DOCUMENTATION: A GROWING NEED... A NEW TOOL

Marcia A. Metcalfe

ABSTRACT

The Software Design and Documentation Language (SDDL) has proven to be an effective, automated documentation tool. This paper presents the purpose, timing, and components of documentation. SDDL is introduced and related to the different timing scenarios; its capability to provide reasonable documentation is discussed and demonstrated by the use of several examples.

I. INTRODUCTION

Documentation has typically been viewed as the drudgery of software development. Weinberg refers to documentation as the castor oil of programming [1]; Kleine says that if design is Cinderella, then certainly documentation is her ugly sister [2]. And it logically follows that documentation became seen in this light since, usually, by the time the documentation stage was reached, the main characters were tired of the program and ready to move on to another project.

However, as more and more programs come into existence, and the need for meaningful documentation also increases, it will be necessary to correct this image if reasonable documentation is to be produced. Without it, the task of understanding, updating, and/or modifying a program becomes increasingly difficult (if not impossible) and costly.

This paper is organized in the following manner: the purpose, timing, and various components of documentation are discussed. Next, the automated tool, SDDL, is presented and related to the different timing scenarios; its capability to provide meaningful documentation is discussed and demonstrated through the use of several examples. Finally, conclusions are drawn.

A. PURPOSE OF DOCUMENTATION

The primary purpose of documentation is to provide undistorted communication between the parties interested in the software, both present and future [3]. Documentation should convey such information as:

- What the program does; how it functions
- How the data/information is represented
- How the various segments/routines relate to one another

This information becomes increasingly valuable as modification is required, and the original developer(s) is no longer available or has forgotten program details [4].

B. TIMING OF DOCUMENTATION

Three documentation timing scenarios will be addressed. The first, "before the fact" documentation, usually presents the program specifications and/or design. The second scenario, "concurrent" documentation, occurs throughout the various phases of software development and provides a working vehicle to prevent distortion of ideas, promotes project control, captures design changes, and permits the orderly development of software. It is useful to the development programmer as well as to the maintenance programmer. The third scenario, "after the fact" documentation typically records the history of development, demonstrates that the program works, and provides a means for maintenance [5].

C. COMPONENTS OF DOCUMENTATION

Given that different users need different information, and that no one document could probably ever provide all the information which in the future may be required, the following documentation components have been identified to accommodate the changing requirements of documentation over the program's lifecycle.

From the very early stages of software design forward, several components become essential. One such component is a high-level description (overview) of the program in prose. It should include such information as what function the program is being designed to perform and what its limitations and assumptions are. As the design progresses, the data structure becomes more defined and the algorithms/procedures which operate on the data are identified and developed. Therefore, data structure diagrams and a calling sequence diagram which shows the interrelationships of the procedures become two additional basic components of documentation.

As the design matures and coding begins, other components surface. These include the procedure (job control runstream) necessary to execute the program. It should contain a written explanation of the various steps, as well as the input/output declarations, definitions, and allocations. Another component includes descriptions of the data files which contain examples of reasonable data in terms of its mode and size/length. Source code listings, preferably with cross reference tables become an invaluable component. Machine specifications can also be documented at this time.

By the time the program code is completed and the usual documentation phase begins, current versions of the previously-mentioned components coupled with actual data files, a sample testcase with output, and references to other related documents (i.e., user's guide, design document, supporting technical papers) will provide a full complement of meaningful documentation.

II. SDDL: A TOOL

The Software Design and Documentation Language which was developed by Henry Kleine [6] at the Jet Propulsion Laboratory of the California Institute of Technology, has as its main objective communication. It facilitates communication between all the characters in the software development process (i.e., managers, customers, designers, development programmers, maintenance programmers, and the machine). SDDL automates the documentation task; it processes input (expressed in natural language or source code) and produces formatted, software documentation.

Further, methodologies have been developed for displaying representations of data, project management, and the direct processing of source code. Automatic features provided by SDDL include a table of contents, module reference tree (calling sequence diagram), and cross reference listing. User-defined features include specific cross reference listings and SDDL keyword definitions.

The remainder of this section will present SDDL's capability to address the changing documentation requirements in a program's lifecycle, and relate them through the various timing scenarios.

A. SDDL AS A DOCUMENTATION TOOL FOR THE DESIGN PHASE ("BEFORE THE FACT")

SDDL has been used successfully to design both SIMSCRIPT and FORTRAN programs. The processing of natural language statements allows a high-level description of the program's function, limitation, and assumptions, which does not have to meet typical programming language syntactical requirements. Figure 1 [7] demonstrates this capability. Capturing the physical representation of the data is also facilitated by SDDL. Figure 2 [7] illustrates this capability; Figure 3 [7] shows a refined design of a data structure.

Automatic features of the SDDL processor include:

- a table of contents
- a module reference tree (forward-calling sequence diagram)
- a module cross reference listing
- logic error detection

Figure 4 [7] is a segment of an automatically-generated module reference tree; it provides information regarding the interrelationships of the various, identified program procedures. A glossary of terms can also be facilitated by the SDDL processor.

B. SDDL AS A DOCUMENTATION TOOL FOR THE CODING PHASE ("CONCURRENT")

SDDL provides a working vehicle which facilitates the coding phase of software development (see Figure 5 [7]). This is the point at which coding conventions can be adopted to allow for the direct processing of source code; and the external data file representation can be documented. Procedures for flagging revisions can be instituted; and user-specified cross reference listings (for global variables, data files, footnotes, etc.) can assist the development programmer. Project management techniques (including a calendar of events, milestones, and progress charts) can be incorporated into the document for use during this phase.

C. SDDL AS A TOOL FOR THE DOCUMENTATION PHASE ("AFTER THE FACT")

Two methodologies for using SDDL during the documentation phase have been developed. The first is the direct processing of SIMSCRIPT, and other high-level languages, source code through SDDL; the second is using the SDDL processor to generate a supporting document to existing source code listings.

1. The Direct Processing of SIMSCRIPT Source Code

Figure 6 is the result of processing SIMSCRIPT source code through SDDL. The document formatting features enhance both the clarity and flow of control in this routine. User-defined cross reference listings can be generated at this point to capture machine portability considerations and I/O devices. The automatically-generated SDDL features provide additional information about the source code.

2. As a Supporting Document for an Existing Program

SDDL can be used to document existing software written in any programming language. Figure 7 [8] shows SDDL being used to capture the physical data representation of an existing FORTRAN program. Meaningful identifiers have been added to clarify the cryptic descriptors; and their instances can be gathered in a user-specified cross reference table. Additionally, variable mode and units of measure have been supplied.

Figure 8 [8] illustrates SDDL being applied to capture the structure/algorithm, at a high level, of an existing FORTRAN program. Natural language statements lend clarity to the routine description; automatic document formatting features lend flow of control visibility.

III. CONCLUSIONS

SDDL can take some of the "drudge" out of documentation by capturing meaningful information during the various phases of software development as well as by transfering the burden to the computer. It allows for details, usually recalled from someone's memory at the end of a project, to be recorded as they occur during the project. This single, automated medium facilitates the communication between the various members in the software development process over time, thereby providing a two-dimensional documentation tool (i.e., between people, over time).

Additionally, SDDL provides a framework for implementing documentation standards. It skews the documentation effort away from the end (when developers are very busy verifying, debugging, and testing) and toward the beginning of the project (when developers are less busy).

When used in a "concurrent" documentation mode, SDDL provides a working vehicle which begins as a designer's tool, then becomes a development programmer's tool, and finally emerges a a maintenance programmer's tool. When used in an "after the fact" documentation mode, SDDL can produce a document in support of existing source code which adds clarity and visibility into the program's actions. Further, SDDL can generate documentation directly from source code.

REFERENCES

1. WEINBERG, Gerald M. (1971) The Psychology of Computer Programming. New York: Van Nostrand Reinhold Company.

2. KLEINE, Henry (1977) "A Vehicle for Developing Standards for Simulation Programming." Presented at the Winter Simulation Conference, Gaithersburg, MD., December 5-7.

3. COOLEY, Belva J. (1977) "Documenting Simulation Studies for Management." Presented at the Winter Simulation Conference, Gaithersburg, MD., December 5-7.

4. HARPER, William L. (1973) Data Processing Documentation: Standards, Procedures and Applications. Englewood Cliffs: Prentice-Hall, Inc.

5. TAUSWORTHE, Robert C. (1976) Standardized Development of Computer Software. Pasadena: Jet Propulsion Laboratory, California Institute of Technology.

6. KLEINE, Henry (1977) Software Design and Documentation Language. Pasadena: Jet Propulsion Laboratory, California Institute of Technology.

7. CHAMBERLAIN, R.G., P.J. FIRNETT, D.A. HEIMBURGER, M.H. HORTON, B.L. KLEINE, M.A. METCALFE (1978) SAMIS III Design Document. Pasadena: Jet Propulsion Laboratory, California Institute of Technology.

8. Jet Propulsion Laboratory (1978) PARAMET (An Electric Vehicle Simulator) User's Manual. Pasadena: Jet Propulsion Laboratory, California Institute of Technology.

```
LINE                                           APRIL 27, 1978                                              PAGE    8
PROGRAM OBJECTIVES                                                                                              RGC
    *****************************************************************************************************
    *    SAMIS III IS AN ECONOMIC MODEL OF A HYPOTHETICAL U.S. INDUSTRY TO MANUFACTURE SILICON SOLAR          *
    * MODULES, WHICH ARE USED TO GENERATE ELECTRICITY DIRECTLY FROM SUNLIGHT BY THE PHOTOELECTRIC EFFECT.  *
    *                                                                                                      *
    *    IT IS INTENDED THAT THE SAMIS III PROGRAM FACILITATE STANDARDIZED COMPARISON OF THE RELATIVE       *
    * ECONOMICS OF COMPETING MANUFACTURING PROCESSES.  IT IS ALSO INTENDED THAT IT FACILITATE ASSESSMENT   *
    * OF COMPLETE SEQUENCES OF PROCESSES WITH RESPECT TO THE LOW-COST SOLAR ARRAY (LSA) PROJECT            *
    * GOALS.  FURTHER, IT IS INTENDED TO PROVIDE INFORMATION THAT WILL HELP IN DETERMINING FRUITFUL        *
    * AREAS OF RESEARCH.                                                                                   *
    *                                                                                                      *
    *    THE INPUTS TO THE SAMIS III MODEL FALL INTO SEVERAL GROUPS:                                       *
    *                                                                                                      *
    * A) DESCRIPTIONS OF THE ECONOMIC CHARACTERISTICS OF EACH MANUFACTURING PROCESS/MACHINE                *
    *    1) PROCESS PARAMETERS (PRODUCT PRODUCED, RATE, DUTY CYCLE, ETC)                                   *
    *    2) EQUIPMENT COST FACTORS                                                                         *
    *    3) FACILITIES AND PERSONNEL REQUIREMENTS (PER MACHINE)                                            *
    *    4) BYPRODUCTS PRODUCED AND UTILITIES AND COMMODITIES REQUIRED (PER MINUTE)                        *
    *    5) PRODUCTS USED IN THE PROCESS (AND THE ASSOCIATED YIELDS)                                       *
    *                                                                                                      *
    * B) DESCRIPTION OF THE TECHNOLOGICAL AND ECONOMIC STRUCTURE                                           *
    *    1) OF FIRMS IN THE INDUSTRY                                                                       *
    *    2) OF PROCESSES IN EACH FIRM                                                                      *
    *                                                                                                      *
    * C) STANDARD DATA                                                                                     *
    *    1) PRICES OF PERSONNEL, COMMODITIES, ETC. AS FUNCTIONS OF QUANTITIES                              *
    *    2) INDIRECT REQUIREMENTS AS FUNCTIONS OF QUANTITIES                                               *
    *    3) RELATIONSHIPS FOR ESTIMATING INITIAL CAPITAL                                                   *
    *    4) INFLATION RATES AND OTHER ECONOMIC PARAMETERS                                                  *
    *                                                                                                      *
    * D) RUN TIME DATA                                                                                     *
    *    1) RANGE OF DEMANDS FOR PHOTOVOLTAIC POWER                                                        *
    *    2) RANGE OF ANOTHER PARAMETER TO BE VARIED [TO BE IMPLEMENTED IN A LATER RELEASE]                 *
    *    3) "SWITCH" SETTINGS (SUCH AS THE INTEGRAL.MACHINES.FLAG)                                         *
    *                                                                                                      *
    *    FROM DESCRIPTIONS OF THE MANUFACTURING PROCESSES, DETERMINISTIC EQUATIONS DESCRIBING THE          *
    * MANUFACTURING COSTS OF EACH PROCESS, AND BUSINESS COSTS OF EACH FIRM HOUSING ONE OR MORE OF THESE    *
    * PROCESSES, THE MODEL ESTIMATES THE PRICES THAT MAY REASONABLY BE EXPECTED FOR SOLAR MODULES AND      *
    * ANY RECOGNIZABLE INTERMEDIATE PRODUCTS THAT ARE USED IN THEIR MANUFACTURE.  COST ELEMENTS ARE        *
    * ALLOCATED TO EVERY PROCESS.                                                                          *
    *                                                                                                      *
    *    BY PERFORMING SENSITIVITY ANALYSES OF VARIOUS PARAMETERS INVOLVED IN THE MODEL, AND BY            *
    * ANALYZING THE EFFECT ON PRICE OF DIFFERENT INDUSTRY CONFIGURATIONS, INFERENCES CAN BE DRAWN WITH     *
    * RESPECT TO RESEARCH PRIORITIES AND THE EFFECTS OF GOVERNMENTAL AND INDUSTRIAL ACTIONS.               *
    *                                                                                                      *
    *    WHILE SAMIS III WAS DESIGNED FOR THE NASCENT SOLAR ARRAY INDUSTRY, THE METHODOLOGY IS NOT         *
    * INDUSTRY DEPENDENT.  APPLICABILITY TO MANUFACTURERS IN OTHER INDUSTRIES CAN BE ACHIEVED BY RATHER    *
    * MODEST AUGMENTATION OF THE STANDARD DATA.                                                            *
    *****************************************************************************************************
END
```

Figure 1. SDDL Processing High-level, Natural Language Statements

```
LINE                                             APRIL 27, 1978                                                    PAGE   12
DATA_STRUCTURE NOMENCLATURE                                                                                               RGC
 ****************************************************************************************************
 *    DATA STRUCTURES ARE DESCRIBED IN TERMS OF ENTITIES, WHICH ARE THE OBJECTS OUT OF WHICH THE MODEL *
 * IS CONSTRUCTED; ATTRIBUTES, WHICH DESCRIBE VARIOUS CHARACTERISTICS OF THE ENTITIES; AND SETS, WHICH *
 * DESCRIBE STRUCTURED RELATIONSHIPS AMONG THE ENTITIES.                                               *
 *                                                                                                     *
 *    ATTRIBUTES CONTAIN VALUES, USUALLY NUMERIC, BUT SOMETIMES ALPHABETIC OR ALPHANUMERIC.  SOME OF    *
 * THESE VALUES ARE CHARACTERISTIC OF THE PARTICULAR ENTITY THEY DESCRIBE (AS, FOR INSTANCE, THE       *
 * EXPENSE ITEM: PRICE.VS.QUANTITY.TABLE).  THIS KIND OF ATTRIBUTE IS IDENTIFIED IN THE DATA STRUCTURE *
 * BY AN S (TO MEAN "SAVED" OR "STATIC") AT THE RIGHT HAND MARGIN.  VALUES OF THESE ATTRIBUTES ARE     *
 * STORED WITH THE ENTITY WHENEVER THE ENTITY IS SAVED ON FILE.  OTHER ATTRIBUTE VALUES ARE CALCU-     *
 * LATED DURING THE COURSE OF A RUN, AND ARE NOT STORED AS PART OF THE ENTITY DESCRIPTION.  THIS       *
 * KIND OF ATTRIBUTE IS IDENTIFIED BY A D (TO MEAN "DYNAMIC" OR "DERIVED") AT THE RIGHT HAND MARGIN.   *
 *                                                                                                     *
 *    SETS ARE "OWNED" BY ENTITIES AND IN CONSEQUENCE, ARE VERY MUCH LIKE ATTRIBUTES.  THEY ARE REPRE- *
 * SENTED IN THE DATA STRUCTURE LIKE ATTRIBUTES, EXCEPT THAT THEIR NAMES ARE PRECEDED BY THE WORD      *
 * "SET".  THE KIND OF ENTITIES WHICH "BELONG TO" THE SET FOLLOW THE SET NAME AND ARE INDENTED.        *
 *                                                                                                     *
 *    THE HIGHEST LEVEL OF STRUCTURE CAN BE AN ENTITY (SUCH AS A PROCESS:), A SET (SUCH AS THE         *
 * COST.ACCOUNT.STRUCTURE), OR AN ATTRIBUTE.  ATTRIBUTES AT THE HIGHEST LEVEL DESCRIBE OR ARE USED BY  *
 * THE MODEL ITSELF, AND ARE CALLED "GLOBAL VARIABLES" (SUCH AS THE NORMAL\INPUT\UNIT).                *
 *                                                                                                     *
 *    IN ADDITION TO THE S OR D DESCRIBED ABOVE, EACH ATTRIBUTE IN THE DATA STRUCTURE DESCRIPTIONS     *
 * HAS ITS UNITS OF MEASURE AND ITS MODE.  THE FOLLOWING CODES ARE USED FOR THE MODES:                 *
 *                                                                                                     *
 * (I)      INTEGER                                                                                    *
 * (R)      REAL                                                                                       *
 * (T)      TEXT                                                                                       *
 * (U)      UNIQUE TEXT                                                                                *
 * (C)      SINGLE CHARACTER                                                                           *
 * (V)      REAL VECTOR                                                                                *
 * (2)      2 BY N TABLE                                                                               *
 * (M)      REAL MATRIX                                                                                *
 * (T V)    TEXT VECTOR                                                                                *
 * (I V)    INTEGER VECTOR                                                                             *
 ****************************************************************************************************

 THE DATA STRUCTURE CONSISTS OF
 THE COST_ACCOUNTS DATA_STRUCTURE-------------------------------------------------------------->( 13)
 THE CURRENT_TECHNOLOGY DATA_STRUCTURE--------------------------------------------------------->( 15)
 THE AVAILABLE_COMPANIES DATA_STRUCTURE-------------------------------------------------------->( 16)
 THE CURRENT_CONFIGURATION DATA_STRUCTURE------------------------------------------------------>( 17)
 THE DIRECTORIES DATA_STRUCTURE---------------------------------------------------------------->( 22)
 THE GLOBAL_VARIABLES DATA_STRUCTURE----------------------------------------------------------->( 23)
 THE PARSER DATA_STRUCTURE--------------------------------------------------------------------->( 24)
 THE TABLE_EXAMPLES---------------------------------------------------------------------------->( 25)

END_DATA_STRUCTURE
```

Figure 2. High-level Data Representation

```
LINE                                          APRIL 27, 1978                                                          PAGE   15
DATA_STRUCTURE FOR CURRENT_TECHNOLOGY                                                                                3/24/78 1.9
    ******************************************************************************************************
    * THE CURRENT TECHNOLOGY SET CONTAINS A "CATALOG" OF THE PROCESSES THAT HAVE BEEN DESCRIBED.  EACH  *
    * PROCESS DESCRIPTION IS THE COMPUTER EQUIVALENT OF A "FORMAT A" (JPL FORM 3037-S).  IN ADDITION    *
    * TO THE INFORMATION PROVIDED ON THOSE FORMATS IS A LOT OF STANDARDIZED DATA, WHICH THE USER CAN    *
    * CHANGE ON A PROCESS BY PROCESS BASIS FOR SENSITIVITY STUDIES OR FOR A MORE REALISTIC DESCRIPTION. *
    ******************************************************************************************************

    SET 'CURRENT.TECHNOLOGY'                                                                                        S
       ENTITY 'PROCESS:'                                                                                            S
          'REFERENT' (SEE TABLE 3 IN DATA_STRUCTURE TABLE_EXAMPLES)                                             (U) S
          'DESCRIPTIVE.NAME'                                                                                    (T) S
          'PRODUCT.REFERENT' (SEE TABLE 4 IN DATA_STRUCTURE TABLE_EXAMPLES)                                     (U) S
          'PRODUCT.NAME'                                                                                        (T) S
          'PRODUCT.UNITS' (SEE TABLE 4 BELOW)                                                                   (T) S
          'OUTPUT.RATE'                                                          OUTPUT UNITS/OPERATING MINUTE (R) S
          'PROCESSING.TIME'                                                 CALENDAR MINUTES AT THE WORK STATION (R) S
          'USAGE.FRACTION' (MACHINE DUTY CYCLE)                            OPERATING MINUTES/FACTORY OPEN MINUTE (R) S
          SET 'MACHINE.DESCRIPTION'                                                                                 S
             ENTITY 'COMPONENT:'                                                                                    S
                'REFERENT' (SEE TABLE 5 IN DATA_STRUCTURE TABLE_EXAMPLES)                                       (I) S
                'DESCRIPTIVE.NAME'                                                                              (T) S
                'PRICE.YEAR'                                                                          DATE YEAR (I) S
                'PURCHASE.COST'                                                              PRICE YEAR DOLLARS (R) S
                'USEFUL.LIFE'                                                                             YEARS (R) S
                'SALVAGE.VALUE'                                                              PRICE YEAR DOLLARS (R) S
                'REMOVAL.AND.INSTALLATION.COST'                                              PRICE YEAR DOLLARS (R) S
                'PAYMENT.FLOAT.INTERVAL'                                                                  YEARS (R) S
                'INFLATION.RATE.TABLE'                                                       YEAR, PERCENT/YEAR (2) S
                'TAX.LIFE'                                                                                YEARS (I) D
                'EQUIPMENT.TAX.DEPRECIATION.METHOD'                                                             (T) S
                'ACCOUNTING.LIFE'                                                                         YEARS (I) D
                'EQUIPMENT.BOOK.DEPRECIATION.METHOD'                                                            (T) S
          SET 'BYPRODUCT.OUTPUTS'                                                                                   S
             ENTITY 'BYPRODUCT:'                                                                                    S
                'EXPENSE.ITEM.REFERENCE' (SEE TABLE 2 IN DATA_STRUCTURE TABLE_EXAMPLES)                         (T) S
                'EXPENSE.ITEM.POINTER'                                                            CORE LOCATION (I) D
                'AMOUNT.PER.CYCLE'                                                                  UNITS/CYCLE (R) S
          SET 'FACILITIES.AND.PERSONNEL.REQUIREMENTS'                                                               S
             ENTITY 'FACILITY.OR.PERSONNEL:'                                                                        S
                'EXPENSE.ITEM.REFERENCE' (SEE TABLE 2 IN DATA_STRUCTURE TABLE_EXAMPLES)                         (T) S
                'EXPENSE.ITEM.POINTER'                                                            CORE LOCATION (I) D
                'AMOUNT.PER.MACHINE'                                                              UNITS/MACHINE (R) S
          SET 'UTILITIES.AND.COMMODITIES.REQUIREMENTS'                                                              S
             ENTITY 'COMMODITY:'                                                                                    S
                'EXPENSE.ITEM.REFERENCE' (SEE TABLE 2 IN DATA_STRUCTURE TABLE_EXAMPLES)                         (T) S
                'EXPENSE.ITEM.POINTER'                                                            CORE LOCATION (I) D
                'AMOUNT.PER.CYCLE'                                                                  UNITS/CYCLE (R) S
          SET 'REQUIRED.PRODUCTS'                                                                                   S
             ENTITY 'REQUIRED.PRODUCT:'                                                                             S
                'PRODUCT.REFERENCE' (SEE TABLE 4 IN DATA_STRUCTURE TABLE_EXAMPLES)                              (T) S
                'YIELD'                                                            OUTPUT UNITS/INPUT UNIT (R) S

END_DATA_STRUCTURE
```

Figure 3. Lower-level Data Structure Diagram

```
   **************  MODULE REFERENCE TREE  ******                                                        PAGE 228
LN  PAGE
 1     1  TEAM_MEETINGS_AND_AGENDA

 2     2  SCHEDULE_AND_MILESTONES

 3     3  PROGRESS_CHART

 4     4  MEMORANDA

 5     6  ACKNOWLEDGEMENTS

 6     8  OBJECTIVES

 7     9  READING_CONVENTIONS
 8   200     CALL_A_ROUTINE

 9    12  NOMENCLATURE
10    13  .  COST_ACCOUNTS
11    15  .  CURRENT_TECHNOLOGY
12    16  .  AVAILABLE_COMPANIES
13    17  .  CURRENT_CONFIGURATION
14    22  .  DIRECTORIES
15    23  .  GLOBAL_VARIABLES
16    24  .  PARSER
17    25  .  TABLE_EXAMPLES

18    28  TOP_LEVEL_COMMANDS

19    29  MANIPULATION_COMMANDS

20    30  MAIN
21   180  .  INITIALIZE_PROGRAM
22   181  .  .  GET_MACHINE_SPECIFICATIONS
23   182  .  .  GET_PARSER_VOCABULARY
24   183  .  .  GET_FILE_DIRECTORIES
25   184  .  .  GET_MODEL_DEFAULTS
26   188  .  .  .  SEARCH_A_SET_FOR_A_REFERENT
27   127  .  .  .  PRINT_A_WARNING
28   129  .  .  .  .  PRINT_ERROR_MESSAGE
29   200  .  .  .  CALL_A_ROUTINE
30   120  .  GET_THE_NEXT_COMMAND
31   123  .  .  INTERPRET_THE_USERS_NEXT_WORD
32   124  .  LIST_COMMAND_CHOICES
33   131  .  .  PRINT_INTRODUCTORY_VERSION_OF_COMMAND_CHOICES
34   122  .  HELP_WITH_TOP_LEVEL_COMMANDS
35   123  .  .  INTERPRET_THE_USERS_NEXT_WORD
36   134  .  .  PRINT_TOP_LEVEL_HELP_MESSAGES
37   129  .  .  .  PRINT_ERROR_MESSAGE
38   139  .  .  .  PRINT_ADDITIONAL_TOP_LEVEL_HELP_MESSAGES
39   200  .  .  .  .  CALL_A_ROUTINE
40   200  .  .  .  CALL_A_ROUTINE
41   130  .  .  PRINT_EXCEPTION_MESSAGE
42   129  .  .  .  PRINT_ERROR_MESSAGE
43   146  .  SET_PROMPT_LEVEL
44   123  .  .  INTERPRET_THE_USERS_NEXT_WORD
45   130  .  .  PRINT_EXCEPTION_MESSAGE
```

Figure 4. Automatically-generated Module Reference Tree

```
                                              APRIL 27, 1978                                        PAGE   30
 PROGRAM MAIN ROUTINE                                                                              3/24/78 2.2

    ********************************************************************
    * THIS IS THE STARTING POINT AND "HOME BASE" FOR THE PROGRAM.       *
    *                                                                   *
    * THIS PROCEDURE RESPONDS TO THE USER'S "SAMIS TOP-LEVEL" COMMANDS. *
    ********************************************************************

    NOW INITIALIZE_PROGRAM------------------------------------------------------------------------------>(180)

    LOOP UNTIL USER TERMINATES PROGRAM EXECUTION

       NOW GET_THE_NEXT_COMMAND------------------------------------------------------------------------->(120)
       * YIELDING;  COMMAND NUMBER

       SELECT CASE PER COMMAND NUMBER

       CASE 1; "?"
          NOW LIST_COMMAND_CHOICES---------------------------------------------------------------------->(124)
       CASE 2; "HELP" OR "EXPLAIN"
          NOW HELP_WITH_TOP_LEVEL_COMMANDS-------------------------------------------------------------->(122)
       CASE 3; "PROMPT"
          NOW SET_PROMPT_LEVEL-------------------------------------------------------------------------->(146)
       CASE 4; "QUERY"
          NOW QUERY_EXISTING_ENTITIES------------------------------------------------------------------->(145)
       CASE 5; "CREATE"
          NOW CREATE_AND_MANIPULATE_THE_ENTITY---------------------------------------------------------->( 32)
       CASE 6; "FIND"
          NOW FIND_AND_MANIPULATE_THE_ENTITY------------------------------------------------------------>( 34)
       CASE 7; "REPLACE"
          NOW REPLACE_THE_ENTITY ------------------------------------------ [DEFERRED TO A LATER RELEASE]>( 47)
       CASE 8; "SIMULATE"
          NOW STRUCTURE_INDUSTRY_AND_VERIFY_COHERENCE--------------------------------------------------->( 49)
          * YIELDING; ABORT FLAG
          IF ABORT FLAG = "ABORT"
    <---------CYCLE
          ENDIF
          NOW SIMULATE---------------------------------------------------------------------------------->( 53)
       CASE 9; "PLOT"
          NOW CALL_A_ROUTINE TO PLOT_A_GRAPH ------------------------------ [DEFERRED TO A LATER RELEASE]>(200)
       CASE 10; "STOP"
 <---------EXITPROGRAM

       ENDSELECT

    REPEAT UNLESS USER HAS TERMINATED PROGRAM EXECUTION

 END PROGRAM MAIN ROUTINE
```

Figure 5. Refined Design of a Routine with Project Management Information, Footnotes, and Various Automatic Formatting Features

```
LINE                                                                                              PAGE  85
    ROUTINE TO LIST.DATA.FOR.A.SYSTEM.ENTITY                                      ''               6.1

       DEFINE EXCEPTION.CODE AND COMMAND AS INTEGER VARIABLES

       NOW INTERPRET.THE.USERS.NEXT.WORD------------------------------------------------->( 40)
       GIVEN 2 ''CONTEXT = ENTITY TYPES
       AND 0 ''NO TEXT TO BE RETURNED
       YIELDING EXCEPTION.CODE AND COMMAND

       IF EXCEPTION.CODE IS NOT ZERO ''AN INPUT ERROR WAS FOUND
          NOW PRINT.EXCEPTION.MESSAGE----------------------------------------------------->( 22)
          GIVEN EXCEPTION.CODE
       ELSE
          IF COMMAND IS LESS THAN 1 OR COMMAND IS GREATER THAN 8,
             NOW PRINT.ERROR.MESSAGE GIVEN 3''OUT OF BOUNDS IN LIST.DATA------------------>( 23)
          ELSE
             SELECT CASE(COMMAND)'
             CASE(1)'
                NOW LIST.VEHICLE.DATA---------------------------------------------------->( 86)
    <------------RETURN
             CASE(2)'
                NOW LIST.CHASSIS.DATA---------------------------------------------------->( 87)
    <------------RETURN
             CASE(3)'
                NOW LIST.ENGINE.DATA----------------------------------------------------->( 89)
    <------------RETURN
             CASE(4)'
                NOW LIST.TRANSMISSION.DATA----------------------------------------------->( 91)
    <------------RETURN
             CASE(5)'
                NOW LIST.DRIVE.POWER.TRAIN.DATA------------------------------------------>( 93)
    <------------RETURN
             CASE(6)'
                NOW LIST.SIMULATION.SPECS.DATA------------------------------------------->( 95)
    <------------RETURN
             CASE(7)'
                NOW LIST.DRIVING.SCHEDULE.DATA------------------------------------------->( 97)
    <------------RETURN
             CASE(8)'
                NOW LIST.ENVIRONMENT.DATA------------------------------------------------>( 98)
             ENDSELECT
          ALWAYS
       ALWAYS

    <--RETURN
    END
```

Figure 6. Actual SIMSCRIPT Source Code Enhanced by the SDDL Processor

```
LINE                                                                        PAGE    7
PROGRAM DATA_STRUCTURE

     ***** NOTE ***** ALL VARIABLES ARE REAL, EXCEPT THOSE WITH AN I AFTER THE
          DESCRIPTION WHICH ARE INTEGER TYPE -- NO LOGICAL VARIABLES ARE USED
                    UNITS ARE INDICATED FOR EACH VARIABLE, EXCEPT "(-)" WHICH
          INDICATES UNITLESS QUANTITIES

     COMMON  !BATRY!  ALL PARAMETERS PERTAINING TO THE BATTERY
        'BATWT'    BATTERY.WEIGHT                                                  (LB)
        'CHLIM'    BATTERY.CHARGING.LIMIT                                           (-)
        'DMASS'    INCREMENTAL.CHANGE.OF.BATTERY.MASS                              (LB)
        'DMSUM'    ACCUMULATED.BATTERY.MASS.OVER.DRIVING.CYCLE                     (LB)
        'DSLIM'    BATTERY.DISCHARGING.LIMIT                                        (-)
        'IDLTM'    IDLING.TIME.ON.DRIVING.CYCLE, I                                (SEC)
        'KDSCH'    BATTERY.CHARGE.DISCHARGE.POLYNOMIAL.COEFFICIENTS                 (-)
        'RECUP'    BATTERY.RECUPERATION.FACTOR                                      (%)
        'REGNF'    REGENERATIVE.BRAKING.FACTOR                                      (%)
        'VVLIM'    LOW.SPEED.REGENERATION.LIMIT                                 (MI/HR)

     COMMON  !CARS!  ALL PARAMETERS BASIC TO THE VEHICLE
        'A'        VEHICLE.FRONTAL.AREA                                         (SQ FT)
        'BKFAC'    REGENERATIVE.BRAKING.ENERGY.FACTOR                               (-)
        'CD'       VEHICLE.DRAG.COEFFICIENT                                         (-)
        'CDA'      VEHICLE.DRAG.COEFFICIENT.FRONTAL.AREA.PRODUCT                (SQ FT)
        'INRTA'    DRIVE.LINE.ROTATING.MASS.FACTOR                                  (-)
        'TIRE'     TIRE.COEFFICIENT.OF.RESISTANCE                                   (%)
        'WT'       TOTAL.VEHICLE.WEIGHT                                            (LB)

     COMMON  !CONST!  CONSTANTS AND FLAGS
        'CFLAG'    PRINT.CHANGE.OPTIONS.FLAG, I                                     (-)
        'G'        ACCELERATION.DUE.TO.GRAVITY                              (FT/SEC/SEC)
        'MHFS'     MI.PER.HR.TO.FT.PER.SEC.CONVERSION.FACTOR               (FT-HR/SEC/MI)
        'PFLAG'    PERFORMANCE.REQUIREMENTS.PRINTING.FLAG, I                        (-)
        'WHERE'    LOOP.BACK.FLAG, I                                                (-)

     COMMON  !ETA!  EFFICIENCY PARAMETERS
        'ECOEF'    EFFICIENCY.LINE.CALCULATION.ARRAY                                (-)
        'EFAVE'    AVERAGE.EFFICIENCY                                               (%)
        'EFFAC'    CALCULATED.EFFICIENCY.POINT                                      (%)
        'EFTIM'    AVERAGE.EFFICIENCY.TIME.PERIOD, I                              (SEC)
        'ESPED'    EFFICIENCY.SPEED.CALCULATION.BREAKPOINTS                     (MI/HR)
        'EVALU'    EFFICIENCY.VALUE.CALCULATION.BREAKPOINTS                         (%)

     COMMON  !NOW!  TRACE PARAMETERS AND INSTANTANEOUS VALUES
        'PRMAX'    TRACE.PRINTING.TERMINATE.TIME, I                               (SEC)
        'PRMIN'    TRACE.PRINTING.START.TIME, I                                   (SEC)
        'PTOT'     INSTANTANEOUS.TOTAL.POWER                                       (HP)
        'V'        INSTANTANEOUS.SPEED                                          (MI/HR)

     COMMON  !OUTPT!  CALCULATED PARAMETERS WHICH WILL BE PRINTED
        'COST'     OPERATING.COST                                                ($/MI)
        'E'        CONSUMED.ENERGY.OVER.CYCLE                                    (KW-HR)
        'EPM'      CONSUMED.ENERGY.PER.DISTANCE.OVER.CYCLE                    (KW-HR/MI)
        'HPWHL'    AVERAGE.POWER.AT.WHEELS.OVER.CYCLE                              (HP)
        'RANGE'    TOTAL.RANGE.ON.A.BATTERY.CHARGE                                 (MI)
```

Figure 7. SDDL Used to Document Data Structures in an Existing FORTRAN Program

```
LINE                                                                                                PAGE   11
PROGRAM MAIN PROGRAM

    $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
    $                                                                                $
    $ THE MAIN PROGRAM ESSENTIALLY IS USED TO CALL THE PRINCIPAL SUBROUTINES $
    $                                                                                $
    $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

    COMMON  !BATRY!, !CAR!, !CONST!, !ETA!, !NOJ!, !OUTPT!, !POWER!,
       !ROUTE!

    ***** NOTE ***** NO VARIABLES ARE TRANSFERRED IN THE MAIN LINE SUBROUTINES IN
         "CALL" STATEMENTS. ALL VARIABLES ARE TRANSFERRED IN THE LABELLED COMMONS

LABEL: START
    NOW INITIALIZE_VARIABLES (INLINE)-------------------------------------------------->( 12)

    PRINT WELCOME MESSAGE

    CALL SUBROUTINE_QUERY TO INPUT CAR PARAMETERS-------------------------------------->( 13)

LABEL: NEW.OR.RECALCULATE.DRIVING.CYCLE
    CALL SUBROUTINE_DRIVE TO SPECIFY DRIVING CYCLE PARAMETERS-------------------------->( 16)

    CALL SUBROUTINE_BEGIN TO PRINT VEHICLE PARAMETERS---------------------------------->( 20)

    IF CYCLE = 0, THAT IS CONSTANT SPEED IS SELECTED

       CALL SUBROUTINE_SCALC TO CALCULATE FOR CONSTANT SPEED--------------------------->( 23)
       CALL SUBROUTINE_SRSLT TO PRINT RESULTS------------------------------------------>( 25)

    ELSE ANY OF THE DRIVING CYCLES HAS BEEN SELECTED

       CALL SUBROUTINE_CCALC TO CALCULATE FOR DRIVING CYCLE---------------------------->( 27)
       CALL SUBROUTINE_CRSLT TO PRINT RESULTS------------------------------------------>( 31)

    ENDIF

    CALL SUBROUTINE_PRFRM TO PRINT PERFORMANCE REQUIREMENTS IF FLAG IS SET------------->( 33)

    CALL SUBROUTINE_CHANGE TO ALLOW USER CHANGES--------------------------------------->( 39)
    *YIELD USER CHOICE

    SELECT CASE PER USER CHOICE
    CASE 1: LOGOFF
<-----EXITPROGRAM
    CASE 2: RESTART
<-----GO TO START
    CASE 3: USER WANTS TO CHANGE DRIVING CYCLE
<-----GO TO NEW.OR.RECALCULATE.DRIVING.CYCLE TO GET A NEW CYCLE
    CASE 4: USER CHANGED AN INPUT PARAMETER
<-----GO TO NEW.OR.RECALCULATE.DRIVING.CYCLE TO RECALCULATE THE DRIVING CYCLE
            PARAMETERS BECAUSE THE VEHICLE PARAMETERS WERE CHANGED
    ENDSELECT CASE

ENDPROGRAM MAIN PROGRAM
```

Figure 8. SDDL Used to Document a Routine/Algorithm in an Existing FORTRAN Program