Proceedings of the 1988 Winter Simulation Conference
M. Abrams, P. Haigh, and J. Comfort (eds.)

State event in combined simulation

Irma Angulo
Dept. of Industrial Engineering
The University of Toledo
Toledo, Ohio 43606

Gholamreza Torkzadeh
Dept. of Info. Sys. & Op. Mgmt.
The University of Toledo
Toledo, Ohio 43606

ABSTRACT

Within the context of combined (discrete and continuous) simulation, modellers attribute considerable importance to the concept of state event representation. Unfortunately, since the methodology introduced in GASP IV, little research efforts have been devoted to the development of a more flexible and a more powerful modelling tool for representation of state event within a support system for model development environment. This paper briefly reviews the existing methodology and suggests a new approach useful for representation of a state event triggered by a complex condition involving two or more continuously changing attributes.

1. INTRODUCTION

Recognizing the need for a model development and management environment in which tools can be used to support modelling and analysis is considered as an important initial step to improve simulation effectiveness (Nance[1981], Henriksen[1983], Zeigler[1984], Overstreet and Nance[1985], and Balmer[1987]). Such an environment is supported by a model management system covering all phases of the model life cycle. Formalizing discrete event simulation modelling development and providing a model management environment for simulation model life cycle has been the focus of considerable attention, particularly formalizing specification languages for discrete event simulation modelling.

The need for simulation languages with the ability to simultaneously model both discrete and continuous systems has been demonstrated and well documented in a wide range of areas (Akatsuka et al.[1972], Desai and Minhram[1974], Golden and Schoeffer[1973], Hurst and Pritsker[1973], Miles et al.[1974], Schlechtendahl[1970], Sigal and Pritsker[1984], Pritsker and Hurst[1973], Talavage and Triplett[1974], Oren [1977], Duket and Standridge[1983], Kristiansen and Landsnes[1985]).

Most combined simulation languages have been written as extensions of discrete simulation languages. Thus, some of them generalize the process system such as DISCO (Helsgaun[1980]), while others generalize the next event system such as SLAM II (Pritsker[1986]), and SIMSCRIPT II.5 (Delfosse[1976], Markowitz [1979]). Also, languages have been written generalizing the three phase system, CGSP (Angulo[1983] and CABL Ellison[1979]).

In their 1985 paper, Overstreet and Nance focus on those phases representing model development activities and the model development system's role in support of model development environment for discrete event simulation. Using their context, this paper focuses only on model formulation and model representation of the model development phases. It describes a tool to support the model formulation and model representation of state events in combined discrete and continuous simulation systems. To the extent that it considers behavioral aspects of models specification, it is compatible with the ideas of Radiya and Sargent[1987] for discrete event simulation.

2. MODEL SPECIFICATION

A simulation model, in general, is composed of two parts: first, the state dimension decomposition or model structure, in terms of the system entities and their attributes (discrete and continuous); and second, time dimension decomposition or model dynamics, describing the rules of interactions between entities. Combined simulation refers to a model in which entities are allowed to have zero, one or more attributes that change continuously with time. The most important aspect of combined simulation arises from the interaction between entities having discretely and continuously changing attributes.

During the last decade, several formalism have been proposed for discrete event simulation (Zeigler[1984], Overstreet and Nance[1985], Radiya and Sargent[1987]). Zeigler's DEVS-formalism, based on systems and set-theoretic concepts, provides a formal basis for specifying the models expressable within discrete event simulation languages. Overstreet and Nance describe a specification language, called condition specification, intended to provide primitives by which time and state relationships can be formalized so that model specification analysis can be conducted and certain documentation can be extracted. Their language specification is developed to assist in analysis of discrete event simulation models. Radiya and Sargent propose a formalism for discrete event simulation based on the theoretical frameworks used for the formal analysis and specification of the different aspects of computer languages which includes the specification and validation of the structural and behavioral aspects of model specified in it.

Modelling methodology should not be limited to any specific form of model expression. The set of primitives, suggested here, could be adapted to fit with specification language tools described by others for discrete event simulation to create a more general modelling environment. In the following sections, we will describe primitives for model structure and model dynamics for representing continuous aspects of modelling.

2.1 Model Structure

Entities, attributes, and sets are the space components of a model. Here we briefly describe continuous attribute variables. For more complete definitions of entities, discrete and continuous attributes, and sets, refer to Angulo[1983] and Angulo and Torkzadeh[1986].

Continuous attribute variable names should be defined as of a specific type. Multiple instances of this specific type can be created dynamically in the same way as the entity with which the attribute is associated. The modeller can manipulate this type of attribute only through a set of operators or commands supporting these functions: establishing a continuous attribute and linking it to a particular entity (and at the same time, setting to zero the value of the attribute, its rate of change, and the tolerance in approximating its value); assigning a value to an attribute; assigning an expression to compute the rate of change of the attribute (this assignment can be made at any time as required); specifying the tolerance needed in approximating the value of the attribute; accessing the value of the attribute; accessing the rate of change; setting the rate of change of the attribute to zero; and producing output values.

2.2 Model Dynamics

A simulation may be constructed from two types of activities: bound activities and conditional activities. Bound activities are entered at certain scheduled times known as event times. Conditional activities, known as state events, on the other hand, consists of two parts: a condition, or a set of conditions; and a description of steps that occur when all conditions are satisfied. The interaction between the continuous and discrete change state variables in a combined discrete-continuous change system necessitates a broader interpretation of an event than is normally used in discrete change languages (Pritsker[1986]).

If the conditions which trigger an activity involve attributes that change continuously, then the time when the conditions are met should be found within the desired accuracy. The existing simulation languages use methods which allow only simple conditions. SLAM II is a popular simulation language with combined discrete continuous capabilities. In SLAM II, there are three fundamental interactions which can occur between discretely and continuously changing variables. First, a discrete change in value may be made to a continuous variable. Second, an event involving a continuous state variable achieving a threshold value may cause an event to occur or to be scheduled. Third, the functional description of continuous variables may be changed at discrete time instants (Pritsker[1986]). The method which is suggested here uses a composite technique that allows a compound condition of arbitrary complexity. A mechanism can be developed to ensure that the composite test is not satisfied at any other time than at which the system halts.

The conditions (testhead) for the commencement of a state event or a conditional activity may involve not only discrete but also continuous attributes. Hence, the testhead, in its more general form, can be broken down into two Boolean functions (for example, DISTEST and CONTEST for the discrete test and continuous test respectively). It should be possible for an activity to have many instances depending on the entities involved in it. A compressed notation should allow the whole class of activities to be expressed in one generalized form (see Figure 1). Using Pascal-like syntax: Figure 1(a) shows an example of how, in general, a conditional activity testhead is structured; Figure 1(b) shows a testhead when only discrete attributes are involved; and Figure 1(c) shows a testhead when only continuous attributes are involved.

```
IF DISTEST (i,j)
    THEN IF CONTEST (i,j)
         THEN ACTION (i,j)
         ELSE WATCH (i,j)

where:
 i activity number
 j activity instance
```

a) Test in terms of mixed discrete and continuous attributes.

```
IF DISTEST (i,j) THEN ACTION (i,j)
```

b) Test in terms of discrete attributes

```
IF CONTEST (i,j)
         THEN ACTION (i,j)
         ELSE WATCH (i,j)
```

c) Test in terms of continuous attributes

Figure 1: General Compressed Notation for a Conditional Activity

The generalized activity must now generate a whole class of specific activities, and this could be done by:

(a) a selection procedure which is applied to each class of entities involved to produce the completely specified set of entities that take part in the activity; and

(b) the arrangement to repeat the process so that all the admissible combinations are dealt with at the same time.

A continuous test can be of any complexity, but it must be broken down to permit the system to clearly detect any change in its true value that may take place. For example, consider C1 and C2 in Figure 2 as continuously changing attributes, and also consider a condition for triggering a state

event that depends on the satisfactory result of two simple tests involving both attributes, represented by the following logical expression:

```
IF ((C1 < L1) AND (C2 < L2))
   THEN BEGINACTION
```

where L1 and L2 are some given values.

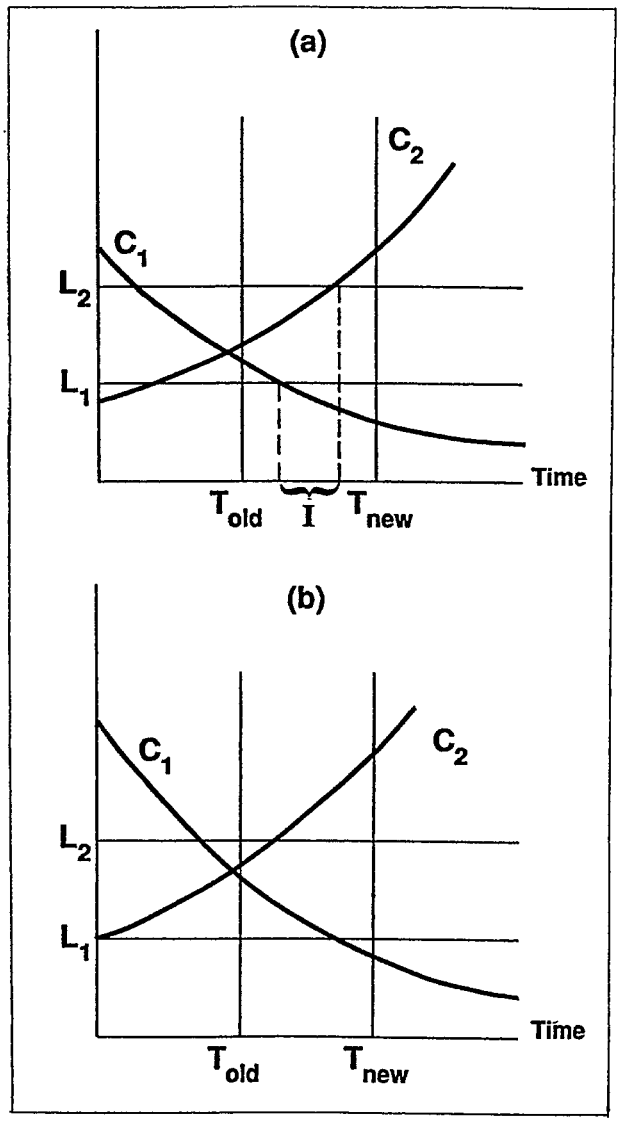

Figure 2: C1 and C2 Variables Evaluated at Two Consecutive Integration Steps

Figure 2(a) shows a situation that may arise if the expression for the condition is evaluated as a whole (as would be the case when the Pascal conditional statement 'IF' is used in the above expression). The values of continuous attributes are updated at each step of integration. At the same time, the system should check the conditions which involve continuous attributes. In the situation shown, in Figure 2 (a), the value of the whole expression is false at T_{old} (Time before last step) and T_{new} (present time) whereas the value is true for some 'time interval', 'I', between T_{old} and T_{new}. In this case, two consecutive changes of the total logical value have taken place within one step and the system, using the above general purpose language expression, has no way of recognizing such a change. This problem can not be avoided by reducing the maximum step size of integration.

If the two tests (i.e, two parts of the above expression) are examined separately, the system could detect the change in status of any one of them and at the same time use some mechanism to determine the relevance of the change with respect to the condition as a whole. Notice that the changes in each part of the expression would not have been relevant if the case was similar to the one shown in Figure 2(b), i.e. the value of C2 crossing L2 before C1 crossing L1.

In order to perform this task the system requires two things. One is the definition of those fundamental logical units, we call 'Boolean Objects', for which the system can recognize any change; and other is the complete testhead expressed in terms of its constituent Boolean Objects stating their logical connection. This information can be provided to the executive by means of a function, for example CONTEST as described in Figure 1. It is possible to build a pre-processor that takes mixed IF clauses of the form: IF C1 < L1 AND C2 < L2 THEN ..., and translates it into the form required by the run-time executive. The function CONTEST contains the whole compound tests in terms of the constituent Boolean Objects.

Following the occurrence of a time event, the executive generates a linked list of Boolean Objects. The list is organized in the following way: A 'pending' testhead (CONTEST) is characterized by its constituent Boolean Objects and its particular instance. For each pending CONTEST instance all its constituent Boolean Objects are added to the list. A recursive procedure is responsible for determining the relevance of any change in the status of the Boolean Objects with respect to the CONTEST to which they belong.

3. DISCUSSION AND CONCLUSION

Considerable effort has been made to formalize discrete event simulation modelling development and to provide a model management environment for simulation model life cycle. In particular, a more intensive effort has been made in formalizing specification languages for discrete event simulation. Expressiveness of specification languages may be further improved by including primitives and constructs to handle modelling of a system with continuous and discrete aspects. Indeed, as Zeigler[1985] contends, multifacetted modelling methodology should, in principle, not be limited to any one form of model expression. However, he maintains that we must have some substrate to work on, and rightly proposes that the discrete event concept is a good one to start with.

Whether to use discrete simulation concepts or continuous simulation concepts as a base for extension to include combined simulation is debatable. The most successful combined discrete and continuous simulation languages so far have been GASP IV, SLAM II and SIMAN. These are extensions of discrete simulation languages. As Markowitz(1979) noted, GASP IV was an important step in the right direction but more work is needed to achieve a satisfactory combined discrete and continuous simulation language.

An event occurs at any point in time beyond which the status of the system cannot be projected with certainty (Pritsker, 1986). There are two types of events that can occur in combined simulations. Time-events are those events which are scheduled to occur at specified points in time. In contrast, state-events are not scheduled, but occur when the system reaches a particular state. This particular state can be described by specifying a set of conditions.

GASP IV, SLAM II and SIMAN provide facilities for handling expressions restricted to certain form. The formation required by these languages are very much in line with the state variable concept. So the conditions are restricted to check the crossing of a state variable against other state variables or certain threshold values. This restricts the representation of a state event to monitor only one state variable. As Radiya and Sargent[1987] argue the resemblance of a model structure to the organization and form of modeller's knowledge of a problem entity increases the expressibility and readability of a model. A model should be constructed such that the model's behavior agrees with the behavior of the problem entity. This paper assesses the problem that arises when a state event is specified by a complex condition which may be needed to comply with model behavior.

Overstreet and Nance[1985] define a model specification in terms of primitives called a condition specification. They define this condition specification in terms of three components: an interface specification; a specification of model dynamics; and a report specification. Their model dynamics is described in terms of object specification and transition specification. For the purpose of comparison with the present work, we will focus on their transition specification. Transition specification is described by a set of ordered pairs; each called a condition action pair (CAP) which consits of a condition and an action. The condition is a Boolean expression composed of standard operators, model attributes, and the special sequencing primitives WHEN ALARM and AFTER ALARM. In general, this condition action pair corresponds to a 'conditional activity' as defined in this work. Cases where the sequencing primitives WHEN ALARM and AFTER ALARM are used correspond to our definition of 'bound activity'. As discussed in this paper, a Boolean expression is not adequate when dealing with a condition which also includes continuous attributes.

The tool suggested here is based on the concept of Boolean Object and allows:

a) the user to express complex conditions (as required) for an state event; and
b) the executive to successfully handle these conditions.

This approach provides a representation of a state event which is closer to the broader definition given earlier in the text and not linked to any particular state variable.

Perhaps a word about the software for implementation is appropriate here. Implementation of this concept requires a strongly-typed language capable of recursiveness and concurrency (e.g. Pascal, Modula-2, and C) that permits, among others, the definition and combination of new data types (e.g., records and enumeration types). Such a language requires more detailed declarations of data types assuring usage of program variables and data consistent with the program's declarations.

REFERENCES

Akatsuka, T., Kato, A., and Yoshida, N. (1972) "Integrated Continuous and Discrete System Simulation Program (CDSP)," *Journal of the Information Processing Society of Japan*, Vol. 13, No. 9. pp. 599-605.

Angulo, I. (1983) "New Facilities for Combined Simulation Modelling," Ph.D. Thesis, University of Lanceaster, U.K.

Angulo, I. and Torkzadeh, G. (198) "New Developments in Combined Simulation," WPS] 86-8, Business Research Center, University of Toledo, Toledo, Ohio, 43606.

Balmer, D.W. (1987) "Modelling Styles and Their Support in CASM Environment," Proceedings of 1987 Winter Simulation Conference, pp. 478-485.

Delfosse, C.M. (1976) "Continuous Simulation and Combined Simulation in SIMSCRIPT II 5, CACI, Inc., Arlington, Va.

Desai, V.K. and Minhram, G.A. (1974) "A Digital Computer Model of Automobile Traffic Passing Through a Signal Controlled Street Intersection," Proceedings, 5th Annual Pittsburgh Conference on Modelling and Simulation, Vol. 15, p. 1169, Pittsburgh, Pennsylvania, April 24-26.

Duket, S. and Standridge, C.R., "Applications of Simulation: Combined Models," *Modeling*, The Simulation Technical Committee Newsletter (IEEE), No. 19, Dec. 1983.

Ellison, D. (1979) "A Combined Executive for Activity Based Language (CABL)", Proceedings of Summer Computer Conference, pp. 77-79.

Golden, D.G., and Schoeffler, J.D. (1973) "GSL - A Combined Continuous and Simulation Language," *Simulation*, Vol. 10, No. 1, pp. 1-8.

Helsgaun, K. (1980) "DISCO - A SIMULA-based Language for Continuous Combined and Discrete Simulation," *Simulation*, Vol. 32, No. 1, pp. 1-12.

Henriksen, J.O. (1983) "The Integrated Simulation Environment (Simulation Software of the 1990's)," *Operations Research*, Vol. 31, No. 6, pp. 1053-1073.

Hurst, N.R., and Pritsker, A.A.B. (1973) "Simulation of a Chemical Reaction Process Using GASP IV," *Simulation*, Vol. 21, No. 3, pp. 71-75.

Kristiansen, T.K. and Landsnes, A. (1985) "Desgin of Refinery Offsite Systems by Simulation," Proceedings, IMACS Conference, Vol. 2, pp. 1187-1192.

Markowitz, H.M. (1979) "SIMSCRIPT: Past, Present and Some Thoughts about the Future," In *Current Issues in Computer Simulation* (Adam, N.R. and Dogramaci, A. eds.) Academic Press.

Miles, G.E., Hintz, T.R., et al. (1974) "Simawev II: Simulation of the Alfalfa Weevil with GASPIV," Modelling and Simulation, Vol. 5, Proceedings of 5th Annual Conference, University of Pittsburgh.

Nance, R.E. (1981) "The Time and State Relationships in Simulation Modeling," *Communications of the ACM*, Vol. 24, No. 4, pp.173-179.

Oren, T.I. (1977) "Software for Simulation of Combined Continuous and Discrete Systems: A State-of-the-Art Review," *Simulation*, Vol. 28, No. 2, pp. 33-45.

Overstreet, C.M., and Nance, R.E. (1985) "A Specification Language to Assist in Analysis of Discrete Event Simulation Models," *Communications of the ACM*, Vol. 28, No. 2, pp. 190-201.

Pritsker, A.A.B., and Hurst, N.R. (1973) "GASPIV: A Combined Continuous - Discrete FORTRAN Based Simulation Language," *Simulation*, Vol. 21, No. 3, pp. 65-70.

Pritsker, A.A.B. (1986) "Introduction to Simulation and SLAM II", Third Edition, Halsted Press, New York.

Radiya, A. and Sargent, R.G. (1987) "A New Formalism for Discrete Event Simulation," Proceedings of 1987 Winter Simulation Conference, pp. 554-558.

Schlechtendahl, E.G. (1970) "DYSYS - A Digital Computer Program for Simulation of General Dynamic Problems," *Nuclear Engineering and Design*, Vol. 14, pp. 104-108

Sigal, C.E., Pritsker, A.A.B. (1974) "SMOOTH: A Combined Continuous - Discrete Simulation Language," *Simulation*, Vol. 21, No. 3, pp. 65-73.

Talavage, J., and Triplett, M. (1974) "Trace Metal Flow in an Urban Environment," Modeling and Simulation, Vol. 5, Proceedings of the 5th Annual Pittsburgh Conference, University of Pittsburgh, April 24-26.

Zeigler, B.P. (1984) "Multifacetted Modelling and Discrete Event Simulation," Academic Press, New York.

Zeigler, B.P. (1976) "Theory of Modeling and Simulation," John Wiley, New York.

AUTHORS' BIOGRAPHY

I. ANGULO:

Assistant Professor in the Department of Industrial Engineering at The University of Toledo. She holds a B.Sc. in Electrical Engineering (Oklahoma State University), M.Sc. in Operations Research (Universidad Central de Venezuela), and Ph.D. in Operations Research (University of Lancaster, England). Her expertise lies in the field of simulation, optimization and quantitative techniques. Her research interest is in the area of combined simulation and has published in Simulation. She is a member of IEEE, ACM, and ASEE.

Dr. Irma E. Angulo
Department of Industrial Engineering
University of Toledo
2801 W. Bancroft Street
Toledo, Ohio 43606
(419) 537-4467

G. TORKZADEH:

Assistant Professor of Information Systems and Management Science at The University of Toledo. He holds a Ph.D. in Operations Research from The University of Lancaster, England. A member of O.R. Society of Great Britain, TIMS, DSI, ACM and SIM. He was involved in research programs pertaining to application of O.R. (in the public sector), distribution, resource allocation/reallocation, and mathematical modelling and published in the J. Opl. Res. Soc., Communications of the ACM, MIQ Quarterly, Journal of Management Information Systems, and Information & Management.

Dr. Gholamreza Torkzadeh
Dept. of Info. Sys. & Operations Mgmt.
University of Toledo
2801 W. Bancroft Street
Toledo, Ohio 43606
(419) 537-2506