MODEL REPRESENTATION IN DISCRETE EVENT SIMULATION: I. PROSPECTS FOR DEVELOPING DOCUMENTATION STANDARDS*

Technical Report CS78001-R

Richard E. Nance†

Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, Virginia 24061

Revised

September 1978

* This report contains modifications and extensions of material presented as an invited address at the Joint ORSA/TIMS Meeting in Atlanta, Georgia, on November 7-9, 1977.

† Research supported in part by the Institute for Computer Science and Technology, National Bureau of Standards under Contract No. T-73221. The opinions expressed herein are those of the author and do not reflect the position or opinions of the National Bureau of Standards.

ABSTRACT

The representation of model dynamics, i.e., the time-related interaction among entities in the model, poses the major unresolved problem in simulation model description. This report identifies the issues relevant to the development of model documentation standards. The argument is advanced that documentation and model specification share common objectives and should be coincident. Prior approaches to the generalization of simulation model specification are reviewed with regard to their potential as documentation tools. Their inadequacies stimulate the proposal for the "conical methodology", which emphasizes a top-down, structured model definition phase followed by bottom-up specification. To implement the conical methodology, a simulation model specification and documentation language (SMSDL) is needed. The characteristics of a SMSDL are identified. Recent publications suggest that certain program generator research and the DELTA Project are pursuing objectives similar to those cited for a SMSDL.

I. THE ISSUES

Perhaps the first issue in the development of model documentation standards is the intended scope of such standards. Simulation is a ubiquitous technique, applied in many areas and many disciplines, and the development of a taxonomy of simulation models is a difficult problem in itself. Should standards be evolved that apply equally well to discrete event simulation, continuous or strategic models, or hybrid (combined discrete event and continuous) models? Should models developed for execution on analog computers be included? If the standard is not to apply to everything labeled as a "simulation model," then how should the subdivision of simulation be accomplished? Should it be along application lines, the computing facility employed, the language or languages used, or some more fundamental procedure for differentiating among model types?

In our opinion, an unresolved basic issue is: What constitutes model documentation? Program documentation and model documentation are not synonymons, for one can always assume the target of the program documentation effort is a user, who is fully knowledgeable of the syntax and semantics of the programming language. Program documentation functions on a single level of communication, from one knowledgeable language user to another. However, adequate model documentation must function on more than one level. Model documentation must be informational to several types of potential user:

(1) the top-level manager funding the modeling effort.

(2) the top-level manager considering the application of the model to a problem in a completely different context,

(3) the systems analyst, who is quite familiar with the simulation technique but unfamiliar with the specifics of the language used, and

(4) the applications programmer, who must consider the detailed differences in implementation necessary for translating the model from one environment into another.

The current versions of simulation programming languages (SPLs) have shown some concern for model documentation, notably, SIMSCRIPT II.5 and SIMULA 67, but neither approaches the level of communication described in the first two cases above.

An important question is: Can documentation standards be both generic and descriptive? One is tempted to view this question as embodied within the issue of the scope of documentation standards, but it actually is more pervasive. If we can transcend the differences in SPLs and application areas, can we retain sufficient descriptive capability for the necessary levels of communication identified above? Accepting the necessity for a compromise between generic and descriptive capability, how do we determine the relative degrees of each? How do we measure the descriptive capability?

The final issue, and to many the most important, concerns the cost and potential benefits of model documentation standards. The recent GAO report [USGAO76] cites the need for model documentation standards for all computerized models used by the federal government. Program documentation has been considered a crucial shortcoming in both the public and private sector uses of computers for several years, and it forms an essential subject area of software engineering [NATIS76, LONDK74, MENKB70]. While the need for program documentation is recognized universally in the computing community, the degree and extent of model documentation implied in the GAO report and envisioned here raises the specter of increased costs

for Federal contractors. An understandable reaction on the part of contractors is the question: Who will be expected to assume the greater portion of the incremental cost? Contractors would view model documentation standards as a potential benefit for the contracting agency but a burdensome cost for them.

We have portrayed the issues purposefully as questions since in most cases the answers are not apparent. As we conclude in an earlier report [NANCR77), the answers are not to be found without some research, notably in the area of the description of model dynamics.

II. PREVIOUS WORK IN SIMULATION MODEL REPRESENTATION

We are excluding the earlier work in comparing, contrasting, and dissecting SPLs from consideration here. While this work undoubtedly contributed significantly to the current progress toward a general representation of simulation models, most papers dwell on the specifics of SPLs. For the sake of completeness, we note those efforts which appear to be of short duration, e.g., Nance's attempt to define in general terms the requisite parts of a simulation analysis [NANCR72], Etschmaier's use of difference equations [ETSCM72], and the recent cellular model description of DeCarvalho and Crookes [DECAR76]. We focus on concerted efforts in the generalization of model representation, found in the work of Lackner - the "calculus of change" concept, the program generator systems, the "systems theoretic" formalization, and the multilevel systems description approach. We also discuss the recently suggested extension of software development techniques to the simulation model context.

The Calculus of Change

The calculus of change concept, advanced in the several papers of Lackner [LACKM62, LACKM64a, LACKM64b], postulates an axiomatic structure for realizing a model state as a consequence of a prior state. The approach is based on the primitive operator "change", which is a relation described "as a class of antecedent-consequent pairs of logical situations" [10, p.28]. The "change" operation is a powerful primitive that is obtained at a cost, in the author's words [LACKM64a, p.37], that "...cannot be easily foreseen and cannot be deduced from theoretical considerations." Dynamic behavior is expressed through rules and laws, specified by the antecedent-consequent pairs. The generality of the calculus of change in describing the sequence of model states is obtained at the sacrifice of describing how the sequence is determined. The description of dynamic relationships among model components is at a level which masks the complexity in realizing a particular transition path. Lackner draws together the time and state relation so generically that the two cannot be distinguished but are woven together in the "change" operation.

Relatively early in the attempt to develop simulation theory, Lackner recognized the principal difficulty in generalizing a simulation model representation [LACKM62, p.10]:

> The fundamental dilemma faced in simulation modeling is the necessity of characterizing the dynamic in static terms; whatever methods are used, limitations are naturally encountered.

He concluded that the program representation of a simulation model lacked the necessary descriptive capability also, since the asynchronous operation of entities and the effects of the varying duration of interdependent processes could not be captured in the program structure. Lackner was

the first to differentiate among simulation models as activity-oriented, process-oriented, and event-oriented, a distinction that was explicated later by Kivat [KIVIP69].

The Program Generator Approach

Program generators are software systems which accept a model description (in a natural-language-like format) as input and produce an executable program in a particular SPL as output. The "Programming By Questionnaire" (PBQ) system developed at RAND [OLDFP66, OLDFP67] was the precursor of modern program generators. Research efforts in this area, with two notable exceptions, are to be found in British universities. Mathewson's development of the DRAFT system at the Imperial College [MATH74, MATH76, MATH77a, MATH77b] has sought to incorporate different target languages within a similar model description component. The CAPS-ECSL program generator, developed in England by Clementson [CLEM73, CLEM77] and implemented in this country by Hutchinson [HUTCG75], like the DRAFT system, is based on an activity-cycle (also referred to as entity-cycle or entity life-cycle) diagram. These diagrams focus on the cyclical activities of each entity, or class of entities, represented in the model. The merger of the activity-cycle diagram for each entity, or class of entities, constitutes the total model description. Figure 1(a) shows the four entity/activity symbols used in the DRAFT system, and a simple example of a group surgery, taken from Mathewson [MATHS77b, pp.3-2 to 3-5].

The construction of an activity-cycle diagram must conform to specified rules governing the combination of symbols. Conformance to the rules assures the definition of a feasible model. The activity-cycle diagram can be divided easily and proves quite useful for relatively small,

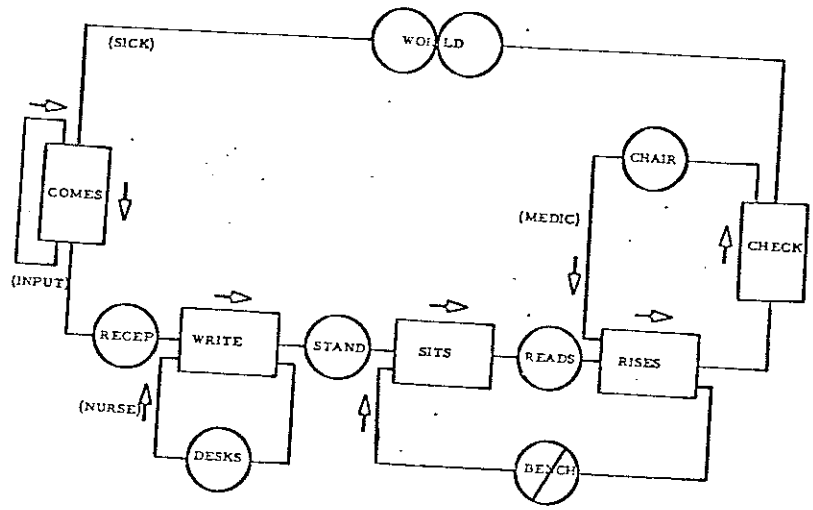
A QUEUE or SET is an ordered list of the objects or entities considered in the model.

The INFINITY QUEUE is QUEUE or SET which can be considered a 'source' (infinite supplier) or 'sink' (infinite storage).

An ACTIVITY is a period of time during which certain actions are taking place. (An ACTIVITY is initiated and terminated by an EVENT - an instant that a change of state occurs in the model.)

A FLAG variable is a counter used to denote a resource whose individual life cycle is not of interest. (The FLAG variable removes the need to attach identifiers to individual QUEUE or SET members if no need exists to distinguish individual behavior.)

(a) Definition of the Symbols Used in Constructing Entitu-Cycle Diagrams.



Each cycle traces out the life of a particular class of entity. The SICK start in the WORLD and, at intervals, one COMES into the RECEPTION. They then WRITE down their personal details and move into a waiting room where they STAND until each individual SITS and READS waiting for his consultation when he RISES, has his health CHECK and returns to the WORLD. At some activities the cycles merge and the interaction causes resources to be utilized. In this model, a SICK patient COMES and enters at RECEPTION. If a NURSE is at the DESKS, the two people join together to write down the relevant personal details. When this is complete the NURSE returns to the DESKS where she may be able to cope with another patient, and the SICK person will STAND until a BENCH is available. Then the patient SITS and READS until a MEDIC, free in his CHAIR, calls the patient who RISES, freeing the BENCH, and enters the consulting room where the health CHECK takes place. On completion the patient returns to the WORLD and the MEDIC sits in his CHAIR, free to check the next patient. There is one artificial cycle, the feed-back loop to the patient arrival activity COMES. The entity INPUT is released whenever a patient is generated and this automatically resets the activity COMES to generate the next patient. By implication further patients are always in the WORLD (infinite queue).

(b) Entity-Cycle Diagram and Narrative Description for a Group Surgery Model

Figure 1. The Entity-Cycle Symbols and An Example from DRAFT (Taken from [MATHS77, pp. 3-1 to 3-5]).

simple models, but as the model size, and in particular the interdependencies among entities, increases, the representation becomes cumbersome and less comprehensible.

Other program generator efforts reported by British researchers are the Sheffield Simplified Simulation System (S4) of Lafferty [LAFFH73] and the Modular Interactive System for Discrete Event Simulation (MISDES) described by Davies [DAVIN76a]. Recent research by Davies has sought to generalize the target language utilized in model construction through interactive dialogue with the user. He has attempted a separation of information required by a program generator, based on that which is essential for model description versus that related to the semantics of the target language [DAVIN76b].

In this country, the work of Heidorn [HEIDG74] exemplified a slightly more ambitious goal. Heidorn's system accepts an English-like specification as input and produces a GPSS program. The approach comprises three steps:

(1) acquisition of the problem statement through an English dialogue,

(2) verification of the model from the output of the consolidated description, and

(3) simulation experimentation using the GPSS model.

Similar objectives have been expressed by Mathewson [MATHS77c] for the "experiment generators" research at the Imperial College.

The success of program generators to date must be tempered by limitations in problem domain, restrictions on the range of discourse, and incompleteness of target language productions. Nevertheless, their impact on systems specification is recognized.

The Systems Theoretic Formalization

The systems theoretic approach, presented in the papers of Zeigler [ZEIGB72, ZEIGB73, ZEIGB75a, ZEIGB75b] and summarized in a book [ZEIBG76], provides the most extensive general theory of simulation modelling. Zeigler's formalism maintains a separation of static and dynamic model description. Each model component is characterized by descriptive variables, and a component interaction section identifies the dynamic relationships in an "informal" description [ZEIGB76, pp.125-143]. The "formal" description reduces to a representation of state transitions, in a sense similar to Lackner's work. Influencer diagrams and model trajectories provide additional dynamic description. A mechanism analogous to a finite state machine forms the nucleus of the model representation.

An example taken from the referenced pages shows the simulation model of a grocery store as consisting of:

Informal Description

Components: ENTRANCE, SHOP-AREA, CHECKOUT, EXIT

Descriptive Variables for each component (only two shown):

ENTRANCE: HELLO - with range prescribed in terms of name assignments

SHOP-AREA: SHOPPING-TIME a random variable whose values range over the positive real numbers (R^+)

TIME-LEFT-LIST - a list of element pairs (χ_1, τ_1) $(\chi_2, \tau_2), \ldots, (\chi_n, \tau_n)$ indicating that customer HELLO = χ_i will leave SHOP-AREA in time τ_i from the current time

Component Interaction which describes the sequential movement of a customer HELLO = x_i through the grocery store, emphasizing the conditions governing the interaction with other entities.

The model trajectory for the grocery store is presented in graphs of the value assignments of descriptive variables, which can be time- and state-related. The formalization produces a discrete event system specification which, conforming to the general representation of a well-described discrete event model, is a 6-tuple < INPUTS, STATES, OUTPUTS, δ, λ, t>, where

δ is a state transition function,

λ is an output function, and

t is the time advance function.

The systems theoretic formalization provides a powerful descriptive structure, emphasizing a high-level model representation. Important information concerning details of simulation program, such as logical and physical data structures, record formating, or the complexity of entity interactions, are not easily derived. Whether the formalism can be extended to this level is an open question.

Extension of Software Development Tools

McLeod [MCLEJ70, MCLEJ73] has suggested a checklist approach to provide uniform documentation of simulation models. This outline format is an extension of procedures from software engineering relating to program documentation. The software design and documentation language (SDDL), developed by Kleine [KLEIH77a] is an interactive system which assists in the design process and promotes complete, consistent program documentation. Recently, SDDL has been applied in the simulation context [KLEIH77b]. While the system has proved useful as a software creation tool, its utility

in promoting the different levels of communication necessary for simulation model documentation is yet to be demonstrated.

III. THE CONICAL METHODOLOGY: RATIONALE AND OVERVIEW

This section summarizes the rationale leading to a proposed approach to the development of model documentation standards, which is presented in an earlier report [NANCR77]. This approach, which we term the "conical methodology", is summarized from an overview perspective.

Rationale for the Conical Methodology

Hypothesis 1: The process of model documentation must be so inseparable from the modelling task itself, that one could not be accomplished without the other.

Hypothesis 2: The economic benefits of model documentation are persuasive, while those of problem (model) specification are convincing.

Hypothesis 3: The social implications of a hierarchy are distasteful; yet, the advantages are acknowledged.

Hypothesis 4: Effective standards are not forced from inadequate examples.

Within these four hypotheses the rationale for the conical methodology is summarized. These hypotheses represent some fundamental beliefs concerning the modelling task, the economical and practical justification of model documentation, the advantages of a hierarchical or structured approach to design, and the belief that compromises do not beget effective standards.

The claim is often made that the only true documentation of a program is the program itself. If that is so, then the only hope for model documentation is that it be inherent to the modelling task. This argument becomes even more significant when one considers the likely reaction of those who will be charged with the model documentation responsibility. Unless the modelling group itself can perceive a clear, distinct benefit, the resulting documentation will be produced as an unessential requirement, warranting no more attention that the hastily constructed program documentation too often encountered.

"Structured design" is a term used to describe a set of proposed general program design considerations [STEVW74]. These techniques and considerations include several familiar terms, e.g., composite design, the HIPO techniques, and structured programming, which characterize the structured design approach as an orderly, hierarchical subdivision of tasks to simplify the requirements of a complex project. This top-down approach, resembling the much earlier constructs of engineering design -- subdivide, analyze, and synthesize, has proved successful in large programming projects.

A model documentation standard based on current SPLs or developed as a compromise among contemporary techniques is doomed to failure. The product of the conical methodology should not be constrained by the existing language capabilities; however, we must accept the responsiblity for recognizing deviations from extant SPLs or extensions in concept beyond those currently accommodated by some language.

Overview of the Conical Methodology

The conical methodology can be described succinctly as an approach which embodies top-down definition of a simulation model coupled with

bottom-up specification of that model. The methodology is described in Figure 2 through an outline illustration. Model definition begins with a statement of the study objectives, including assumptions regarding these objectives. The modelling environment is described in terms of the creating group, the scope of the effort, the model boundaries, and the interaction of the model with the exogenous factors. The model itself is defined in terms of attributes which are typed as value or relational. For the purpose of an overview, we define these attribute types as follows:

(1) value attributes describe an aspect of a model component, and

(2) relational attributes relate a component to one or more other components.

I. Statement of Study Objectives.

A. Definitions.
B. Assumptions regarding objectives.

II. Modeling Environment.

A. Modeling effort.
1. Organization creating model, dates, individuals, etc.
2. Scope of effort in time and money.
B. Model assumptions.
1. Boundaries.
2. Interaction with environment.
a. Input description.
b. Assumptions on model/environment feedback or cross effects.
c. Output and format decisions.

III. Model Definition.

A. Model attributes.
1. Value attributes.
2. Relational attributes.
B. Submodels.
(1.) Submodel at the first level.
(a) Value attributes.
(b) Relational attributes.
((1)) Submodel at the second level.
((a)) Value attributes.
((b)) Relational attributes.
.
.
.
(...(1)...) Object at level n.
(...(a)...) Value attributes.
(...(b)...) Relational attributes.

(2.) Submodel at the first level.
.
.
.

IV. Model Validation and Verification Procedures

A. Validation tests.
B. Verification criteria and tests.

V. Model Experimentation.

A. Hypotheses to be tested.
B. Experimental design.

VI. Implementation Requirements.

Figure 2. The Conical Methodology Approach to Model Specification/ Documentation -- Outline Illustration.

From the model level, or most global level, the definition proceeds through a partitioning process. This strictly hierarchical approach breaks the model into submodels at succeedingly lower levels, the submodel at each level is characterized by its value and relational attributes. The partitioning continues, producing a conical structure, until eventually a level is reached where no further submodel is necessary, and this is called the object level. From the object level, one begins to assign values or determine the source of values as well as the relations among components within the submodel. Thus, the model definition phase is succeeded by the model specification phase, and a bottom-up development is followed through an association of objects to form submodels and submodels to form submodels at higher levels. At each level the submodels are specified in terms of the determination of, or provision for, assignments to value and relational attributes.

The conical methodology could be applied using various descriptive mechanisms, ranging from a natural language to a simulation programming language. However, the full utility of the methodology can be realized only through a descriptive mechanism possessing certain important characteristics, which are described in a following section. An example illustrating the application of the conical methodology through such a descriptive mechanism is given in a paper to follow [NANCR79].

One very important component of the full utility of the conical methodology is the provision of a limited degree of model validation and verification through testing of the consistency of attribute types and the identification of component relations. Further, verification procedures can be applied to submodels rather than the entire model. Please note

that we use the terms "validation" and "verification" in accordance with the following definitions:

> validation is the testing of model validity, i.e., the assurance of internal model consistency or the inclusion of all relationships prescribed in the description of the modelling environment to accomplish the study objectives, and
>
> verification relates to the correspondence between the model and the physical system under study; i.e., the truthful representation of the system commensurate with the objectives of the study.

We recognize the disparity between our definitions and those currently used by others, but we prefer a usage more in accordance with the dictionary and the language precedent.

IV. THE SIMULATION MODEL SPECIFICATION AND DOCUMENTATION LANGUAGE

Implementation of the conical methodology is intended through a rather unique language that embodies the structured, interactive, procedural approach implied by the above description. We refer to this language as the Simulation Model Specification and Documentation Language (SMSDL) which should be an interactive, conversational language similar in constructs to OPS-4 [JONEM68] or CONSIM [NELSS77]. The semantics of SMSDL are to be derived from a very precise set of terms that explicate the time and state relationships in a model description.

Characteristics of a SMSDL

In addition to the above properties, a SMSDL should possess the following characteristics:

(1) The semantics of a SMSDL must facilitate model specification and model documentation. The two are inextricable if both are begun in the initial stages of model construction, which *is* the proper point for each. A SMSDL is an *a priori* construction tool serving the purposes of effecting communication among project members and providing a technical base.

(2) A SMSDL must permit the model description to range from a very high to a very low level. Description at the highest level promotes communication among managers, those who must make decisions concerning acquisition and implementation. At the lowest level the language should permit the description of logical and physical data structures employed in representing model components. A SMSDL must permit the level of detail to be dictated by model complexity, cost, application area, or external specification.

(3) The degree of detail -- the level of description -- should be controllable. This control should be sufficient to permit clear specification, and each level should support a "natural" refinement of the preceding level.

(4) A SMSDL must exhibit broad applicability to diverse problem areas. The semantics of the language should not reflect any area of application.

(5) A SMSDL should be independent of but not incompatible with extant SPLs. The relation of an SMSDL to a SPL is conceptually clear: A SPL model description, together with the explicit representation dictated by a particular translator, constitutes the lowest descriptive level of a SMSDL.

(6) A SMSDL should facilitate the validation and verification of simulation models. The semantics of the language and the discipline embodied in its use should force the testing of model validity as well as enable progressive model verification.

The DELTA Project

Near the conclusion of the NBS-sponsored project concerning simulation model documentation standards [NANCR77] we became aware of the DELTA Project. Subsequently, Professor Kristen Nygaard provided a copy of Report No. 4 [HOLBE77].

Initially, the objectives of the DELTA Project were to define and formalize the concepts necessary for communication of systems descriptions among diverse groups, such as systems analysts, scientists, programmers, and trade union members. The project centered on higher-level descriptive elements, unrelated to computer implementation. However, as the project evolved, the extension to an implementation level became more appealing; so that three language levels are considered:

DELTA - the general system description language,

GAMMA - a high-level programming language, conceptually and and syntactically very similar to DELTA, and

BETA - a systems programming language, reflecting a computer organization

At the center of the DELTA research is a core of well defined terms. Those terms are knitted into a descriptive structure, inspired in part by the SIMULA 67 language. The decomposition and composition capabilities of DELTA are carefully developed; the model dynamics are designed to deal with the complexities of parallellism and simultaneity in component interaction; and the descriptive mechanism is bound in a natural-like language. A more detailed explanation of DELTA is not possible within this paper; however, some "flavor" of the language and the program execution environment can be gained from Figure 3, taken from [HOLBE77, p.490].

```
IDSG: SYSTEM BEGIN
      ACTING OBJECTS: a list of all acting objects in
                      the system;
      EFFECTIVE DESCRIPTORS: a list of all effective property
                             descriptors;
      OPERATED VARIABLES: a list of variables for which the
                          system generator should define
                          values;
      SUPERVISED CONDITIONS: a list containing the supervised
                             condition of each concurrency
                             imperative under execution;
      EVENT LIST: a list of EVENT NOTICEs;
      TIME: REAL;
      PRIME TASK BEGIN
          PROCEDURE REGISTER EVENTS : BEGIN  ....  END;
          PROCEDURE EVENT EXECUTION : BEGIN  ....  END;
          generate the initial DELTA system;
          WHILE the system object is acting REPEAT
          (*REGISTER EVENTS;
            WHILE EVENT LIST is not empty REPEAT
            (*choose an EVENT NOTICE from EVENT LIST;
              EVENT EXECUTION PUT (*EVNOTICE :- selected
                                   EVENT NOTICE*);
              PAUSE LET {EFFECTIVE DESCRIPTORS}
                        DEFINE OPERATED VARIABLES;
              REGISTER EVENTS*);
            WHILE all SUPERVISED CONDITIONS are fulfilled
            LET {EFFECTIVE DESCRIPTORS} DEFINE OPERATED
            VARIABLES as the value of TIME increases
            continuously;
          *)
      END TASK;
      END SYSTEM;
```

Figure 3. A formal Description of the Idealised System Generator for a DELTA Program.

V. SUMMARY

The need for model documentation standards is claimed from highly influential sources, and the benefits to be accrued from standardization are considered high. Within this report we have noted the likely perception of increased costs on the part of Federal contractors or any simulation model development group. This important criticism can be tempered by the benefit of reduced costs in model development through the improved specification afforded by the conical methodology. However, the conical methodology is not implementable through a SMSDL at this time.

The means of describing model dynamics in an effective, precise, and succinct form remain the major unresolved problem in the development of a SMSDL. Mathewson [MATHS78] has critiqued the current capabilities of program generators, noting the degree to which the requirements of a SMSDL are approached. We believe that the crucial prerequisite for further progress is a clear, rigorous, general definition of modelling perspectives that enable an understanding of the relationships among the three current simulation language approaches: event scheduling, activity scan, and process interaction. Based on this understanding, we believe that a procedural definition of model documentation could be derived and that measures of language capability for model documentation could be developed. Relative measures of model complexity would provide a flexible scale for documentation requirements in addition to providing a more formal comparison tool.

Beyond the considerable obstacle of describing model dynamics, the constant competitive requirements of sufficient generality and adequate descriptive power remain in the design of a SMSDL. DELTA [HOLBE77] is certainly the most ambitious undertaking to date. Language design is an art, and

the creation of a language as distinct in its objectives and characteristics as a SMSDL is a challenging task for the most artful of designers.

REFERENCES

CLEMA73 Clementson, A. T., "Extended Control and Simulation Language", University of Birmingham, Birmingham, England, 1973.

CLEMA77 Clementson, A. T., "Computer Aided Programming for Simulation", University of Birmingham, 1973, (date verified by private correspondence of June, 1977).

DAVIN76a Davies, N. R., "A Modular Interactive System for Discrete Event Simulation Modeling", Proceedings of the Ninth Hawaii International Conference on System Sciences, January 1976, 296.

DAVIN76b Davies, N. R., "On the Information Content of a Discrete Event Simulation Model", Simulation, Vol. 27, No. 4, October 1976, pp. 123-128.

DECAR76 DeCarvalho, R. S. and J. G. Crookes, "Cellular Simulation", Operations Research Quarterly, Vol. 27, No. 1, 1976, pp. 31-40.

ETSCM72 Etschmaier, M. M., "Mathematical Modelling for the Simulation of Discrete Systems", Technical Report No. 3, Department of Industrial Engineering, University of Pittsburgh, June 1972.

HEIDG74 Heidorn, G. E., "English as a Very High Level Language for Simulation Programming", Proceedings of the Symposium on Very High Level Languages, SIGPLAN Notices, Vol. 9, No. 4, April 1974, pp. 91-100.

HOLBE77 Holbaek-Hanssen, Erik, Petter Handlykken, and Kristen Nygaard, System Description and the DELTA Language, Report No. 4 (Publication no. 523), second printing, February 1977.

HUTCG75 Hutchinson, G. K., "An Introduction to CAPS - Computer Aided Programming in Simulation", draft paper dated February 28, 1975 (received from author).

JONEM68 Jones, Malcom M., "Incremental Simulation on a Time-Shared Computer", Unpublished Ph.D. thesis, Project MAC Report TR-48, MIT, 1968.

KIVIP69 Kiviat, P. J., "Digital Computer Simulation: Computer Programming Languages", RAND Report RM-5883-PR, January 1969.

KLEIH77a Kleine, Henry, "SDDL" Software Design and Documentation Language", Publication 77-24, Jet Propulsion Laboratory, Pasadena, California, July 1, 1977.

KLEIH77b Kleine, Henry, "A Vehicle for Developing Standards for Simulation Programming", Proceedings 1977 Winter Simulation Conference, Gaithersburg, Maryland, December 5-7, 1977, pp. 730-741.

LACKM62 Lackner, M. R., "Toward a General Simulation Capability", Proceedings of the SJCC, May 1962, pp. 1-14.

LACKM64a Lackner, M. R. and P. Kribs, "Introduction to a Calculus of Change", System Development Corp. TM-1750/000/01, February 12, 1964.

LACKM64b Lackner, M. R., "Digital Simulation and System Theory", System Development Corp. SP-1612, April 6, 1964.

LAFFH73 Lafferty, H. H., "Sheffield Simplified Simulation System - S4 Manual", The University of Sheffield, Department of Applied Mathematics and Computing Science, January 10, 1973.

LONDK74 London, K. R., Documentation Standards, Petrocelli Books, New York, 1974.

MATHS74 Mathewson, S. C., "Simulation Program Generators", Simulation, Vol. 23, No. 6, December 1974, pp. 181-189.

MATHS76 Mathewson, S. C. and J. E. Beasley, "DRAFT/SIMULA", Proceedings Fourth Simula Users Conference, National Computer Conference, 1976.

MATHS77a Mathewson, S. C. and J. H. Allen, "DRAFT/GASP - A Program Generator for GASP", Proceedings Tenth Annual Simulation Symposium, Tampa, 1977.

MATHS77b Mathewson, S. C. DRAFT, Department of Management Science, Imperial College of Science and Technology, 1977.

MATHS77c Mathewson, S. C., "Technical Comment 'On the Information Content of a Discrete-Event Simulation Model'", Simulation, March 1977, p. 96.

MATHS78 Mathewson, S. C., "Computer Aided Simulation Modelling and Experimentation," Proceedings of the Eighth Australian Computer Conference, August 28-September 1, 1978 (privately communicated).

MCLEJ70 McLeod, J., "Toward Uniform Documentation - PHYSBE and CSMP", Simulation, Vol. 14, No. 5, May 1970, pp. 215-220.

MCLEJ73 McLeod J., "Simulation: From Art to Science for Society", Simulation, Vol. 21, No. 6, December 1973 (SIMULATION TODAY, Center Section).

MENKB70 Menkus, B., "Defining Adequate Systems Documentation", Journal of Systems Management, Vol. 21, No. 12, December 1970, pp. 16-21.

NANCR72 Nance, R. E., "Towards A Unifying Structure for Discrete Event Simulation", SIMULETTER, Vol. 3, No. 2, January 1972, pp. 4-9.

NANCR77 Nance, Richard E., "The Feasibility of and Methodology for Developing Federal Documentation Standards for Simulation Models", Final Report to the National Bureau of Standards, June 26, 1977.

NANCR79 Nance, Richard E., "Model Representation in Discrete Event Simulation: II. Fundamental Constructs in Model Specification", in preparation.

NATIS76 National Bureau of Standards, "Guidelines for Documentation of Computer Programs and Automated Data Systems", Federal Information Processing Standards Publications (FIPS) 38, Washington, D.C., February 15, 1976.

NELSS77 Nelson, Sallie S., "Control Issues in the Development of a Conversational Simulation Language," Unpublished Ph.D. thesis, U. Pittsburgh, 1977.

OLDFP66 Oldfather, P. M., A. S. Ginsberg and H. M. Markowitz, "Programming by Questionnaire: How to Construct a Program Generator", RAND Report RM-5129-PR, November 1966.

OLDFP67 Oldfather, P., A. S. Ginsberg, P. L. Love, and H. M. Markowitz, "Programming by Questionnaire: The Job Shop Simulation Program Generator", RAND Report RM-5162-PR, July 1967.

STEVW74 Stevens, W. P., G. J. Myers and L. L. Constantine, "Structured Design", IBM Systems Journal, Vol 13, No. 2, 1974, pp. 115-139.

USGAO75 U.S. GAO, "Ways to Improve Management of Federally Funded Computerized Models", LCD-75-111, Washington, D.C., August 23, 1976.

ZEIGB72 Zeigler, B. P., "Towards a Formal Theory of Modeling and Simulation: Structure Preserving Morphisms", Journal ACM, Vol. 19, No. 4, October 1972, pp. 742-764.

ZEIGB73 Zeigler, B. P., "A Conceptual Basis for Modeling nd Simulation", SIMULETTER, Vol. 5, No. 1, October 1973, pp. 23-32.

ZEIGB75a Zeigler, B. P., "Introduction to a Theory of Modeling and Simulation", Proceedings of the Sixth Annual Pittsburgh Conference on Modelling and Simulation, 1975.

ZEIGB75b Zeigler, B. P., "Postulates for a Theory of Modelling and Simulation", Proceedings Summer Computer Simulation Conference, Vol. 1, 1975, pp. 49-54.

ZEIGB76 Zeigler, B. P., Theory of Modelling and Simulation, John Wiley & Sons, New York, 1976.