LAUNCH VEHICLE OPERATIONS SIMULATOR

John W. Blackledge
IBM Corporation, Federal Systems Division
Kennedy Space Center, Florida 32920

Prepared for Presentation
to the
1974 Automatic Support Systems
Symposium for Advanced
Maintainability, October 29-31,
San Diego, California
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASTP</td>
<td>Apollo-Soyuz Test Project</td>
</tr>
<tr>
<td>ATOLL</td>
<td>Acceptance, Test, or Launch Language</td>
</tr>
<tr>
<td>CEXEC</td>
<td>Continuous Model Executive</td>
</tr>
<tr>
<td>COMMON</td>
<td>Cathode Ray Tube</td>
</tr>
<tr>
<td>CRT</td>
<td>Cathode Ray Tube</td>
</tr>
<tr>
<td>DDAS</td>
<td>Discrete Model Executive</td>
</tr>
<tr>
<td>DEXEC</td>
<td>Discrete Model Executive</td>
</tr>
<tr>
<td>DI</td>
<td></td>
</tr>
<tr>
<td>DO</td>
<td></td>
</tr>
<tr>
<td>FORTRAN</td>
<td></td>
</tr>
<tr>
<td>GOAL</td>
<td>Ground Operations Aerospace Language</td>
</tr>
<tr>
<td>GSE</td>
<td>Ground Support Equipment</td>
</tr>
<tr>
<td>I/O</td>
<td>Input/Output</td>
</tr>
<tr>
<td>IU</td>
<td>Instrument Unit</td>
</tr>
<tr>
<td>IVB</td>
<td>IV B</td>
</tr>
<tr>
<td>KSC</td>
<td>Kennedy Space Center</td>
</tr>
<tr>
<td>LCC</td>
<td>Launch Control Center</td>
</tr>
<tr>
<td>LSE</td>
<td></td>
</tr>
<tr>
<td>LVDC</td>
<td>Launch Vehicle Digital Computer</td>
</tr>
<tr>
<td>LVOS</td>
<td>Launch Vehicle Operations Simulator</td>
</tr>
<tr>
<td>ML</td>
<td>Mobile Launcher</td>
</tr>
<tr>
<td>NASA</td>
<td>National Aeronautics and Space Administration</td>
</tr>
<tr>
<td>SCALE/FORTTRAN</td>
<td></td>
</tr>
<tr>
<td>SGOS</td>
<td>Shuttle Ground Operations Simulator</td>
</tr>
<tr>
<td>SIB</td>
<td>Saturn I B</td>
</tr>
<tr>
<td>SIU</td>
<td>Saturn Instrument Unit</td>
</tr>
<tr>
<td>SIVB</td>
<td>Saturn IV B</td>
</tr>
<tr>
<td>SLCC</td>
<td>Saturn Launch Control Complex</td>
</tr>
<tr>
<td>SYSGEN</td>
<td>System Generator</td>
</tr>
<tr>
<td>VAB</td>
<td>Vertical Assembly Building</td>
</tr>
<tr>
<td>XREF</td>
<td>Cross Reference</td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS

Abstract ................................................................. 1
Introduction ............................................................. 1
The Saturn Launch Control Complex (SLCC) ......................... 1
LVOS System Description ............................................. 1
LVOS Software ......................................................... 2
Modeling Language ..................................................... 3
Model Execution ....................................................... 5
The ASTP (Saturn IB) System Model .................................. 5
Conclusion ..................................................................... 5
References ..................................................................... 6

LIST OF ILLUSTRATIONS

Figure ................................................................. Page
1 Saturn Launch Vehicle Checkout System ......................... 2
2 Launch Vehicle Operations Simulation ....................... 3
Abstract

The Saturn Launch Vehicle Operations Simulator (LVOS) simulates the hardware operations of the Saturn vehicle and ground equipment. The LVOS math model contains approximately 10,000 equations written by NASA and contractor engineers. The models, representing the Saturn stages and ground support equipment, are compiled by a Boolean equation compiler (using three dimensional logic) and a modified FORTRAN IV compiler. This higher level language enables engineers to code models directly without having to rely on programmers to translate models to assembly language.

The simulator executive system responds to almost 1400 switch actions and computer commands originating in the firing room at Launch Complex 39 at Kennedy Space Center. The model responses include 3000 discrete and 1200 analog functions. A fast, compact matrix execution algorithm is used for Boolean logic equations. Continuous model equations are grouped into FORTRAN subroutines and executed as independent subroutines.

Introduction

The Saturn Launch Vehicle Operations Simulator (LVOS) was developed for NASA at the Kennedy Space Center. LVOS simulates the Saturn launch vehicle and its ground support equipment. The simulator was intended to be used primarily as a launch crew trainer; but is also being used for test procedures and software validation. A NASA/contractor team of engineers and programmers implemented the simulator after the Apollo XI lunar landing during the low activity periods between launches.

The Saturn Launch Control Complex (SLCC)

Complex 39 at the Kennedy Space Center (KSC) consists of two launch pads, fuel farms, the Vertical Assembly Building (VAB), the Launch Control Center (LCC) and three mobile launchers. These facilities are used to erect, integrate, test, and launch Saturn rockets. One mobile launcher (ML) has been equipped with a pedestal to accommodate the shorter Saturn IB rocket.

The LCC at Complex 39 is equipped with three complete firing rooms. Each firing room can interface with any of the three mobile launchers. A mobile launcher may be parked in any of the three high-bay areas of the VAB or on either launch pad.

In each firing room there are over 100 control consoles and a three computer complex which are used to test and launch vehicles. Commands originating in the firing room from switch actions, or computerized test procedures, travel over 5 miles to the mobile launcher computer via a hardline data link. The mobile launcher computer stimulates the vehicle and ground support with these commands. System responses return to the firing room via the computer system and five 72 kilobit telemetry systems. In the firing room, responses are monitored on strip charts, console lights and meters, printer outputs, and on computer controlled CRTs. Figure 1 is a block diagram of the Saturn launch vehicle checkout system.

Testing and launch operations are performed by NASA/contractors engineers and technicians from the LCC firing rooms. These operations are performed according to predefined test procedures. In the early phases of the program most tests were a sequence of manual actions with a few automatic test programs. As the program matured, engineers began developing automatic test procedures using ATOLL (Acceptance, Test, Or Launch Language). The Apollo XVII launch operations were controlled by almost 150 automatically linked ATOLL test procedures. These procedures perform a variety of functions such as stage power up, engine testing, propellant monitoring, and emergency detection system testing.

LVOS System Description

The LVOS system replaces the mobile launcher, launch vehicle, and ground support equipment with the capability of responding to commands and test procedures executed in the LCC firing rooms. A laboratory computer is used in place of the mobile launcher computer to provide easier operation and avoid modifying operational hardware.
The model responses include 3000 discrete functions and 1200 analog measurements. The Launch Vehicle Digital Computer (LVDC) functions are simulated to respond in the same manner as the preflight software. No attempt was made to simulate post time (after liftoff).

LVOS Software

The LVOS software system is composed of six programs. Each of the programs contain its own utility and I/O support routines and can be loaded and executed independently. The programs are:

1. LVOS Math Model Compiler
2. SCALE/FORTRAN
3. Procedure Generator
4. System Generator (SYSGEN)
5. Discrete Model Executive (DEXEC)
6. Continuous Model Executive (CEXEC).

In addition to these programs which are used to compile and execute the system models, a program was developed to postprocess the SYSGEN output tape and provide an overall map of the vehicle model. A description of these programs follows.

LVOS Math Model Compiler - The LVOS compiler translates Boolean equations into a three level logic matrix and associated tables. Continuous model equations are preprocessed and formatted for the Fortran compiler. Linkages between continuous and discrete models are established and analog output channel assignments are stripped out and formatted for the executive system. The compiler can compile up to 10,000 input cards with 1250 discrete equations and 23 continuous model segments. (Continuous model segments are groups of continuous model equations which are compiled as independent FORTRAN subroutines.)

SCALE/FORTRAN - The SCALE program converts numeric constants to scaled integers and passes the continuous model card images to the FORTRAN compiler. The FORTRAN IV compiler has been modified to treat all constant data as single precision integers. The output of the SCALE/FORTRAN system is a relocatable binary tape of continuous model subroutines.

Procedure Generator - The procedure generator program compiles automatic procedures for real-time execution. Procedures are used to initialize models to specific configurations, control model execution,
and to insert faults for training exercises. Subroutines
in the discrete model executive are used to execute
procedures in the real-time system.

System Generator - The SYSGEN routines accept the
compiler output and the Fortran output tape for multiple
models and merge these models into one system tape.
Up to 14 discrete models and 120 continuous model
segments can be linked into one system model.

Continuous models are converted from a relocatable
format to load modules which are automatically re-
locatable by setting a base register. Cross reference
tables are established for CModel to CModel com-
munication. This feature allows the CModel executive
to determine when one CModel segment has modified a
variable parameter which is used in another CModel.
Hardware I/O assignments are also resolved.

Discrete models are merged and formatted for the
executive system. A name directory is constructed
from the variable names in all of the models and tested
for conflicts in name types and uses.

Discrete Model Executive - The discrete model execu-
tive controls the execution of all discrete models and
procedures. All model data is loaded by the DEXEC
and written to the disk. The models are all initialized
to an all OFF state and control is turned over to the
instructor console. Normally an automatic procedure
is started to bring the model to a specific configuration.

Continuous Model Executive - The continuous model
executive schedules and executes CModel segments and
supports the real time I/O interfaces with the ML
computer and the telemetry system. Execution of
continuous models is caused by a change in a Logic
Function switch in a discrete model or by a cross
reference variable change in another CModel. A time
integral, once initiated, will sustain a CModel execu-
tion until the integral change rate goes to zero.

SYSGEN Post Processor - The SYSGEN post processor
program processes a merged model tape and produces
a printed output of the model with the following informa-
tion:

1. A numeric list of all discrete inputs by
name and model
2. A numeric list of all discrete outputs by
name and model
3. A list of all analog variables by position in
COMMON
4. An alphabetic list of all names, variable
types, and models that use names.

Modeling Language

The LVOS modeling language is oriented toward the
engineer-user. He must learn basically three types
of statements before he can write models: continuous
model statements, discrete model statements, and
compiler directives. Each type of statement will be
described briefly below. All statements are written
in free form adhering to the FORTRAN card format.
Comments may be included within the model text by
starting a card with an asterisk (*) in card column 1
or on the same card with a statement within quote
marks.

Compiler Directive - A compiler directive is a com-
mand to the LVOS compiler to change its mode, format
the output listing, or make a hardware input/output
assignment. Compiler directives begin with an
asterisk (*) in card column one followed by a keyword.
MODE CONTROL
*INT  The following cards are to be included in the
  CModel segment named
*CMODEL  The following cards are
           to be included in the
           CModel segment named
*DMODEL  The following cards are
to be compiled as
          DModel equations

FORMAT CONTROL
*TITLE  Title information to
         appear at the top of
         each page
*PAGE   Begin a new page

HARDWARE I/O ASSIGNMENTS
*ANALOG INPUT varlablename
*ANALOG OUTPUT varlablename
*DISCRETE INPUT varlablename (nnnn)
*DISCRETE OUTPUT varlablename (nnnn)
*INTERFACE DISCRETE IN varlablename
*INTERFACE DISCRETE OUT varlablename

          (nnnn) specifies the actual discrete
          number

Continuous Model Syntax – The continuous modeling
language uses a subset of FORTRAN IV and some
special operators which are invoked as FORTRAN
function or subroutines. The FORTRAN subscripting
and array operations are prohibited and all input and
output to peripheral devices is processed by the CEXEC.
DO loops and backward GOTO statements are also
prohibited.

The following special purpose operators are available
for communication with discrete models and to provide
functions for real time applications;

Logic Function  (LFCNSW) allows a
Switch  continuous model
decision to be made
depending on the cur-
rent status of a discrete
function.  

Discrete LIMIT  (DLIMIT) provides a
Test  method of setting a
discrete argument
based on the value of a
continuous variable.

INTegrateL  (INTGRL) time integra-
tion of continuous
functions. A basic
Euler integration tech-
nique is used.

CLAMP  Limit a continuous
variable to upper and
lower bounds.

The continuous model special operators are similar to
those used in DSL/90 and CSMP.

Continuous model variable names may be up to 16
characters in length and use the same character set
as the discrete model variables. Continuous model
variables must start with an alphabetic character.

Continuous model variables and constants are written
in engineering units; scaling is automatically provided
by the CEXEC.

Procedure Language Syntax – The procedure compiler
processes input statements from card images. The
basic format of procedure statements is

  [#LABEL]  OPERATOR  OPERAND1
             [OPERAND2]  [OPERAND3]

Statement labels are optional. All operators require
at least one operand; some operators can have two or
three operands.

Discrete Model Syntax – Discrete model equations are
written in the following format;

  FUNCTION = EXPRESSION

Expressions are made up of arguments connected by
the logical operators AND (&) and OR (+). Arguments
may be negated with the logical NOT prefix (-) and also
may be grouped logically with parentheses.

A function may be defined only once in a model, however
it may be used as an argument in many other equations.
Arguments may be discrete inputs, functions or just a
variable name.

A discrete function may be delayed by an increment to
time by writing the equation as

  FUNCTION (nn.n) = EXPRESSION

where nn.n is the delay time is seconds.

Functions and arguments may have symbolic names up
to 16 characters in length. The alphabet and the num-
bers 0 through 9 are allowable characters. With these
basic syntax rules and an understanding of the discrete
model execution algorithm, an engineer is ready to
write discrete models.
There are three types of procedure operators

SYSTEM CONTROL OPERATORS

START MODEL  DINH  PULSE  CREL
STOP MODEL  DREL  CINH

PROCEDURE CONTROL OPERATORS

CALL  START PROC  END PROC  PROC

BASIC OPERATORS

SET  STAT  FAIL  GOTO
WAIT  RESET  IF

A complete description of these operators is not appropriate at this point. The function of most of the operators is obvious from the operator name, however the function of some will be discussed here briefly.

The DREL and DINH operators are used to prevent lengthy time delays in the discrete model execution when initializing models. The CINH and CREL operators are used to inhibit and release CModels when the results of the particular models is not required. The PULSE operator causes a one-time execution of a CModel.

Model Execution

Discrete models are executed on a demand basis. Commands from the firing room (switch actions or automatic procedure commands) will cause a discrete model (or models) to be queued for execution. Other actions may also cause models to be executed; expired time delays, discrete limit changes from CModels, interface discrete changes from other models, or procedure SET commands.

Continuous models are executed whenever a discrete model sends a logic function switch change. For example, a switch action may command a valve to open resulting in a pressure change in a system. A discrete function change indicating the valve opening will be sent to the corresponding CModel activating the equations for the pressures that change. Once a continuous model begins executing, it may queue itself for future execution if a time integration function within the model is active. CModels will otherwise remain dormant until queued by an outside stimulus.

One CModel may queue another CModel for execution by changing a cross reference variable which the other CModel uses. The CModel executive traps any changes in the Cross Reference (XREF) area of common storage. A change in an XREF variable will cause the CEXEC to queue all models associated with that XREF variable.

The ASTP (Saturn IB) System Model

LVOS is currently being used to train the launch crew for the ASTP mission scheduled for July 1975. To accomplish this training, the model integration team at KSC brought together stage models generated by system engineers from the SIB, SIVB, SIU stages and the GSE (or LSE). The IU, IVB, and LSE models had been previously generated for a Saturn V system and later modified for the Saturn IB system. Each model was compiled and debugged independently and the linked together and tested in Firing Room 3.

Currently there are six Individual models for the ASTP launch vehicle. They are:

1. SIB stage Part I
2. SIB stage Part II
3. SIVB stage
4. SIU stage
5. Integration model
6. LSE.

Each stage has a stage model coordinator for the integrated model. His responsibility is to assure that his model is compatible with the models of the other stages.

In the integrated model there are six discrete models as listed above, and 98 continuous model segments. Of the 98 continuous model segments seven are used to simulate the flight computer and are written in assembler language.

Procedures have been generated to initialize models for various test configurations and to start at predefined break points in the launch countdown. Some of the test configurations are Countdown Demonstration Test, Malfunction Overall Test, Flight Readiness Test, and Launch Countdown. The breakpoints in the countdown are at T minus 9 hours, T minus 4 hours, and T minus 1 hour 15 minutes.

Conclusion

The LVOS system has proven the feasibility of using a high level language for large scale real-time simulation. Use of the simulation language described in this paper has demonstrated that a high level language can result in a very low cost simulation system for training, and procedure validation. The favorable results of this project convinced NASA/KSC to select the language and techniques for the Shuttle Ground Operations Simulator (SOOS). SOOS will be used to validate ground applications programs for the Space Shuttle written in GOAL (Ground Operations Aerospace Language), as well as being used to train the Shuttle launch crews.
References


