CLASS - COMPOSITE LANGUAGE APPROACH FOR SYSTEM SIMULATION

Harold G. Hixson
Computer Sciences Division
Headquarters, Air Force Logistics Command
Wright-Patterson Air Force Base, Ohio 45433

Summary. There is an obvious need for a language which permits the incorporation of
 discrete and continuous representation in the
same model. A composite language (CLASS) is
described, with a typical application for
which this approach is appropriate. In addi-
tion, the flexibility of a composite language
in offering the use of either an event or
entity flow orientation, is shown to be advan-
tageous to the modeler. The choice depends
on his needs in describing various logic
patterns in different sections of the model
under construction.

Introduction

Background

Currently available simulation languages
such as SIMSCRIPT, SIMULA, GPSS, CSMP,
and SL-15, have provided generalized simula-
tion capability, but only within the
specialized area addressed by each. For
example, SIMSCRIPT is a tool for discrete
event simulation and SL-1 is a tool for con-
tinuous system simulation. GPSS is a power-
ful language but it is almost exclusively
limited to network flow simulation analogous
to the movement of objects through space.
It has a very limited capability for event
scheduling and canceling, which are basically
time-related mechanisms.

Modeling Needs

The need for a language with the ability
for simultaneous modeling of both discrete
and continuous system simulation has
been documented in several references during the past three years. Two languages
which enable the modeler to combine features
of event- and flow-oriented simulation have
been developed and are being used to model
computer systems. The need for the latter
combination has not been expressed frequently
but is nevertheless being recognized.

Language Development

Approach

Specification and development of a new
language with universal simulation capability
may be premature at this time. In order to
provide an interim technique with a comparable
scope of applications, this paper presents
CLASS, a composite language approach to system
simulation. The implementation of this tech-
nique is based on SIMSCRIPT II and provides
all capabilities of that language as imple-
mented in its original IBM 360 computer
version, much of the capability of the
General Purpose Simulation System - the IBM
360 version of GPSS, and the standard Con-
tinuous System Simulation Language (CSSL).
The specification for CSSL has been published
by Simulation Councils, Incorporated and the
language has been implemented by many comput-
er manufacturers and software houses.

Implementation. Due to limited resources,
the implementation of CLASS is planned for
accomplishment in three (3) phases. The
first (1970), provides SIMSCRIPT II, a limited
continuous system simulation capability, and
a limited GPSS capability. Some of the
features implemented in this phase are listed
in Figures 1 and 2. The second phase (1971)
would re-implement the continuous system
simulator features to attain both a full CSSL
capability (including user choice of integra-
tion methods), and an open-ended (algebraic/
statement/macro) orientation, (Figure 4), so
that the user can add constructs to the lan-
guage conveniently. The third phase (1972),
will expand the GPSS capability by the addi-
tion of the blocks described in Figure 3,
and also improve the compatibility between the three
specialized parts of the CLASS package, and
add a time vs variable plotting feature to
augment tabular output.

Interface. The arrangement of a multi-
language model will be with the continuous
event logic separated into submodels which normally communicate through
the use of global variables. The event and
algebraic language statements (SIMSCRIPT II)
and block/macro logic (GPSS) may be mixed
throughout the discrete event portions of the
model.

Example. An application using this
approach is the simulation of a process con-
trol system. The process being controlled
by a digital computer is the testing of newly
repaired aircraft engines. Simulation results
are to be used in acquiring equipment and
developing a system to operate eight (8)
engines concurrently in different test cells. Figure 5 depicts the major components of the
testing system and their interface mechanism. Figure 6 shows a sample of the digital logic
of one module of the control program. The
significant aspects of the continuous system
representing one aircraft engine in operation
are included in Figure 7. Samples of the
simulation output results are provided for
the continuous system submodel, (Figure 8)
and the discrete event submodel (Figure 9).

Results. Outputs from this simulation are
used for sizing the configuration of the
process control computer, evaluating the
cost-effectiveness of competitive computer
equipment, testing the feasibility of concurrent
gig engine testing, and checking the logic
of the process control digital programs. All
of these actions are thus accomplished with-
out the expense of benchmark programs, mock-
ups, prototypes, or the leasing of existing
commercial facilities similar to those being
designed and procured for this purpose. Space
limitations do not permit a full explanation.
of the results obtained from this model but the computer configuration evaluation results are similar to those obtained through the use of the System and Software Simulator\textsuperscript{14}, the Systems and Computer Evaluation and Review Technique\textsuperscript{14}, and the Extendable Computer System Simulator\textsuperscript{15}. The main reason for coupling the process model to the digital control simulation is to more precisely describe the simulated computer workload and improve the accuracy of the results. Byproducts of the simulation include checks of both test procedures and computer control logic.

Potential. There are many other potential applications of CLASS which would not involve the simulation of a computer. These include, for instance, testing of new manufacturing technology or chemical process validation.

Proposals of Marriage

Other continuous and discrete event language unions have been proposed—SIMULA\textsuperscript{2}, with MIMIC\textsuperscript{16}, GASPI\textsuperscript{17} with PACTO\textsuperscript{18}, and an extension of CSSL called SSL\textsuperscript{19}. The CLASS approach was developed for two reasons related to these proposals. It has more power and user convenience than other mergers of existing languages. And it can be used as an interim tool until a new language such as SSL can be developed.
Bibliography


12. SCI Simulation Software Committee, The SCI Continuous System Simulation Language (CSSL), Simulation, Vol. 9, No. 6, December, 1967


16. H. E. Petersen and F. J. Sansom, MMIC - A Digital Simulator Program, SESCA Internal Memo 65-12, Wright-Patterson AFB, Ohio (1965)

17. A. Pritsker and P. J. Kiviat, GASP II - A FORTRAN Based Simulation Language, Prentice-Hall 1969


Basic Single Values:

INDEPENDENT VARIABLE
CONSTANT
ABSOLUTE VALUE

Arithmetic Operations:

SUMMER (+)
MULTIPLIER
DIVIDER

Roots and Powers:

SQUARE ROOT
LOGARITHM
EXPONENTIAL
POWER OF VARIABLE

Trigonometric Functions:

SINE
COSINE
TANGENT
ARC SINE
ARC COSINE
ARC TANGENT

Calculus Operations:

INTEGRATOR
DERIVATIVE

Logical Elements:

AND/NAND/NOT
EOR/IXOR/NOR
EQUIVALENCE
RESET (Flip-Flop)
COMPARATOR

Switches (Relays):

INPUT SWITCH
OUTPUT SWITCH
FUNCTION SWITCH
BANG-BANG
DEAD SPACE
LIMITER
NEGATIVE CLIPPER
POSITIVE CLIPPER

Time Dependent Operations:

DELAY (Lag)
IMPULSE
PULSE
RAMP
STEP
HYSTERESIS

Special Operations:

STORE/ZERO-ORDER HOLD
MAXIMUM/MINIMUM
QUANTIZER
FUNCTION GENERATOR

Figure 1
Phase 1 - Continuous System Simulation Features
Transaction Oriented Blocks:
  GENERATE/TERMINATE
  BUFFER
  LINK/UINK
  JOIN/REMOVE
  EXAMINE/SCAN
  ALTER
  COUNT/SELECT
  ASSIGN/INDEX
  MARK
  LOOP
  TEST/GATE
  SPLIT
  ASSEMBLE/GATHER
  ADVANCE

Facility-Oriented Blocks:
  SEIZE/RELEASE

Storage-Oriented Blocks:
  ENTER/LEAVE

Queue-Oriented Blocks:
  QUEUE/DEPART

Standard Numerical Attributes:
  Parameters (Transactions)
  Transit Time (Transactions)
  Priority (Transactions)
  Current Contents (S,Q,C,G)*
  Available Units (S only)
  Facility Status (F only)
  Average Contents (S,Q,C)
  Maximum Contents (S,Q,C)
  Utilization (F,S,Q)
  Entry Count (F,S,Q,C)
  Average Residence Time (F,S,Q)

*Note: Abbreviations used are:
  F - Facility
  S - Storage
  Q - Queue
  C - Chain
  G - Group

Run Control Cards:
  RESET
  CLEAR
  START

Figure 2

Phase 2 - GPSS Features (SIMSCRIPT II Implementation)
**BLOCK NAME**

MXTOFN*

MXTOP*

PTOMX*

TRANSFER**

TRANSFER**

CAPACITY***

**BLOCK VALUES**

MX,ROW,COL,POINTS,FN

MX,ROW,COL,FIRSTP,LASTP

MX,ROW,COL,FIRSTP,LASTP

SBR,BLK,POINTER,NEST

F,POINTER,1,NEST

LIMIT

* The first three blocks permit the dynamic loading of function
"r" values from matrix savevalues, and communication between
matrix savevalues and transaction parameters.

** The underlined items are used as codes to describe the type
of transfer and its direction (to or from a recursive subroutine).

*** This block permits dynamic modification of STORAGE Capacities.

---

**Figure 3**

Phase 3 - Additional GPSS Blocks

---

```
LET LIM = LIMIT(K,INT1,INT2)
LET LAZY = DELAY(1.0,LIM)
LET DADT = DERIVATIVE(IC1,LAZY)
LET FG1 = FGEN(1,LAZY)
LET THRUST = FGEN(2,FG1)
LET FUELFLOW = FGEN(3,FG1)
LET KA = GAIN*DADT
LET INT1 = INTGRAL(IC3,KA)
LET INT2 = INTGRAL(IC4,INHIB,FUELFLOW,THROTTLE)
```

---

**Figure 4**

CLASS Features in SIMSCRIPT II Statement Form
Figure 5
Automated Jet Engine Test facility

LET ULIMIT = XN2 + DEL
LET LLIMIT = XN2 - DEL
ADVANCE 10 MILLISECONDS
IF RPM GT ULIMIT GO TO HIGH ELSE
IF RPM LT LLIMIT GO TO LOW ELSE
GO TO RESET
'HIGH' CALL THROTTLE
ADVANCE 13 MILLISECONDS
GO TO NEXT
'LOW' IF XF GT XO GO TO LL ELSE
GO TO SECOND
'LL' CALL THROTTLE
ADVANCE 13 MILLISECONDS
GO TO THIRD

Figure 6
Process Control Program Logic
Key: Symbols used are uniform graphics for analog and hybrid simulation, reference (I9).

Figure 7
Simulation Submodel Representing Aircraft Engine Operation
Figure 8
Continuous System (Aircraft Engine) Simulation Measurements
<table>
<thead>
<tr>
<th>Number of Engines Tested Concurrently</th>
<th>Digital Computer - Central Processing Unit Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>19%</td>
</tr>
<tr>
<td>2</td>
<td>23%</td>
</tr>
<tr>
<td>3</td>
<td>29%</td>
</tr>
<tr>
<td>4</td>
<td>36%</td>
</tr>
<tr>
<td>5</td>
<td>44%</td>
</tr>
<tr>
<td>6</td>
<td>54%</td>
</tr>
<tr>
<td>7</td>
<td>65%</td>
</tr>
<tr>
<td>8</td>
<td>77%</td>
</tr>
<tr>
<td>9</td>
<td>89%</td>
</tr>
<tr>
<td>10</td>
<td>100%</td>
</tr>
</tbody>
</table>

Figure 9
Discrete Event Submodel Results