A GENERAL PURPOSE TOOL FOR INTERACTIVE SIMULATIONS

Frank Scarpino  
Air Force Avionics Laboratory

Joe Clema  
General Dynamics  
Fort Worth

ABSTRACT

A framework and methodology from which a set of forty-seven Fortran subprograms (the SCENARIO) have been developed may be used for the setup of any modularly developed Fortran simulation. The SCENARIO provides a natural and convenient interface between the human and the computer. The principles, requirements, programming, implementation, and use are described. The full system runs on a DEC SYSTEM-10 computer located in the Air Force Avionics Laboratory at Dayton, Ohio. A scaled-down version of the system is in operation on a CDC 6600 in the General Dynamics Division at Fort Worth, Texas.

INTRODUCTION

"The future growth of the computer industry and the acceptance of computer methods will depend largely on the successful establishment of effective man-machine communications."

James Martin, 1973

In the design, development, and implementation of large-scale, complex computer simulations, the man-machine interface has often been relegated by default to clumsy I/O methods, such as cards and the restrictive formats of cards.

It is generally possible to develop powerful man-machine timesharing interfaces on modern medium and large-scale computers. Displays, editing facilities, diagnostic messages, library capabilities, and file manipulations must be provided in these interfaces. The I/O man-machine interface need not be all things to all users for all time but should provide those capabilities common to most users and should provide facilities for the easy addition of capabilities for the individual user.

Some people feel little progress will be made in significantly increasing the tools and techniques for "upgrading of the quality of software production" (Ref. 2) during the next few years; however, improvement of the man-machine interface not only increases the usability of the simulation software but increases the capacity for developing, integrating, debugging, modifying, and maintaining the system. Certain features to be described also improve the testing, reliability, and generality of the man-machine interface. The interactive system should significantly speed the development of a new set of application models.

The software has been written for a DEC SYSTEM-10 in Fortran Extended. The application software is composed of airplane models, flight control system, sensors, environment models, and support software. These application models provide a real-time (or non-real-time) Avionic Simulation (AVSIM) for the United States Air Force so that Air Force engineers can synthesize, integrate, test, and evaluate total avionic systems. The SCENARIO software enables the user to choose the desired models; for instance, he might choose models that simulate an A-7 or the new F-16 Lightweight Fighter. Models of varying fidelity exist. Thus, if an engineer is interested in studying the Doppler radar model, he might choose the detailed Doppler model and a simple airframe and flight control system. An engineer interested in the attack radar could choose a simple Doppler model or no Doppler model at all. The engineer, in less than two minutes clock time, can inter-

*This work was supported by the Air Force Avionics Laboratory, Wright-Patterson AFB, under Contract F33615-73-C-1208.
actively establish his configuration for a simulator run. The software used in a real-time simulation (emulation of the actual time during which a physical process occurs in the real world) in this application is composed of several assembly language routines. All software, including the Assembly language routines, should be easily portable to other large-scale and medium-scale machines with less than six man-months work needed to implement the total system on a different computer. The non-real-time software could be converted in one to two man-months.

The setup software has been named the SCENARIO (see Figure 1), and it has the properties described in Table 1. The SCENARIO is composed of a main body and a PRESCENARIO in which is made of a DEC-10 "Command Stream". This command stream is not essential to the user in setting up a run but should be easily implemented on most third-generation systems. The PRESCENARIO portion of the setup permits the user to define the particular application models to be run. This stage will be defined in more detail in the following topics. The only severe limitation is that the application models must, as would be expected, share their own set of COMMON Blocks with the SCENARIO COMMON Blocks. Any duplication of variable names contained in COMMON must be eliminated when the SCENARIO is applied to a new set of application models.

**REQUIREMENTS**

"Increasingly in the next decade, man must become the prime focus of system design."

James Martin, 1973

With the advent of conversational computing available on large-scale digital hardware, many advantages of running simulations in an interactive environment may be enjoyed by the user. The interface between the human and his program may be divided into four stages: (1) setup, (2) verification, (3) dynamic execution, and (4) post-run analysis. A general-purpose system would provide interactive capabilities for each of these stages while providing versatility, flexibility, and efficiency. Many considerations, evaluations, and trade-offs are necessary in the development of suitable software that performs the necessary user functions. Reference 3 in the Bibliography provides a detailed discussion of the considerations and requirements used for the development of interactive simulations. Those applicable to the SCENARIO are defined in Figure 2.

Emphasis has been placed on human factors and usability by Air Force engineers. In the design, development, and implementation stages, the following requirements were constantly followed: (1) operator inputs should be minimal and concise; (2) software should be modular, structured, and well documented; (3) software should be amenable to growth; (4) the system should provide rapid response to operator inputs; (5) the system should be fool-proof. While the first four items are relatively easy to quantify, item five involves anticipation of every conceivable error a user might make. If ample time and testing are allowed, item five may be satisfied for most situations (Ref. 12).

<table>
<thead>
<tr>
<th>TABLE 1 SCENARIO MODEL FEATURES</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>KEY PROGRAM FOR SIMULATOR SETUP AND INITIALIZATION</strong></td>
</tr>
<tr>
<td>00 REAL-TIME OR NON-REAL-TIME OPERATIONS</td>
</tr>
<tr>
<td>00 DEFAULTS FOR ALL MODES, PARAMETERS, VARIABLES, AND OPTIONS</td>
</tr>
<tr>
<td>00 NO OPERATOR INPUTS REQUIRED (OTHER THAN &quot;RUN&quot;) UNLESS OPERATOR DESIRES TO MODIFY DEFAULTS: EXAMPLE OPTIONS</td>
</tr>
<tr>
<td>000 DATA &quot;STIMULATION&quot;, SELF-CONTAINED</td>
</tr>
<tr>
<td>000 REAL-TIME, NON-REAL-TIME</td>
</tr>
<tr>
<td>000 MAN-IN-LOOP, NO MAN-IN-LOOP</td>
</tr>
<tr>
<td>000 AIRFRAME (F-10, A-7), SIMPLE, COMPLEX</td>
</tr>
<tr>
<td>000 SENSOR MODES</td>
</tr>
<tr>
<td>000 TARGETS</td>
</tr>
<tr>
<td>000 DATA ACQUISITION ANALYSIS OPTIONS</td>
</tr>
<tr>
<td>000 ASSISTANCE OF OPERATOR DURING SETUP IF REQUIRED</td>
</tr>
<tr>
<td>000 TUTOR NODE PROVIDES DETAILED PROMPTING MESSAGES FOR A NEW OPERATOR</td>
</tr>
<tr>
<td>000 PROVISION OF PROMPTING INFORMATION ONLY WHEN EXPERIENCED OPERATOR REQUESTS IT</td>
</tr>
<tr>
<td>1. SIMULATOR CONFIGURATION AND MODEL DESCRIPTIONS</td>
</tr>
<tr>
<td>2. PARAMETER-MODEOPTION DESCRIPTION AND VALUE</td>
</tr>
<tr>
<td>3. ECHO-RECORD ON DISK OPERATOR INPUTS</td>
</tr>
<tr>
<td>4. EXAMPLES AND DESCRIPTIONS OF COMMANDS</td>
</tr>
<tr>
<td>000 AUTOMATIC DISPLAY OF ERRORS AND PROVISION OF INFORMATIVE DIAGNOSTICS</td>
</tr>
</tbody>
</table>

**IMPLEMENTATION**

"The default technique can ordinarily be expected to reduce input errors based on the simple idea that what you don't do you can't do wrong."

Tom Gilb and Gerald Weinberg, 1976
Considered of primary importance during the design of the setup software was the desire to humanize the man-machine interface. Thus, the human should have to perform as little I/O as possible, and all inputs should be edited. Informative diagnostics should be provided when errors are detected, and displays and prompting messages should guide the user rapidly through a setup. Defaults are furnished for all parameters, modes, and variables to accomplish the limited-input requirement. In addition, defaults diminish the number of user errors.

Default values are provided by use of BLOCK DATA or through initialization procedures in which ENTRY POINTS are used in each application model (see Figure 3 and Ref. 5, 6, 7, 8, and 9). Each application model contains two other ENTRY POINTS in addition to the initialization ENTRY POINTS. These three code segments interface with the...
SCENARIO and provide the man-machine inter-
action. These segments, portrayed in Fig-
ure 3 are described as (1) "IN", (2) "TT", 
and (3) "EX" routines, which are labels for 
the (1) initialization segment, (2) tele-
type (CRT) man-machine interface segment, 
and (3) execution segment. These three 
segments are automatically built for each 
model through the use of the PRESCEN-
ARIO module. After a user establishes those 
parameters (usually 20 or 30 parameters) in 
an application model that contains para-
eters he desires to be able to change 
through the SCENARIO, approximately 15 
minutes in the time-sharing mode will be 
taken to build the interface by use of the 
PRESCENARIO. Prompting messages enable the 
ewner to develop easily his interface 
with the software.

The initialization ("IN") segment ini-
tializes any local or global variables 
specified by the user when he built his 
SCENARIO interface. The man-machine inter-
face ("TT") segment provides the editing, 
error checks, diagnostics, and displays 
which describe commands, parameters, modes, 
variables, range, and bounds of those items 
that the user may change interactively. 
Finally, "EX" routines execute any para-
eters dependent upon operator inputs 
during the simulator setup. All calls to 
these ENTRY POINTS are performed by the 
SCENARIO modules, and their execution is 
invisible to the user.

As seen in Figure 1, the SCENARIO structure 
itis highly modular. This structure facil-
itated development, integration, debugging, 
and maintenance. Approximately 14K 36-bit 
words more than the real-time application 
models are required to run the SCENARIO, 
as implemented at AFAL. The core varies 
as a function of the number of models and 
the number of parameters that may be modi-
ified. Most of the SCENARIO code is rolled 
out of core through use of overlays during 
application model execution.

The SCENARIO editing features enable the 
user to enter his inputs in any Fortran 
mode. If the variable IRNADC is to be ini-
tialized to the value 7, the user may input 
7, 7.0, 7.0, +.7E+1 as they are equivalent 
SCENARIO inputs. The format of the inputs 
is not a problem for the user as he does 
not need to worry whether a parameter is 
INTEGER, REAL, DOUBLE PRECISION, etc. The 
SCENARIO software handles any required con-
version.

CONFIGURATION SETUP

"Although computer technology 
hasevolved rapidly over the 
past 30 years, human/computer 
interfaces are still inefficient, 
clumsy, and generally quite 
unsuitable by the non-computer 
expert."

Harold Sackman, 1970

In the development and use of the applica-
tion models, it is desirable to have con-
trol software that provides flexibility 
and ease of use. Three key modules in 
this control group are (1) the EXEC, (2) 
the DIRECTOR, and (3) CALIPER (see Figure 
1). The EXEC is a Fortran subroutine 
that enables the user to specify easily 
the application models he desires to run, 
and the rate groups he desires to run in 
(see Fig. 4).

EXECUTIVE (EXEC)

The real-time application models may be 
specified to run in any one of seven rate 
groups called A group through G group. 
The user needs to do three things in devel-
opment the configuration he desires to run: 
(1) specify the applications models in a 
Fortran EXTERNAL type of statement; (2) 
name the models in the proper position in 
the CALL DIR statement; and (3) specify 
the number of models in each of the seven 
rate groups.

Assume the following models are to be run:
A group (64 times/sec) consisting of FD16, 
TARG, CCVFC, RADAR models; B group (32 
times/sec): COUP, FGSIS, PROJ, ANALY, 
FCS16 models; C group (16 times/sec): AIRIG, 
BOMB, ADC models; D group (8 times/sec): 
ATMO model; E group (4 times/sec): GRAV 
model; F group (2 times/sec): no models; 
G group (1 time/sec): GUST model. Then, 
Figure 5 is an example of the EXEC state-
ments that would be placed in the sub-
routine to create this desired config-
uration.

The models named in the CALL DIR state-
ment are automatically loaded from system 
libraries and linked. The Air Force 
maintains two airplane libraries at the 
Air Force Avionics Laboratory and is able 
set up an A-7 or an F-16 run dependent 
on the application models specified in 
the CALL to the DIRECTOR.
Figure 3  APPLICATION MODEL ENTRY POINTS

Figure 4  TIME SLICE SEQUENTIAL SCHEDULER
Central Purpose Tool...Continued

SUBROUTINE EXEC

EXTERNAL RADAR, ANALY, BOMB, GUST, FC16,
ADC, FD16, GRAV, ATMS, TANG, COUP,
CCVFC, AIRTG, FC35, PROJ

THE FOLLOWING PARAMETERS ARE THE NUMBER
OF MODELS IN EACH RATE GROUP.

RA = 4
RB = 5
NC = 3
ND = 1
NE = 1
NF = 0
NO = 1

CALL DIR (FD16, TANG, CCVFC, RADAR, A5, A6, A7, A8, A9, A10,
COUP, FC16, PROJ, ANALY, FC35, B6, B7, B8, B9, B10,
AIRTG, BOMB, ADC, C4, C5, C6, C7, C8, C9, C10,
ATMS, D2, D3, D4, D5, D6, D7, D8, D9, D10,
GRAV, E2, E3, E4, E5, E6, E7, E8, E9, E10,
F1, F2, F3, F4, F5, F6, F7, F8, F9, F10,
GUST, G2, G3, G4, G5, G6, G7, G8, G9, G10)

Figure 5 SAMPLE EXEC STATEMENTS

DIRECTOR (DIR)

The DIRECTOR provides software that monitors the execution of the application model. Subroutine DIR interfaces with the operator, the EXEC, the SCENARIO, subroutine CALP, and the DECSYSTEM real-time operating system. The DIRECTOR passes the application names to CALP and provides the operator with the capability of putting the simulator in (1) hold, (2) reset, or (3) execution.

The operator desires to be able to perform various functions during each of the stages of a simulator run. Figure 6 summarizes these functions for each phase, including the real-time execution stage controlled by the DIRECTOR.

CALIPER (CALP)

Subroutine CALP provides the software that sequentially calls the application models. The Fortran EXTERNAL statement provides the facility for configuring a simulator run in the EXEC and, then, passes the application models' names through the DIRECTOR to CALIPER. CALIPER calls the models on the basis of their specified frequency (see Fig. 4). The frame time (major cycle) (Ref. 14, 16) is a parameter that may be modified through the SCENARIO before run-time. If the operator specifies a frame size of one, the fastest rate group (the A group) will execute 64 times/sec, the B group 32 times/sec, and so forth to the slowest rate group (the G group), which would execute once per second. If the operator had specified a frame of two seconds for the major cycle, the A group would in reality be executing 32 times/sec, etc. When the frame is set at 1.28, the A group executes 50 times/sec, the B group at 25 times/sec, and so on.

The CALIPER routine with variable frame size provides flexibility for both real-time and non-real-time runs. In most real-world applications, all models do not need to execute at the same frequency. By use of the control system that has been described, it is possible to execute models at various frequencies and to emulate the real-world requirements of the models.

AVSIM COMMANDS

Operator commands provide many higher level program capabilities (see Ref. 14, 15, 16 for a detailed description). In Table 2 and Figure 7 are described the commands available during setup. Several of the more useful commands will be described under this topic. Operator displays are available by means of the HELP command, which produces a description of all AVSIM commands and how they are used. In addition to this display, there is the MODELS command, which causes descriptions of all models that the user has configured in his run to be listed. Another display available through the LIST command produces a description of all models and all parameters available to the user through the SCENARIO so that they can be changed. All parameters may be displayed by typing the name of the model. A single parameter is described when the user types the model name and the parameter name. The model name must be typed along with the parameter name to facilitate the addition of new models, which may contain local variable names that are the same as existing parameter names. A simple modification enables naming the desired parameter to be described with the restriction that the variables to be employed in the SCENARIO setup not have multiple meanings.

For example, assume a model named TARGET is to be added to the AVSIM Library. The COMMON Blocks are modified to correspond to the AVSIM COMMON Blocks. The PRESCENARIO software is used in building the three ENTRY points "IN", "TT", and "EX". The RADAR model is then added to the AVSIM Library. The command RADAR would then produce a list

480 December 6 - 8 1976
Figure 6 FOUR STAGES OF SIMULATOR USAGE

Figure 7 SIMULATOR COMMANDS

<table>
<thead>
<tr>
<th>TABLE 2 SIMULATOR COMMANDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEY</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>15</td>
</tr>
<tr>
<td>16</td>
</tr>
<tr>
<td>17</td>
</tr>
<tr>
<td>18</td>
</tr>
<tr>
<td>19</td>
</tr>
<tr>
<td>20</td>
</tr>
<tr>
<td>21</td>
</tr>
<tr>
<td>22</td>
</tr>
<tr>
<td>23</td>
</tr>
<tr>
<td>24</td>
</tr>
<tr>
<td>25</td>
</tr>
<tr>
<td>26</td>
</tr>
<tr>
<td>NUMBER</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
</tbody>
</table>

Figure 8 OUTPUT FROM RADAR COMMAND

of information about this model and a description of all parameters the user can modify at setup, as the user specified during his PRESCENARIO construction (see Fig. 8).

The command RADAR/ASRCE will produce the following output:

```
ASRCE RANGE 1-2; ANTENNA POINTING SOURCE SELECTION VARIABLE
CURRENT VALUE IS 1.
```

To enter a new value for the parameter ASRCE, the user must type the following command:

```
RADAR/ASRCE/2.
```

In addition to these displays, there is a Summary Report, which is automatically printed at run time. The Summary Report contains a list of all critical information in the setup and a record of the values of SCENARIO parameters and information about each model.

Any two characters or more are all that are necessary to input the higher level commands. The commands BE, BEG, and BEGIN initiate an AVSIM run. Multiple commands can be used for the same purpose. EXECUTE, START, GO, and RUN perform the same function as BEGIN.

Fortran logical unit numbers are handled through the use of parameters defined before execution or during run time (see Fig. 9 and Table 3). By the use of one parameter for all CRT operator inputs and a second parameter for all CRT operator outputs, different logical devices may be assigned for simulator inputs and outputs. Two capabilities are immediately provided: (1) the operator is capable of initiating a run from multiple input devices (tape, disk, cards, CRT) and of sending outputs to multiple output devices (tape, disk, CRT); (2) operator inputs may be echoed to the CRT and recorded on tape or disk for later use. Thus, even though a large number of parameters are to be modified in a large number of runs, the default values on the Master File need not be changed. The operator records his inputs on a disk file.
TABLE 3 SIMULATOR I/O FILES

<table>
<thead>
<tr>
<th>UNIT NO.</th>
<th>PORT VAR NAME</th>
<th>FILE NAME</th>
<th>DESCRIPTION</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NCRT</td>
<td>AVSIM.DAT</td>
<td>INPUT UNIT FOR TAPE/DISK INPUT</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>NCRT</td>
<td>AVSIM.SAM</td>
<td>INPUT FILE FOR SETTING UP MONI PRINTOUT TIMES</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>NPRINT</td>
<td>AVSIM.LPT</td>
<td>OUTPUT FILE FOR LINE PRINTER</td>
<td></td>
</tr>
</tbody>
</table>
| 4        | NDISKI        | AVSIM.REC | OUTPUT FILE FOR RECORDING OPERA-
TOR INPUTS (HNSKWT MUST BE ON) |          |
| 5        | NCRT          |           | TELETYPING INPUTS |          |
| 6        | NCRT          |           | TELETYPING OUTPUTS |          |
| 9        | NCRT          | TESTER.TST | AVSIM SELF-TEST INPUT FILE |          |
| 10       | NCRT          | AVSIM.TST | OUTPUT SELF-TEST FILE |          |
| 11       | LOG           | AVSIM.LOG | RECORDS DATA DURING REAL-TIME RUN( NOT YET IMPLEMENTED) |          |

Figure 9 SIMULATOR I/O FILES

named AVSIM.REC (Fig. 9).

When the simulator is next run, the operator initiates the run with the input values from this disk file. After this initiation from the disk, inputs are again read from the CRT; thus, the operator still has the option of modifying any or all SCENARIO parameters before the simulator run. Files (Fig. 9) are provided for automated self-

test, line printer output, parameters to be operator-monitored during the real-time run, and a file for real-time data logging.

SUMMARY

"To a certain extent, our problems in furthering man-machine communica-
tions arise from our impatience, rather than basic technological weaknesses."

Charles T. Meadow, 1970

In the rush to implement a major system, the emphasis has generally been placed on the application with little consideration given to human factors when the interface between the man and his program is implemented. In large, complex simulations, there exist many instances when human intervention is desirable, as described earlier. Particularly at setup time, there are many displays, options, and parameters, which a user will desire to control. If he is an experienced user of the system, he does not want to be forced into cycling through all the options, etc. The user desires only enough prompting information to change those options necessary to setup and execute his run. The SCENARIO is a set of software control modules that have such a structure. It provides a high degree of effective cooperation between the man and his application models by displaying only enough dialogue for the user to initiate any desired modifications. The user inputs commands to which the SCENARIO responds. The structure is expandable and easily adaptable to individual needs.

No matter how well a system performs an application, it will be little used if it is difficult to setup. Systems of the future will be designed from the outside to the inside; the user will be considered first and effective man-machine dialogue will become a major consideration as important as the application itself.


