The Conical Methodology: A Framework for Simulation Model Development

Richard E. Nance

TR 87-8

Technical Report SRC-87-002*

THE CONICAL METHODOLOGY: A FRAMEWORK FOR SIMULATION MODEL DEVELOPMENT†

Richard E. Nance

Systems Research Center and
Department of Computer Science
Virginia Tech
Blacksburg, VA 24061

December 1986

* Cross-referenced as Technical Report TR-87-8, Department of Computer Science, Virginia Tech, Blacksburg.

† To appear in the *Proceedings of the Conference on Simulation Methodology and Validation* to be held as part of the *1987 Eastern Simulation Conference* (Orlando, FL, 6-9 April). The Society for Computer Simulation, San Diego, Calif.

Technical Report TR-87-8*

THE CONICAL METHODOLOGY: A FRAMEWORK FOR SIMULATION MODEL DEVELOPMENT†

Richard E. Nance

Department of Computer Science
and Systems Research Center
Virginia Tech
Blacksburg, VA 24061

December 1986

* Cross-referenced as Technical Report SRC-87-002, Systems Research Center, Virginia Tech, Blacksburg.

† To appear in the *Proceedings of the Conference on Simulation Methodology and Validation* to be held as part of the *1987 Eastern Simulation Conference* (Orlando, FL, 6-9 April). The Society for Computer Simulation, San Diego, Calif.

ABSTRACT

The Conical Methodology, intended for large discrete event simulation modeling, is reviewed from two perspectives. The designer perspective begins with the question: What is a methodology? From an answer to that question is framed an inquiry based on an objectives/principles/attributes linkage that has proved useful in evaluating software development methodologies. The user perspective addresses the role of a methodology vis a vis the software utilities (the tools) that comprise the environment. Principles of a methodology form the needs analysis by which the requirements for tool design can be derived. A comparison with software development methodologies and some applications of the Conical Methodology comprise the concluding summary.

CR Categories and Subject Descriptors: I.6.1 [**Simulation and Modeling**]: Simulation Theory; I.6.m [**Simulation and Modeling**]: Miscellaneous

Additional Key Words and Phrases: Conical methodology, model development, model specification

1. INTRODUCTION

The need for an orderly, disciplined process for simulation analysis is well accepted. The means for introducing and imposing structure, the degree of formality, and the necessity for rigorous adherence to process guidelines are issues that remain unresolved. The Conical Methodology [Nance 1981] represents one approach to model development, intended from its inception to be utilized in large simulation modeling and experimentation.

1.1. Chronology of the Conical Methodology

The roots of the Conical Methodology (CM) are found in an effort to assess the feasibility of developing standards for simulation model documentation [Nance 1977]. A complementary project investigated standards for model documentation in general [Gass 1979]. The rationale for the CM is expressed in a set of hypotheses, which in abbreviated form stipulate that:

(1) model specification and model documentation should be accomplished within the same activity (inseparable),

(2) a hierarchical structure for model development is advantageous, and

(3) the clarification of nomenclature is a necessary precursor to documentation standards.

The hierarchical nature of the "simulation model specification/documentation structure" is illustrated [Nance 1977, p.16], and from it is derived the description "conical."

The characteristics of a Simulation Model Specification and Documentation Language (SMSDL) are described in the 1977 report. Three aspects of the modeling task supported by a SMSDL are apparent in this early description:

(1) a clear separation between *model specification* (development) and *model implementation* (execution),

(2) the characterization of model components as *objects,* and

(3) the description of objects and the relationships among objects in terms of *attributes.*

The structure and conceptual basis of the CM emerge during the doctoral research of Overstreet, and without a doubt he has contributed significantly to the current form through several ideas described in his dissertation [Overstreet 1982]. Views of the evolving methodology are provided

in [Nance 1979; Nance 1981a], and the explication, with definitions, principles, and justification, is given in [Nance 1981b].

Since 1983 the definition of a Model Development Environment (MDE) based on the CM has involved Osman Balci in a pivotal role. Suggested extensions, clarifications, and refinements, that survive the intellectual hammering of Balci, Overstreet, and others, now progress to empirical evaluation in a prototype suite of software tools [Moose 1983; Hansen 1984; Nance et al. 1984]. At this juncture a status review with critical commentary seems especially appropriate.

1.2. Organization

The treatment to follow is narrowly focused on the Conical Methodology. The earlier report [Nance 1981] refers to several approaches to simulation model development, comparing the alternatives in terms of the phases of the model life cycle addressed by each. Among the more recent papers are several treating simulation analysis from a methodological perspective, comparing modeling approaches in general, or noting software development implications [Barger and Nance 1986; Hooper 1986; Geoffrion 1986]. Hopefully, the decision to draw a rather narrow bead in this work, to conform with both space and subject constraints, is understood and accepted.

Section 2 provides a review of the CM, primarily from the perspective of the designer, and concentrates on the *raison d'etre* of a methodology (any methodology). The user view is taken in Section 3, which explores the dual contribution of principles and their foundational role in a simulation support environment. The final section takes a commentary form, responding to the questions about the differences between model and software development and citing cases where the CM has been applied.

2. METHODOLOGY REVIEW: THE DESIGNER PERSPECTIVE

The terms "method," "approach," "technique," and "methodology" are frequently used in what appears to be an indiscriminate or interchangeable fashion. This is unfortunate, for although related, the terms do connote differences in meaning.

2.1. What Is a Methodology?

We consider "methodology" to be the most inclusive of the above terms; that is, a methodology [Arthur et al. 1986, p. 4]:

(1) organizes and structures the tasks comprising the effort to achieve global objectives,

(2) includes methods and techniques for accomplishing individual tasks (within the framework of global objectives), and

(3) prescribes an order in which certain classes of decisions are made and the ways of making those decisions that lead to the desired objectives.

Particularizing with respect to simulation, a methodology should respond to the needs of the simulation user (manger, modeler, analyst, programmer) by identifying the objectives and by guiding the modeler in the execution of the process to achieve those objectives.

In the paragraphs to follow, we characterize a methodology as: (1) enunciating clearly defined *objectives* and (2) defining the *principles* to be employed to achieve those objectives (see [Arthur et al, 1986]). Further, the principles employed should produce recognized *attributes* in the product. The principle/attribute relationship provides a basis for scoping the needs for software utilities and designing the requisite assistance tools.

2.2. Simulation Analysis: The Methodological Needs

Drawing from the software engineering experience, the model life cycle is considered to be the initiation of the methodological needs. Figure 1 from [Nance 1984, p. 76] is an expansion of an earlier version that stresses the model development phases [Nance et al. 1981]. The Conical Methodology focuses on the model development phases — the circular set of activities beginning with the system and objectives definition. Within these phases are the model construction, verification, and validation processes. The role of the CM is defined in terms of the guidance imparted to these processes.

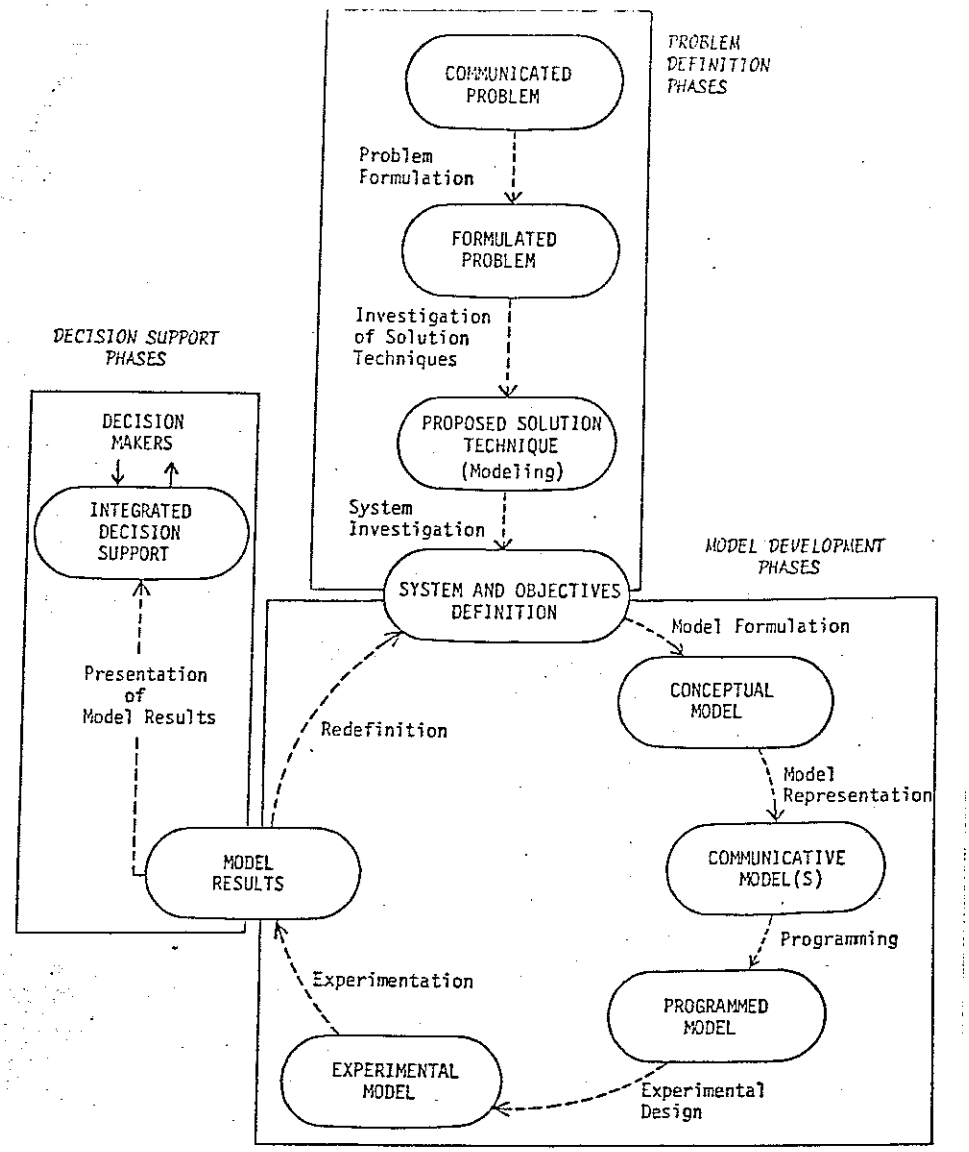


Figure 1. Phases in the Chronological Periods of the Model Life Cycle

2.2.1. Objectives of the Conical Methodology

The model development phases, from system and objectives definition to experimental model, can be properly viewed as *model transformations,* wherein both the input and output of each phase are model specifications. The progression toward the experimental model consists of a series of processes (albeit with provisions for considerable feedback) through which non-procedural forms are transformed into an ultimate procedural and executable form. The Conical Methodology is a conceptual "blueprint" for conducting these processes to achieve five primary objectives:

(1) *Model correctness*: The capability for assuring that the model behavior "mimics" the system behavior within some specified tolerance consistent with the modeling requirements.

(2) *Testability*: the comparison between model specifications in different forms and between the model and system behavior.

(3) *Adaptability*: the ability to change the successive model specifications with relative ease so that the applicability can be rapidly extended with little cost.

(4) *Reusability*: the ability to extract model components and reuse them in subsequent modeling tasks.

(5) *Maintainability*: the property of model specifications that enables their modification to meet originally unstated needs.

The CM prescribes a top-down *model definition stage,* illustrated in an outline format in Figure 2. The definition stresses:

(1) model decomposition — partitioning a model into component submodels,

(2) assignment of attributes to the objects defined (submodels), and classification of each attribute by

(3) *type,* according to the taxonomy tree shown in Table 1.

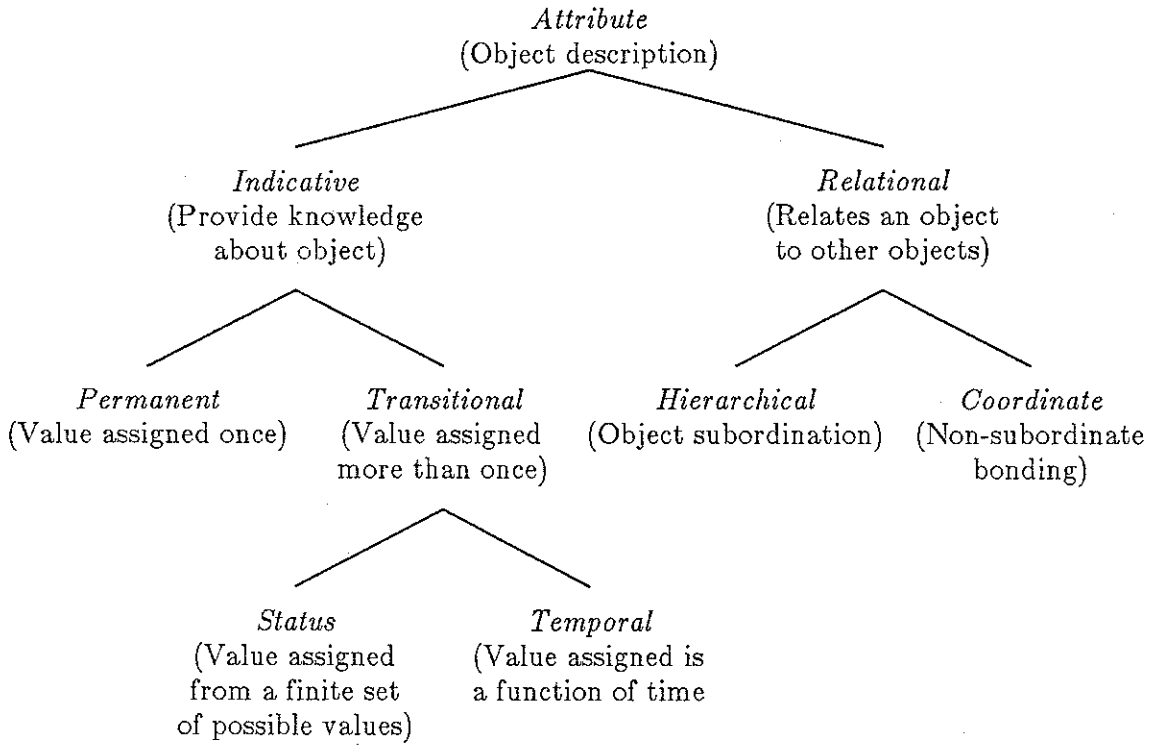
Attribute dimensions and the range of attribute values are additional information extracted during the definition process. In essence, a static description of the model is produced in the definition stage.

Model dynamics are imparted in the *specification stage,* during which the modeler stipulates the expression evaluation for each defined attribute. The typing, dimensionality, and value information supplied by the modeler enable subsequent diagnosis for consistency (type) and correspondence (dimensions and value range).

2.2.2. Conical Methodology Principles

Principles form the nucleus of a methodology. From principles are derived the directions and procedural guidance that enable achievement of the targeted objectives. The Conical Methodology *emphasizes* the principles described in Table 2. (Note that other principles may be employed but are not emphasized.) To assist in the understanding of the principles, Table 2 also includes the instructive guidance derived from each principle and identifies related software engineering principles.

Table 1. The Tree of Conical Methodology Types



3. METHODOLOGY SUPPORT: THE USER PERSPECTIVE

The CM adheres to the philosophy of direct and open communication with users. Practically, this statement means that both the objectives and principles are presented, explained, and even justified to the user (or prospective user). The novice user may not appreciate the significance, but growing appreciation is likely to accompany experience.

3.1. The Dual Contribution of Principles

Recent work in the evaluation of software development methodologies draws a causality linkage among objectives, principles, and *attributes* [Arthur et al. 1986]. Briefly, the authors argue that the achievement of a specific objective relies on conformance with designated principles that result in the presence of desirable attributes in the software (programs and documentation). An approach to

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

Table 2. The Conical Methodology Principles

Conical Methodology Principle	Procedural Guidance Derived From the CM Principle	Related Software Engineering Principles
(1) Top-down model definition is followed by bottom-up model specification.	(1.1) Definition must precede specification.	Hierarchical Decomposition
	(1.2) The transition from specification to definition must be swift and easy.	Functional Decomposition
(2) Documentation and specification are inseparable.	(2.1) Model documentation is produced in model specification.	Concurrent Documentation
	(2.2) The model specification and consequent documentation should support different views (aspects) of the modeling task.	
(3) Iterative refinement and progressive elaboration are essential in large modeling efforts.	(3.1) The degree of detail in submodel description should be controlled by the modeler. Submodel "stubbing" should be supported so that later addition of detail is facilitated.	Abstraction Information Hiding Hierarchical Decomposition Functional Decomposition Stepwise Refinement
	(3.2) The functional expansion (progressive elaboration) of the model should be supported.	
(4) Verification must begin with communicative models and continue throughout the development process.	(4.1) Diagnosis of model representations should begin as early as possible, certainly prior to the program form.	Life Cycle Verification
	(4.2) Automated or semi-automated diagnosis is a requirement.	
(5) Model specification should be independent of model implementation.	(5.1) The execution (implementation) details should be ignored in the model development (specefication) process.	Abstraction Division of Concerns

empirical examination gathering evidence of the presence or absence of the attributes is described.

The above claim that principles form the nucleus of a methodology is substantiated in the evaluation procedure for software development methodologies. Examination of the middle column of Table 2 reveals basic beliefs about how the modeling activity *should be* carried out.

Equally important is the contribution of principles to the statement of requirements for *tools*

to enable the modeling activity to be done properly. The middle column of Table 2 essentially lays out the design goals for the software utilities needed in an environment supporting model development. Thus, the principles state *how* the objectives of the CM are to be achieved and at the same time *what* is needed so that the development process can realize those objectives.

3.2. Environments: The Integration of Simulation Utilities

> A MDE (model development environment) should provide an integrated and complete collection of computer-based tools which offer continuous and cost-effective support to all phases, processes and stages of the model development life cycle.... A MDE should implement a model development methodology [Balci 1986].

All too often a project or an organization, driven by one, two or three existing utilities, attempts to structure an "environment" around these tools. In the worst of circumstances the existing tools are not compatible. Such an undertaking is doomed to failure. The key terms in the above quotation are "integrated" and "cost-effective." A true "environment" can exist only with both of these qualities.

The Conical Methodology has served as the foundation for a prototype MDE under development since 1983. Key characteristics of this environment are illustrated in Figure 3, which depicts a layered architecture that at the level of the Minimal Model Development Environment (MMDE) includes eleven software tools. Communication among the tools uses the Kernel Interface, which relies on the kernel (KMDE) functions layered over the host computer system. An in-depth discussion of the component tools is beyond the scope of this paper; the interested reader should consult [Nance et al. 1984] or [Balci 1986].

The principles of the CM and, more specifically, the derived procedural guidance shown in Table 2 have influenced the definition of simulation support tools, for example

- the identification of a Model Analyzer that includes the diagnostic analysis to be applied to communicative model specifications long before an executable version exists [Overstreet and Nance 1986] , and
- the stipulation of a Model Translator to be applied to a communicative model (specification) to produce an executable version (implementation).

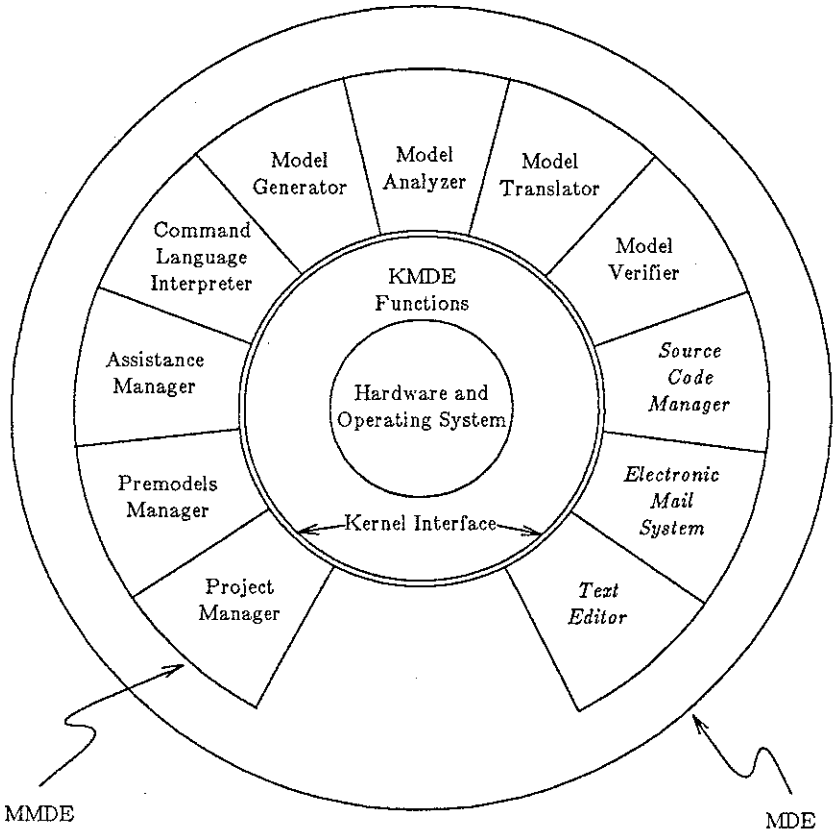


Figure 3. The Structure of Model Development Environments.

The influence of the CM is even more apparent in a tool such as the Model Generator, where the top-down definition followed by bottom-up specification is clearly enforced.

What is most significant in the message conveyed in this section (and perhaps the entire paper) is the necessity for maintaining the *distinction between the methodology and an environment.* The Conical Methodology exists as a set of principles from which is derived the specification of software to assist simulation users. The user sees the software tools first-hand. If what is seen is unappealing or perceived as unhelpful, no argument based on principles is likely to be convincing.

4. A CONCLUDING COMMENTARY

Prescribing a methodology for accomplishing any task is an inherently presumptuous undertaking. It smacks of an evangelical message that this is *the* way. Critics are quick to respond with comments such as:

(1) It looks like nothing more than _____ with different terms or a few more claims.

(2) If it is so good, what has it been used for?

We conclude with a few remarks directed at these understandable reactions.

4.1. The CM Resembles Software Development Methodologies

The Conical Methodology shares some features with software development methodologies, the most obvious being the top-down approach to model (program) development. The similarities between modeling and programming have only recently achieved wide recognition in the software engineering and programming languages communities. With the exception of Lehman, who has long held the view that program development should be viewed as a modeling activity [Lehman 1980], the remainder of the community has awaited the recognition of the applicability of the entity-relationship (E-R) modeling technique, generally attributed to Peter Chen [Chen 1976], and the object oriented paradigm (see [Unger 1986] for an historical perspective). The current trend is toward the perception of programming and software development as a problem-solving activity just as is modeling.

The Conical Methodology casts the programming activity as subordinate to the modeling activity. The *programmed model* is developed from preceding *communicative models,* and software engineering techniques may not apply in producing the earlier representations. Additionally, the CM differs noticeably from software development methodologies in:

(1) the explicit relationship between top-down and bottom-up construction techniques,

(2) the emphasis on model and data validation,

(3) the inclusion of experimental design and statistical analysis requirements, and

(4) the explicit relationship between documentation and specification.

4.2. Applications of the Conical Methodology

The CM has been applied in the development of numerous small or "toy-size" models that have been used for academic instruction or examples (see [Overstreet and Nance 1985] for example). However, it has proved capable of modeling the data transfer laboratory requirements for Naval experimentation [Nance 1982], the evaluation of personal computers for an acquisition decision [Evaluation Committee 1985], and the design of an expert system shell [Lee 1985].

The major test of the CM remains: the success of the model development environment for which it provides the architectural framework. Early experiences with the prototypes are encouraging, and we anticipate positive results in the future expansion of the simulation support environment.

ACKNOWLEDGMENTS

This research was sponsored in part by the Naval Sea Systems Command and the Office of Naval Research under Contract N60921-83-G-A165 through the Systems Research Center at Virginia Tech.

REFERENCES

Arthur, J.D., R.E. Nance, and S.M. Henry (1986), "A Procedural Approach to Evaluating Software Development Methodologies: The Foundation," Technical Report SRC-86-008, Systems Research Center, Virginia Tech, Blacksburg, Va., Sept.

Balci, O. (1986), "Requirements for Model Development Environments," *Computers & Operations Research 13,* 1 (Jan. - Feb.), 53-67.

Barger, L.F. and R.E. Nance (1986), "Simulation Model Development: System Specification Techniques," Technical Report SRC-86-005, Systems Research Center, Virginia Tech, Blacksburg, Va., Aug.

Chen, P.P. (1976), "The Entity-relationship Model — Toward a Unified View of Data". *ACM Transactions on Database Systems 1,* 1 (Mar.), 9-36.

Evaluation Committee (1985), "A Questionaire for Evaluating Personal Computers for Computer Science Students at Virginia Tech," Department of Computer Science, Virginia Tech, Blacksburg, Va., Sept.

Gass, S.I. (1979), Computer Science and Technology: Computer Model Documentation: A Review and an Approach," National Bureau of Standards Pub. No. 500-39, Washington, D.C., Feb.

Geoffrion, A.M. (1986), "Modeling Approaches and Systems Related to Structured Modeling," Working Paper No. 339, Western Management Science Institute, UCLA, Los Angeles, July.

Hansen, R.H. (1984), "The Model Generator: A Crucial Element of the Model Development Environment," Technical Report SRC-85-004, Systems Research Center, Virginia Tech, Blacksburg, Va., Aug.

Hooper, J.W. (1986), "Language Assessment Criteria for Discrete Simulation," In *Proceedings of the 1986 Winter Simulation Conference* (Washington, D.C., Dec. 8-10), IEEE, Piscataway, N.J., pp. 404-408.

Lee, N.S. (1985), "GUESS1: A General Purpose Expert Systems Shell," Master's Thesis, Department of Computer Science, Virginia Tech, Blacksburg, Va., Mar.

Lehman, M. M. (1980), "Programs, Life Cycles, and Laws of Software Evolution," *Proceedings of the IEEE 68*, 9 (Sept.), 1060-1076.

Moose, R.L., Jr. (1983), "Proposal for a Model Development Environment Command Language Interpreter," Technical Report SRC-85-012, Systems Research Center, Virginia Tech, Blacksburg, Va., Dec.

Nance, R.E. (1977), "The Feasibility of and Methodology for Developing Federal Documentation Standards for Simulation Models," Final Report to the National Bureau of Standards, Department of Computer Science, Virginia Tech, Blacksburg, Va., June.

Nance, R.E. (1979), "Model Representation in Discrete Event Simulation: Prospects for Developing Documentation Standards," In *Current Issues in Computer Simulation,* N.R. Adam and A. Dogramaci, Eds. Academic, New York, pp. 83-97.

Nance, R.E. (1981a), "The Time and State Relationships in Simulation Modeling," *Communications of the ACM 24,* 4 (Apr.), 173-179.

Nance, R.E. (1981b), "Model Representation in Discrete Event Simulation: The Conical Methodology," Technical Report CS81003-R, Department of Computer Science, Virginia Tech, Blacksburg, Va., Mar.

Nance, R.E., A.L. Mezaache, and C.M. Overstreet (1981), "Simulation Model Management: Resolving the Technological Gaps," In *Proceedings of the 1981 Winter Simulation Conference* (Atlanta, Ga., Dec. 9-11), IEEE, Piscataway, N.J., pp. 173-179.

Nance, R.E. (1982), "Data Transfer Architectures: Development of the Capability for Comparative Evaluation," NSWC Technical Report 82-345, Naval Surface Weapons Center, Dahlgren, Va., Mar.

Nance, R.E., O. Balci, and R.L. Moose, Jr. (1984), "Evaluation of the UNIX Host for a Model Development Environment," In *Proceedings of the 1984 Winter Simulation Conference* (Dallas, Tex., Nov. 28-30), IEEE, Piscataway, N.J., pp. 577-584.

Overstreet, C.M. (1982), *Model Specification and Analysis for Discrete Event Simulation,* PhD Dissertation, Department of Computer Science, Virginia Tech, Blacksburg, Va., Dec.

Overstreet, C.M. and R.E. Nance (1985), "A Specification Language to Assist in Analysis of Discrete Event Simulation Models," *Communications of the ACM 28,* 2 (Feb.), 190-201.

Overstreet, C.M. and R.E. Nance (1986), "World View Based Discrete Event Model Simplification," In *Modelling and Simulation Methodology in the Artificial Intelligence Era,* M.S. Elzas, T.I. Ören, and B. P. Zeigler, Eds., North-Holland, pp. 165-179.

Unger, B.W. (1986), "Object Oriented Simulation — Ada, C++, Simula," In *Proceedings of the 1986 Winter Simulation Conference,* (Washington, D. C., Dec. 8-10), IEEE, Piscataway, N.J., pp. 123-124.

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: SRC 87-002/ (CS TR-87-8)	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): The Conical Methodology: A Framework for Simulation Model Development	5. TYPE OF REPORT & PERIOD COVERED: Interim
6. PERFORMING ORG. REPORT NUMBER: SRC 87-002/ (CS TR-87-8)	
7. AUTHOR(s): Richard E. Nance	8. CONTRACT OR GRANT NUMBER(s): N60921-83-G-A165
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Systems Research Center and Dept. of Com. Sci. Virginia Tech Blacksburg, VA 24061	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Sea Systems Command SEA61E Washington, D.C. 20362	12. REPORT DATE: December 1986
	13. NUMBER OF PAGES: 17
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office): Naval Surface Weapons Center Dahlgren, VA 22448	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This report is distributed to scientists and engineers at the Naval Surface Weapons Center and is made available on request to other Navy scientists. A limited number of copies is distributed for peer review.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

To Navy research and development centers and university-based laboratories.

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Simulation, Model Development, Methodology, Software Engineering, Environments, Principles, Objectives, Conical Methodology, Model Specification

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The Conical Methodology, intended for large discrete event simulation modeling, is reviewed from two perspectives. The designer perspective begins with the question: What is a methodology? From an answer to that question is framed an inquiry based on an objectives/principles/attributes linkage that has proved useful in evaluating software development methodologies. The user perspective addresses the role of a methodology vis a vis the software utilities (the tools) that comprise the environment. Principles of a methodology form the needs analysis by which the requirements for tool design can be derived. A

comparision with software development methodologies and some applications of the Conical Methodology comprise the concluding summary.