The Validation by Measurement Theory of Proposed Object-Oriented Software Metrics

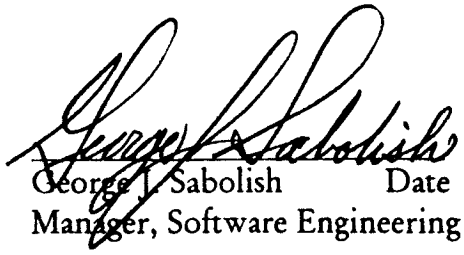
Ralph D. Neal

Virginia Commonwealth University
Department of Information Systems
School of Business
Richmond, Virginia 23284-4000

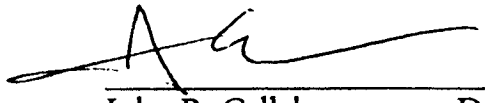
Current Address: WVU/NASA Software IV & V Facility
100 University Drive
Fairmont, WV 26554

Office Tel.	(304) 367-8355
Home Tel.	(304) 368-0252
Fax	(304) 367-8211
E-mail	rneal@cerc.wvu.edu

According to the terms of Cooperative Agreement #NCCW-0040, the following approval is granted for distribution of this technical report outside the NASA/WVU Software Research Laboratory



George J. Sabolish Date
Manager, Software Engineering



John R. Callahan Date
WVU Principal Investigator

Abstract

Moving software development into the engineering arena requires controllability, and to control a process, it must be measurable. Measuring the process does no good if the product is not also measured, i.e., being the best at producing an inferior product does not define a quality process. Also, not every number extracted from software development is a valid measurement. A valid measurement only results when we are able to verify that the number is representative of the attribute that we wish to measure. Many proposed software metrics are used by practitioners without these metrics ever having been validated, leading to costly but often useless calculations. Several researchers have bemoaned the lack of scientific precision in much of the published software measurement work and have called for validation of software metrics by measurement theory. This dissertation applies measurement theory to validate fifty proposed object-oriented software metrics (see Li and Henry, 1993; Chidamber and Kemerrer, 1994; Lorenz and Kidd, 1994).

I. Background and Objectives

The need for software metrics

Software development historically has been the arena of the artist. Artistically developed code often resulted in arcane algorithms or spaghetti code that was unintelligible to those who had to perform maintenance. Initially only very primitive measures such as lines of code (LOC) and development time per stage of the development life cycle were collected. Projects often ran over estimated time and over budget. In the pursuit of greater productivity, software development evolved into software engineering. Part of the software engineering concept is the idea that the product should be controllable. DeMarco [1982] reminds us that what is not measured cannot be controlled.

Measurement is the process whereby numbers or symbols are assigned to attributes of entities in such a manner as to describe the attribute in a meaningful way. We cannot take measurements and then apply them to just any attributes. Unfortunately this is exactly what the software development community has been doing. [Fenton, 1994]

Because people observe things differently (and often intuitively feel differently about things), a model is usually defined for the entities and attributes to be measured. The model requires everyone to look at the subject from the same viewpoint. Fenton [1994] uses the example of human height. Should posture be taken into consideration when measuring human height? Should shoes be allowed? Should we measure to the top of the head or the top of the hair? The model forces a reasonable consensus upon the measurers.

As has already been stated, control of a process or product requires that the process or product is measurable; therefore, control of software requires software measures [Baker, et al.,

1990]. It does no good to measure the process if the product is not measured. Being the best at producing an inferior product does not define a quality process.

The need for metric validation

Choosing metrics becomes a horse and cart or a chicken and egg type of question. Which do we do first; choose the metrics of interest or validate the metrics? Since these metrics are already in use, I have chosen to validate them first. The next step will be to choose from among the measures (valid metrics) a suite of them that is the smallest set of measures that is both necessary and sufficient to measure the important dimensions of the software. The steps involved are:

1. Identify important dimensions of the software.
2. Classify measures by the dimension(s) they measure.
3. Use multivariate statistical methods to investigate the parallelism/ orthogonality of the captured measures.

It is not beneficial to measure the same dimension of an object by more than one method. Each method will have its own degree of accuracy and its own cost of application. Once the necessary degree of accuracy has been established, the most cost effective method that delivers that level of accuracy should be the measurement of choice. When building models with unvalidated metrics the degree of accuracy cannot be known.

Fenton [1994] argued that much of the software measurement work published to date is scientifically flawed. Fenton is not the only scientist who has observed this lack of scientific precision. Baker, et al., [1990] said as much when they wrote that research in software metrics often is suspect because of a lack of theoretical rigor. Li and Henry [1993a] argued that validation is

necessary for the effective use of software metrics. Schneidewind [1992] stated that metrics must be validated to determine whether they measure what it is they are alleged to measure. Weyuker [1988] stated that existing and proposed software measures must be subjected to an explicit and formal analysis to define the soundness of their properties.

McCabe failed to validate his complexity metric. Gilb referenced empirical testing as his source of verification and validation, i.e., there was no theoretical validation of Gilb's metrics. Halstead's equations were tested statistically. McCall defined metrics based on heuristics. A metric was accepted by McCall if a chosen sample fell within a 90% confidence interval [McCall, et al., 1977]. DeMarco employed no theoretical base in the validation of his metrics. Li and Henry [1993] used statistical analysis to validate the prediction of maintenance effort by the group of metrics that they published. No theoretical validation was attempted by Li and Henry. Chidamber and Kemerer mentioned measurement theory in their evaluation of each metric but made no attempt to assign a scale to the metrics (see the paragraph on scales in section II for an explanation of the importance of scale to the valid interpretation of a measurement). Lorenz and Kidd [1994] only used heuristics to validate their metrics.

Software metrics and measurement theory

Measurement theory was first used in software metric research to validate the myriad complexity metrics which dominated the early research in the field. Correlations were expected to exist between the complexity of a project and the achievement of acceptable parameters in its development. This was the rationale for the interest in software complexity and the development of metrics to measure this complexity [Anderson, 1992].

When defining a measure, first one must designate precisely the attribute to be measured, e.g., the height of humans. Then a model is specified that captures the attribute, e.g., stand up straight, take off your shoes, do not include hair height in the measurement. The congruence that comes from the model must represent the attribute being measured, i.e., the intuitive order of the objects, with respect to the attribute being measured, must be preserved by the model. Finally, an order-preserving map from the model to a number system is defined, e.g., if we observe that Harry is taller than Dick, any measurement that we take of their height must result in numbers or symbols that preserve this relationship. [Baker, et al., 1990]

Before a model can be proposed, it must be known what is being measured. This basic measurement principle has been ignored in much of the software metric work of record. It is fundamental to measurement theory that the measurer have an intuitive understanding, usually based on observation, of the attribute being measured [Fenton, 1991].

The object-oriented paradigm

An object combines both data structure and behavior in a single entity. Object-oriented software is organized as a collection of explicit objects. By contrast, data structure and behavior are loosely connected in traditional programming [Rumbaugh, et al., 1991]. Authors have not been in agreement about the characteristics that identify the object-oriented approach. Henderson-Sellers [1991] listed information hiding, encapsulation, objects, classification, classes, abstraction, inheritance, polymorphism, dynamic binding, persistence, and composition as having been chosen by at least one author as a defining aspect of object-orientation. Rumbaugh, et al. [1991] added

identity, Smith [1991] added single type and Sully [1993] added the unit building block to this list of defining aspects.

The old software metrics do not take into consideration these new concepts. Therefore, these characteristics necessitate the advent of new metrics to measure object-oriented software. The recent explosion of object-oriented software metrics (Li and Henry, 1993; Chidamber and Kemerer, 1994; and Lorenz and Kidd, 1994) has hit the scene with little validation beyond regression analysis of observed behavior.

Research objectives

"*Validation* of a software measure is the process of ensuring that the measure is a proper numerical characterization of the claimed attribute" [Baker, et al., 1990]. Fenton [1991] described two meanings of validation. Validation in the narrow sense is the rigorous measurement of the physical attributes of the software. Validation in the wide sense determines the accuracy of any prediction system using the physical attributes of the software. Accurate prediction is possibly the most valuable outcome to be gained from software measurement. Prediction systems are validated by empiric experiments. Accurate prediction relies on careful measurement of the predictive attributes and careful observation of the dependent attributes. A model which accurately measures the attributes is necessary but not sufficient for building an accurate prediction system [Fenton, 1994].

In the past, validation in the wide sense has been conducted without first carrying out validation in the narrow sense. In this dissertation we intend to validate in the narrow sense the

object-oriented software metrics that have appeared in the literature. This is a necessary step before these metrics can be used to predict such managerial concerns as cost, reliability, and productivity.

> Fenton [1991] states: "Good predictive theories only follow once we have rigorous measures of specific well understood attributes."

II. Research Approach and Methodology

Introduction

There are two fundamental problems in measurement theory; the first is the *representation problem.* The representation problem is to find sufficient conditions for the existence of a mapping from an observed system to a given mathematical system. Another aspect of the representation problem is pointed out by Weyuker [1988]. How unique is the result of the measurement? A measurement system must provide results that enable us to distinguish one class of object from another class of object.

The other fundamental problem of measurement theory is the *uniqueness problem.* Uniqueness theorems define the properties and valid operations of different measurement systems and tell us what type of scale results from the measurement system. A uniqueness theorem contributes to a theory of scales which says that the scale used dictates the meaningfulness of statements made about measures based on the scale [Hong, et al., 1993; Roberts, 1979]. A statement involving numerical scales is meaningful if the truth of the statement is maintained when the scale involved is replaced by another (admissible) scale.

The empirical/formal relational system. A relational system is a way of relating one entity (or one event) of a set to another entity (or event) of the same set. In the physical sciences the relations take the form longer than, heavier than, of equal volume, etc. In the social sciences (and thus in software metric measurement) the relations take the form is preferred to, is not preferred to, is at least as good as.

> Definition 2.1: *The ordinal relational system is an ordered tuple (A,R1,...,Rn) where A is a nonempty set of objects and the Ri, i=1,...,n are k-ary relations on A.* [Zuse, 1990]

The extensive structure. The extensive structure is an expansion of the ordinal relation system to include binary operations on the objects of the set. The extensive structure is required to measure objects on the interval or ratio scales. The binary operation in the empirical relational system usually is designated concatenation, denoted by • . The usual manifestation of the binary operation in the formal relational system is addition (+) although multiplication may be the proper operation under some circumstances.

> Definition 2.2: *The extensive relational system is an ordered tuple (A,R1,...,Rn,• 1,...,• m) where A is a nonempty set of objects, the Ri, i=1,...,n are k-ary relations on A and the • j, j=1,...,m are closed binary relations.* [Zuse, 1990]

Homomorphism. A software measurement can be a homomorphism only if the meaning and interpretation of the empirical relationship is clear [Zuse, 1990]. Let • denote is larger than (or is preferred to). Given the empirical scale • (A,• ,•) which we wish to measure using the real numbers, we must map • to • (B,>,+) while preserving the relation • and the operation • , i.e., • : A• B is a valid mapping from A to B iff $a1 \bullet a2 \bullet b1 > b2$. In order to know whether or not the

relation and the operation have been preserved, the meaning and interpretation of • ,A,• ,and • must be precisely defined.

The weak order. Suppose you must select from a list of alternatives. For each pair of alternatives *a1* and *a2*, you prefer *a1* to *a2*, you are indifferent between *a1* and *a2*, or you prefer *a2* to *a1*. If you always prefer *a1* to *a2*, you are said to have a strict preference. If, however, you sometimes prefer *a1* to *a2* and sometimes you are indifferent between *a1* and *a2*, you are said to have a weak preference. When you have a weak preference and the measurements exhibit the axioms of completeness, reflexiveness, and transitivity, the alternatives are said to constitute a weak order.

Meaningfulness. When does it make sense to state:

- Program A is more complex than program B?
- Program A is twice as complex as program B?
- Program A is twice as maintainable as program B?
- Program A displays more quality than program B?
- The quality of program A was increased by 20%?

Following Zuse [1990], a statement is meaningful if and only if the truth of the statement holds against all admissible transformations. Therefore, the meaningfulness of these statements depends on the scale assignable to the metric used to measure the attribute of question.

Table 1

Properties of Measurement Scales

Scale	Basic empirical operations	Admissible transformations
Ratio	=,<,>, equality of intervals, and ratios	M'=• M, • >0 similarity transformation
Interval	=,<,>, and equality of intervals	M'=• M+• , • >0 positive linear transformation
Ordinal	=,<, and >	M'=*f*(M) where *f*(M) is any monotonic increasing transformation
Nominal	=	M'=*f*(M) any one-to-one transformation

Scales. When groups of objects are measured on the nominal scale: many statistics can not be used; proportions can be taken; the mode is the only meaningful measure of centrality. When groups of objects are measured on the ordinal scale: rank order statistics and non-parametric statistics can be used (assuming that the necessary probability distribution can be reasonably assumed to be present); the median is the most powerful meaningful measure of centrality. When groups of objects are measured on the interval scale: parametric statistics as well as all that apply to ordinal scales can be used (it must be reasonable to accept that the necessary probability distribution is present); the arithmetic mean is the most powerful meaningful measure of centrality. When groups of objects are measured on the ratio scale: percentage calculations as well as all statistics that apply can be used; the arithmetic mean is the most powerful meaningful measure of centrality.

Desirable properties of measures

Intuition. A measure should make sense based upon the professional experience of the measurer. Objects that appear better in the attribute being measured (based on the observer's experience) should score higher on the metric being used. Objects which appear similar should score roughly about the same.

Monotony. Monotony (or consistency) goes along with intuition. The measurement must be such that very nearly the same score is achieved regardless of the measurer. Also, the order that the objects appear in, in relation to each other, must be consistent from measurement to measurement.

Mathematical foundation. It is important that the measure be grounded in mathematical theory. This foundation is necessary but not sufficient to make the metric an appropriate gauge of the property being measured.

Understandability. The measurement process as well as the meaning of the metric should be understandable by interested persons [Tsai, et al., 1986 (as cited in Zuse, 1990)].

Variation. If all articles score the same on a metric, then that metric measures nothing. In order to measure a property there must be variation in measurement from object to object.

Dispersion. A measure is not precise enough if all articles fall into only a few categories. Ideally, the measure should be sensitive enough to measure the appropriate property on a continuum.

Especially grievous is the case that assigns the property to a set with discrete units of limited cardinality [Weyuker, 1988 (as cited in Zuse, 1990)].

Before a model can be proposed, it must be known what is being measured. This basic measurement principle has been ignored in much of the software metric work of record. It is fundamental to measurement theory that the measurer have an intuitive understanding, usually based on observation, of the attribute being measured [Fenton, 1991].

The basis of the methodology to be followed will be Zuse's model.

Zuse's model

Before a metric can be said to possess scale, 1) enough atomic modifications must be defined to completely describe any changes that can affect the metric, 2) the partial properties of the metric must be ascertained, and 3) the intuition of the measurer must agree with the partial properties established.

The concatenation operator for each metric must be defined based on the properties of the metric. Since Zuse always evaluated static measures of software code, he used the sequential and alternative structures of flowgraphs to define the concatenation operation.

> Definition 2.3: *A flowgraph G=(E,N,s,t) is a directed graph with a finite, nonempty set of nodes N, a finite, nonempty set of edges E, a start node s• N, and a terminal node t• N. Each node x• N lies on some path in G from s to t along the edges. An edge is an ordered pair of nodes (x,y).* [Zuse, 1990].

Figure 1 is a flowgraph. Nodes 3, 7, and 11 are called predicate (decision) nodes. Nodes 4, 5, 8, and 9 are called processing nodes. An atomic modification to a flowgraph is defined as adding, deleting, or transferring edges or nodes in the flowgraph [Zuse, 1990]. Specifically, we define:

AM1 as adding (deleting) an edge at an arbitrary location,

AM2 as adding (deleting) a node and an edge at an arbitrary location, and

AM3 as transferring an edge from one location in a flowgraph to another location.

Every metric increases, decreases, or remains the same in reaction to each of these atomic modifications. The partial property of the metric is defined as the sensitivity of the metric to an atomic modification, i.e., the measure **M** has the partial property <=> (either it is less desirable, you have indifference, or it is more desirable) with respect to the atomic modification AM.

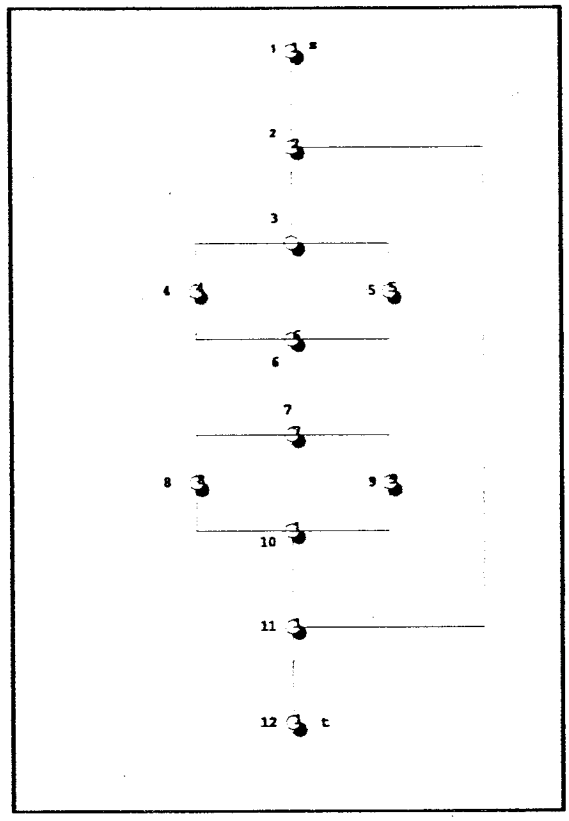


Fig. 1 A FlOWGRAPH

A measure can be placed on the ordinal scale if the user accepts the partial properties of the atomic modifications defined for that measure and the axioms of the weak order (completeness, reflexiveness, and transitivity) hold. A measure can be used as an interval scale if all conditions of the ordinal scale are met and the distance defined on the interval is consistent for all intervals. A measure can be

placed on the ratio scale if all conditions of the ordinal scale are met and the user accepts the binary concatenation operation(s) defined on the measure.

Let us now consider Zuse's methodology more specifically.

Description of the measures. The original definition (as provided by the author of the measure) is given for each metric. Each metric is then defined using a uniform method. The flowgraphs of Zuse will be used whenever static code is being measured. Other, appropriate, structures will be defined as needed for each metric being validated.

Examples of the calculation of the measures. Simple and uniform examples are given for each metric.

Partial property description of the measures. Atomic modifications are used on each metric to describe its partial properties. Atomic modifications to flowgraphs consist of adding, deleting, and moving edges and nodes. Other atomic modifications will be developed as necessary for other structures.

Complete description of the measures as an ordinal scale. Atomic modifications are defined sufficient to describe the criteria for the use of the metric as an ordinal scale then the measures are examined to determine if the axioms of the weak order (completeness, reflexiveness, and transitivity) hold.

Consideration of the measures as an interval scale. The mapping which results from the atomic modifications are compared to determine if a uniform difference between integer results can be discerned.

Extensive structure and ratio scale. Binary concatenation operations to flowgraphs consist of sequential and alternative addition of two flowgraphs. When it is necessary to define another structure, other binary concatenation operations must also be defined. The ways the metrics respond to the binary concatenation operations, as defined, are investigated to determine whether or not the metric possesses the properties of the extensive structure. The rules are given for the use of the metrics as a ratio scale.

Metric summary. The properties of the metric are summarized and compared to the properties of similar metrics.

The seven steps of Zuse's model are applied to each metric to determine what meaningful statements may be made using the information gleaned from the metric.

III. Expected Contribution

Contribution and significance of this study

Many object-oriented metrics are being proposed. Because they have not been validated using measurement theory, it is not clear that these metrics are valid measures of the attributes that they claim to measure. Some of these metrics are touted as predictive without being rigorously defined. This study looks at each of the object-oriented metrics and scrutinizes them for validity in the narrow sense of Fenton [1991].

Does the metric measure what its author proposes to measure? If not, what can be said about the metric in terms of what is being measured? Is there another metric which does measure the desired attribute? Are the statistics used with the metric valid considering the scale attributed to the metric? Is the measurement an assessment measurement or meant to be a predictive measurement? Does the metric hold up under vigorous scrutiny of the conditions of representation and uniqueness? Do intuitive and empirical understandings survive under all allowable transformations?

The answers to these questions are pertinent to the valid use of these metrics. Since the collection of data for the calculation of metrics is very expensive [Deutsch and Willis, 1988], this study will help the practitioner by separating those object-oriented metrics that are not worth the cost of calculation from those that are and by differentiating those metrics that are valid for assessment purposes from those that are valid for use in prediction systems.

Additionally, the software engineering community should gain insight into further use of the metrics, other metrics which might replace them, the valid statistics that each metric supports, and future research that needs to be carried out.

REFERENCES

Albrecht, A. J., *Measuring Application Development Productivity,* Proceeding of Joint Share/Guide/IBM Application Development Symposium, 1979.

Anderson, O., *Industrial Applications of Software Measurements*, Information and Software Technology, Vol. 34, No. 10, Oct., 1992.

Baker, Albert L., James M. Bieman, Norman Fenton, Davis A. Gustafson, Austin Melton, and Robin Whitty, *A Philosophy of Software Measurement*, The Journal of Systems and Software, Vol. 12, 1990, p. 277-281.

Boehm, Barry W., Software Engineering Economics, Prentice-Hall, Englewood Cliffs, New Jersey, 1981.

Chidamber, Shyam R., and Chris F. Kemerer, *A Metric Suite for Object Oriented Design*, MIT Center for Information Systems Research Working Paper #249, January 1993, Revised July 1993.

Chidamber, Shyam R., and Chris F. Kemerer, *A Metric Suite for Object Oriented Design*, IEEE Transactions on Software Engineering, Vol. 20, No. 6, June 1994.

DeMarco, Tom, Controlling Software Projects, Yourdon Press, New York, NY, 1982.

Deutsch, Michael S., and Ronald R. Willis, Software Quality Engineering: A Total Technical and Management Approach, Prentice-Hall, Englewood Cliffs, NJ, 1988.

Ellis, Brian, Basic Concepts of Measurement, Cambridge University Press, Cambridge, UK, 1966.

Fenton, Norman, *Software Measurement: A Necessary Scientific Basis*, IEEE Transactions on Software Engineering, Vol. 20, No. 3, March 1994.

Fenton, Norman, Shari Lawrence Pfleeger, and Robert L. Glass, *Science and Substance: A Challenge to Software Engineers*, IEEE Software, July 1994.

Fenton, Norman, Software Metrics: A Rigorous Approach, Chapman & Hall, London, UK, 1991.

Fenton, N., and B. Littlewood (eds.), Software Reliability and Metrics, Elsevier Applied Science, London, 1990.

Fenton, Norman, and Austin Melton, *Deriving Structurally Based Software Measures*, Journal of Systems and Software, Vol. 12, p. 177-187, 1990.

Fenton, N.E. and A.A. Kaposi, *An Engineering Theory of Structure and Management*, Alvey Project SE, (ALV/SE69/SBP/071/02), 1987.

Fenton, Norman E., *The Structural Complexity of Flowgraphs*, in Graph Theory with Applications to Algorithms and Computer Science, John Wiley & Sons, NY, 1985.

Gilb, Tom, Software Metrics, Winthrop Publishers, Inc., Cambridge, Massachusetts, 1977.

Halstead, M. H., Operating and Programming Systems: Elements of Software Science, Elsevier, New York, 1977.

Hecht, M.S., Flow Analysis of Computer Programs, Elsevier, New York, 1977.

Henderson-Sellers, B., A Book of Object-Oriented Knowledge, Prentice Hall, NY, 1992.

Hong, Sa Neung, Michael V. Mannino, and Betsy Greenberg, *Measurement Theoretic Representation of Large, Diverse Model Bases*, Decision Support Systems, 10, 1993.

Kitchenham, Barbara A., *Metrics and Measurement*, in Software Engineers Reference Book, John A McDermid (ed.), Butterworth/Heinmann, Oxford, UK, 1991.

Krantz, David H., R. Dunce Luce, Patrick Suppes, and Amos Tversky, Foundations of Measurement: Volume I, Academic Press, NY, 1971.

Kyburg, Henry E., Jr., Science and Reason, Oxford University Press, New York, 1990.

Kyburg, Henry E., Jr., Theory and Measurement, Cambridge University Press, Cambridge, UK, 1984.

Li, Wei, and Sallie Henry, *Object-Oriented Metrics that Predict Maintainability*, Journal of Systems and Software, Vol 23, p.111-122, 1993a.

Li, Wei, and Sallie Henry, *Maintenance Metrics for the Object-Oriented Paradigm*, Proceedings of the First International Software Metrics Symposium, May 1993b.

Lorenz, Mark, and Jeff Kidd, Object-Oriented Software Metrics, Prentice Hall, Englewood Cliffs, NJ, 1994.

McCabe, T. J., *A Complexity Measure*, IEEE Transactions on Software Engineering, Vol. 5, 1976.

McCall, J.A., P.K. Richards, and G.F. Walters, Factors in Software Quality, RADC-TR-77-369, Vol I-III, Rome, NY, November, 1977.

Perlis, Alan, Frederick Sayward, and Mary Shaw eds, Software Metrics: An Analysis and Evaluation, The MIT Press, Cambridge, Massachusetts, 1981.

Pfanzagl, Johann, Theory of Measurement, Physica-Verlag, Würzburg, Germany, 1971.

Putnum, Lawrence H., and Ware Myers, Measures for Excellence, Yourdon Press, Englewood Cliffs, NJ, 1992.

Roberts, Fred S., Measurement Theory with Applications to Decisionmaking, Utility, and the Social Sciences, Addison-Wesley Publishing Company, Reading Massachusetts, 1979.

Rubey, R.J., and R.D. Hartwick, *Quantitative Measurement of Program Quality*, Proceeding ACM National Conference, Brandon/Systems Press, Inc., Princeton, NJ, 1968.

Rumbaugh, James, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorensen, Object-Oriented Modeling and Design, Prentice Hall, Englewood Cliffs, NJ, 1991.

Savage, C. Wade, and Philip Ehrlich, *A Brief Introduction to Measurement Theory and to the Essays*, in Philosophical and Foundational Issues in Measurement Theory, Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1992.

Schneidewind, Norman F., *Methodology for Validating Software Metrics*, IEEE Transactions on Software Engineering, Vol. 18, No. 5, May 1992.

Schneidewind, Norman F., *Validating Metrics for Ensuring Space Shuttle Flight Software Quality*, IEEE Computer, August 1994.

Schneidewind, Norman F., *Software Quality Assessment and Standards*, IEEE Computer, June 1993.

Smith, David N., Concepts of Object-Oriented Programming, McGraw-Hill, NY, 1991.

Stevens, S.S., *On the Theory of Scales and Measurement*, Science, 103, p.677-680, 1946.

Stevens, S.S., *On the Averaging of Data*, Science, 121, p.113-116, 1955.

Sully, Phil, Modeling the World with Objects, Prentice Hall, NY, 1993.

Torgerson, Warren S., Theory and Methods of Scaling, John Wiley & Sons, Inc., New York, 1967.

Tsai, W.T., M.A. Lopez, V. Rodriguez, D. Volovik, *An Approach Measuring Data Structure Complexity*, COMPSAC 86, pp. 240-246, 1986.

Weyuker, Elaine J., *Evaluating Software Complexity Measures*, IEEE Transactions on Software Engineering, Vol. 14, No. 9, September 1988.

Zuse, H., and P. Bollmann, *Software Metrics: Using Measurement Theory to Describe the Properties and Scales of Static Software Complexity Metrics*, Sigplan Notices, Vol. 24, No. 8, August, 1989.

Zuse, Horst, Software Complexity: Measures and Methods, Walter de Gruyter, Berlin, 1990.