Proceedings of the 1987 Winter Simulation Conference
A. Thesen, H. Grant, W. David Kelton (eds.)

THE CONTROL AND TRANSFORMATION METRIC: TOWARD THE MEASUREMENT OF SIMULATION MODEL COMPLEXITY

Jack C. Wallace
Georgia Tech Research Institute
Georgia Institute of Technology
Atlanta, GA 30332, U.S.A.

ABSTRACT

Current complexity metrics based upon graphical analysis or static program characteristics are not well suited for discrete event simulation model representations, owing to their inherent dynamics. This paper describes a complexity metric suitable for model representations. A study of existing metrics provides a basis for the development of the desired metric, and a set of characteristics for a simulation model complexity metric is defined. A metric is described based on the two types of complexity that are apparent in model representations. Experimental data are presented to verify that this metric possesses the desired characteristics. Based on evaluation of this data and the desired characteristics, this metric appears to offer an improvement over existing metrics.

1. INTRODUCTION

1.1 Background

Current complexity metrics based on graphical analysis or static program characteristics (such as the number of lines of code) are not well suited for simulation model representations, due to the inherent dynamics of models. A metric suitable for models represented by Condition Specifications or World View Specifications (described below) is developed in this paper. The use of these representational forms is motivated by the desire to introduce informative diagnosis early in the modeling process. These forms "introduce an intermediate form between a conceptual model (the model as it exists in the mind of the modeler) and an implementation of that conceptual model. A model implementation, even using a simulation programming language, often includes many factors ... which may obscure the conceptual model being implemented". (Overstreet 1985, p.200)

To present what follows, some preliminary definitions are useful. The following definitions are taken from Overstreet (1982).

1.2 Definitions

A condition specification (CS) consists of two components; a description of the communication interface for the model and a specification of the dynamics of the model. The communication interface can be derived from the description of internal dynamics for the model and is assumed to be system generated. The internal dynamics of the model are described (formed) by a set of ordered pairs called condition action pairs (CAPs). Figure 1 shows an example of a condition specification.

An Action Cluster (AC) is a collection of CAPs with the same condition. An AC specifies a model action.

Action Cluster Name: Action(s)
Condition

Initialization:
initialization
	READ (n,max_repairs, meanuptime, meanrepairtime);
	CREATE (repairman);
	FOR i = 1 (TO n DO
	CREATE (facility(i));
	failed(i) : = false;
	SET ALARM (failure(i), Neg-exp (meanuptime));
	END for;
	num_repairs : = 0;
	location : = idle;
	status: = avail;

Termination: num_repairs > = max_repairs	STOP;
Failure (i:1. . n): WHEN (failure(i))	failed(i) : = true;

Begin_Repair (i:1 ..n):
WHEN (arrfac)
	SET ALARM (endrepair, Neg_exp (meanuptime));
	status : = busy;
	location : = fac(I);

End_Repair (i:1 ..n):
WHEN (endrepair)
	SET ALARM ((failure, Neg_exp (meanuptime));
	failed (i) : = false;
	status : = avail;
	num_repairs : = num_repairs + 1

Travel_to_Idle: (FOR ALL 1< +i< +n, failed(I)) & status = avail & location = idle	SET ALARM (arrival, traveltime (location, idle)): status : = travel;
Arrive_Idle: WHEN (arrival)	status : = avail; location : = idle;

Travel_to_Facility:
status = avail &
(FOR SOME 1< = i< = n,failed(i))
	i : = Closest_failed_fac (location);
	SET ALARM (arrfac, traveltime(location, fac(j)));
	status : = travel;

Figure 1: Minimal Distance Repairman Condition Specification

A control attribute of an AC is any attribute that appears in the condition of the AC. A control attribute can be either time-based or state-based.

An output attribute of an AC is any attribute that is modified by the actions of the AC. (An attribute can be both a control and output attribute for a given AC.)

An Action Cluster Incident Graph (ACIG) is a graph

representing the relationships among ACs in the model. For each AC X in the CS, a set of ACs can be identified that can occur as a result of the actions of X. An ACIG can be algorithmically constructed to represent the interconnections among action clusters. This graph may include connections (arcs) which could never occur. No algorithm can exist to produce a completely simplified ACIG (Overstreet 1982). However, extensive simplification can be done and a reasonable graph is easily constructed. Throughout this paper, all ACIGs are assumed to be simplified. Figure 2 shows the ACIG corresponding to the CS in figure 1.

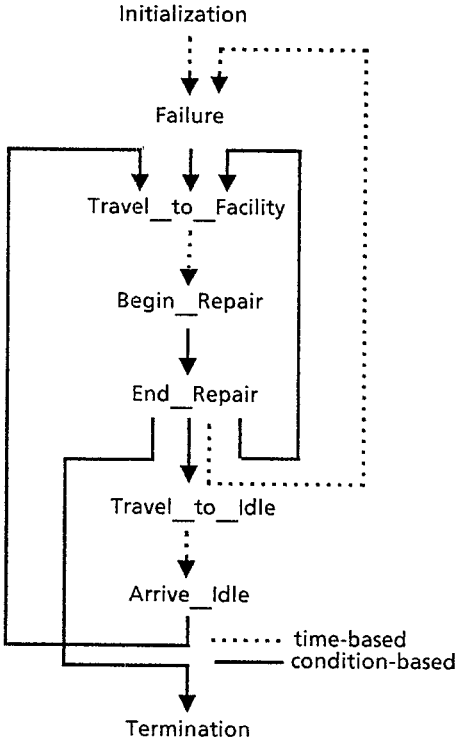

Figure 2: Minimal Distance Repairman Action Cluster Incidence Graph

A node in an ACIG represents an AC.

World views are the different sets of representational constructs that provide ways of describing model behavior. A CS that has been translated into one of the world views is referred to as a World View Specification (WVS) described in Overstreet and Nance (1986). Three world views are widely recognized and used (Kiviat 1969). Each of the world views provides a different type of locality. Weinberg defines locality as "that property when all relevant parts of a program are found in the same place" (Weinburg 1971, p.229). The world views are described in the following paragraphs. A CS can be algorithmically transformed into the three world views as shown in Overstreet and Nance (1986).

In an event scheduling world view, the modeler specifies when actions are to occur in a model. This world view is based upon the scheduling of actions at certain points in time (locality of time).

In an activity scanning world view, the modeler specifies why actions are to occur in a model. This world view is based upon the reasons that cause model actions to occur (locality of state).

In a process interaction world view, the modeler specifies the components (or subsystems) of a model and describes the actions of each component individually. This world view is based upon the definition of system behavior by the definition of the action sequences of each component (locality of object).

2. IN SEARCH OF A METRIC

2.1 Desired Characteristics

The following paragraphs describe some of the characteristics required of a complexity metric. These requirements are identified to meet the specific needs of a modeler. Significant among these needs is the form of the model representations to be used.

Psychological Complexity. The metric must measure psychological complexity as opposed to computational complexity. As defined by Curtis (1980, p.1147): computational complexity "relies on the formal mathematical analysis of such problems as algorithm efficiency and use of machine resources," while psychological complexity "is concerned with the characteristics of software which affect programmer performance".

Compared with psychological complexity, a computational complexity metric is easier to develop and validate. However, the life-cycle costs of a model are often more dependent on factors influencing the difficulty in modifying a model or establishing credibility. Computational complexity is directly related to execution efficiency (time and space), but may be unrelated to maintaining, modifying or understanding a model, all of which are prominent in model life-cycle costs. A programmed model should run as efficiently as possible, but absolute measures vary from computer to computer and translator to translator.

A psychological complexity metric cannot be proven correct conclusively. Opinions of what constitutes psychological complexity (complexity of understanding) vary. A program (model) property that causes someone to view the program (model) as being complex may actually be perceived to introduce simplicity by others. Such differences in opinion complicate the development and validation of a metric. Despite the absence of a uniform view of complexity, the factors contributing to it, and the dependency on representational forms, measures of model complexity, particularly psychological complexity, are essential in guiding the model development task and estimating the effort required.

Generality of Use. The metric must be usable not only on a CS but also on WVSs. This generality aids a modeler in the decision on which world view to use for implementation in a simulation programming language. The desired metric cannot be depended upon to give absolute information on which world view (or language) executes most efficiently, but should give information on

how understandable a programmed representation of the model may be.

Lower Measures for Condition Specification. The metric should give lower values for a CS than for any corresponding WVS, since the CS is the most primitive representation. Each node in a WVS consists of one or more nodes from the CS. Each of the CS nodes "retains" all arcs when translated.

Dynamic Characteristics. Simulation models are inherently dynamic; therefore, the metric must attempt to assess dynamic characteristics and not be limited to static properties of a model representation such as lines of code or number of operators. Static characteristics must be considered when they affect understandability. The metric must measure in some way the temporal transitions ("flow") of the model from one state to another.

Simple and Understandable. The metric must be simple and comprehensible. Confidence in a metric is reinforced by intuitive appeal. A metric which is difficult to explain or which depends on vague properties is unlikely to find support. An objective of the Model Development Environment (MDE) Project at Virginia Tech (under which this metric was developed) is to create modeling tools that are powerful yet simple to comprehend and to provide a modeler with as much support as possible (perhaps described as maximum aid with minimum effort). An overly complicated metric violates this idea. How helpful can a metric be if the user does not understand the basis of the metric?

Baseline. The metric should admit a potential baseline measure. If the metric is used on just one model and not used to compare two models (or versions of a model), can some information still be gained by the modeler? Can a modeler determine if the model is sufficiently complex to justify modification or respecification? Or is the model "simple" enough to justify continuing toward implementation? The metric should be in a form such that, given a model representation, a lower limit on the complexity of the model can be determined. This limit need not be the lower bound on the complexity. In fact, it does not seem possible algorithmically to determine a lower bound on the complexity of a model representation for a given conceptual model.

2.2 Considerations in Metric Definition

Depending on the intended application, any metric is likely to exhibit some inadequacy. The challenge is to construct a metric that incorporates as many desirable characteristics as possible while avoiding the undesirable. In a sense, a metric often is a synthesis of tradeoffs, and the objective is to achieve the best balance. The strategy followed in this work is based on the recognition that psychological complexity can neither be conclusively defined nor precisely measured. What is measurable are the properties of a model representation that are believed to reflect psychological complexity, as defined by previous research in program complexity.

3. EXISTING METRICS

Current complexity literature deals almost exclusively with program complexity. What little exists on model complexity deals with either graphical analysis or the use of existing program metrics. However, a model represented by a CS or any of the three WVS is not a program or a graph. A program is an executable model representation. As such, certain factors needed for execution are included. These additions could constrain the model in a way that increases (or decreases) the complexity of the model. A model represented by a CS or WVS is not executable and, in a sense, is a "purer" representation of the model. As described in Zeigler (1976) "the criterion for accepting ... a measure of complexity is that it relates to the difficulty with which the behavior of a model may be inferred..." . In a model represented by a program, the behavior of the model is merged with details of execution.

However, a study of some existing program metrics can be useful. Examination of existing metrics provides a starting point for development of the desired model complexity metric.

Study of these metrics with respect to a CS reveals two basic elements of complexity to be considered. Transformation complexity measures the complexity of the function (transformation) performed by a cluster. Control complexity measures the complexity of the interconnections among clusters. These interconnections are represented by an ACIG.

Arc-to-node ratio (ANR), McCabe's (1976) metric, Myer's (1977) metric and the knot metric (Woodward, et, al. 1979) can be classified as control complexity metrics. Chapin's (1979) metric, the structure metric (Henry and Kafura 1981) and Halstead's (1977) metric are transformation metrics. Hansens' (1978) metric, Mill's (1973) metric, Zolnowski and Simmon's (1977) metric and the chunks metric (Davis 1984) combine control and transformation complexity in various ways. Other metric definitions have been studied, however the referenced metrics provide the most insight into the problem of model complexity. A more detailed discussion of these metrics and their applicability to models can be found in Wallace (1985).

A model complexity metric must incorporate both control and transformation complexity. A model is inherently dynamic since control continually passes from one model component to another. Consequently, a metric that measures only transformation complexity is not sufficient. However, the complexity of model actions also contributes significantly to the overall complexity of a model; thus a metric that only evaluates control complexity is also not sufficient. Both control and transformation complexity must be incorporated in the model complexity metric.

4. THE CONTROL AND TRANSFORMATION METRIC

4.1 Definition of the Control and Transformation Metric

Considering the CS in its ACIG representation, where each action is a node, the CAT metric is defined as:

$$MC = \sum_{i=1}^{n} (2*RW_i + 1.5*W_i + R_i) * \frac{A_i}{N}$$

where

- MC = model complexity
- RWi = number of variables in node i read and written
- Wi = number of variables in node i written only
- Ri = number of variables in node i read only
- Ai = number of arcs entering or leaving node i
- N = total number of nodes

The parenthetical expression defines the transformation complexity portion of the metric. The type classification RW, W and R are from Henry and Kafura's (1981) structure metric. The weights are similar to those found in Chapin (1979). Control complexity A_i/N is determined by counting the number of arcs entering and leaving node i (FanIn + FanOut). The complexity of each node is summed to determine model complexity.

One last adjustment is made to the metric. The initialization and termination nodes are ignored by the metric (in the WVS, actions that evolve from these two nodes are ignored). This is to avoid undue influence by nodes that are not representative of dynamic behavior. Otherwise, a very large initialization and/or termination node (each executed only once) can cause the measure to be an inaccurate reflection of the complexity contributed by nodes that are repeatedly executed; i.e. those action clusters that represent the repetitive nature of model behavior.

Skeptics can protest that the use of the number of nodes as a divisor can make it possible for a "large" model representation to have a lower complexity than a "small" model. This formulation could also encourage unwarranted decomposition of the model into smaller ACs in order to minimize the resulting complexity measure. However, each AC represents some action of the model itself and not just anarbitrary action that only partially represents some "happening" in the model. If an action of a model requires extensive transformations then this should be reflected in the representation.

The main purpose of including the division in the metric is to incorporate the benefits of the ANR. The ANR represents the uncertainty of the control path. Depending upon what kind of arcs are counted, the ANR reflects the uncertainty of where control came from, where control can pass, or both. For example, a ratio of 5/8 means that of the eight nodes in the model representation, five can pass control to and/or from this node. A ratio of 9/10 shows what might be referred to as a complexity bottleneck; this node can affect almost all other nodes in the model. A ratio of 9/100 is not as serious; even though this node affects the same number of its counterparts, only a small percentage are affected.

Fan-in represents the number of places from which the node could receive erroneous data. If an error is detected at the node, fan-in represents the number of immediate predecessor nodes to be examined in order to find what could have caused this error. If the node is modified, the fan-in represents the number of additional nodes having some effect on this change: what can cause this change to be exercised (perhaps mistakenly). A high fan-in shows the potential of the node to be called many times. For example, a node with fan-in of one is not likely to be activated as often as a node with fan-in of ten.

Fan-out, on the other hand, represents the number of nodes to which control can pass after completion of the current node's actions. This is appealing since it represents the potential for error propagation at the first level. A modification to a node can have an immediate effect upon all nodes that can be subsequently activated. When examining a representation, the fan-out represents the difficulty in determining which of the immediate successors is actually given control.

4.2 Review of Requirements

Psychological Complexity: the CAT metric measures usage of variables and flow of control among nodes. These factors could affect computational efficiency, but are more related to psychological complexity.

Generality of Use: CSs and WVSs are represented by action clusters of similar construction. ACIGs can be constructed from either form. Therefore, the CAT metric applies equally well to CSs and WVSs.

Lower Measures of Condition Specification: since nodes are either combined or unchanged during transformation from a CS to a WVS, the average transformation complexity of each node in a CS will be lower than for any corresponding WVS. This also results in a larger number of nodes for the CS than for any corresponding WVS (hence a larger divisor). Consequently, the CS exhibits a lower complexity than for any corresponding WVS.

Dynamic: the CAT metric incorporates flow of control into the metric through use of the control complexity factor.

Simple and Understandable: the CAT metric is very simple and straightforward and uses reasonable weights for the three types of data.

Baseline: There is a potential for a baseline. Since nodes represent an action of the model, then each of these actions would tend to occur in any version of the given conceptual model. The minimum code in a node involves one variable being set to a constant (i.e. TCi = 1.5 for all i). The difficulty is determining the minimum number of arcs for the model. Since all nodes are assumed to exist, all arcs are also assumed to exist. Since this assumption could very easily be wrong, the baseline is not necessarily the lower bound for the conceptual model. Conversely, it may be impossible to even approach this baseline measure since it is doubtful that all model actions can be accomplished by setting only one variable to a constant.

5. EXPERIMENTAL RESULTS

This experimentation is to compare the CAT metric with other metrics and to evaluate how the CAT metric reacts to selected models (Nance 1971). The test set is devised to evaluate the ability of a metric to react to specific properties. Five CSs are used as a test set. These model have implicit rankings among themselves. Some of these rankings appear in the CS representations while others appear after transformation into a WVS. The five models are:

- Minimal distance machine repairman.
- First fail, first fix (4F) machine repairman.
- Patrolling machine repairman - in this model the repairman patrols a circuit encompassing all machines. If a machine has failed, he stops to fix it, otherwise he continues his circuit.
- Infinite server machine repairman - two types of machine failure can occur and there is always an available repairman.

- Manufacturing model - this model is described in Henrikson (1981).

These five models provide a broad spectrum for experimentation. The manufacturing model is constrained almost exclusively by state-based conditions and as such fits very well into an activity scan world view. The infinite server machine repairman problem is almost exclusively time dependent and as such fits very well into an event scheduling world view. Therefore, there is an implicit ranking between these two models after translation into WVSs. The other three CSs are not as close to a particular world view. However, they have a distinct ordering in terms of complexity of CSs. The 4F version is a simplification of the minimal distance problem, since the functions to implement the queue are much simpler than the functions determining the closest failed facility. The action clusters of the patrolling model are simpler than the 4F model, yet the two AGIGs differ only slightly. (The CS, WVSs and ACIGs for this test set can be found in Wallace 1985.)

Differences among metrics must be attributable to distinguishable properties of the CAT metric. Differences between the CAT metric and other metrics should be attributable to desirable characteristics of the CAT metric. These differences must be easily defended, i.e., a casual examination of the model representation and/or the ACIG should confirm the preference for the result given by the CAT metric.

Experimental results are obtained by applying the metrics to CS representations. The CS representations are transformed into the three world views and the metrics are reapplied. The metrics used are:

- The CAT metric.
- Number of lines.
- Halstead's metric.
- ANR. Arc-to-node ratio. This is determined using the ACIG.
- McCabe's metric (cyclomatic complexity). This is determined using the ACIG (as opposed to using the predicates in the model representation).
- Chunks metric.

5.1 Comparison of Condition Specification Results

Table 1 summarizes the results of applying the various metrics to the CSs in the test set. All values for a particular metric are normalized by dividing by the lowest value for that metric. Normalization assists in relative comparisons, which in some cases are more meaningful.

The ANR and McCabe metrics (control complexity metrics) give comparable results. The lines of code and Halstead metrics (transformation complexity metrics) also give comparable results. In terms of ranking, the transformation metrics rank the models identically with only one exception. However, the CAT and control metrics different from the transformation metrics. Comparison of the CAT and control metric values reveal differences between them. Nevertheless, closer examination of the CAT metric permits meaningful distinctions to be attached to these differences.

Table 1: Complexity Measure Results by Condition Specification

By Condition Specification	CAT	# of lines	Hal-stead	ANR	Mc-Cabe	Chunks
min dist	2.49	2.50	2.02	1.18	1.33	6.14
4F	2.25	2.38	1.95	1.18	1.33	4.89
patrolling	1.21	1	1	1.24	1.67	2.02
manufacturing	2.14	1.75	1.57	1.48	2	4.34
infinite server	1	1.88	1.54	1	1	1

The CAT and control metrics show the infinite server model to be least complex, while the transformation metrics rank it above the patrolling model. The difference stems from the lower control complexity of the infinite server CS that diminishes the overall effect of higher transformation complexity. The difference between the CAT and control metrics applied to the patrolling model stems from higher control complexity, which offsets the effect of a low transformation complexity.

The CAT metric determines the modified manufacturing CS as (a distant) third, but all other metrics choose it as fifth or sixth. This CS has 18 lines but only 2.25 lines per node. The manufacturing, minimal distance and 4F CSs each have more lines per node. The control complexity for the manufacturing CS is not much higher than for the other models. Consequently, the CAT metric exhibits a lower value based on low complexity per node.

Another interesting result is the CAT metric selects the 4F CS as fifth, but the transformation metrics pick it fourth and the ANR second. However, examination of both the CS and the ACIG reveals that the nodes with the highest control complexity also have the most actions (i.e., higher transformation complexity). Therefore, the CAT metric results in a higher value than any other metric. The CAT metric chooses the minimal distance CS as most complex for the same reasons. Most importantly, the CAT metric ranks the three repairman models in the correct order based on prior examination and the experience in development of analytical models.

5.2 Comparison of Results by Problem Type

Table 2 summarizes the results by problem type. In all cases, the CS (or modified CS) exhibits less complexity than any of its translated versions, which fulfills one of the requirements. As expected, the manufacturing model fits best into an activity scan world view according to the CAT metric, since model transitions are based primarily on state conditions.

In the infinite server model, the event scheduling world view gives a lower result than the other world views, again as expected, since the infinite server model is primarily time dependent. The transformation metrics evaluate the process interaction world view as least complex, but the CAT metric evaluates it as a distant third because the event version has many nodes and few transformations per node.

Table 2: Results by Problem Type

By Problem Type	CAT	# of lines	Hal-stead	ANR	Mc-Cabe	Chunks
Min dist						
CS	1	1	1	1	1	1
event	1.95	2.15	1.84	1.13	1	1.56
activity	2.03	1.10	1.09	1.13	1	1.27
process	3.39	2.05	1.50	1.50	1.25	2.29
4F						
CS	1	1	1.10	1	1	1
event	1.92	1.79	1.58	1.13	1	1.16
activity	2.09	1.11	1	1.13	1	1.43
process	3.34	1.79	1.39	1.50	1.25	2.29
Patrolling						
CS	1	1	1.19	1	1.25	1
event	3.34	2.00	1.70	1.19	1.25	1.16
activity	2.83	1.25	1	1.07	1	1.43
process	4.65	2.63	1.99	1.43	1.25	3.43
Manufacturing						
CS	1.12	1	1	1.11	1.20	1
event	7.71	3.86	2.66	1.78	1.40	4.22
activity	1.49	1.64	1.18	1.07	1	1.59
process	2.94	1.86	1.26	1.55	1.20	2.08
Infinite Server						
CS	1	1	1.35	1.13	1.50	1
event	1.35	1.73	1.58	1.17	1.50	3.53
activity	5.07	1.45	1.55	2.00	2.50	5.28
process	3.38	1.18	1	1	1	3.64

In the remaining machine repairman problems, relatively equal emphasis is placed on time-based and state-based conditions. Consequently, little difference is observed between the results for the event scheduling and activity scan world views.

In the single-repairman models, the process interaction world view is easily the most complex WVS. This outcome is at least intuitive, and an explanation can be substantiated somewhat from the literature (Nance 1971).

Essentially, extra information is involved in separating a model into processes. While this view may be the most natural for some models and modelers, this extra information is necessary either in an explicit or implicit form. Extra information must impose a cost, which is reflected in the results of the single-repairman models. Much of the extra cost appears in the form of high control complexity, resulting from the interaction among objects.

The manufacturing model is skewed towards an activity scan world view. The extra cost for separating the model into processes results in the process interaction WVS exhibiting more complexity than the activity scan WVS. However, this extra cost is not so great as to result in process interaction displaying greater complexity than event scheduling, since the manufacturing model is so poorly suited to event scheduling.

The infinite server model is skewed towards an event scheduling world view. The extra cost of using a process interaction world view results in process interaction displaying more complexity than event scheduling, but not so great as to allow activity scan to produce a lower result.

6. SUMMARY

A measure of model complexity is essential for estimation of project resource requirements, comparison and implementation decision, and assessment of alternative representational forms. While program complexity is a recognized area of research, little work is evident in model complexity. A review of the program complexity literature shows that current metrics, which are based on graphical analysis or static program characteristics, fail to capture the influences on complexity stemming from the inherent dynamics of a model. These metrics deal almost exclusively with program complexity. However, a model represented by a CS or WVS is not executable and cannot be considered to be a program. A CS or WVS is, in a sense, a "purer" representation of a model since details needed for execution are not included. The use of a CS of WVS also allows informative diagnosis early in the model development process.

The CAT metric evolves from the recognition of two types of complexity: transformation and control, both of which are prominent in model representations. The metric is designed to fulfill the defined set of requirements.

The CAT metric incorporates the inherent dynamics of models through the use of control complexity. The metric is designed to give a modeler information on the difficulty in implementing and maintaining a model. The metric uses reasonable weights for the three types of data defined and uses a very simple function. The metric is applicable to models represented by a CS or WVS. The CAT metric has a potential for a baseline, therefore a modeler can gain additional information even if there is only one version of a model.

The CAT metric responds well to the test set of models. The metric discerns the implicit ranking in both the CS and WVS representations. Differences in values among the CAT metric and other metrics are attributable to desirable characteristics of the CAT metric. Based on the experiments described in this report, the CAT metric represents an improvement over extant software metrics for measuring the complexity of model representations.

ACKNOWLEDGMENTS

The author wishes to thank Dr. Richard Nance for his aid in defining the scope of this paper. The author also thanks Barbara Call of the Georgia Tech Research Institute for her yeoman work on the typing, formatting and graphics which were done under serious time constraints.

REFERENCES

Chapin, N. (1979). A Measure of Program Complexity. Proceedings 1979 AFIPS National Computer Conference, 995-1002.

Curtis, B. (1980). Measurement and Experimentation in Software Engineering. Proceedings of the IEEE 68, 1144-1157.

Davis, J.S. (1984). Chunks: A Basis for Complexity Measurement. Information Processing and Management 20, 119-127.

Halstead, M.H. (1977). Elements of Software Science, P.J. Denning, ed., Elsevier North-Holland, Inc., New York, NY.

Hansen, W.J. (1978). Measurement of Program Complexity by the Pair (Cyclomatic Number, Operator Count). SIGPLAN Notices 13, 29-33.

Henriksen, J.O. (1981). GPSS - Finding the Appropriate World View. 1981 Winter Simulation Conference Proceedings, 505-516.

Henry, S. and D. Kafura (1981). Software Metrics Based on Information Flow. IEEE Transactions on Software Engineering SE-7, 510-518.

McCabe, T.J. (1976). A Complexity Measure," IEEE Transactions on Software Engineering SE-2, 308-320.

Mills, H.S. (1973). The Complexity of Programs. Program Test Methods, W.C. Hetzel, ed., Prentice-Hall, Inc. Englewood Cliffs, NJ, 225-238.

Myers, G.J. (1977). An Extension to the Cyclomatic Measure of Program Complexity. SIGPLAN Notices 12, 61-64.

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

Nance, R.E. (1971). On Time Flow Mechanisms for Discrete System Simulation. Management Science 18, 59-73.

Overstreet, C. M. and R. E. Nance (1986). World View Based Discrete Event Model Simplification. Modelling and Simulation Methodology in the Artificial Intelligence Era, M. S. Elzas, T.I. Oren and B. P. Zeigler, ed., Elsevier North-Holland, Inc., New York, NY.

Overstreet, C.M. and R.E. Nance (1984). Graph-based Diagnosis of Discrete Event Model Specifications. Technical Report CS83028-R, Department of Computer Science, Virginia Tech.

Overstreet, C.M. (1982). Model Specification and Analysis for Discrete Event Simulation. Ph.D. Dissertation, Computer Science Department, Virginia Tech.

Wallace, J.C. (1985). The Control and Transformation Metric: A Basis for Measuring Model Complexity. Technical Report TR-85-15, Systems Research Center, Virginia Tech, Blackburg, Virginia.

Weinberg, G.W. (1971). The Psychology of Computer Programming Van Nostrand Reinhold Co., New York, NY.

Woodward, M.R., M.A. Hennell and D. Hedley (1979). A Measure of Control Flow Complexity in Program Text. IEEE Transactions on Software Engineering, SE-5, 45-50.

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

Zolnowski, J.M. and D.B. Simmons (1977). Measuring Program Complexity. Proceedings of the 1977 Fall COMPCON, IEEE, 336-340.

AUTHOR'S BIOGRAPHY

JACK C. WALLACE is a Research Scientist for the Computer Systems and Technology Division, Electronics and Computer Systems Laboratory, Georgia Tech Research Institute. He received a B.S. in Computer Science from the University of Dayton in 1982, and an M.S. in Computer Science and Applications from Virginia Tech in 1985. His current research interests include modeling and simulation, software metrics, computer graphics and human-computer interfaces.

Jack C. Wallace
GTRI/ECSL/CSTD
Georgia Institute of Technology
Atlanta, GA 30332, U.S.A.
(404) 894-3523