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 addition, the flexibility of a composite language in offering the use of either an event or entity flow orientation, is shown to be advantageous 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[1], SIMULA[2], GPSS[3], CSMP[4], and SL-1[5], have provided generalized simulation 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 continuous system simulation. GPSS is a powerful 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 event and continuous system simulation has been documented in several references[6,7,8] 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.[9,15] 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 technique is based on SIMSCRIPT II and provides all capabilities of that language as implemented in its original IBM 360 computer version,[10] much of the capability of the General Purpose Simulation System - the IBM 360 version of GPSS[11], and the standard Continuous System Simulation Language (CSSL)[12]. The specification for CSSL has been published by Simulation Councils, Incorporated and the language has been implemented by many computer 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 integration methods), and an open-ended (algebraic/statement/macro) orientation, (Figure 4), so that the user can add constructs to the language conveniently. The third phase (1972), will expand the GPSS capability by the addition of the blocks described in Figure 3, 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 and discrete 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 control 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 engine testing, and checking the logic of new process control digital programs. All of these actions are thus accomplished without 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[13], the Systems and Computer Evaluation and Review Technique[14], and the Extendable Computer System Simulator[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[2] with MIMIC[16], GASP[17] with PACTOLUS[18], and an extension of CSSL called SSL[6]. 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

1. P. J. Kiviat, R. Villanueva, and H. M. Markowitz, The SIMSCRIPT II Programming Language, Prentice-Hall Inc., 1969

2. O. Dahl, B. Myhrhaug, and K. Hygaard, Some Features of the SIMULA 67 Language, Digest of the Second Conference on Applications of Simulation, December, 1968

3. G. Gordon, System Simulation, Prentice-Hall Inc., 1969

4. R. D. Brennan and M. Y. Silberberg, Two Continuous System Modeling Programs, IGM Systems Journal, Vol. 6, No. 4, 1967

5. D. H. Brandin and S. A. Schram, Introduction to Continuous System Simulation, Proceedings of the Conference on Applications of Continuous System Simulation Languages, July, 1969

6. D. Fahrland, Combined Discrete Event / Continuous Systems Simulation, SRC-68-16, Systems Research Center, Case Western Reserve University, July, 1968

7. H. Hixson, Equipment Maintenance Studies Using a Combination of Discrete Event and Continuous System Simulation, Proceeding of the Third Conference on Applications of Simulation, December, 1969

8. D. Fahrland, Combined Discrete Event / Continuous Systems Simulation, Vol. 14, No. 2, February, 1970

9. D. G. Weamer, QUIKSIM, a Block Structured Simulation Language Written in SIMSCRIPT, Proceedings of the Third Conference on Applications of Simulation, December, 1969

10. P. Kiviat, H. J. Shukiar, J. B. Urman, and R. Villanueva, The SIMSCRIPT II Programming Language : IBM 360 Implementation, RM-5777-PR, The RAND Corporation, July, 1969

11. General Purpose Simulation System/360 User's Manual (H20-0326-1), IBM Corporation, 1967

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

13. L. J. Cohen, System and Software Simulator, AD 679269 through AD 679272, Clearinghouse, U. W. Dept. of Commerce, 1968

14. D. J. Herman, SCERT ' A Computer Evaluation Tool, Datamation, February, 1967

15. N. R. Nielsen, ECSS, An Extendable Computer System Simulator, RM-6132-NASA, The RAND Corporation, February, 1970

16. H. E. Petersen and F. J. Sansom, MIMIC - 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

18. R. D. Brennan and H. Sano, PACTOLUS - A Digital Analog Simulator for the IBM 1620, Proceedings of the Fall Joint Computer Conference (1964)

19. Uniform Graphics for Simulation, Simulation, Vol. 9, No. 6, December, 1967

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/IOR/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/UNLINK

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	BLOCK VALUES
MXTOFN*	MX,ROW,COL,POINTS,FN
MXTOP*	MX,ROW,COL,FIRSTP,LASTP
PTOMX*	MX,ROW,COL,FIRSTP,LASTP
TRANSFER**	SBR,BLK,POINTER,NEST
TRANSFER**	P,POINTER,1,NEST
CAPACITY***	LIMIT

* The first three blocks permit the dynamic loading of function "Y" 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 = INTGRL(IC3,KA)
LET INT2 = INTGRL(IC4,INHIB,FUELFLOW,THROTTLE)
```

Figure 4

CLASS Features in SIMSCRIPT II Statement Form

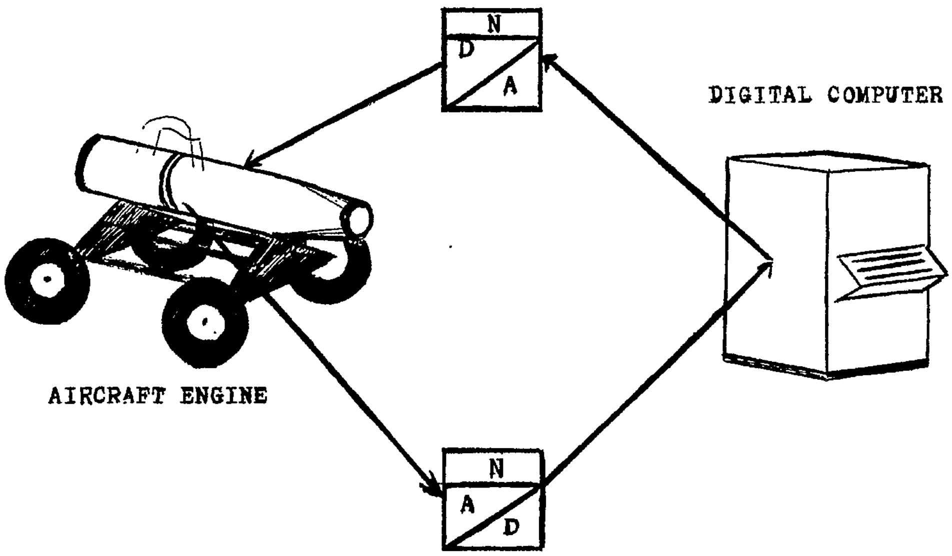

Figure 5

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 (19).

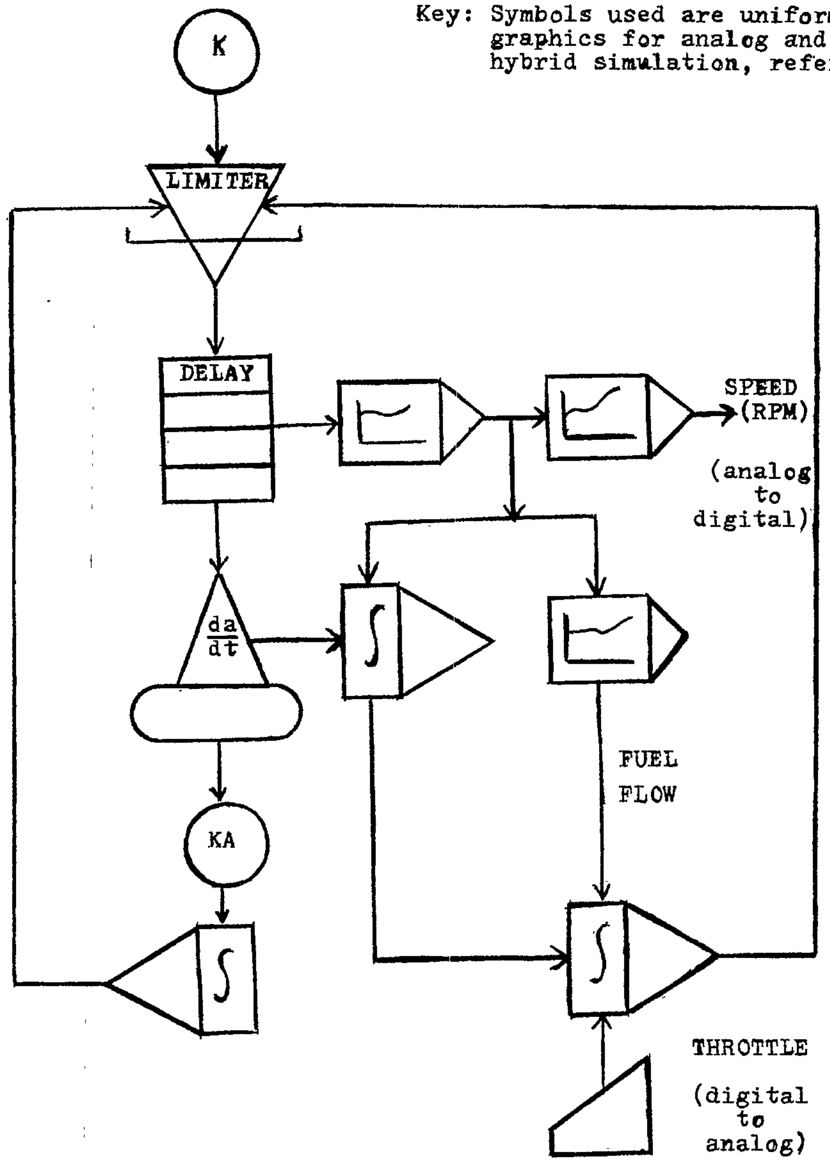

Figure 7

Simulation Submodel Representing Aircraft Engine Operation

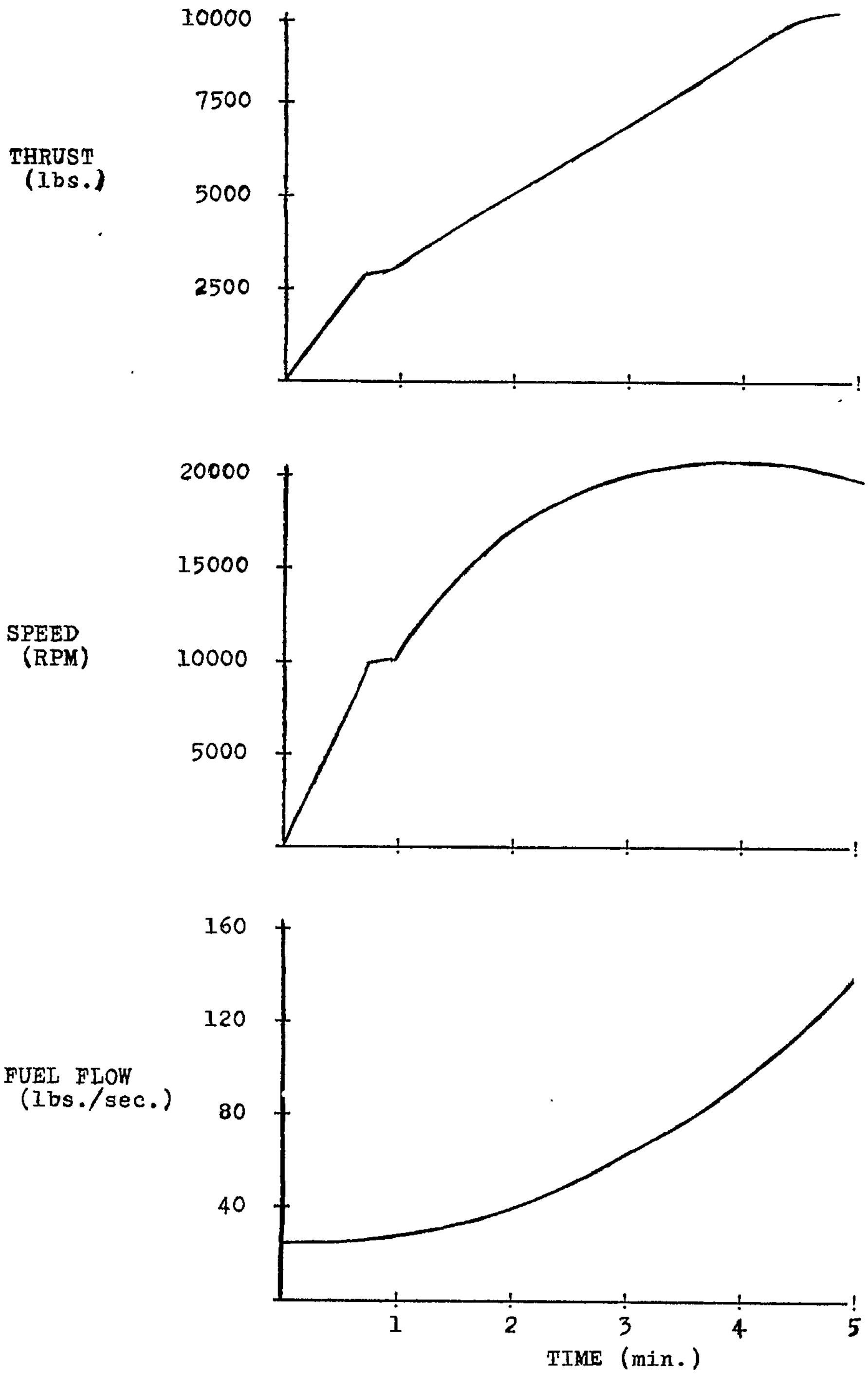

Figure 8

Continuous System (Aircraft Engine) Simulation Measurements

Number of Engines Tested Concurrently	Digital Computer - Central Processing Unit Utilization
1	19%
2	23%
3	29%
4	36%
5	44%
6	54%
7	65%
8	77%
9	89%
10	100%

Figure 9

Discrete Event Submodel Results