CONF-9010256--6

WSRC-MS--90-150

DE91 007241

ARCHY: A TOOL FOR FORTRAN CODE MAINTENANCE AND DEVELOPMENT (U)

BY

J. E. AULL

NUCLEAR REACTOR TECHNOLOGY AND SCIENTIFIC COMPUTATIONS
WESTINGHOUSE SAVANNAH RIVER COMPANY
AIKEN, SC 29808

DISCLAIMER

WESTINGHOUSE COMPUTER SYMPOSIUM
FOX CHAPEL YACHT CLUB
FOX CHAPEL, PENNSYLVANIA
OCTOBER 1990

ARCHY: A TOOL FOR FORTRAN CODE MAINTENANCE AND DEVELOPMENT (U)

by

J. E. Aull

Nuclear Reactor Technology and Scientific Computations
Westinghouse Savannah River Company
Savannah River Site
Aiken, South Carolina 29808
(803) 725-2622

ABSTRACT

Analysis and Reverse Engineering of Code Using Hierarchy and Yourdon (ARCHY) diagrams is a tool for development and maintenance of FORTRAN programs. When FORTRAN source code is read by ARCHY, it automatically creates a database that includes a data dictionary, which lists each variable, its dimensions, type, category (set, referenced, passed), module calling structure, and common block information. The database exists in an ASCII file that can be directly edited or maintained with the ARCHY database editor. The database is used by ARCHY to produce structure charts and Yourdon data flow diagrams in PostScript format. ARCHY also transfers database information such as a variable definitions, module descriptions, and technical references to and from module headers.

ARCHY contains several utilities for making programs more readable. It can automatically indent the body of loops and conditionals and resequence statement labels. Various language extensions are translated into FORTRAN-77 to increase code portability. ARCHY frames comment statements and groups FORMAT statements at the end of modules. It can alphabetize modules within a program, end-of-line labels can be added, and it can also change executable statements to upper or lower case.

ARCHY runs under the VAX-VMS operating system and inputs from VAX-FORTRAN, IBM-FORTRAN, and CRAY FORTRAN source files.

INTRODUCTION

This paper describes a set of software tools called ARCHY that were developed in the Reactor Physics Group of the Savannah River Laboratory, which is operated for the U.S. Department of Energy by Westinghouse Savannah River Company. In the following sections, the need for reverse engineering tools within the computer industry is assessed. How ARCHY is used to analyze and maintain existing applications and how it is used to develop new applications is discussed. The last section provides a summary of the benefits of ARCHY.

DISCUSSION

Why Reverse Engineering Tools Are Needed

This section describes the problem that ARCHY was built to solve and examines the potential market within the computer industry for a product like ARCHY.

There is a substantial investment within industry in computer applications that are written in Third Generation Languages (3GL) such as FORTRAN. A study by Digital Equipment Corporation estimated that it would cost more than $2 trillion to replace all of their 3GL code.[1] Many of today's scientific applications are large programs written in FORTRAN before the advent of structured programming or design methodologies such as the Yourdon method. In many cases the documentation for these programs is often nonexistent or sketchy at best and the design used to develop the program is completely missing. Many of these programs are being maintained by experts who are approaching retirement.

The need to maintain FORTRAN programs is likely to continue for some time as most scientists and engineers are well grounded in the language. In Levesque[2], it is related that John Backus was once asked by an interviewer what would be the nature of the language running on supercomputers in the year 2000. He replied, "I can't tell you anything about its nature, but I know we will call it FORTRAN."

Maintenance of existing 3GL applications is expensive. Numerous surveys have shown that 70 to 80 percent of the effort within the life cycle of an application is devoted to maintenance.[3] A survey by Digital Equipment Corporation[1] found that there is an average of 170,000 lines of 3GL code per maintenance programmer with enhancements being performed at the rate of 1 to 2 lines per day. Clearly there is a need for tools that can enhance software maintenance productivity.

Computer-Assisted Software Engineering (CASE) tools are hard to cost justify. CASE tools are designed to ease maintenance by automatically generating code based on the specification or design of an application. Then when the program must be changed, the specification is changed and a modified program is regenerated using the CASE tool. The costs for CASE hardware, software, and training for a typical scenario are estimated at $30,000 per programmer.[3] In addition to initial cost there is the inevitable loss of productivity as programmers convert to use of the CASE tool. Programmer productivity is difficult to measure and use of a CASE tool often imposes adherence to a particular design methodology, which can be a management problem. Furthermore many CASE tools are designed only for use in developing new applications and provide no help in the maintenance phase of the software life cycle.

There is a great need for CASE tools to ease maintenance of existing FORTRAN programs by making programs more readable as well as providing tools for documenting programs and the underlying design within the programs. The remainder of this paper describes how ARCHY meets this need as well as describing how it may be used to develop new software.

WHAT ARCHY DOES

ARCHY Is Used to Reverse Engineer Code

Often the responsibility for maintaining a large FORTRAN program is transferred to an individual and the program is accompanied by minimal or nonexistent documentation. Upon examination of the source code, the unfortunate new maintainer may find it incomprehensible because of statement numbers in random order and lack of comment lines and indention. This section shows how ARCHY is used to assist a maintainer in such a predicament.

Figure. 1 is a dataflow diagram that shows an overview of the sequence in which various ARCHY tools are used. Each step in the sequence is then expanded in more detail. The diagram uses the Yourdon symbology wherein files or data stores are represented by parallel lines, transforms (i.e., programs) are represented by circles, and the flow of data between a transform and file is represented by an arrow.

File 1 (Figure 1) is a messy program that needs to be cleaned up in order to be easily comprehended by the maintainer. It is fed into the cleanup utilities that produce a program with the body of loops and IF-THEN-ELSE statements indented, statement numbers in sequence, modules alphabetized and separated, comment lines framed, and end-of-line labels appended. Next the unfamiliar source code is processed by the DECIFE program, which creates a database of entities within the program including modules with their calling structure, variables, and common blocks. This database (file 3 in Figure 1) is then accessed by various report writing programs such as STRUCT (not shown), which produces a structure chart that shows what submodules are called by each module. The maintainer then enters descriptions of modules, variables, and common blocks into the database using the DB program. Lastly, INCORP is used to put descriptive information from the database into the program in the form of comment statements at the head of each module.

The cleanup utilities are best illustrated by an example of what a program looks like before and after cleaning. Programs always behave in an identical manner after cleaning as they do before cleaning. The only benefit of cleaning is to make a program easier to read and thus easier to understand. Figure 2 shows a program before cleaning (a) and the same program after cleaning (b). The indention of the body of control structures, resequencing of statement numbers, alphabetizing of modules, and framing of comment lines all serve to enhance program readability. Other features of the cleanup utilities include the conversion of certain non-standard language extensions to FORTRAN-77. These include the VAX TAB format, Hollerith constants, and end-of-line comments.

Figure 3 shows the type of information that is stored in an ARCHY database when a program is analyzed by the DECIFE utility. The database is organized into seven tables, of which, three are shown in Figure 3. The other four tables contain information used to generate dataflow diagrams. The database is actually just an ASCII file, which means it is portable and can be easily browsed or modified using any text editor. The ARCHY system also includes a menu-driven database editor, which ensures that the database is in the proper format and performs referential integrity checks. The database is useful as a point of reference when examining a particular module or variable. Various reports based on the database contents are printed by ARCHY including a data dictionary of variables, module descriptions, and structure charts.

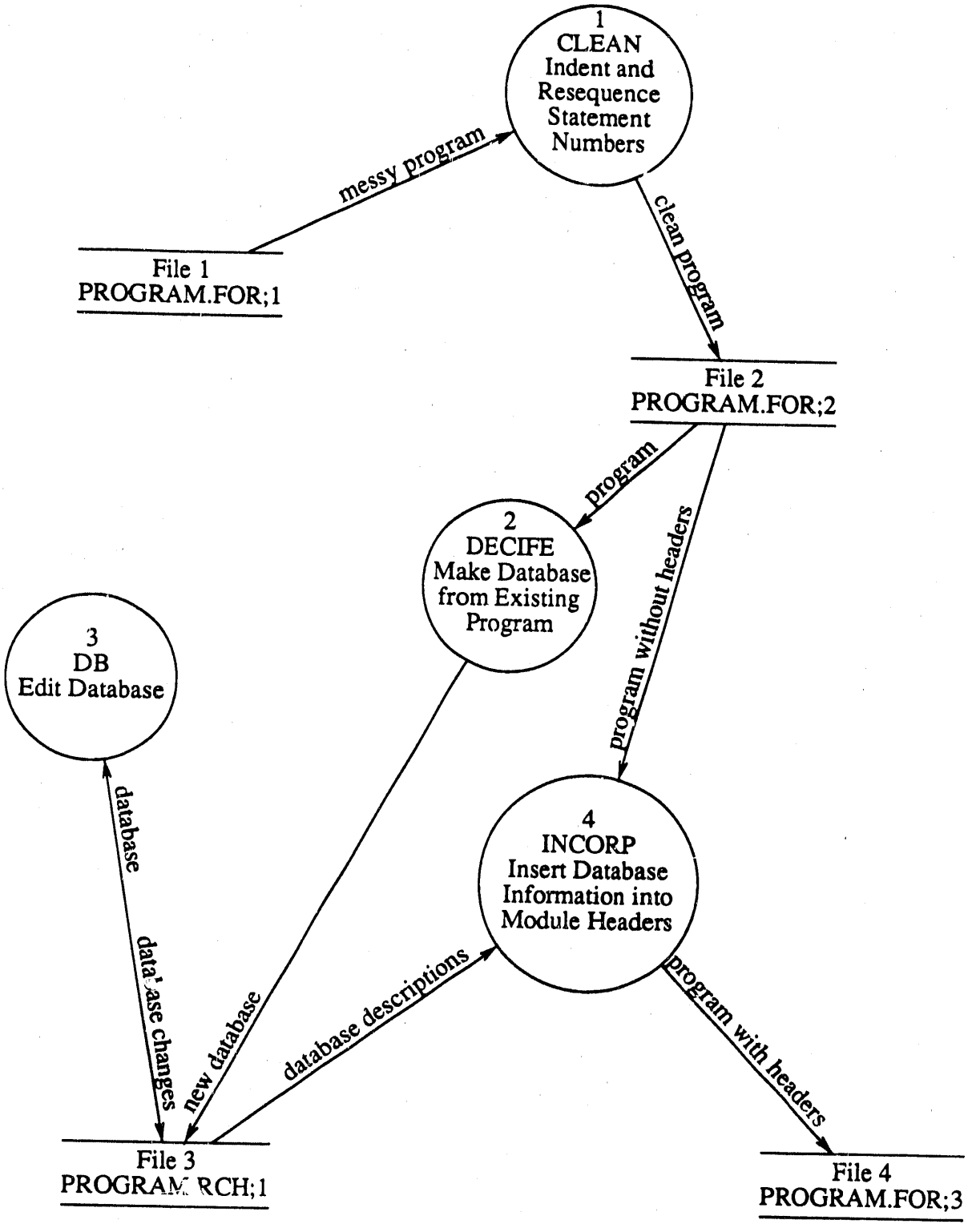

Figure 1. Reverse Engineering Using ARCHY

a. Before Cleaning

```
      PROGRAM MAIN
  200 DO  I = 1,3
      IF (I.EQ.2) THEN
      WRITE(6,250)
  250 FORMAT(' UNEQUAL')
      ELSE
      GOTO 40
      ENDDO
   40 WRITE(8,300)
  300 FORMAT(1X,7HTHE END)
      CALL A
      STOP
      END
      SUBROUTINE A
      WRITE(6,111)
  111 FORMAT(1X,'PROGEND')
      RETURN
      END
```

b. After cleaning

```
      program main                                                    MAIN0001
   10 do  i = 1,3                                                     MAIN0002
        if (i.eq.2) then                                             MAIN0003
          write(6,9010)                                              MAIN0004
        else                                                         MAIN0005
          goto 20                                                    MAIN0006
        endif                                                        MAIN0007
      enddo                                                          MAIN0008
   20 write(8,9020)                                                  MAIN0009
      call a                                                         MAIN0010
      stop                                                           MAIN0011
 9010 format(' UNEQUAL')                                             MAIN0012
 9020 format(1x,'THE END')                                           MAIN0013
      end                                                            MAIN0014
c*****************************************************************#
c*****************************************************************#
      subroutine a                                                   A   0001
      write(6,9010)                                                  A   0002
      return                                                         A   0003
 9010 format(1x,'PROGEND')                                           A   0004
      end                                                            A   0005
```

Figure 2. Program Before and After Cleaning

Modules
- Description (purpose, history, theory, algorithm)
- Called modules
- Variables (external or internal, referenced or set)
- Parameters

Data Dictionary of Variables
- Definition
- Dimensions
- Type
- Modules Using (in case of duplicates)

Common Blocks
- Description
- Module
- Variables

Figure 3. ARCHY Database Structure

The structure chart output by the STRUCT program of ARCHY in PostScript format is extremely useful to the maintenance programmer. An example of such a structure chart is shown in Figure 4. Each box in the chart represents a module, and boxes under a box are the ones called from that module. In this example, CLEAN, ALPHA, and LABEL are all modules that are called by the BEAUTIFY module. The descriptions in each box are stored in the database and must be input by the user.

As the maintainer gains understanding of the program, variable definitions, common block descriptions, maintenance history, and information about the theory and algorithm used in the code may be entered into the database. This descriptive information is then transferred to the program in the form of module headers by using the INCORP program. An example of a module header is shown in Figure 5. The header format can also be read by the DECIFE program, which stores the descriptive information in the ARCHY database. This feature is useful for two reasons:

- It allows the programmer to modify descriptions in the header as the program is modified and then feeds the modified header information to the database instead of modifying database descriptions separately from the program modification; and

- It provides the ability to read database information from module headers thus allowing the maintenance programmer to make use of an existing header at the beginning of the task of maintaining a program.

The maintainer simply edits the header of the program to make it conform to the ARCHY format, then runs DECIFE, and the descriptions are stored in the appropriate fields in the new ARCHY database.

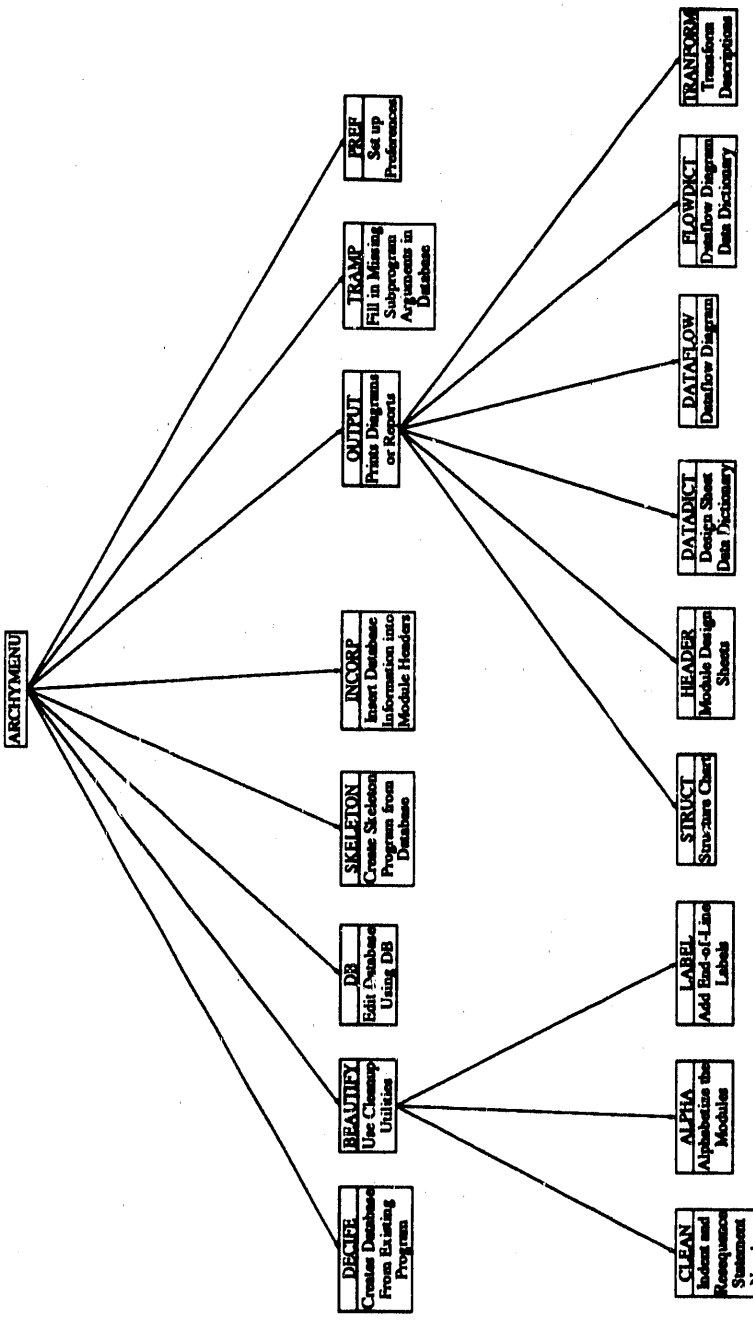

Figure 4. Structure Chart

```
c*************************************************************************#
c                                                                         #
c Module name:                                                            #
c   GETCALLS                                                              #
c                                                                         #
c Purpose:                                                                #
c  Read XRF file and store the modules called by each module.             #
c                                                                         #
c History:                                                                #
c  John Aull; WSRC; 6/01/90  Mod- SDCN#17;6/28/90;JEA                     #
c                                                                         #
c Structure chart description:                                            #
c  GET LIST OF WHAT MODULES ARE CALLED.                                   #
c                                                                         #
c Called modules:                                                         #
c  FINDLEN132 FINDCHAR                                                    #
c                                                                         #
c External variables - used:                                              #
c   NMOD                                                          I*4     #
c     NUMBER OF MODULES                                                   #
c                                                                         #
c                                                                         #
c External variables - set:                                               #
c                                                                         #
c   MODULE (100,50)                                             CHAR*32   #
c     MODULE NAME ALONG WITH ALL THE MODULES IT CALLS.                    #
c                                                                         #
c                                                                         #
c Internal variables:                                                     #
c   ACHAR                                                       CHAR*132  #
c     Line read from .XRF file.                                           #
c                                                                         #
c   IST                                                           I*4     #
c     Starting point for scan.                                            #
c                                                                         #
c   LENL                                                          I*4     #
c     LENGTH OF A STRING.                                                 #
c                                                                         #
c                                                                         #
c*************************************************************************#
```

Figure 5. Module Header

ARCHY Database Is Maintained as the System Is Modified

The ARCHY database is easily updated as the program goes through extensive modification. The procedure for updating the database during this program modification is shown in Figure 6. The first transform represents the modification of a program by a programmer. As the program is modified, the programmer may not bother to make the corresponding changes to the ARCHY database. After completing the modification, a new database is generated using DECIFE, and then the MIX program is executed, which is able to extract the descriptive material from the old database and merge it with the new database.

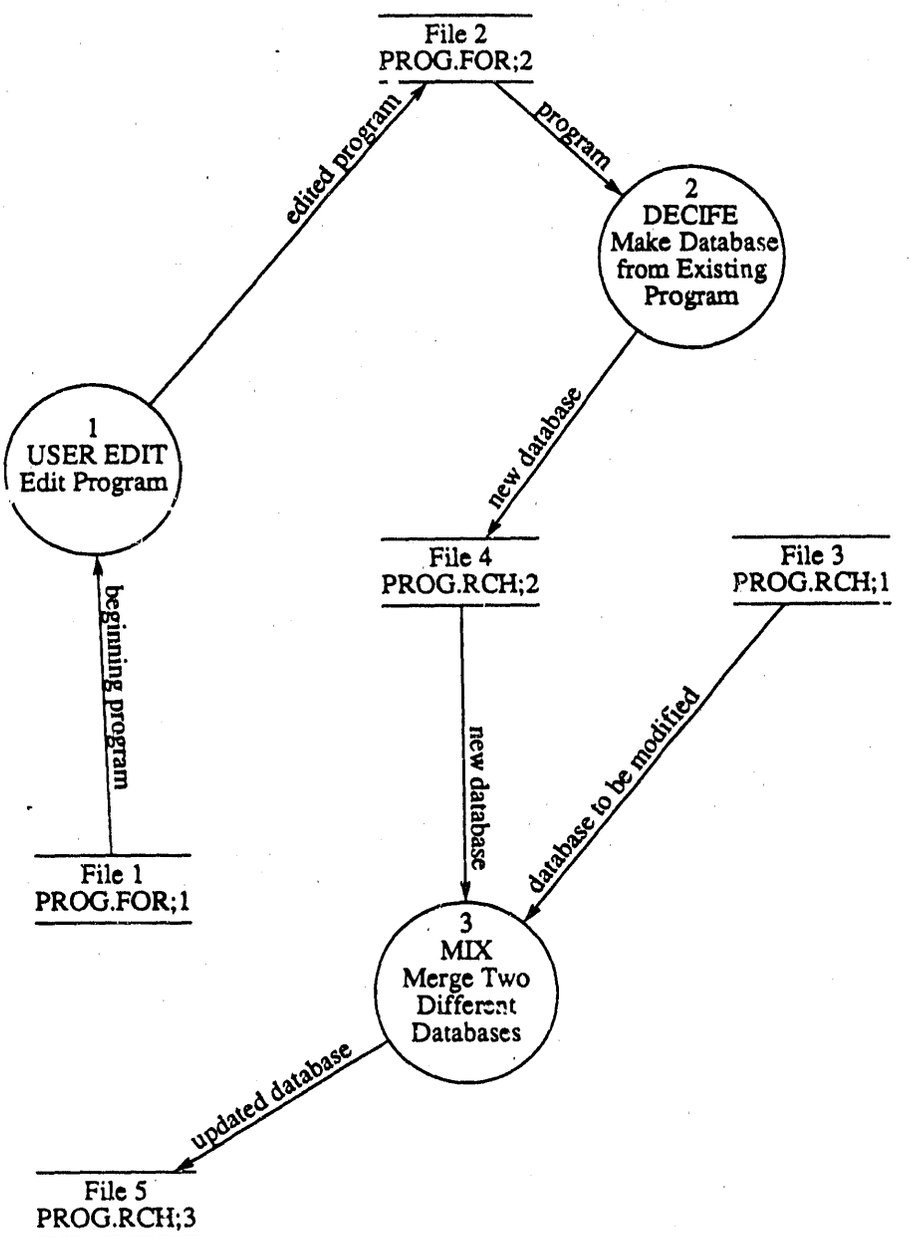

Figure 6. Major Program Modification Using ARCHY

ARCHY Is Used For Development of New Software

ARCHY is also useful in development of new applications as illustrated in Figure 7. This process begins with the entry of descriptive information about files, transforms, and dataflows, which are stored in separate tables in the database. The designer must also specify the location of these symbols, which are stored in a fourth table. The DATAFLOW utility then generates a dataflow diagram in a PostScript file. In fact, Figure 7 is a dataflow diagram that was generated by the DATAFLOW program. Next, the designer enters modules and variables into the database. The SKELETON utility generates a skeleton program consisting of subroutine calls to each defined module along with the the SUBROUTINE statement for the module and type statements for the variables within that module. Descriptive information like that in the module headers is printed out. This information is referred to as a module design sheet and serves as an instruction sheet for the programmer developing the module.

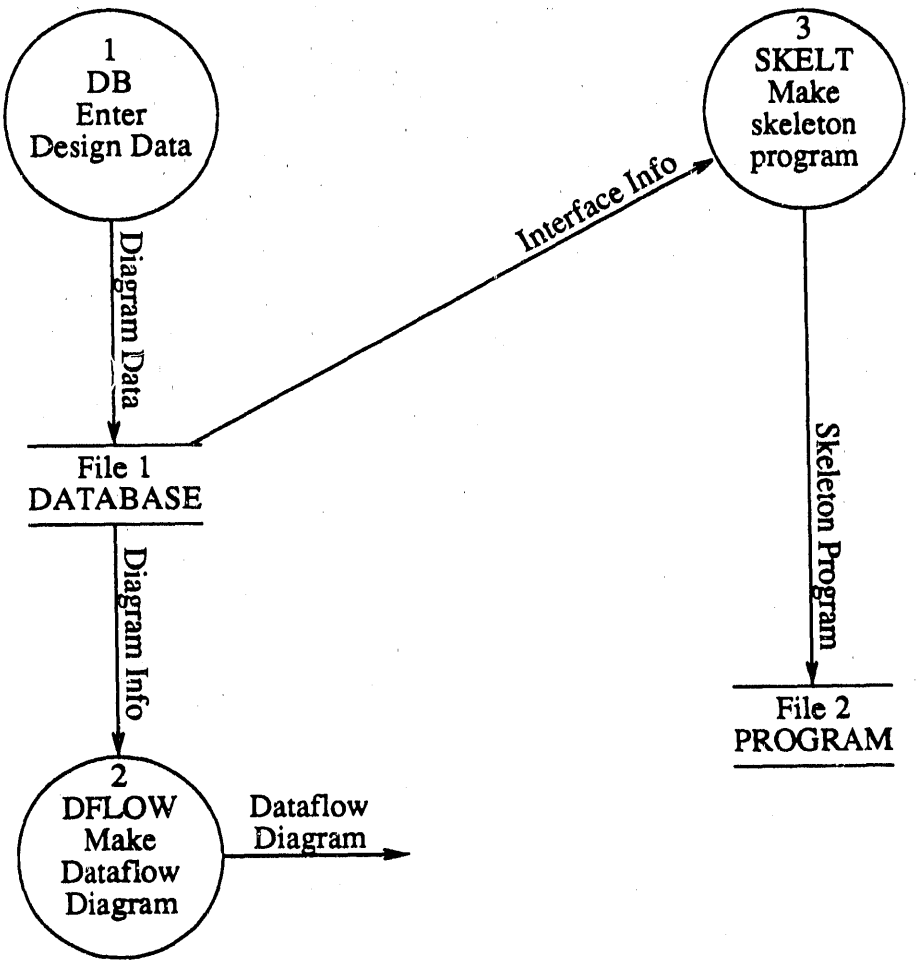

Figure 7. New Development with ARCHY

TRAMP is another ARCHY utility that is useful in new code development. As development proceeds, it is often found that a variable that exists in one module is needed in another module. TRAMP adds the missing variable to all the intervening subroutine calls so that it is delivered to the module that needs it.

Details of ARCHY Execution

ARCHY runs on a VAX under VMS 5.3. It is written in VAX FORTRAN and each of the utilities mentioned previously is actually a FORTRAN program. The system is menu-driven and the menu program makes use of the DEC SMG library. As previously mentioned, the ARCHY database is actually an ASCII file. There is one database for each program file and by default, the database is named with the same name as the program with an ".RCH" extension. ARCHY is used to analyze IBM FORTRAN and Cray FORTRAN files but they must be copied to the VAX before being fed into ARCHY.

SUMMARY

ARCHY comprises a useful set of software tools that ease maintenance of existing FORTRAN programs by making programs more readable as well as providing the means for documenting programs and the underlying design within programs. ARCHY provides a relatively painless entry into the realm of computer-assisted software engineering and, in addition, provides an environment for development of new software.

ACKNOWLEDGEMENT

The information in this article was developed during the course of work under Contract DE-AC09-89SR18035 with the U.S. Department of Energy.

REFERENCES

1. S. A. Stern. "CASE and Software Re-Engineering". *Languages and Tools SIG Session Notes, U.S. DECUS Symposium*, New Orleans, LA, pp. 156-163 (1990).

2. J. M. Levesque and J.W. Williamson. *A Guidebook to Fortran on Supercomputers.* Academic Press, CA (1989).

3. "Justifying Technology Decisions". *Digital Review*, pp. 34-38 (Oct. 2, 1989).

END

DATE FILMED

03/05/91