************************************************************
```

Figure 8.–DO loop analysis report.

Additional information, such as an exit from within a loop to a statement external to the loop, is also shown by the histogram. A nest of three loops is represented by three vertical bars. The longest bar represents the initial DO loop, the next longest represents the second-level loop, and the shortest represents the third-level loop. The second part of this listing is the DO loop analysis summary, which specifies the loop level and the number of loops of different levels used in the program.

Figure 9 is the assigned GO TO analysis by program. This listing presents the sequence, page and box numbers, and statement number of all the assigned GO TO statements in a FORTRAN program. Additionally, variable names used in the branch list for each assigned GO TO are presented. The right side of the report lists all references to particular assigned GO TO statements. If one of the variables in the branch list is not defined within the program, this variable name will be listed with a dollar sign indicator. This is particularly helpful since undefined variables used in assigned GO TO statements will result in unpredictable destinations for the branch. The logic analysis section of this report presents program conditions that are probable program errors (e.g., undefined labels, unreferenced statements, undefined variables, or transfers into a DO loop).

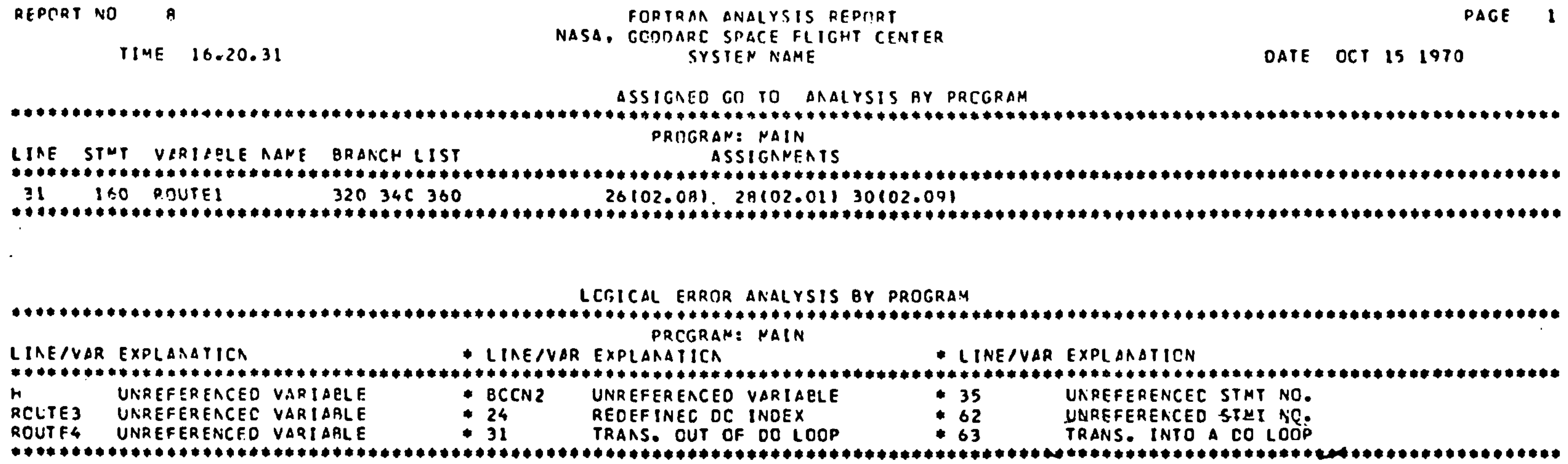
Figure 10 is the statement usage and complexity factor report, which presents a weighted summary of statement types within a program. On the left side of the report is the statement type (such as assigned GO TO, computed GO TO, dimension, value, and computational) and the number of each type within a program. The listing also contains the information needed for the complexity factor analysis. The assigned weight factors and the weighted values automatically assigned to the different types of statements. The user may override the default values and assign his own weighted factors at execution time. The product of the number of statements of a particular type and the weight factor for that type is the usage factor. At the bottom of this report is a summary which shows the total number of statements in the FORTRAN program, the total weight (the sum of all the usage factors), and the program complexity (the computed value of the total weight divided by the total number of statements). Program complexities range from 0.1 to 0.9. A factor of 0.5 would indicate that the program is of average complexity. The complexity factor is a useful guide for effective programmer assignment.

HISTORY AND CONTROL OF PROGRAMS

A program represents a considerable asset to an organization because it is usually costly to develop and is used to control functions within an organization ranging from the performance of simple accounting operations to the control of space flight programs.

Many programs have a life span far in excess of 5 years. A case in point is the IBM 650 program, which was simulated on the IBM 1401 after the IBM 650 was removed. The IBM 1401 is now being simulated on the IBM 360 and will shortly be simulated on the IBM 370. Rumor has it that the IBM 650 program was actually simulating an IBM 604 tabulating function.

Programs survive intact over long periods of time because they are infrequently run and, therefore, not economical to reprogram, or nobody really knows their contents (the fear factor). In general, today's software technology is in such a deplorable condition for



```
REPORT NO     8                          FORTRAN ANALYSIS REPORT                                  PAGE   1
                                     NASA, GODDARD SPACE FLIGHT CENTER
        TIME  16.20.31                          SYSTEM NAME                             DATE  OCT 15 1970

                                      ASSIGNED GO TO  ANALYSIS BY PROGRAM
*********************************************************************************************************
                                               PROGRAM: MAIN
LINE  STMT  VARIABLE NAME  BRANCH LIST                 ASSIGNMENTS
*********************************************************************************************************
 31    160  ROUTE1          320 340 360          26(02.08), 28(02.01) 30(02.09)
*********************************************************************************************************


                                    LOGICAL ERROR ANALYSIS BY PROGRAM
*********************************************************************************************************
                                               PROGRAM: MAIN
LINE/VAR EXPLANATION                  * LINE/VAR EXPLANATION                  * LINE/VAR EXPLANATION
*********************************************************************************************************
H        UNREFERENCED VARIABLE        * BCCN2    UNREFERENCED VARIABLE        * 35       UNREFERENCED STMT NO.
ROUTE3   UNREFERENCED VARIABLE        * 24       REDEFINED DO INDEX           * 62       UNREFERENCED ~~STMT~~ NO.
ROUTE4   UNREFERENCED VARIABLE        * 31       TRANS. OUT OF DO LOOP        * 63       TRANS. INTO A DO LOOP
*********************************************************************************************************
```

Figure 9.–Assigned GO TO analysis.

```
REPORT NO    9                        FORTRAN ANALYSIS REPORT                                  PAGE   1
        TIME  16.20.31            NASA, GODDARD SPACE FLIGHT CENTER                      DATE  OCT 15 1970
                                             SYSTEM NAME

                                          PROGRAM: MAIN
                                  STATEMENT USAGE AND COMPLEXITY FACTORS

                            TYPE              NUMBER     ASSIGNED      USAGE
                                              IN PGM     WT FACTOR     FACTOR

                       ASSIGN GO TO             1          1.0          1.0
                       COMPLEX                  0          0.2           .0
                       COMPUTED GO TO           1          1.0          1.0
                       CONTINUE                 1          0.2           .2
                       DATA                     1          0.5           .5
                       DIMENSION                1          0.3           .3
                       DO                       3          0.5          1.5
                       EQUIVELENCES             0          0.6           .0
                       FORMAT                   3          0.3           .9
                       FUNCTION DEF.            2          0.7          1.4
                       GO TO                    7          0.9          6.3
                       IF                       4          0.5          2.0
                       INTEGER                  2          0.2           .4
                       LOGICAL                  1          0.2           .2
                       READ                     1          0.1           .1
                       REAL                     1          0.2           .2
                       SUB CALL                 2          0.8          1.6
                       STMT FUNCT CALL          3          0.7          2.1
                       WRITE                    3          0.1           .3
                       ASSIGNMENTS             18          0.1          1.8
                       NAMELIST                 1          0.4           .4

                       .......REPORT SUMMARY.........

                        A. TOTAL STATEMENTS     56

                        B. TOTAL WEIGHT....     22.2

                        C. COMPLEXITY..(B/A)    .40
```

Figure 10.–Statement usage and complexity factor report.

the latter reason. Programs such as The LIBRARIAN, an adjunct to the AUTOFLOW system, are available to monitor program activity; produce histories of changes; retain copies of old versions of programs; protect programs against unauthorized use; and provide complete indexes that give dates of modifications, reasons for changes, and other information necessary for the orderly maintenance of programs and data.

UNDERSTANDING THE PROGRAM

The next questions to be asked concern the function, organization, and reason for organization of a program. All these questions can be answered by "picking the brains" of the programmer and the designer.

Given the aversion of most programmers to documentation, the tape recorder can be a very effective means of obtaining vital information. It is probably much easier for many programmers to sit down and record on a cassette all the details of program development than for them to take the time to write everything down. The taped information can be easily transcribed and converted to a machine-readable form for input to a system such as TEXT EDITOR. This system can be used to produce a finished document for permanent retention as the program history and enables a user to specify format, alter content, and expedite production of hard-copy documentation with a minimum of manual effort. In short, the programmer need only talk about his projects, and a final record of such discussions can be automatically produced.

The final issue that is critical for the overall effectiveness of documentation is whether it actually reflects the current status of program development. Outdated documentation can be only partially useful at best, and totally misleading at worst. The systems discussed, AUTOFLOW, The LIBRARIAN, and TEXT EDITOR, assure all users that the documentation will be not only accurate, standardized, and complete but also timely and readily available whenever needed.

CONCLUSION

In summary, the critical needs in the area of effective program documentation involve the integration of normal programming activities with the requirement for more comprehensive documentation. The ultimate solution to these needs lies in automated documentation systems that can reduce clerical effort on the part of the programmer, provide timely and accurate documentation whenever needed, analyze program design and structure, expedite maintenance and debugging operations, protect source programs from loss or damage, and provide an understanding of the program. Computer programs can do this and can do it better, faster, and more economically.

DISCUSSION

MEMBER OF THE AUDIENCE: I understand that AUTOFLOW is applicable to FORTRAN; is it also applicable to other programming languages?

GOETZ: AUTOFLOW can be applied to all of the major languages in use today, including second-generation programming languages and various types of FORTRAN.

MEMBER OF THE AUDIENCE: To your knowledge, does anyone else employ the tape recorder in the way that you have discussed, and what benefits does it offer to programming personnel?

GOETZ: Although I am certain that it must be used elsewhere, I cannot provide any specific organization names. The technique makes it easier for the programmer to record information. The information generated is actually of better quality than that which would be produced if the programmer were required to write his documentation, since the programmer becomes too self-conscious when he is writing.

MEMBER OF THE AUDIENCE: Do you have any intention of writing a manual describing the entire procedure that could be marketed?

GOETZ: We have no current plans for doing that.

MEMBER OF THE AUDIENCE: You have mentioned that AUTOFLOW is available for several different language systems. Does this diversity also extend to different computers?

GOETZ: AUTOFLOW is not available for many machines; it is available for the Spectra 70 series, the Honeywell series, and the IBM 7090 and 360 series.

MEMBER OF THE AUDIENCE: Is there an extended AUTOFLOW available for the CDC 6600?

GOETZ: No. The AUTOFLOW system is written in assembly language and cannot be transferred between machines. No AUTOFLOW was written for the CDC 6600. We do accept 6600 programs—assembly language and the various FORTRANS, I believe—but the AUTOFLOW system does not operate with them. Also, the extended versions of the FORTRAN analysis are hypothetical systems that have not yet been constructed. The flowcharts and reports used in my paper were manually produced.

MEMBER OF THE AUDIENCE: What use is made of the tape recorder in the development of the user documentation?

GOETZ: The program documentation, providing the internal logic of the program, can best be obtained with the use of the tape recorder, but the user documentation is something quite different. It should be well organized and produced in a more formal way than the program documentation.

MEMBER OF THE AUDIENCE: Do the American National Standards Institute (ANSI) flowchart standards constrain the actual communication of information because of restrictions placed on the size and proportion of symbols and the lack of symbols needed to terminate and then continue a line that is not related to the flow of the data or the logic of the program? Since symbols in modern languages can have as many as 30 characters, the standards, to a certain extent, inhibit communication because the programmer must limit what he says.

GOETZ: Our current standards do not quite conform to ANSI standards. The width of a process box, for instance, must be related to its length, according to ANSI standards, but AUTOFLOW will produce a process box of virtually any size, so it could be 50 or 100 lines long. We are upgrading our system so that it will conform completely to ANSI standards, which will restrict or inhibit somewhat the flowchart produced. The user will then have the option of having ANSI or AUTOFLOW standards.

MEMBER OF THE AUDIENCE: Do you consider the ANSI standards to be adequate or archaic?

GOETZ: We think that they are somewhat archaic, but they are standards, and we are willing to conform. Therefore, we are producing the option.

MEMBER OF THE AUDIENCE: Consider a program that was written without AUTOFLOW in mind. If the program were then analyzed by AUTOFLOW, which would be the most useful: analysis portion or the flowchart portion?

GOETZ: It would depend upon who would be using the report. For the original programmer, the analysis portion will suffice in many cases. For debugging and making program alterations, the flowchart is especially useful and would probably be a necessary aid if those functions were being performed by someone who was not the original programmer. The level of the programmer's training would also be a consideration.

MEMBER OF THE AUDIENCE: To what extent is AUTOFLOW used to document and maintain itself?

GOETZ: The entire system is written in Assembly language and contains chart codes in the comments portion of the program. By putting these chart codes in the program and considering what the assembly language coding represents, we obtain very good narrative statements and comments. The very low personnel turnover that we have reduces considerably the need for producing flowcharts for maintenance purposes.