Simulation Modeling and Statistical Computing

N. Adam Guest Editor

The Time and State Relationships in Simulation Modeling

Richard E. Nance
Virginia Tech

Time and state descriptions form the core of a simulation model representation. The historical influence of initial application areas and the exigencies of language implementations have created a muddled view of the time and state relationships. As a consequence, users of simulation programming languages work in relative isolation; model development, simulation application, model portability, and the communication of results are inhibited and simulation practice fails to contribute to the recognition of an underlying foundation or integrating structure. A model representation structure has been forged from a small set of basic definitions which carefully distinguish the state and time relationships. This paper focuses on the coordination of the time and state concepts using "object" as the link. In addition to clarifying the relationships, the structure relates the concept of "state-sequenced simulation" to the variations in time flow mechanisms. In conclusion, some speculations are offered regarding alternative algorithms for time flow mechanisms.

Key Words and Phrases: model representation, definitions, event scheduling, activity scan, process interaction, simulation programming languages, model documentation, state-sequenced simulation, Conical Methodology, time flow mechanisms

CR Category: 8.1

This work was done while the author was on study/research leave at the Combat Systems Department of the Naval Surface Weapons Center, Dahlgren, Virginia 22448.

Author's present address: Richard E. Nance, Department of Computer Science, Virginia Tech, Blacksburg, VA 24061.

I. Introduction

The recognition of similar program structures and functions led to the development of several simulators between 1960–63, e.g., the General Simulation Program (GSP) [43], the Sequence Diagram Simulator [8], GPSS [13], SIMPAC [23], Control and Simulation Language (CSL) [26], SIMSCRIPT [27], GASP [18], and SIMULA [35]. The emergence of these general tools for discrete event modeling, while offering advantages over the problem-oriented (general purpose) languages (POL*s*), introduced different modeling perspectives as well as the influences of varying problem domains.

Lackner [24, p.3] describes a modeling perspective as a *Weltansicht* or world view that "... must be at least implicitly established to permit the construction of a simulation language." In a subsequent work [25], he identifies the world views that categorize the different simulation programming languages (SPL*s*). Later, Kiviat [19, 20] compares and contrasts the three world views:

(1) Event-oriented, which describes the model states in terms of the consequences of events, programmed to describe "... how system state changes take place" [19, p. 52].
(2) Activity-oriented, which describes the actions of objects comprising the model and the conditions for these actions to take place.
(3) Process-oriented, which considers "... a set of events that are associated with a system behavior description" so as to combine "... the run-time efficiency of event scheduling with the modeling efficiency of activity scanning" [20, p. 22].

(Lackner's original categorization differentiated between event-oriented and activity-oriented views, considering the process perspective as identical with the latter.)

One unfortunate legacy of the different perspectives has been a geographical communications gap. The activity-oriented view, stimulated by the utility of the "wheel chart" [44] (later called "activity cycle" and "entity cycle" diagrams), predominated in the U.K. The popularity of ALGOL contributed to the success of SIMULA among European simulation practitioners. Notwithstanding the success of GPSS with its easily learned process interaction view, the event-oriented view dominated the U.S. scene.

A more unfortunate legacy is the inability to generalize the simulation modeling task. Language implementations of world views have often obscured important distinctions and slight but significant differences in terminology. Kiviat [19, p. 11] notes that the long-term results are the "different formal mechanisms" and the "slightly different interpretation of the theory" afforded (or enforced) by each SPL.

Language differences or modeling differences are not a problem as such, but when an *interpretation* is perceived as a *theory*, confusion arises as to the primacy of concepts and the generality of application. The seemingly independent and concurrent development of several SPL*s* during 1960–63 and shortly thereafter magnified the confusion caused by the "interpretation/theory inversion."

Is the simulation community really suffering from this language-induced inversion of theory and interpretation? Many would answer in the negative. However, simulation still carries the label of an expensive, uncertain problem-solving technique that represents the "court of last resort." This view persists despite the considerable advances in output analysis and improved capability of both hardware and software systems since 1970. Documentation of the shortcomings of computerized models, with simulation models prominent among those cited, is found in [6].

The concern with simulation extends beyond those who use the technique or those who underwrite the project costs. Others, admittedly a much smaller group, are concerned with the lack of recognition of a fundamental structure—a theory—after so many years of practice. The possibility is suggested that a causal relationship exists among these four factors:

(1) The absence of a theory, i.e., some fundamental definitional and integrating concepts spanning individual SPL domains;
(2) The theory/interpretation inversion in language-directed modeling;
(3) The multiplicity of languages, emanating from quite different application contexts;
(4) The high cost and questionable accuracy of simulation.

Efforts to reduce the deleterious effects of these factors include the application of computer-assisted modeling tools [22, 29, 38], research in model representation and documentation [32, 33], and the emergence of methodological approaches to simulation model development [14, 37, 34]. To date, Zeigler's work [46, 47] has progressed the farthest toward a descriptive theory of simulation modeling; however, other efforts such as using systems theory [16, 17] or the SIMSCRIPT-influenced entity/attribute/set approach [28] indicate a growing recognition of the need for an integrating general framework.

The objectives of this paper are:

(1) To illustrate the imprecision and subtle representational differences present in the approaches to discrete event simulation;
(2) To demonstrate the counterproductive influences of the imprecisions and differences;
(3) To present a set of definitions that clearly separate yet coordinate the time and state relationships;
(4) To consider the implications of this approach to simulation model development.

II. Time and State: The Model Coordinates

A. The Definitional Disorder

Early papers comparing SPL*s* utilized tabular presentations to assist the reader in understanding the variations in terminology relating to the same concept. These tables introduced the terminology used in particular SPL*s*, but did little to reveal semantic differences among languages. Fundamental differences in modeling views were hidden. However, the most serious misperceptions were fostered by the subtle distinctions in meaning applied to the same term in different SPL*s*.

A long list of these "pseudohomographs" serves no useful purpose. Instead, we focus on the term *event* and its relationship to the terms *activity* and *process* since these are principal terms in distinguishing among the three world views. At least four currently used SPL*s* are represented in the list below.

—(An event is) the instantaneous change in the value of one or more state variables [12, p. 3].

—This change of state is an event [4, p. 48].

—The events in a simulation are those particular times when something happens or should have happened [42, p. 83].

—An event occurs at any point in time beyond which the status of a system cannot be projected with certainty [41, p. 13].

—An event is an instant in time at which an activity starts or stops. The basic unit of action in a SIMSCRIPT II simulation is an activity [21, p. 282].

—Activity—a process that proceeds over a period of time.

—Event—a change in the state of the system that indicates the start or completion of an activity [5, p. 58–59].

—A process is an activity that proceeds over time. Its initiation, alteration, or conclusion is called an *event*. If, and only if, an event occurs, the state of the system changes [9, p. 76].

These examples describe an event as a point in time, a state change, or a state change at a point in time. An event is characterized as bounding a process or bounded by a process. An event is also described as initiating or terminating an activity, and the activity (initiation or termination) is characterized as causing an event. An event is defined in terms of a change in the system state in one case, and the change in state is predicated on the occurrence of an event in another definition. If the focus is shifted from event to process, for example, the situation can be made even more confusing (from [3, p. 91]):

—Descriptive units are the modules from which a description of the behavior of a system is constructed.

—*Activity.* The sequence of state changes prescribed by a descriptive unit.

—*Process.* A single instance or occurrence, during a simulation, of the activity prescribed in a descriptive unit.

—*Interaction Point.* A point within an activity at which control may be transferred to some other process.

—*Event.* The activity between two interaction points. An event always occurs instantaneously in simulated time.

The point is not that some definitions are "wrong" and others are "right," because each can be justified in its presentational context. The issue is: Can the differences among the definitions be reconciled within a clear, coherent understanding of *event* in the context of discrete event simulation? We think not—at least, not within the current understanding of the term.

B. Significance of the Disorder

The differences in terminology have contributed to the isolation of language users, albeit not as substantially as the geographical origins of the SPL or the availability of vendor-supplied language support. Installations acquired and supported only a single SPL. University courses emphasized a specific SPL available for student use, although possibly mentioning others (see [31]). Thus, few installations and fewer simulation analysts have used more than one SPL [2], and most simulations are written in a POL (also see [45]). This situation focused attention on the resolution of syntactic and semantic problems in a particular SPL, diverting interest from more fundamental issues. The persistence of the specific language focus is documented in the classical paper of Conway [7] as early as 1963.

Ironically, the diversion of attention from concepts and principles is now exacerbated by the spreading awareness of SPL developments. The subtle differences in terminology contribute more to this unfortunate circumstance than the glaring, more obvious examples. Students, novice users, and even experienced users assume a definitional uniformity that does not exist. Language extensions, especially when motivated by the desire to accommodate other world views, are achieved at the expense of altered definitions or "slight" redefinitions of the original terms. The consequence is a deviation from the original clarity of concept and cohesion in structure. Thus, the apparently beneficial spread of SPL awareness has proved detrimental in the sense that the efforts to increase language generality have contributed to the warping of concepts and the distortion of structures. This result contributes to increased impedance in the communication between model developer and model user, a significant factor in "cost overruns" and model disutility [6, p. 7–9].

The specificity of a SPL view imposes an insularity, beyond that characterized as user isolation above, by its mixing of syntactic, semantic, and modeling requirements during the model development and testing phases. The preoccupation with the *program representation* distracts from the more crucial aspects of the model itself. Zeigler [48] cites several reasons for the "model/program confusion" and emphasizes this misperception by citing a reference that defines a model as a program. Simulation analysts are given language training that contributes to the two problems cited above: Isolation and the relegation of concepts to a minor role. Insularity is promoted by the supposition that a model is a program. Furthermore, model portability is severely restricted since "documentation" is perceived to be "program documentation." Even if the documentation is adequate for the program representation, it is not likely to sufficiently meet the requirements for *model documentation* (see [33, p. 83–84]).

In summary, the definitional disorder discourages the realization of simulation foundations and inhibits the development and portability of models. It contributes to the difficulties in communication among model developers, model users, and potential model users (who are unable to assess utility or applicability). Therefore, the creation of order should concern simulation practitioners no less than those seeking a theory.

III. Coordinating the Time and State Relationships

In this section a set of definitions are offered that correspond closely to the terminology suggested by Kiviat [19, 20]. The differences arise primarily in the more detailed treatment of attributes and the clear distinction between state and time. Also essential to an understanding of the coordination of state and time is the structure imposed on the model during the development process. A brief explanation of the model development approach is provided.

A. Objects and Attributes

In typical fashion a simulation model is considered to be comprised of objects that are described in terms of their attributes (see [20]). In the definitions that follow, two conventions are utilized: (1) The use of indentation to reflect subordination among terms, and (2) The italicizing of a term that is subsequently defined.

—A model of a system is comprised of *objects* and the *relationships* among objects.

—An object is anything that can be characterized by one or more *attributes* to which *values* are assigned.

—Attributes are object descriptors of two types: *Indicative* and *relational.*

> Indicative attributes describe an aspect of the object; i.e., provide some knowledge about the object.
>
> Relational attributes relate the object to one or more other objects; i.e., describe the relationships among objects.

—Values assigned to attributes conform to an attribute typing much the same as in a POL; logical, numerical, and character string are examples of value types.

These definitions are a selected subset of those used in the Conical Methodology (see [34]), which requires a detailed classification of attributes.

B. Definition and Specification

The application of the Conical Methodology guides the model development progressively (but not necessarily sequentially) through two stages: Definition and specification. Definition proceeds through a top-down decomposition of the model into submodels, submodels into lower level submodels, and so forth until a base (atomic)

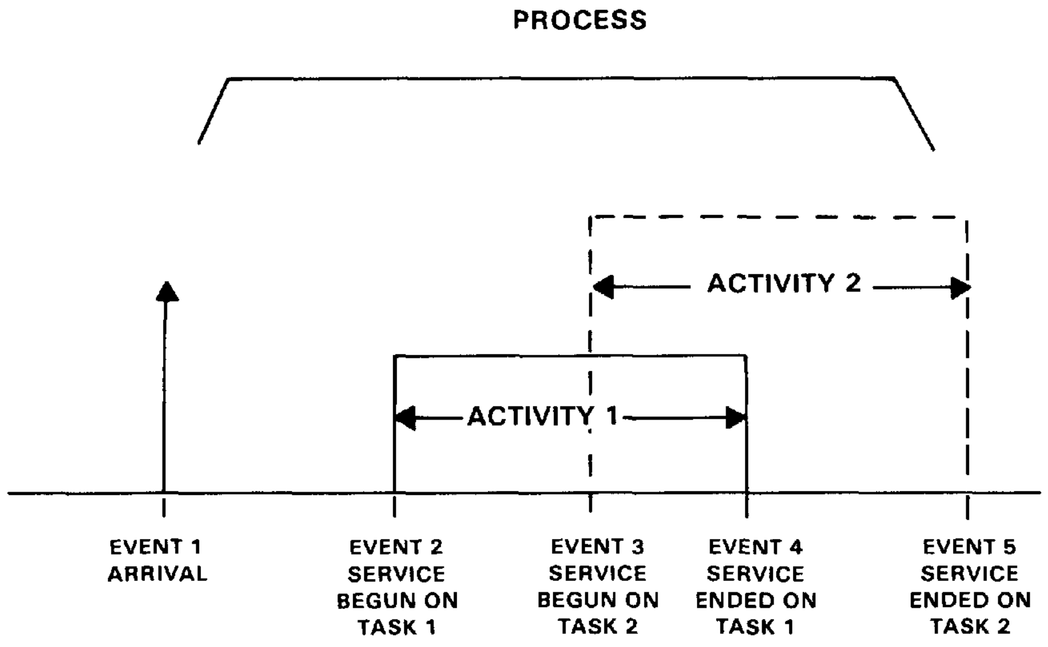

PROCESS
ACTIVITY 2
ACTIVITY 1
EVENT 1 ARRIVAL
EVENT 2 SERVICE BEGUN ON TASK 1
EVENT 3 SERVICE BEGUN ON TASK 2
EVENT 4 SERVICE ENDED ON TASK 1
EVENT 5 SERVICE ENDED ON TASK 2

Fig. 1. Event, Process, and Activity. (From FISHG73, p. 24)

level is reached. Specification is accomplished by the description of the value assignment expressions for each attribute of an object and the synthesis of objects to form higher level objects (submodels). Note that, by the above definitions, each submodel is an object, and the model itself is an object.

Among other things, the top-down definition and bottom-up specification establish structural relationships of a hierarchical nature among objects. Thus, a scoping of objects and object attributes can be derived. Further, the relational attributes provide the means for describing other types of relationships among objects (termed "coordinate" to distinguish them from "hierarchical").

C. Time and State Definitions

Every simulation model must have an indexing attribute, that is, an attribute of the model object that enables state transitions. Time is the most common indexing attribute, and we employ the common term "system time" in the definitions below.[1]

—*An instant* is a value of system time at which the value of at least one attribute of an object can be assigned (altered).
—*An interval* is the duration between two successive instants.
—*A span* is the contiguous succession of one or more intervals.
—*The state of an object* is the enumeration of all attribute values of that object at a particular instant.

The above definitions establish three measures of time: A point (instant), a duration between two points (interval), and a contiguous succession of durations (span). Instants where no attribute value for an object changes have no meaning for that object (but must have meaning for some object). The hierarchical scoping imposes the recognition of an instant for all objects superior to the object experiencing a change in attribute value. Of course, every instant has meaning for one object—the model.

The time and state concepts are brought together in the definitions of the familiar terms below:

—*An activity* is the state of an object over an interval.
—*An event* is a change in object state, occurring at an instant, that initiates an activity precluded prior to that instant.

An event is *determined* if the only condition on event occurrence can be expressed strictly as a function of system time. Otherwise, the event is *contingent*.

—*An object activity* is the state of an object between two events describing successive state changes for that object.
—*A process* is the succession of states of an object over a span (or the contiguous succession of one or more object activities).

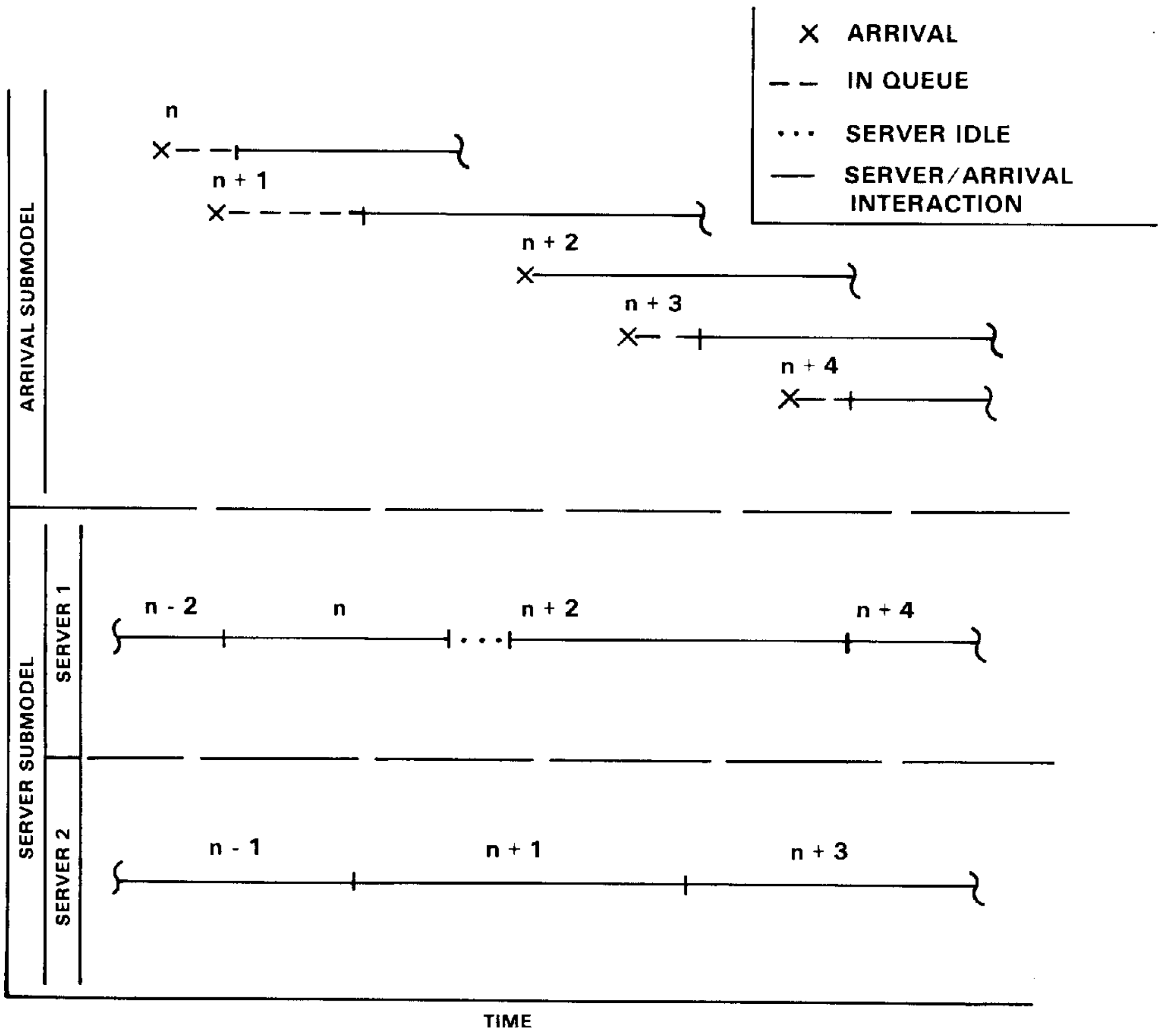
Keep in mind that an activity for a designated object is bounded by two successive events *for that object*. Intervening events, related to state changes for other objects, define intervals during which no state change occurs for the designated object. The object activity is terminated by the first instant such that an attribute value changes for the designated object.

IV. Implications and Speculations

The coordinated treatment of time and state, with clear distinctions between the two, has several implications for model development. The most important benefit is the removal of ambiguity surrounding the terms *event*, *activity*, and *process*. An equally important benefit which is currently being investigated but is not discussed here, is the potential for creating a primitive model description language that explicates the relations among the three world views of current SPL*s* (see [39]). Another impor-

[1] A model using space as an indexing attribute seems quite plausible.

Fig. 2. Two-Server Queueing Model Illustrating the Linking of the Activity and Process Concepts to Objects.



tant implication is the explanation of the relation of the concept of "state-sequenced simulation" to the existing time flow mechanisms (TFM*s*). Finally, the coordinated treatment of time and state permits some additional insights regarding the world views, the timing mechanisms, and the classification of SPL*s*.

A. Improved Precision in Definitions

A comparison by Fishman [10, p. 24] of the three world views indicates the current imprecision.

> ... an event signifies a change in state of an entity. A process is a sequence of events ordered on time. An activity is a collection of operations that transform the state of an entity.

Figure 1, also taken from [10, p. 24], illustrates the three concepts.

The revised definitions of Sec. III extend these concepts by making the object relationship explicit. For example, the interval between the instants for Events 2 and 3 define an activity for the object of Task 1, but the Task 1 object activity is defined for the interval initiating at the instant for Event 2 and terminating at the instant for Event 4.

The definitions also remove the implied restriction of *process* to temporary entities (objects), which is revealed in the characterization "... the process interaction approach emphasizes the progress of an entity through a system from its arrival event to its departure event" [10, p. 25]. In accordance with the treatment of *process* in SIMULA, it is considered applicable for describing both permanent and temporary entities. The revised definition above is suitably general, only imposing the requirement that, just as an object activity is bound by events related to the designated object, a process is comprised of one or more object activities.

Figure 2 applies an extension of the Fishman diagram to a two-server queuing model to illustrate the above points. Activities and processes are defined in terms of events and objects (or object attributes, since objects are defined in terms of their attributes). The arrival submodel describes the "customer" objects labeled *n*, *n*+1, ... from some arbitrary instant. The process for each "customer" can begin with a queueing activity followed by a service activity (e.g., customer *n*) or can consist only of a service activity (e.g., customer *n*+2).

During the time segment shown, the second server process includes no idle time; however, the server submodel process does contain an idle period. The hierarchical relation of the server submodel process to that for each server is quite apparent. With respect to the issue of simultaneity, the revised definitions specify that instants are unique; thus, intervals must take on positive values. The simultaneity complications treated in the interesting paper by Parnas [40] are avoided. All attribute value assignments at the same instant collectively define the event. Using the terminology of probability theory, an event can be simple—a single attribute value change, or compound—more than one. The ordering of value assignments comprising a compound event remains the responsibility of the modeler; however, continuing research effort is exploring techniques of model analysis to assist in the detection of potential ordering difficulties as well as model incompleteness and inconsistency.

B. State-Sequenced Simulation

In a 1975 presentation, Norton and Schaefer [36] described a state compression technique which would remove the requirement of strict time sequencing for some events. The result was to obviate the need for a time flow mechanism. This technique was referred to as "schedule independent simulation." At that time analysis of the requirements for schedule independence supported the conclusion that the technique had limited applicability. However, due in part to the obscurity surrounding the modeling concepts, the approach could not be related to the existing concepts of TFM*s* and the dynamic aspects of model representation.

A similar result is obtained with a technique described by Fishman [11, p. 126–129], based on a paper by Hordijk, Iglehart, and Shassberger [15]. Utilizing the property of exponential interevent times, the simulation execution can be reduced to the generation of states of a Markov chain, treating the time between transitions independently.

The schedule independent and Markov chain approaches are grouped under the term, "state-sequenced simulation." Figure 3 is intended as a conceptual aid to understanding the time and state relationships, in general, and in relating state-sequenced simulation to TFMs, in particular. Figure 3 is an expansion of an illustration of the continuum of TFM algorithms, presented nearly a decade ago [30], and re-examined recently by Atkins [1]. Time and state are depicted as orthogonal axes in Figure 3, and each axis is scaled conceptually as increasing, i.e., the time axis increases from very small to very large time increments. The state axis is scaled from very detailed changes (simple events) to highly aggregated changes (involved compound events). Thus, a TFM using fixed time increments lies along the time axis while a TFM might be postulated that strictly considers state sequences, with time treated implicitly, as lying along the state axis. The latter possibility is indicative of a state-sequenced approach. The relative position of a TFM on either axis is dependent on model attribute values. Employment of a state-sequenced approach is dependent on the property of time invariance: The division of the execution sequence into a repetitive subsequence of state changes that are not conditioned on time. A model with this property admits an execution that treats system time as a subordinate attribute.

The related Markov chain treatment described by Fishman is an example of the state-sequenced approach to the modeling situation where the state transitions remain probabilistic but the interevent times can be handled deterministically. This approach appears to be closely related to the use of a conditions list, which is described in the following paragraph.

Consider another implication of Figure 3: The possibility of forming a conditions list, analogous to the events list of an event scheduling SPL. An indexing variable would test for a listed condition (evaluate an expression of attribute values) to determine the state transition. This possibility, a more restrictive structure than used in activity scanning SPL*s* (see [47, p. 160–161]), represents a practical means for handling the typically complex problem of determining the time invariant subsequence of state changes. Is it possible that the conditions list could better match the inherent structure of some models than an events list?

V. Summary

In reference to the objectives stated in the Introduction, it has been demonstrated that the proposed definitions remove some difficulties caused by the confounding of state and time in the discrete event simulation literature and among SPL*s*. The separation of state and time

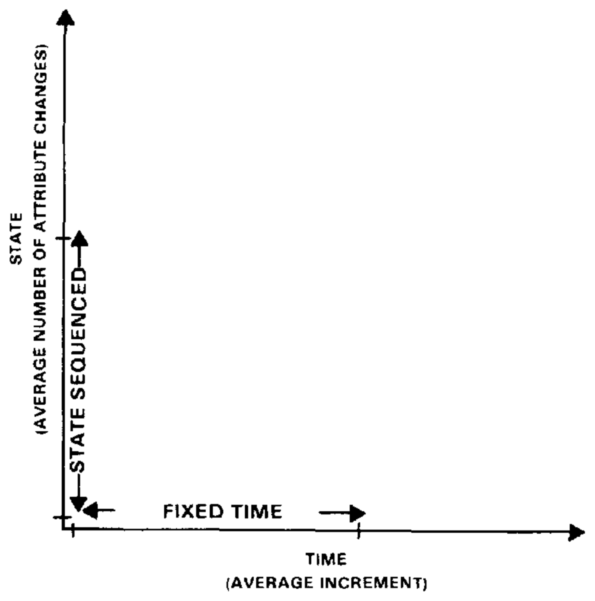


Fig. 3. Conceptual Illustration of the Conjunction of Space and Time in a TFM and the Contrast in the Fixed-Time and State-Sequenced Approaches

enables a more precise coordination of the concepts, helping to explain the relation of state-sequenced simulation to the explicit time treatments and stimulating some speculations on alternative TFM algorithms. This precision is a necessary precursor to the creation of an axiomatic approach to simulation model development.

Acknowledgments. The author acknowledges stimulating conversations with C. M. Overstreet, B. P. Zeigler, S. C. Mathewson, and J. G. Laski which contributed to the ideas developed in this paper.

Received 4/80; revised 10/80 accepted 12/80

References

1. Atkins, M. Stella A comparison of SIMULA and GPSS for simulating sparse traffic. *Simulation, 34,* 3 (March 1980), 93–100.
2. Beckwith, Richard E. Survey of simulation education: I. The perceived need. ORSA/TIMS Joint National Meeting, San Juan, Puerto Rico, October 16–18, 1974.
3. Blunden, G.P. and Krasnow, H.S. The process concept as a basis for simulation modeling. *Simulation, 9,* 2 (August 1967), 89–93.
4. Bobillier, P.A., Kahan, B.D. and Probst A.R., *Simulation with GPSS and GPSS V.* Prentice-Hall, Englewood Cliffs, N.J., 1976.
5. Colella, A.M., O'Sullivan, M.J., and Carlino, D.J. *Systems Simulation: Methods and Applications.* D.C. Heath, Boston, Mass., 1974.
6. Comptroller General of the U.S., Ways to improve management of Federally funded computerized models. LCD-75-111, General Services Administration, August 23, 1976.
7. Conway, Richard W. Some tactical problems in digital simulation. *Management Sci., 10,* 1 (October 1963), 47–61.
8. Dietmeyer, D. L., Gordon, G., Runyon, J.P., and Tague, B.A. An interpretive simulation program for estimating occupancy and delay in traffic-handling systems which are incompletely detailed. Proc. IEEE Pacific General Meeting, San Diego, CA, 1960.
9. Emshoff, James R. and Sisson, Roger L. *Design and Use of Computer Simulation Models.* MacMillan, New York, 1970.
10. Fishman, George S. *Concepts and Methods in Discrete Event Digital Simulation.* John Wiley, New York, 1973.
11. Fishman, George S. *Principles of Discrete Event Simulation.* John Wiley, New York, 1978.
12. Franta, William R. *The Process View of Simulation,* Elsevier-North Holland, New York, 1977.
13. Gordon, Geoffrey A general purpose systems simulator. (unpublished manual) IBM Corporation ASDD Commercial Department, October 25, 1960.
14. Holbaek-Hanssen, E., Handlykken, P., and Nygaard, K. System description and the DELTA language. Rept. No. 4 (Publication No. 523), Norwegian Computing Center, Oslo, Norway, 1977.
15. Hordijk, A., Iglehart, D.L., and Shassberger, R. Discrete time methods for simulating continuous time Markov chains. Rept. No. 35, Department of Operations Research, Stanford Univ, Stanford, CA, Jan. 1975.
16. Kindler, Evzen On the way to a mathematical theory of simulation. *Elektronische Informationsverarbeitung und Kybernetik, 12,* 1 (1976), 497–504.
17. Kindler, Evzen Dynamic systems and theory of simulation. *Kybernetika, 15,* 2 (1979), 77–87.
18. Kiviat, P.J. GASP—A general activity simulation program. Project No. 90.17-019(2), Applied Research Lab., United States Steel Corp., Monroeville, PA, July 1963.
19. Kiviat, Philip J. Digital computer simulation: Modeling concepts. RAND Memo. RM-5378-PR RAND Corp., Santa Monica, CA, Aug. 1967.
20. Kiviat, Philip J. Digital computer simulation: Computer programming languages. RAND Memo RM-5883-PR, RAND Corp., Santa Monica, CA, Jan. 1969.
21. Kiviat, P.J., Villanueva, R., and Markowitz, H. M. *SIMSCRIPT II Programming Language.* CACI Inc., Los Angeles, CA, 1973.
22. Kleine, Henry. Software design and documentation language. JPL Publication 77–24, Jet Propulsion Lab., Pasadena, CA, July 1, 1977.
23. Lackner, Michael R. SIMPAC: A research tool for simulation. SDC Document SP-220, System Development Corp., Santa Monica, CA, March 13, 1961.
24. Lackner, M.R. Toward a general simulation capability. Proc. of the SJCC, San Francisco, CA, May 1–3, 1962, 1–14.
25. Lackner, Michael R. Digital simulation and system theory. System Development Corp. SDC Document SP-1612, Santa Monica, CA, April 6, 1964.
26. Laski, John C. and Buxton, John N. C.S.L. Control and simulation language. Esso Petroleum Company, Ltd. and IBM United Kingdom, Ltd., London, England, 1962.
27. Markowitz, H.M., Hausner, B., and Karr, H.W. SIMSCRIPT: A simulation programming language. Rept. RM-3310-Pr; RAND Corp., Santa Monica, CA, 1962.
28. Markowitz, Harry M. SIMSCRIPT. Res. Rept. RC 6811 (29158), IBM Thomas J. Watson Research Center, Yorktown Heights, NY, July 26, 1979.
29. Mathewson, S. C. Computer aided simulation modeling and experimentation. Proc. Eighth Australian Comptr Conf. Canberra, Australia, Aug. 28–Sept 1, 1978, 9–13.
30. Nance, Richard E. On time flow mechanisms for discrete systems simulations. *Management Sci. 18,* 1 (Sept. 1971), 59–73.
31. Nance, Richard E. A comparison of discrete event simulation courses based on a small sample survey. Tech. Rept. CS76010-E, Dept of Comptr Sci, Virginia Tech, Blacksburg, Virginia 24061, July 1976.
32. Nance, Richard E. The feasibility of and methodology for developing Federal documentation standards for simulation models. Final Rept. to the Natl. Bur. Stand., Dept of Comptr Sci., Virginia Tech, June 26, 1977.
33. Nance, Richard E. Model representation in discrete event simulation: Prospects for developing documentation standards. In *Current Issues in Computer Simulation,* N. Adam and A. Dogramaci, Eds., Academic Press, New York, 1979, 83–97.
34. Nance, Richard E. Model representation in discrete event simulation: The Conical Methodology. Tech. Rept. CS81003-R, Dept of Comptr Sci, Virginia Tech, Blacksburg, VA, March 15, 1981.
35. Nygaard, K. SIMULA, An extension of ALGOL to the description of discrete event networks. Proc. IFIPS Cong., North-Holland, New York, 1963, 520–522.
36. Norton, Suzanne and Schaefer, Brian M. A new methodology of discrete systems simulation: Schedule independent simulation. *Bull Operations Res. Soc. Am.,* Chicago, IL, April 30 and May 1–2, 1975, B-69.
37. Ören, Tuncer I. and Zeigler, Bernard P. Concepts for advanced simulation methodologies. *Simulation, 32,* 3 (March 1979), 69–82.
38. Ören, Tuncer I. Computer-aided modeling systems (CAMS). Plenary Address, Simulation '80 Symp., Interlaken, Switzerland, June 25–27, 1980.
39. Overstreet, C. Michael, Research progress report. Dept. Comptr Sci, Virginia Tech, (limited availability), 1978.
40. Parnas, David L. On simulating networks of parallel processes in which simultaneous events may occur. *Comm. ACM, 12,9* (Sept. 1969), 519–531.
41. Pritsker, A.B., and Young, R.E. *Simulation with GASP-PL/I.* John Wiley, New York 1975.
42. Reitman, Julian *Computer Simulation Applications.* Wiley-Interscience, New York, 1971.
43. Tocher, K.D. and Owen, D.G. The automatic programming of simulations. Proc. Second Int. Conf. on Operational Research, 1960, 50–68.
44. Tocher, K.D. Some techniques of modeling building proceedings. IBM Scientific Computing *Symposium Simulation Models and Planning. IBM Data Process Division, 1966 (Conference held in White Plains, New York, Dec. 7–9, 1964), 119–155.*
45. Tocher, K.D. Keynote address. Proc. Winter Simulation Conf., Volume II., San Diego, California, Dec. 1979, 640–654.
46. Zeigler, Bernard P. Towards a formal theory of modeling and simulation: Structure preserving morphisms. *J ACM, 19,* 4 (April 1972) 742–764.
47. Zeigler, Bernard P. *Theory of Modeling and Simulation.* John Wiley, New York, 1976.
48. Zeigler, Bernard P. Integrated model pluralism: An alternative to a universal modeling language. Department of Applied Mathematics, The Weizmann Institute of Science, Rehovot, Israel, 1979.