INTRODUCTION TO THE SIMSCRIPT II PROGRAMMING LANGUAGE

Philip J. Kiviat
The RAND Corporation
Santa Monica, California

ABSTRACT

SIMSCRIPT II is a new computer programming language designed and implemented at The RAND Corporation[1]. It has been modeled after SIMSCRIPT[2], the simulation programming language introduced by RAND in 1963, but goes far beyond the design goals of that language. The design goals of SIMSCRIPT II were stated in[3]; briefly, they were to produce a readable, user-oriented language strongly oriented to the efficient debugging and running of large simulation models. In addition, the implementation was to be as computer independent as possible. As it is difficult in a short abstract to describe a complex language adequately, the following strategy has been adopted. Section 1 describes the basic properties of the SIMSCRIPT II language in concise, but hopefully communicative, terms. Section 2 illustrates the language through excerpts taken from a job shop simulation model. Section 3 discusses the present status of the language, its implementation and its use.

The SIMSCRIPT II Programming Language

SIMSCRIPT II is documented in a series of levels. The first five levels constitute the language initially released to the public. Levels 6 and 7 are at present RAND research topics,aand will be integrated with the previous levels as they become operational. This paper deals only with levels 1 through 5.

Level 1 is a basic programming language designed for non-programmers. Its lack of declarations of any sort, free-form syntax, forgiving nature with regard to programming errors, free-form data input statement, pictoral output statement and relative freedom from naming conventions make it simple to learn and use.

Level 2 is an algebraic compiler roughly comparable to FORTRAN in power. Its special features are full dynamic storage allocation for n-dimensional arrays, recursive subprogram structures, and comprehensive logical branching and looping phrases. Subprograms can be used both as functions and as procedures. Data can be global to an entire program or local to a subprogram in a recursive or "own" sense.

Level 3 brings SIMSCRIPT II up to the power of ALGOL or PL/I. Included are additional logical testing and control statements, statements that search collections of data for conditionally specified values and compute functions of these values, an internal buffer for data conversions, numerous non record-oriented input/output statements, a report generator and an ALPHA variable mode.

Level 4 adds the SIMSCRIPT concepts of entities, attributes and sets to levels 1 through 3. Program definitions are made in an English-like free-form language, and provide full control of the assignment of data to data structures and computer words as well as the generation of code. The concepts of entities and sets are extended, as are the properties of attributes. Entities can be simple or compound; attributes can be functions as well as values. A TEXT mode enables arbitrarily long character strings to be read, manipulated and displayed. New programming concepts such as implied subscripting, left-handed functions and monitored variables are introduced[4,5,9]

In Level 5 facilities are added for simulation. Event control is extended to allow activities as well as event representations. Interrupt control is provided through monitored variables for "process-like" applications[6,7]. Events can be ranked and given priorities; multiple external event tapes can be used. Events can be triggered both internally and externally. A library of statistical functions is provided, as are facilities for dealing with other statistical problems such as antithetic variates, parallel random number streams and discrete sampling distributions. Two classes of global declarations provide automatic data collection and analysis and program monitoring.

Excerpts From a Simulation Model

The following sections of code illustrate a complex SIMSCRIPT II preamble (definitional section), an event routine and a subprogram. Unfortunately there is not space available to provide a complete program or to explain the nuances of all the statements shown. The complete program and an explanation of it appear elsewhere[1,8]. Basically, the program models a typical job shop in which partially completed orders flow from one group of machines to another. The entities defined are products, production centers, jobs and operations. The events are sales, end of processes and periodic reports.

The program is not difficult to follow if certain basic SIMSCRIPT II conventions are understood. These are:

- Terminal periods in all words are ignored.
- All statements start with a key word.
- Comments are preceded by ''.
- System functions end in .F, system variables in .V.
- The DEFINE TO MEAN statement performs word substitutions during the lexical scan of the compiler.
- Some preamble statements are purely definitional, e.g., EVERY, DEFINE, BREAK TIES; others generate executable programs, e.g., BEFORE, TALLY, ACCUMULATE.

Attributes need not have their subscripts explicitly stated, if not stated they are implied, e.g., VALUE implies VALUE(JOB).

Status Report

SIMSCRIPT II has been implemented on the IBM 360 family of computers. At present, its compiler runs under OS/360 and requires at least 200K bytes of core. The compiler is written entirely in SIMSCRIPT II; the support programs that run along with compiled SIMSCRIPT II programs performing input-output and interfacing with the operating system are in the main written in assembly language.

The generality of the SIMSCRIPT II language and the straightforwardness of the syntax-directed compiling scheme we have chosen keeps our compiler from being "fast". Work is underway to improve its performance. However, its error detection and correction and forced execution features make it necessary to compile fewer programs than one has to in FORTRAN or PL/I, as one can debug with fewer accesses to the computer.

Execution performance is quite good, as attention is paid to the generation of efficient code.

REFERENCES

1. Kiviat, P. J., R. Villanueva, "The SIMSCRIPT II Programming Language", The RAND Corporation, R-460-PR, 1968.

2. Markowitz, H. M., B. Hausner and H. W. Karr, SIMSCRIPT: A Simulation Programming Language, Englewood Cliffs, N.J., Prentice-Hall, 1963.

3. Kiviat, P. J., "Introduction to the SIMSCRIPT II Programming Language", P-3314, The RAND Corporation, February 1966.

4. Balzer, R. M., "Dataless Programming", Proceedings of the 1967 Fall Joint Computer Conference, Thompson Books, Washington, D.C., 1967.

5. Barron, D. W., J. N. Buxton, D. F. Hartley, E. Nixon, and C. Strachey, "The Main Features of CPL, The Computer Journal, Vol. 6, No. 2, July 1963.

6. Dahl, O. J. and K. Nygaard, "SIMULA - An ALGOL-Based Simulation Language", Communications of the ACM, IX, September 1966.

7. Blunden, G. P. and H. S. Krasnow, "The Process Concept as a Basis for Simulation Modeling", SIMULATION, Vol. 9, No. 2, August 1967.

8. Kiviat, P. J. "Simulation Programming Using SIMSCRIPT II", P-3861, The RAND Corporation, 1968.

9. McNeley, J. L., "Compound Declarations", Paper presented at the 1967 IFIP Working Conference on Simulation Languages, Oslo, Norway.

SAMPLE SIMSCRIPT II SIMULATION PROGRAM
A JOB SHOP SIMULATION

```
PREAMBLE
NORMALLY MODE IS INTEGER AND DIMENSION IS 0

PERMANENT ENTITIES....
     EVERY PRODUCT HAS A SALES.FREQUENCY AND A NAME AND OWNS A STRUCTURE
          DEFINE SALES.FREQUENCY AS A REAL RANDOM LINEAR VARIABLE
          DEFINE NAME AS AN ALPHA VARIABLE
     EVERY PRODUCT,PRODUCT HAS A PRODUCT.SALES(*/2)
     EVERY PRODUCTION.CENTER HAS A (MAX.IN.QUEUE(1/2), MAX.QUEUE(2/2)) IN ARRAY 1
        A (WNUM(1/2), MNUM(2/2)) IN ARRAY 2, A WSUM, A MSUM, A NUMBER.IDLE
        AND OWNS A QUEUE
          DEFINE NUMBER.IDLE AS A VARIABLE MONITORED ON THE LEFT

TEMPORARY ENTITIES....
     EVERY JOB HAS A VALUE IN WORD 2, A DUE.DATE, AN ARRIVAL.TIME,
        AN EXPEDITE.FACTOR.FUNCTION, MAY BELONG TO A QUEUE, OWNS A ROUTING
        AND MAY BELONG TO A WAITING.SET
          DEFINE EXPEDITE.FACTOR AS A REAL FUNCTION
          DEFINE VALUE, DUE.DATE AND ARRIVAL.TIME AS REAL VARIABLES
          DEFINE ROUTING AS A FIFO SET WITHOUT P AND N ATTRIBUTES
          DEFINE QUEUE AS A SET RANKED BY HIGH VALUE
     EVERY OPERATION HAS A (CODE(1/2) MACHINE.DESTINED(2/2)) IN WORD 1
        AND A PROCESS.TIME AND BELONGS TO A STRUCTURE AND A ROUTING
          DEFINE STRUCTURE AS A SET RANKED BY LOW CODE WITHOUT M ATTRIBUTE
                 AND WITHOUT R ROUTINES
          DEFINE PROCESS.TIME AS A REAL VARIABLE

EVENT NOTICES INCLUDE WEEKLY.REPORT
     EVERY SALE HAS A PRODUCT.TYPE, A PRICE AND A PRIORITY
          DEFINE PRICE AS A REAL VARIABLE
     EVERY END.OF.PROCESS HAS AN ITEM AND A PRODUCER

BREAK SALE TIES BY HIGH PRICE THEN BY LOW PRIORITY
EXTERNAL EVENTS ARE END.OF.SIMULATION AND SALE
EXTERNAL EVENT UNITS ARE LOCAL.SALES AND IMPORT.SALES
PRIORITY ORDER IS END.OF.PROCESS, SALE, WEEKLY.REPORT AND END.OF.SIMULATION

BEFORE FILING AND REMOVING FROM QUEUE CALL QUEUE.CHECK
BEFORE DESTROYING JOB, CALL STAY.TIME
       DEFINE STAY AS A REAL DUMMY VARIABLE
TALLY AVG.STAY AS THE WEEKLY MEAN, VAR.STAY AS THE WEEKLY VARIANCE, SUM.STAY AS
         THE WEEKLY SUM, SUM.SQUARES.STAY AS THE WEEKLY SUM.OF.SQUARES, AND
         NUM.STAY AS THE WEEKLY NUMBER OF STAY
ACCUMULATE WSUM AS THE WEEKLY SUM, WNUM AS THE WEEKLY NUMBER, AVG.QUEUE AS THE
         WEEKLY MEAN, MAX.QUEUE AS THE WEEKLY MAXIMUM AND FREQ(0 TO 25 BY 1)
         AS THE WEEKLY HISTOGRAM OF N.QUEUE
ACCUMULATE MSUM AS THE MONTHLY SUM, WNUM AS THE MONTHLY NUMBER, AVG.IN.QUEUE AS
         THE MONTHLY MEAN, MAX.IN.QUEUE AS THE MONTHLY MAXIMUM OF N.QUEUE

THE SYSTEM OWNS A FINISHED.GOODS.INVENTORY
DEFINE FINISHED.GOODS.INVENTORY AS A SET RANKED BY DUE.DATE
DEFINE LOCAL TO MEAN DEFINE I,J,K,L,M AND N AS SAVED INTEGER VARIABLES
DEFINE WEEK TO MEAN *HOURS.V*7 HOURS
DEFINE PRIORITY.FREQUENCY AS A 2-DIMENSIONAL ARRAY
DEFINE TITLE AS A TEXT VARIABLE
DEFINE WEEK.COUNTER AND TAPE.FLAG AS INTEGER VARIABLES
DEFINE AVERAGE AS A REAL FUNCTION WITH 1 ARGUMENT
END
```

```
EVENT SALE(PRODUCT,PRICE,PRIORITY) SAVING THE EVENT NOTICE
DEFINE SF TO MEAN SALES.FREQUENCY
LOCAL
IF SALE IS EXTERNAL, READ PRODUCT, PRICE AND PRIORITY AS B 30,I 5, D(10,3), I 5
REGARDLESS ADD 1 TO PRODUCT.SALES(PRODUCT, TRUNC.F(PRICE)+1)
CREATE A JOB
     LET VALUE=PRICE
     LET DUE.DATE=TIME.V + PRICE + PRIORITY
     LET ARRIVAL.TIME=TIME.V
IF SALE IS INTERNAL,
     FOR EACH PIECE OF STRUCTURE, FILE PIECE IN ROUTING    GO TO JOB
''PROCESS SPECIAL ORDERS
OTHERWISE UNTIL MODE IS ALPHA, DO THE FOLLOWING....
          READ N
          FOR EACH PIECE IN STRUCTURE WITH CODE(PIECE) = N,
          FIND THE FIRST CASE, IF NONE GO TO LOOP
          FILE PIECE IN ROUTING
'LOOP'      LOOP
'JOB' NOW ATTEND.TO.JOB
IF SALE IS EXTERNAL, DESTROY THE SALE    RETURN
OTHERWISE....
     SCHEDULE THE SALE(PRODUCT, PRODUCT*RANDOM.F(1), PRIORITY.FREQUENCY(PRODUCT,
          TRUNC.F(PRICE+1)) IN SF HOURS
RETURN    END
```

```
EVENT FOR WEEKLY.REPORT SAVING THE EVENT NOTICE
RESCHEDULE THIS WEEKLY.REPORT IN 1 WEEK
ADD 1 TO WEEK.COUNTER
NOW REPORT
RESET WEEKLY TOTALS OF STAY
FOR EACH PRODUCTION.CENTER, RESET WEEKLY TOTALS OF N.QUEUE
OF MOD.F(WEEK.COUNTER.4)=0, FOR EACH PRODUCTION.CENTER, RESET MONTHLY TOTALS
     OF N.QUEUE    ELSE
RETURN    END
```

```
ROUTINE TO REPORT
START NEW PAGE
BEGIN REPORT
BEGIN HEADING
     IF PAGE IS FIRST, PRINT 1 LINE AS FOLLOWS
         J O B  S H O P  S I M U L A T I O N
        SKIP 2 OUTPUT LINES
     ELSE PRINT 2 LINES AS FOLLOWS
         PRODUCTION CENTER QUEUEING REPORT
            CNTR  AVG.QUEUE  MAX.QUEUE
END HEADING
FOR EACH PRODUCTION.CENTER, PRINT 1 LINE WITH PRODUCTION.CENTER,
     AVG.IN.QUEUE AND MAX.IN.QUEUE THUS
          **     **.*        **
END REPORT
END
```

SIGNIFICANT FEATURES

LEVELS 1 - 3

C	FULL DYNAMIC STORAGE ALLOCATION
	RESERVE - RELEASE ARRAYS, PROGRAMS
	NON-RECTANGULAR ARRAYS
C	RECURSION
	FREE FORM INPUT-OUTPUT
C	NON-RECORD ORIENTED I/O
	INTERNAL BUFFER
	REPORT GENERATION
	PREAMBLE DECLARATIONS
	LANGUAGE STYLE
C	MUCH USE IN COMPILER

SIGNIFICANT FEATURES

LEVEL 4

C	GREATER CONTROL OF STORAGE THAN SIMSCRIPT
	COMPOUND ENTITIES
	SYSTEM ATTRIBUTES AND SETS
C	MORE COMPLETE SET PROCESSING
C	MORE CONTROL OVER PROGRAM GENERATION
	IMPLIED SUBSCRIPTS } DATALESS PROGRAMMING
	LEFT-HANDED FUNCTIONS }
	MONITORED VARIABLES
	SUBPROGRAM MODE
	TEXT MODE

SIGNIFICANT FEATURES

LEVEL 5

EXTENDED TIME REPRESENTATION
MULTIPLE EVENT TAPES
EVENT PRIORITIES
"COMMON" EVENTS
STATISTICAL FUNCTIONS AND FEATURES
BEFORE AND AFTER
AUTOMATIC ACCUMULATION