(12)

NSWCDD/TR-92/119

AEGIS MEASURES DEFINITION

BY EDWARD J. DUDASH DAVID E. McCONNELL
COMBAT SYSTEMS DEPARTMENT

MARCH 1992

NAVAL SURFACE WARFARE CENTER
Dahlgren Division
Dahlgren, Virginia 22448-5000

NSWCDD/TR-92/119

AEGIS MEASURES DEFINITION

BY EDWARD J. DUDASH DAVID E. McCONNELL
COMBAT SYSTEMS DEPARTMENT

MARCH 1992

NAVAL SURFACE WARFARE CENTER
DAHLGREN DIVISION
Dahlgren, Virginia 22448-5000

FOREWORD

Changes taking place within the Department of Defense environment require that greater efforts be made in improving the quality of products and ensuring that the highest levels of productivity are achieved. Achieving highest levels of quality and productivity requires that objective standards of measurement be employed. The Naval Surface Warfare Center, Dahlgren Division (NSWCDD) AEGIS Program Office has been supporting process improvements for several years through utilization of a process improvement panel that has focused on formalizing and completing process procedures and standards.

Continuing and building upon the existing process formalization activities has required that the AEGIS Program Office additionally sponsor the formulation and implementation of a software measurement program as the basis for quantifying improvement. This document serves as the initial definition of measurement data that will be employed. It is expected that the effort begun with this document will continue with a follow-on implementation plan.

The authors acknowledge the following NSWCDD personnel for their support and insight into measurement of computer programs and processes: Jim Blackwelder, Paul Garnett (SYSCON), Ken Novell, Gary Richard, Don Robinson, Chuck Sperry (CSC), and Ann Storey.

Approved by:

[illegible]

LEATON M. WILLIAMS III, Head
Combat Systems Department

ABSTRACT

As part of an effort to improve the development and maintenance processes for AEGIS computer programs at the Naval Surface Warfare Center, Dahlgren Division (NSWCDD), a software measurement program has been initiated. The measurement program is based on a classification of software engineering entities representing the process employed, the products involved, the resources required, and the management controls specified. A standard set of attributes is defined across the software engineering entities as a means of obtaining complete measurement coverage. A life cycle process model is specified to standardize the collection of required data. Then a set of measures that span the entities and attributes and a method for validation of the defined measures are described. The defined measures are intended to be employed as part of a mechanism to provide the objective management necessary to support continuous improvement of AEGIS lifetime processes.

CONTENTS

Chapter		Page
1	INTRODUCTION	1
	Purpose	1
	Background	1
	Goals	2
	Approach	3
2	MEASUREMENT PROGRAM OVERVIEW	5
	Orientation	5
	Entity Classification	5
	Attribute Definition	6
	Attribute Relationships	7
3	LIFE-CYCLE PHASES	9
	Project Definition	9
	System Requirements	10
	Computer Program Requirements	11
	High-Level Computer Program Design	12
	Detailed Computer Program Design	13
	Code and Unit Test	14
	Integration and Formal Test	15
	Preparation and Installation	17
	Operation and Evaluation	17
	Management Support	18
4	MEASURES DEFINITION	21
	Product Measures	21
	Process Measures	24
	Resource Measures	25
	Control Measures	26
5	MEASURES VALIDATION	29
	Measures Validation Criteria	29
	Validation Plan	30
6	CONCLUSIONS	31

CONTENTS (CONTINUED)

Chapter — Page

GLOSSARY 33

BIBLIOGRAPHY 34

REFERENCES 35

APPENDIX
A. STANDARDS FOR SOFTWARE DEVELOPMENT . . A-1

DISTRIBUTION . . . . (1)

ILLUSTRATION

Figure — Page

1 MEASUREMENT PROGRAM CLASSIFICATION . . 4

TABLE

Table — Page

1 ATTRIBUTE/PHASE MATRIX 8

CHAPTER I

INTRODUCTION

PURPOSE

The Naval Surface Warfare Center, Dahlgren Division (NSWCDD) AEGIS Program supports ongoing efforts to assess and improve its computer program development and maintenance processes. This document describes one of these efforts, a software measurement program organized around software engineering entities. The attributes exhibited by the entities have been used as a basis for building a classification scheme for the entities and relating them to software development life-cycle phases. The resulting classification scheme can be used to build a measurement program based on the use of software measures that will quantify product quality, process quality, resource utilization, and management control.

BACKGROUND

As a participant in quality-improvement efforts, the AEGIS Program has pursued training opportunities and the exchange of experiences at a number of government and industrial sites. Concurrently with searching out new ideas and new ways to reinforce old standards of excellence, a programwide effort to describe the existing process was undertaken. A formal definition of the current computer program development and maintenance process had to precede efforts to improve the process. This major effort of definition and documentation was completed in mid-1991 by a team representing the various Dahlgren Division AEGIS organizations.[1] Only now is the program prepared to initiate changes to achieve a uniformly implemented process that is measurable and controllable.

Having formally defined our present methodology, we viewed establishing a measurement program as a critical technology that offered opportunity for improvement. The maturity of every scientific or engineering field is marked by its ability to measure progress, effort, cost, and quality. As far back as 1977, software had become so complex that it was beyond the capability of most engineers to understand the practical consequences of software changes merely by reading the program listings.[2] It was in that year that Gilb attempted to describe the emerging technology of software metrics, which he calls a "powerful language for describing the relationships which we desire, expect, or experience in any group of subsystems." To him, metrics simply meant measures. Measurement of component parts is still a necessary forerunner to understanding.

The software industry is emphasizing measurement as a necessary technique in achieving quality processes and products. In July 1990 *The Journal of Systems and Software* published a special issue on a metrics workshop, for example. Within the Navy, in December 1988, the Deputy Commander for Weapons and Combat Systems (NAVSEA 06) of the Naval Sea Systems Command directed[3] its managers to use software management indicators to provide a management overview of software management status; these indicators were developed from metrics. GE Aerospace, development contractor for the AEGIS Weapon System,[4] has developed an initial set of computer program metrics as a means of assessing the quality and reliability of their computer program products.

The initial motivations for establishing a measurement program in the AEGIS Program at NSWCDD came from two different avenues. One was managers who wanted to be able to understand current resource requirements in terms of both expenses and personnel. These same managers wanted to be able to estimate budgets early on in a project and make projections of resource requirements using objective data. Also frequently asked were questions concerning the productivity of personnel and the quality of products. The second avenue concerns similar needs in the technical area, only with a slightly different perspective. Here we have the following questions asked: "How do I best utilize my resources to get this task done?" "How do I meet this schedule and still produce a good product?" "What processes are most effective in reducing defects?" Any attempt to answer these questions must utilize a reliable, well-integrated, and supported data (metric) collection system.

GOALS

General goals were selected that address the software engineering entities: product, process, resource, and control.

Improve Product Quality

The first step in improvement is understanding. By focusing on defect analysis (such data as where the defect originated, where the defect was found, and a measure of defect severity), one has the best chance of understanding and improving the product. Defect analysis provides data on stability of the product, which not only comments on readiness of the software to proceed to the next stage of development but also contributes to schedule assessment.

Software measures can be used to provide feedback to an engineer concerning complexity of design or code. An early alert can lead to simplified designs or trigger remedial or avoidance actions. Complexity and defect measures can be used to predict error-prone modules and expected numbers of errors. The outputs of a software development program supported by measures can be represented graphically. Graphic representation aids our understanding of product complexity and highlights areas where improvements are needed.

Improve Process Quality

Process-oriented activities are usually the most costly and time consuming for a program to accomplish. Moreover, maintenance activities are more difficult to control than the other life-cycle phases and take the lion's share of the budget.[5] This high cost of maintenance is, in part, due to deficiencies in the other life-cycle phases; as much as 12 percent of the budget is expended on correcting production defects.[6] Thus, by improving the effectiveness of development processes and achieving more control over maintenance processes we can reduce system cost. Understanding the value of processes, through the quantitative view of software measures, is invaluable to managers who must allocate resources, define priorities, and set schedules. Over the long term, process improvements have a more positive effect on quality than product improvements.

Improve Resource Utilization

The very act of measuring the software engineering process leads to improvements in productivity. The use of common terminology in the metrics lexicon and the sharing of success stories leads to more consistent environment and use of tools. Measures provide insight into the efficiency of various aspects of the development process. Measures will be used as a quantitative means of assessing changes to our process and the value of tools. Knowing the value of processes and tools allows for a more efficient use of resources. Better utilization of resources can be directly related to reducing budget requirements. Also, a better understanding of the above directly supports justification of real budget requirements.

Improve Management Control

Measurement and analysis is a means for more accurate estimations of project milestones, as well as a useful mechanism for monitoring progress and expenditure of resources. Software measures provide insight into progress, or a lack of progress, and a better understanding of trends that can predict problem areas and processes needing additional resources.

Through measurement, we can optimize available planning data through which management will be able to estimate and schedule more effectively and make critical decisions early in the development process. A measured understanding of current practices and needs can be the basis of a successful long-term strategy[7].

APPROACH

Chapter 2 presents a classification of software engineering entities into four groups (product, process, resource, and control); presents attributes exhibited by these entities; and, finally, relates the attributes to software engineering life-cycle phases. See Figure 1. Chapter 3 describes those life-cycle phases as they relate to NSWCDD AEGIS software development and maintenance. Chapter 4 presents definitions for our initial set of software measures, and Chapter 5 lays the foundation for their validation. The authors' conclusions appear in Chapter 6, and the appendix contains a list of applicable standards.

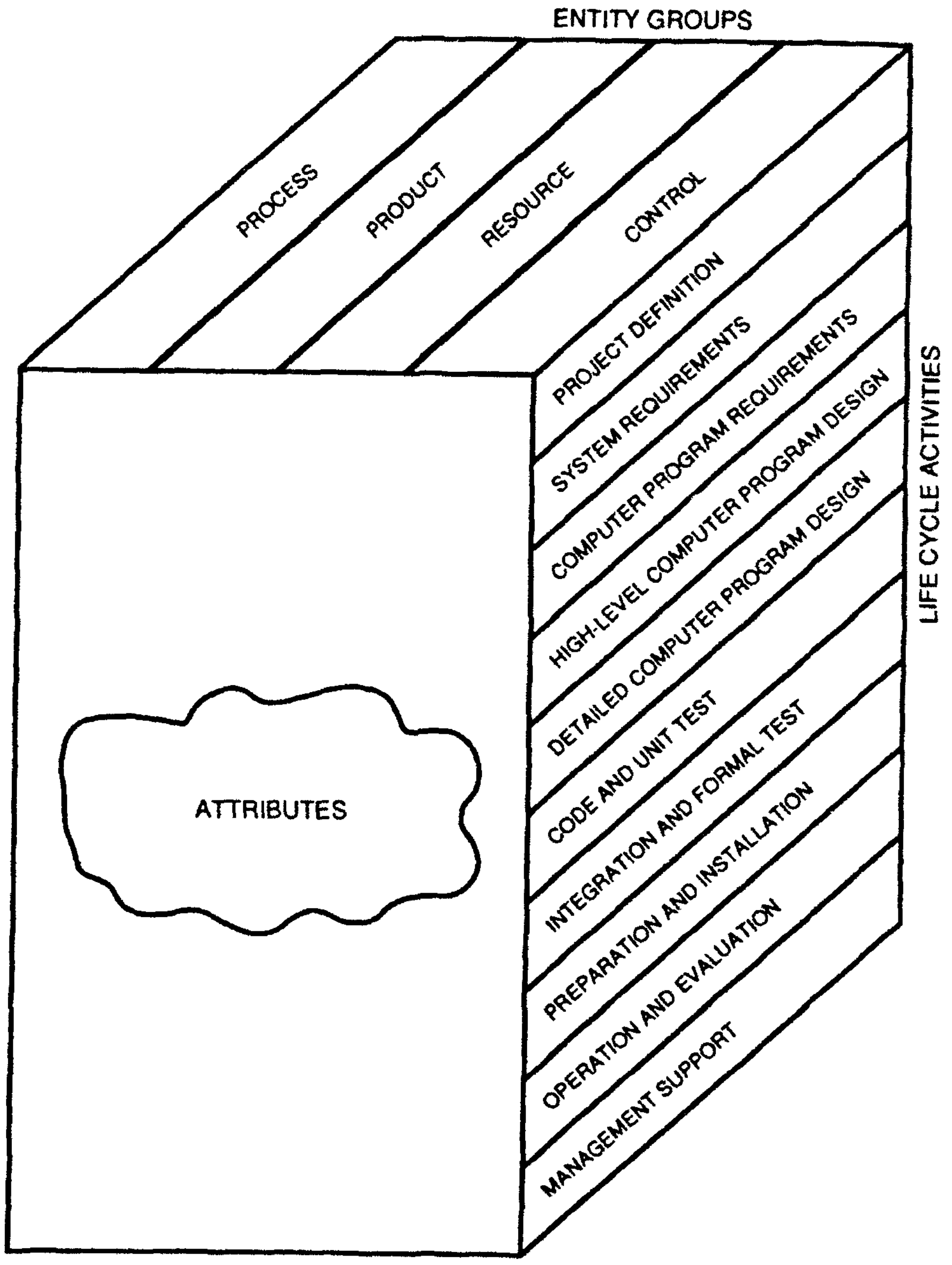

FIGURE 1. MEASUREMENT PROGRAM CLASSIFICATION

CHAPTER 2

MEASUREMENT PROGRAM OVERVIEW

ORIENTATION

The approach being taken is based upon several key ideas. One recognizes that complex systems must be decomposed into more elementary units for effective analysis. This concept applies to developmental processes and measurement programs as well as to functional and physical system components. A second recognizes that empirical techniques and data are often required in understanding effects within systems. Thirdly, a useful measurement program must address all parts of the system it is measuring. Finally, there must be consistent usage of terms and concepts.* The fact that the field of software engineering in general does not practice this consistency has been a major contributor to the current state of ambiguity and lack of cohesion in the use of tools and methods.

The framework of our measurement program is based upon a structured approach that requires well-defined processes with understood relationships.

ENTITY CLASSIFICATION

The production and support of a system such as AEGIS includes both technical and management aspects. The technical aspects can be represented by the processes used and the products produced. The management aspects can be represented by the project controls and resources.

The entities in software engineering[8] that are amenable to measurement fall into the following groups:

Product:
Any physical item that may be produced by an activity. Examples include code and documentation.

Process:
Any activity or collection of activities associated with software generation. Examples include code walkthroughs and system testing.

* The IEEE Glossary will be helpful in achieving consistency (see the appendix).

Resource:

The items that act within or are applied by the process to cause the creation of the products. Examples include personnel and computer equipment.

Control:

The items or aspects that relate to planning, organizing, staffing, directing, coordinating, reporting, and budgeting. Examples include baseline schedules and budgets.

ATTRIBUTE DEFINITION

In our classification scheme, the four groups of software engineering entities (product, process, resource, and control) making up the measurement domain are organized with respect to their essential attributes (see Table 1). This approach allows us to develop a framework for collecting metrics in a way that will highlight their interrelationships. The following represents a classification allowing for the definition of the specific attributes to be collected for the software engineering entities:

Schedules:

This attribute refers to time frames within which activities are accomplished, actual and planned. Collection will be for process and control. Examples are:

- Number of months for code and unit test (process)
- Number of days slip for a project activity (control)

Cost:

This attribute refers to those expenses, other than labor, such as training, hiring, equipment, travel, and marketing. Collection will be for resource. An example is:

- Cost of metrics collection training (resource)

Labor:

This attribute refers to staff hours including management, both planned and actual. Collection will be for process.
An example is:

- Hours of technical labor for specification change notice (SCN) update (process)

Size:

This attribute refers to the physical or functional magnitude of a component or aggregation of components. Collection will be for product, process, and resource. Examples include:

- Executable source statements per procedure (product)
- Number of test procedures (process)
- Number of direct charge personnel per project (resource)

Quality:

This attribute refers to a realization of the goodness of a component or aggregation of components. Collection will be for product, process, resource, and control. Examples include:

- Ratio of defects found in a design review to total defects one year after release (process)
- Count of failures during 25-hour stress test (product)
- Average training days per employee (resource)
- Ratio of planned budget to actual (control)

Complexity:

This attribute refers to the level of difficulty in understanding or assessing implications. Greater complexity is considered a negative trait. Collection will be for product, process, and control. Examples include:

- Cyclomatic complexity of source code (product)
- Number of organizational nodes involved in project (control)
- Number of test cases/steps required by system test for the project (process)

Volatility:

This attribute refers to the frequency and extent of change associated with the entity under consideration. Collection will be for product, process, resource, and control. Examples include:

- Number of specification change (SC) pages after Critical Design Review (CDR) (product)
- Personnel turnover (resource)
- Average number of months between procedure updates (process)
- Average baseline life (control)

Productivity:

This attribute refers to the efficiency with which an activity is accomplished with respect to some unit rate factor. Collection will be for control, process, and resource. Examples include:

- Number of unit test defects identified per labor hours expended (process)
- Number of defect-free procedures delivered (one year after release) per programmer hour (control)
- Average trouble report response time per year (resource)

ATTRIBUTE RELATIONSHIPS

Table 1 presents the relationships between the attributes and the appropriate life-cycle phases. Chapter 3 defines each of these phases and discusses each in terms of the supporting activities performed for which metrics will be collected. Chapter 4 defines the software measures to be used in the AEGIS Program for each attribute.

LIFE CYCLE PHASE	PRODUCT ATTRIBUTE								PROCESS ATTRIBUTE								RESOURCE ATTRIBUTE								CONTROL ATTRIBUTE							
	S	C	L	Z	Q	X	V	P	S	C	L	Z	Q	X	V	P	S	C	L	Z	Q	X	V	P	S	C	L	Z	Q	X	V	P
PROJECT DEFINITION									*		*							*		*					*				*	*	*	
SYSTEM REQUIREMENTS				*		*	*		*		*	*	*			*																
COMPUTER PROGRAM REQ				*	*	*	*		*		*	*	*			*																*
HL COMP PROG DESIGN				*	*	*	*		*		*	*	*			*																
DT COMP PROG DESIGN				*	*	*	*		*		*	*	*			*																
CODING & DEBUG				*	*	*	*		*		*	*	*			*																*
INTEGRATION & TEST				*	*		*		*		*	*	*	*		*		*														
PREP & INSTALLATION					*				*		*	*	*	*		*		*														
OPERATION/EVALUATION					*								*												*				*			*
MANAGEMENT SUPPORT															*	*		*			*		*	*					*	*	*	*

S = SCHEDULE	Q = QUALITY
C = COST	X = COMPLEXITY
L = LABOR	V = VOLATILITY
Z = SIZE	P = PRODUCTIVITY

TABLE 1. ATTRIBUTE/PHASE MATRIX

CHAPTER 3

LIFE-CYCLE PHASES

Chapter 2 described a classification scheme based on four entities (product, process, resource, and control) and their organization with respect to their essential attributes. In this chapter we discuss classification by software engineering life-cycle phases and the activities making up those phases. Our life-cycle phases are based on industry and government standards and procedures and have been tailored to meet the needs of the NSWCDD environment. Specific software engineering standards are listed in the appendix.

The life-cycle methodology within which the measurement program is implemented includes various control points. These control points indicate how well the methodology and its included activities are supporting the development of the required products and where improvement should be focused. The life-cycle phases addressed by this measurement program are the following:

- Project Definition
- System Requirements
- Computer Program Requirements
- High-Level Computer Program Design
- Detailed Computer Program Design
- Code and Unit Test
- Integration and Formal Test
- Preparation and Installation
- Operation and Evaluation
- Management Support

Each of these phases will be discussed in terms of its constituent key activities and control points.

PROJECT DEFINITION

The Project Definition phase involves the formulation of plans and the allocation of responsibilities. It includes any required procurement activities associated with products that are to be purchased rather than developed.

Project Planning

This activity includes the preparation of a project plan that documents the project scope in terms of project milestones, required project activities, key management controls, resources constraints, and allocated responsibilities.

Application Procurement

This activity pertains to the acquisition of already developed concepts, tools, and applications as opposed to the systems engineering and development of alternative solutions that would subsequently transition to follow-on phases. Such procurements would not normally have associated life-cycle support activities but would largely consist of procurement activities.

SYSTEM REQUIREMENTS

The System Requirements phase results in the system specification for the embedded computer system being developed. Further, it provides the initial definition for interface with other external systems.

Engineering Studies and Analyses (Needs Assessment)

This activity involves foundational engineering studies and analyses leading towards but preceding actual system conceptual definition.

Pre-Specification Analysis

This activity involves developing system conceptual definitions. It may include operational sequence diagrams, functional flow diagrams and descriptions, and configuration definition documents either as working papers or as formally controlled system models. It may also include prototyping, simulation development, report generation, and analysis.

Pre-System Design Review System Specification Development

This activity is concerned with the documentation of the system requirements and functional capabilities of the system. It specifies all performance parameters that the system must support. It includes all activities including documentation preparation and internal review necessary, preliminary to formal review.

Preliminary Prime Item Specification Development

This activity is concerned with the documentation of the preliminary element requirements and allocated functional capabilities of the system. It specifies all performance parameters that the element must support. It includes all activities, including preliminary documentation preparation and internal review, prior to computer program requirements analysis.

Interface Design Specification Preparation

This activity documents the specification of the functional signals between interfacing systems to the level necessary to begin development of computer program performance requirements. The specification progressively evolves over this and subsequent phases as additional details (represented by additional sections to the document) are successively added as necessary to continue the design process. The parts of the definition added during this phase represent the highest level of interface definition; namely, the identification of the required functional data, its significance, and its corresponding semantics.

System Design Review

This activity represents a formal control point in the management of the computer program development process. It constitutes a formal review of the system specification and the interface requirements.

Post-Review System Specification Development

This activity includes all efforts associated with producing the system specification including documentation activities and reviews precipitated by direction emanating from formal review.

COMPUTER PROGRAM REQUIREMENTS

The Computer Program Requirements phase results in the detailed performance specification of the computer program. The completed specification provides the necessary responses from and to the external signals defined in the interface design specification by defining the intermediate functions necessary to derive or transform data as well as to query or update required stored data.

Computer Program Requirements Analysis

This activity involves defining, analyzing, and selecting alternatives that meet the system-level requirements. The analysis typically includes various trade-off studies that have a bearing on alternative solutions. It may include modeling and simulation.

Prime Item Specification Development

This activity is concerned with the documentation of the element prime item requirements as a necessary aspect of completing the preliminary design review. It fully specifies all allocated system performance parameters to the element. It documents ali element-level test requirements.

Preliminary Computer Program Specification Development

This activity includes all efforts associated with developing the computer program performance specification including documentation activities and internal review and walk-throughs prior to formal review.

Interface Design Specification Development

This activity represents a continuation of the development and refinement of the external interface definition to the level necessary to begin development of computer program high-level design. In the previous phase, the external signals were defined and documented permitting the functional definition of the interfacing computer programs. This activity permits the physical definition of the interfacing computer programs to proceed in the next phase.

Preliminary Design Review

This activity represents a formal control point in the management of the computer program development process. It constitutes the formal review of the computer program performance requirements and the requirements for designing the interface to other systems.

Post-Review Computer Program Specification Development

This activity includes all efforts associated with developing the computer program performance specification including documentation activities and internal review and walkthroughs subsequent to formal review.

HIGH-LEVEL COMPUTER PROGRAM DESIGN

The High-Level Computer Program Design phase results in the physical architecture of the computer system, functional allocation of requirements, and top-level physical design of the computer program. The completed physical design description provides the necessary specification of the physical structures and their interfaces to permit detailed procedural design during the next phase.

Computer Program Architecture and Design Analysis

This activity includes analysis supporting the computer program architecture and preliminary allocation of computer program functions to processors and physical computer program structures.

Pre-Critical Design Review Computer Program Design Specification Development

This activity documents the high-level design of the computer program providing the allocation of computer program functions and stored data to physical computer program modules, computer program procedures, and data structures.

Internal Computer Program Design Walkthroughs

This activity represents the informal verification of the high-level design prior to the formal critical design review.

Interface Design Specification Updates

This activity pertains to changes to the interface design specification that result from more detailed analysis of the computer program physical design. Updates to the specification during this activity normally indicate incompleteness in the analysis performed in the previous phase. However, they could result from changes to user requirements.

Critical Design Review

This activity is a formal control point in the quality assessment of the development process. It results in an approved computer program design. After completion of this review, the computer program design is placed under formal configuration control.

Post-Review Computer Program Design Specification Development

This activity performs required changes to the approved and controlled computer program design documentation. These changes may result from deficiencies recognized in later design or program development phases or from the analysis and resolution of problem reports in any subsequent phase.

DETAILED COMPUTER PROGRAM DESIGN

The Detailed Computer Program Design phase results in the specification of the procedures and data structures required to implement functions and specified equations defined in the requirements specification.

Detailed Computer Program Design Analysis

This activity includes researching alternative design approaches that precede the actual documentation of the selected design. It also includes developing appropriate detailed design conventions that would be common at the detailed design level across all procedural units. Finally, it includes developing any common standards and test beds necessary for unit testing.

Pre-In Process Review Detailed Computer Program Design Development

This activity documents the computer program procedural designs and local data designs. These designs would typically be represented using a variety of techniques such as structure charts, data flow diagrams, structured English, cross-reference tables, and data dictionaries.

Module and Procedure Design Reviews and Walkthroughs

This activity, or series of activities, represents a formal control point in the quality assessment of the development process. It results in an approved detailed computer program design represented by the program design document. After completion of this review the document is placed under formal configuration control.

Post-Review Detailed Computer Program Design Development

This activity accomplishes the required changes subsequent to the review of the approved and controlled design document and test cases. These changes may result from deficiencies recognized in later program development stages or through the analysis and resolution of problem reports in any subsequent phase.

CODE AND UNIT TEST

The Code and Unit Test phase involves the creation of the source code necessary to implement and validate the detailed design.

Code

This activity pertains to the converting of the computer program detailed design into source code that can be successfully converted into machine language for execution upon the designated target computer to meet the computer program requirements.

Code Reviews and Walkthroughs

This activity, although normally conducted informally by the development group, is a formal control point in the quality assessment of the development process and requires concurrence by a peer review group in accordance with element standard operating procedures.

Unit Test

This activity involves the testing of the computer program code units in accordance with approved test plans upon completion of code reviews and walkthroughs as prescribed in element standard operating procedures. This activity, although normally conducted informally by the element, is a formal control point in the quality control of the development process. Upon successful completion of module testing, the computer program unit is placed under formal element configuration control.

Post-Unit Testing Program Modifications

This activity performs required changes subsequent to unit testing to the approved and controlled unit code.

Program Integration

This activity involves integrating the modules into the total computer program and performing informal integration tests to ensure that interfaces are in accordance with the computer program design.

Initial System Integration and Checkout (Informal)

This activity involves testing by the computer program developers after having successfully integrated individual modules. It includes testing with other element computer programs and external systems, using simulators as required, to validate that the computer program will function in its intended environment.

INTEGRATION AND FORMAL TEST

The Integration and Formal Test phase includes all formal, controlled system integration and testing for the complete system. The environment for testing will often include real-time simulations when the actual interfacing system or equipment is not available. The product of this phase is a fully assessed quality product that is ready for installation in the intended environment.

Element Test Plan and Procedures Development

This activity involves the preparation, review, and approval of all test plans and procedures necessary for supporting computer program testing in accordance with the test requirements in the computer program performance specification. These plans and procedures will document all supporting services, resources, analyses, and reports required to complete this activity.

Element Testing

This activity includes the formal, controlled testing by the respective combat system element to ensure that the computer program configuration item is in accordance with its computer program performance requirements. Although this activity is a formal control point in the quality control of the development process, the actual control occurs after completion of the next activity when the analysis and reporting on the results of the computer program testing activity have been completed.

Element Testing Data Analysis and Reporting

This activity includes all data analysis and reporting for the computer program testing. It is performed in accordance with the requirements stated in the computer program test plan.

System Integration Testing

This activity includes the preliminary system-level testing to ensure that all required interfaces are being supported and that all observable functions are supported and work appropriately. The primary intent is to do a preliminary assessment sufficient to gain the confidence in the system to schedule the necessary resources and begin the formal system-level testing.

System Test Plan and Procedures Development

This activity involves the preparation, review, and approval of all test plans and procedures necessary for supporting system testing in accordance with the test requirements in the system-level performance specification. These plans and procedures will document all supporting services, resources, analyses, and reports required to complete this activity.

System Testing

This activity includes the formal, controlled testing by the system integration and testing organization to ensure that the system is in accordance with its specification. Although this activity is a formal control point in the development process, the actual computer program certification occurs after completion of the next activity when the analysis and reporting on the results of the system testing activity have been completed.

System Testing Data Analysis and Reporting

This activity includes all data analysis and reporting for the system-level testing. It is performed in accordance with the requirements stated in the system test plan.

PREPARATION AND INSTALLATION

The Preparation and Installation phase is the final step in producing the product for installation and checkout. It defines the particular attributes of the system to be produced for a given ship or site and the preparation of appropriate documentation related to it.

Build Definition and Preparation

This activity includes all preparations required to provide well-defined descriptions of the precise computer program version that is to be created from all the components (both computer program and documentation) of the various versions that are available. It includes identification of all corrections (various types of patches) as well as environment-specific adaptation files. It includes controls in terms of various signed instructions from the elements* contributing products to the build.

Computer Program Assembly and Packaging

This configuration management (CM) activity includes the gathering and packaging of all AEGIS Weapon System (AWS) tactical and support computer program media that will be transported to and left on the ship as part of a custody transfer or baseline update. This media consists of all magnetic tape and disk packs as well as the preparation of load program documentation not available from previous phases. The load program documentation, as a minimum, includes the following: load program descriptions, data extraction/data reduction guidelines, computer program description document, delivery description documents, disk/tape listings, operating manuals, quick reference guides, version description documents, and technical bulletins.

OPERATION AND EVALUATION

The Operation and Evaluation phase includes those activities subsequent to computer program installation, shipboard or at a land-based engineering facility. It includes user training, analysis in support of users, external support to fleet exercises and testing, and problem assessment and resolution.

Problem Report Validation and Impact Assessment

This activity involves the review and assessment of the impact of formally submitted problem reports. Reports determined to represent valid problems are assessed further as to what parts of the system are affected by the problem.

* The term element refers to the organization responsible for a major component of the AWS or its support system as well as the computer program itself.

Problem Report Resolution Assessment

This activity involves the analysis of reported problems to determine changes required to the computer programs and documentation, resources required to make the changes, origin of defect (i.e., computer program development phase and activity where the defect was most likely introduced), and the recommended priority.

User Analysis and Training Support

This activity involves all analysis and training support for users subsequent to system installation and checkout.

External Support

This activity includes all support provided to other organizations responsible for the following events: installation and checkout (INCO), engineering test, acceptance test, ship trials, technical evaluation (TECHEVAL) support, operational evaluation (OPEVAL) support, and combat system ship qualification trials (CSSQTs).

MANAGEMENT SUPPORT

Management Support includes activities that span all phases of the development and maintenance process rather than consisting of a distinct phase itself. From a measurement viewpoint, management metrics will be collected throughout the effort.

Project Management

This activity includes the preparation of such products as project plans, resource estimates, and progress and milestone reports.

Administration

This activity includes all office management activities not specifically included within other activities. It includes general recordkeeping, personnel appraisal activities, and support for general office procedures and controls.

Quality Assurance

This activity relates to a formal quality assurance function for the program in terms of both products and processes. A substantial part of the quality assurance role is achieved through the development of required instructions and procedures and inspections to ensure that instructions have been implemented and that procedures are being followed. Problem report and test reports are reviewed to identify problem areas. Metrics data are collected and analyzed to identify trends and weaknesses in the processes that need management attention. The quality assurance presence is manifested throughout all activities.

Configuration Management

This activity relates to a formal control and accounting function for the program. It includes the following tasks:

Configuration Identification pertains to the identification and definition of all valid equipment and computer program components of development products, facilities, and shipboard items.

Status Accounting pertains to the definition of requirements for and maintaining the configuration management data base records of system, equipment, computer program, and component identification including location, change properties, status, planned change installations, and audit results.

Change Control pertains to the configuration of systems or components and the control of changes to them. Development of configuration management instructions and procedures provides the primary means of maintaining control over the configuration.

Audits pertains to the conduct of configuration audits or reviews including ship or other site installation and change validation.

Facilities

This activity pertains to the design, acquisition of space and equipment, operation, and support of facilities required for performing development and fleet support functions for the AEGIS Program. It includes the following tasks:

Site Planning includes all planning tasks necessary for development or upgrade and activation of new facilities.

Site Design involves producing plans and designs for a specific facility or site.

Site Activation involves the construction of new facilities, the procurement of equipment and computer programs needed to operate the facility, the physical installation and checkout of the equipment, the design and conduct of test procedures to assure proper operation of the computer programs, and the demonstration of required operational capabilities.

Maintenance Methodology

This activity produces and maintains a formal, controlled methodology for the AEGIS Program. The methodology function is responsible for the preparation of documentation that identifies the processes, data required and produced within each activity, the controls over the activity, and the organization responsible for the activity. The measurement program within the maintenance methodology activity is responsible for analyzing software measures to determine changes to the process that are needed to improve product quality and productivity. The measurement program provides an empirical basis for implementing process improvements. The measurement program provides the definition and refinement of software measurement, but its implementation is a line management function.

CHAPTER 4

MEASURES DEFINITION

Earlier in this report we introduced the terms software engineering entity, any object or concept from software engineering that is amenable to measurement; and attribute, a characteristic or property exhibited by an entity. Relationships between entities exhibiting the same attribute can reveal important information. The term used in referring to the data collected to quantify essential software attributes is metric; a measure is information stemming from a metric used in assessment and decision-making. The literature defines a valid software measure[9] as a collected metric where the following criteria are met:

a. It is an attribute of the software engineering entity.

b. Abstractions for defining the attribute exist.

c. The attribute determines important relationships between the entities possessing it.

d. A mapping exists from the abstraction to a usable number system that preserves order relationships.

Within this section we present the definitions for our initial set of sofware measures.* Our criteria for validating these measures appear in Section 5. Their proper use through classification, comparison, and mathematical analysis provides the means of assessing where improvements are needed and whether changes to the processes have led to improvements. The set of metrics required for these measures, its collection, analysis, and application in our methodology will be addressed in the measurement plan that will be developed subsequent to this report.

PRODUCT MEASURES

SIZE Number of high-level source statements, excluding comments, in thousands (KNCSS).

Average number of executable source statements per procedure (ESS).

* Consult the glossary if a term or abbreviation is unfamiliar.

Number of direct code source statements, excluding comments, in thousands (KNCDCS).

The memory required to be allocated so that the operational program can be run on the tactical system (load file size).

Number of executable source statements in thousands (KESS).

Number of lines of source code in thousands (KLOC).

Number of unique test cases required to verify conformance to requirements.

Function points, a generic measurement of computer program size.*

Number of SCs and interface design specification change requests (ICRs) approved for a project.

Number of unique computer program procedures in final quality assurance (QA) build per element.

Number of change pages associated with an SC or ICR for a project.

QUALITY

Ratio of the number of tests completed on first pass to the total tests run.

Ratio of number of defects written against the document to number of pages in the document subsequent to a formal control point (i.e., SDR, PDR, CDR, computer program certification panel (CPCP), IPR) through one year after CPCP.

Ratio of number of defects written against the computer program to KNCSS subsequent to QA load file build through CPCP and again through one year after CPCP (defect density).

Ratio of number of high-priority defects written against the computer program to KNCSS subsequent to QA load file build through CPCP and again through one year after CPCP.

Ratio of number of non-comment cells of patch (per element) to load file size (patch density).

* See Function point, in the glossary.

Ratio of number of defects found during formal system test to KNCSS. (test defect density).

Mean time between system failures (MTBF).

Count of all software defects found through one year after CPCP.

Count of user problems and change requests - number of computer program change requests (CPCRs) written.

Comment correctness based on statistical sampling per program.

COMPLEXITY

McCabe cyclomatic complexity[10] of an individual source procedure.

Average complexity of the procedures whose cyclomatic complexity is in the top 10 percent for each program within a project. (NAVSEA Code C06 complexity).

Number of unique source procedures which require noncomment changes to implement an individual CPCR.

Ratio of the number of interfaces among procedures to the number of procedures per program (design complexity).

Ratio of the maximum requirements subparagraph nesting level, within a document section, to the total number of subparagraphs in that section. The largest of these section ratios serves as the document complexity.

Reading comprehension level for specifications (Grammatik IV*).

Number of module interfaces affected.

VOLATILITY

Number of noncomment lines of source code added, changed, and deleted after QA build.

Number of function points added and deleted after QA build.

Number of SC and ICR changes after formal review per project.

Number of revisions to project plan.

Number of element program memory cell changes after unit test is completed through CPCP per project.

* Commercial computer program.

Yau and Collofello's Stability Metric - A measure of the maintainability characteristics of software computed by summing the McCabe complexity of all procedures impacted by a changeto one of the interface variables in a procedure.[11]

PROCESS MEASURES

SCHEDULE	Number of months of calendar time required from completion of a project plan through CPCP.
	Calendar time in months between planned activity completion and actual activity completion.
	Number of months of calendar time for a project activity.
	Relative percentage of calendar time spent in each of the following activities: requirements, design, code and unit test, and integration and test on a per-project basis.
LABOR	Relative percentage of direct labor hours spent in each of the following activities: requirements, design, code and unit test, and integration and test per project.
SIZE	Total hours of formal system test prior to CPCP.
	Number of unique process steps required for the activity per project.
	Total direct labor hours spent in support of formal reviews per project.
QUALITY	Ratio of defects found during an individual review to total defects found through one year after CPCP.
	Ratio of defects introduced in correcting CPCRs to total CPCRs implemented through one year after CPCP.
	Ratio of number of CPCRs generated from test observation reports (TORs) to number of TORs written up to CPCP per project.
	Ratio of test procedure TORs written to CPCRs identified by test activity.
	Sum of the absolute differences of the ideal unit test ratios and the actual unit test ratios. The ideal unit test ratio is determined from the unit complexity and the aggregate unit complexity. The actual

	unit test ratio is determined from the unit test procedure page count and the aggregate unit test procedure page count.
	Number of waivers associated with product at CPCP.
COMPLEXITY	Number of test cases required to assure requirements compliance.
	Number of different functional areas required to accomplish an activity.
VOLATILITY	Percentage of procedures, instructions, and directives that change per year.
PRODUCTIVITY	Average calendar time in days from initial Change Review Board (CRB) status date to CPCP for Class I CPCRs corrected.
	Ratio of defects identified per activity to total direct labor hours charge.

RESOURCE MEASURES

COST	Equipment expense.
	Training expense.
	Travel expense.
	Staffing expense.
	Commercial software expense.
	Total operations expense, including contract support, for the AEGIS Computer Center (ACC) per CPU hour of operation.
SIZE	Average number of direct-charge in-house personnel per project.
	Average number of direct-charge contract personnel per project.
QUALITY	Average number of job-related training days per year per employee.
	Average number of credit hours per employee per year (with a grade of C or better).
	Average technical personnel experience level.

	Average management personnel experience level. (Experience level is based on the following weighting intervals in years; years < 3, 3 < = years < 10, years > = 10.)
VOLATILITY	Ratio of number of personnel transfer actions to average number of people on board per year.
PRODUCTIVITY	Average trouble report response time per year.

CONTROL MEASURES

SCHEDULE	Summation of calendar weeks slip over the major control points (SDR, PDR, CDR, IPR, and CPCP) between planned and actual per project.
QUALITY	Ratio of planned budget to actual per project.
	Percentage of control functions addressed in writing per project.
	Percentage of written program directives addressing control functions.
	Ratio of actual program activities to those documented in plans.
	Ratio of planned personnel resources to actual personnel assigned per project.
COMPLEXITY	Number of organizational nodes involved in project (organizational dispersion).
	Ratio of AEGIS budget to total budget per organization.
	Ratio of sum over ship classes of number of ships times number of baselines to sum over ship classes of number of ships per year.
VOLATILITY	Number of replans per project.
	Average number of calendar months between baseline releases within a class of ships (baseline life).
PRODUCTIVITY	Ratio of changed lines of code to total project activity labor hours.
	Average square foot of floor space per employee.
	Average square foot of shelf space per employee.

Ratio of the number of changed function points to total project labor hours.

Ratio of the number of changed pages of documentation to total project activity labor hours.

Ratio of the number of changed defect-free procedures per product to total changed procedures per project per programmer hour (through one year after CPCP).

CHAPTER 5

MEASURES VALIDATION

The software engineering entity groups pertinent to the AEGIS Program have been specified as product, process, resource, and control. Measurement of these entities is in terms of eight attributes: schedule, cost, labor, size, complexity, volatility, and productivity.

In some cases these attributes are directly measurable. When this is the case, such measures will be viewed as validated by definition. In other cases, the measure only indirectly represents an attribute for a particular software engineering entity. Then, measure validation is appropriate and will be applied. Our concept for the validation of measures requires that a specified measure provide an appropriate indication of what it is intended to represent.

MEASURES VALIDATION CRITERIA

Validation is a process that assesses a measure in terms of several statistical properties. It assesses the acceptable degree to which magnitude, order, change, and prediction using measured data correlate with expected values. We suggest a measure validation method based on six validation criteria as defined by studies done at the Naval Postgraduate School.[12] Each of those criteria--associativity, consistency, discriminative power, tracking, predictability, and repeatability--support certain quality factors.[*] Brought out in the study is the finding that nonparametric statistical methods are shown to play a role in evaluating measures against the validation criteria. This is especially important since it is consistent with the nonlinear, nonnormal, and potential variable nature of metric data. The following qualitative descriptions are given for each of their criteria.

Associativity. A measure that is validated according to this criterion can be used to allow magnitude comparison to measures obtained from different entities to estimate the degree to which they differ in quality; e.g., the quality of Component 2 is twice that of Component 1.

[*] A quality factor is an attribute of software engineering that contributes to its quality. A quality factor can also be a measure.

Consistency. A measure that is validated according to this criterion can be used to compare relative ranking among entities; e.g., Component 2 is better than Component 1.

Discriminative Power. A measure that is validated according to this criterion can be used as a substitute for an unavailable quality factor to identify an unacceptable quality; e.g., a lack of design traceability is an indication of poor reliability.

Tracking. A measure that is validated as tracking a quality factor on a previous similar project could serve as a substitute for tracking that quality factor on the project of interest; e.g., procedure complexity is used to track future maintainability.

Predictability. A measure from an early phase (e.g., design) that is validated according to this criterion could be used to make predictions about a different but related attribute in a later phase (e.g., testing).

Repeatability. A measure that has successfully demonstrated one of the previous five criteria over a given percentage of projects establishes confidence that the measure can represent a given quality factor; e.g., a complexity measure must be able to predict maintainability for eight of ten components.

VALIDATION PLAN

The specific measures that require validation will be defined in the measurement plan. Each measure requiring validation will be identified, as well as the particular validation criteria and associated tests.

CHAPTER 6

CONCLUSIONS

A software measurement program supporting improvement in AEGIS development and maintenance processes must be based on a structured approach that will satisfy needs in both management and technical areas. This report has presented a general set of program goals; management involvement in establishing specific measurement goals that will be actively supported is critical. Those goals will bound and help define the methods to be employed, resources needed, time frames for implementation, and management support required.

A structured approach begins with the definition of the entities of software development and maintenance (grouped by product, process, resource, and control). The entities serve as a basis for defining the attributes necessary to quantify many of the less tangible aspects of the software development and testing process. The report has shown these attributes can be related to life cycle phases supported by DOD-STD-1679A[13] and the various activities accomplished within those phases.

An initial set of software measures for the AEGIS Program has been defined. The use of measures can lead to the realization of the following benefits:

- More objective software management
- More understandable processes
- A measurement of progress
- A common terminology
- Elimination of the causes of defects
- Better estimates
- Improved planning and scheduling
- Identification of good/poor engineering practices
- Identification of components that are error prone or costly to maintain
- Encouragement of good management practices

The measures presented are not intended to be implemented all at once, but in an incremental fashion. An approach consistent with the Software Engineering

Institute's* (SEI's) maturity levels** should be considered. This approach would allow the AEGIS Program to implement those measures it is **now** capable of collecting and utilizing to support process improvement. As the program advances to a higher maturity level, other measures should be added. Over the next few years, measures relating to the following project aspects (grouped in order of SEI maturity level) will be collected and analyzed:

- Project size, project effort, project management, project planning, configuration management, quality assurance
- Software size, effort, tracking, requirements volatility, personnel experience, personnel turnover
- Standards usage, product complexity, product quality practices and mechanisms, error statistics, training
- Functional testing coverage, process productivity, defect distribution, progress indicators, analysis of CM tracking DB data, internal review standards
- Process quality, tool insertion.

We must always keep in mind that a software measurement program can be successful only if it is part of an overall strategy for process improvement. Software process improvement can be achieved only with the active support and involvement of management.

* Federally funded research and development center established in 1984 at Carnegie Mellon University, Pittsburgh, Pennsylvania.

** An assessment of the level of control exhibited by an organization over its software engineering process. Level 1 is the initial stage, which is ad hoc and chaotic. Level 2 is the repeatable stage, which is intuitive and dependent on individuals. Level 3 is the defined stage, which is qualitative and has a defined and institutionalized process. Level 4 is the managed stage, which has a measured process. Level 5 is the optimizing stage, where improvement is fed back into the process.

GLOSSARY

Computer Program Certification Panel (CPCP) - A group convened by Baseline Management and chaired by N05 whose purpose is to determine the readiness of the baseline upgrade for shipboard installation.

Control - Management functions related to planning, organizing, staffing, directing, coordinating, reporting, and budgeting.

Defect - A product anomaly. Examples include such things as (1) omissions and imperfections found during early life-cycle phases and (2) faults contained in software sufficiently mature for test or operation.

Design Review Process includes a System Design Review (SDR), which reviews the system operational requirements and the allocated performance requirements; a Preliminary Design Review (PDR) following the SDR, which is a technical review of the performance requirements and the basic design approach for each element; and a Critical Design Review (CDR), where the design for each element is formally reviewed and the system is committed to development. In-process reviews (IPRs) are conducted as required.

Entity - An object or concept from software engineering that is amenable to measurement; i.e., product, process, resource, or control.

Failure - (1) The termination of the ability of a functional unit to perform its required function. (2) An event in which a system or system component does not perform a required function within specified limits. A failure may be produced when a fault is encountered.

Fault - (1) An accidental condition that causes a functional unit to fail to perform its required function. (2) A manifestation of an error in software. A fault, if encountered, may cause a failure. Synonymous with bug (IEEE Standard Glossary).

Function Point - Weighted and adjusted total of the number of inputs, outputs, inquiries, logical data files, and interfaces that are associated with an application. Independent of the source code statement.

Process - Any activity or collection of activities associated with software generation.

Product - Any physical item which may be produced by an activity.

Project - Activity having begin and end dates.

Resource - Nonconsumable items that act within or are applied by the process to cause the creation of the products.

BIBLIOGRAPHY

The Institute of Electrical and Electronics Engineers, Inc. "IEEE Guide for the Use of IEEE Standard Dictionary of Measures to Produce Reliable Software." *IEEE Std 982.2-1988.*

The Journal of Sysiems and Software. July 1990

Roberts, F. S. *Measurement Theory with Application to Decision Making, Utility, and the Social Sciences.* Reading, Mass: Addison Wesley, 1979.

REFERENCES

1. AEGIS Maintenance Methodology Steering Panel, AEGIS Program Office, *AEGIS Maintenance Methodology Manual: Process Definition,* NAVSWC MP 91-19, 2 August 1991, Dahlgren, VA, 22448.

2. Gilb, Tom, *Software Metrics,* Winthrop Publishers, Inc., Cambridge, MA, 1977, p. 132.

3. "Procedures Guide for Software Management Indicators," *Software Quality Improvement (SQI) Program,* Department of the Navy, Naval Sea Systems Command, Combat Systems Engineer - 06D2Q. Washington, DC, 11 September 1990.

4. Government Electronic Systems Division, *AEGIS Computer Program Metrics Plan,* Moorestown, NJ, 1 August 1991.

5. Boehm (1979) and Brown (1980), as cited in Kafura, Dennis, and Reddy, Geereddy, R., *The Use of Software Quality Metrics in Software Maintenance,* TR-85-33, Virginia Polytechnic Institute, Blacksburg, VA, August 1985.

6. Perry, W. E., "Software Still Is Arriving With These Annoying Bugs," *Government Computer News,* 1987.

7. Brady, Robert B., and Caswell, Deborah L., *Software Metrics: Establishing a Company-Wide Program,* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1987.

8. Bush, Martin E., and Fenton, Norman E., "Software Measurement: A Conceptual Framework," **J. SYSTEMS SOFTWARE**, 12:223-231.8.

9. Baker, Albert L., et al., "A Philosophy for Software Measurement," **J. SYSTEMS SOFTWARE**, 1990; 12:277-281.

10. McCabe, Thomas J., "Structured Testing: A Software Testing Methodology Using the Cyclomatic Complexity Metric," McCabe & Associates, Inc., under sponsorship of National Bureau of Standards, Report NBS SP 500-99, December 1982.

11. Yau, S., and Collofello, J., Some Stability Measures for Software Maintenance," **IEEE Transactions on Software Engineering**, Vol SE-6, November 1990.

12. Norman F. Schneidewind, "Validating Software Metrics," Naval Postgraduate School, Monterey, CA, September 1990. Unpublished communication with NSWCDD.

13. *Software Development,* 22 October 1983, Military standard under which AEGIS was developed.

APPENDIX A. STANDARDS FOR SOFTWARE DEVELOPMENT

The Institute of Electrical and Electronics Engineers, Inc. **IEEE Standard Glossary of Software Engineering Terminology**. An American National Standard. ANSI/IEEE Std 729-1983. 1983.

The Institute of Electrical and Electronics Engineers, Inc. **IEEE Guide for the Use of IEEE Standard Dictionary of Measures to Produce Reliable Software**. IEEE Std 982.2-1988.

The Institute of Electrical and Electronics Engineers, Inc. **Standard for a Software Quality Metrics Methodology**. Draft. P-1061/D21. 1 April 1990.

Department of Defense. **Military Standard: Defense System Software Development** and related data item descriptions. DOD-STD-2167A. 29 February 1988.

Department of Defense. **Military Standard: Software Development** and related data item descriptions. DOD-STD-1679A. 22 October 1983.

Department of Defense. **Mititary Standard: Defense System Software Quality Program**. DOD-STD-2168. 29 April 1988.

Department of Defense. **Military Standard: Specification Practices.** MIL-STD- 490A. 4 June 1985.

AEGIS Shipbuilding Program (PMS-400). **AEGIS Combat System Design Review Manual. Volume I, Forward Fit; Volume 2, Backfit**. May 1991.

DISTRIBUTION

	Copies
ATTN PMS 400B	1
AEGIS PROGRAM MANAGER	
DEPARTMENT OF THE NAVY	
WASHINGTON DC 20362	
EDWARD PRIMM CODE 6000A	1
NAVAL SURFACE WARFARE	
CENTER	
PORT HUENEME DIVISION ECO	
DAM NECK	
VIRGINIA BEACH VA 23461	
MAILSTOP 138-303 (MALLOY)	1
127-302 (BLAZEWITZ)	1
GE AEROSPACE	
GOVERNMENT ELECTRONICS	
SYSTEMS DIVISION	
GENERAL ELECTRIC CO	
BORTON LANDING RD	
MOORESTOWN NJ 08057	
ATTN C GRAHAM	1
D KEHN	1
COMPUTER SCIENCES CORP	
304 W RT 38	
MOORESTOWN NJ 08057	
ATTN C SPERRY	1
COMPUTER SCIENCES CORP	
STE 201	
4001 OAK MANOR OFFICE PARK	
KING GEORGE VA 22485	

	Copies
ATTN D PAULSON	1
P GARNETT	1
D HAUN	1
R WILLIAMS	1
SYSCON CORPORATION	
PO BOX 1480	
DAHLGREN VA 22448	
ATTN K HYLAND	1
OAK RIDGE ASSOCIATED UNIV	
PO BOX 117	
OAK RIDGE TN 37831-0117	
ATTN DR R E NANCE	1
SYSTEMS RESEARCH CENTER	
VIRGINIA POLYTECHNIC INSTITUTE	
AND STATE UNIVERSITY	
BLACKSBURG VA 24061-0251	
ATTN JOHN ZIMMERMAN	1
SOFTWARE PRODUCTIVITY	
RESEARCH INC	
77 S BEDFORD ST	
BURLINGTON MA 01803	
DEFENSE TECHNICAL	
INFORMATION CENTER	
CAMERON STATION	
ALEXANDRIA VA 22304-6145	12
ATTN GIFT AND EXCHANGE	
DIVISION	
LIBRARY OF CONGRESS	
WASHINGTON DC 20540	4

DISTRIBUTION (CONTINUED)

INTERNAL DISTRIBUTION:

	Copies
E231	3
E232	2
F42	1
F42 (STROCK)	1
G72 (LUSHER)	1
K52 (FARR)	1
L10 (BLACKWELDER)	1
N05	1
N054	1
N055	1
N15	1
N15 (RUSHLOW)	1
N20	1
N20B	1
N20B (STONE)	1
N20P	10
N20P (RICHARD)	1
N21	1
N21 (HENSHAW)	1
N21 (HORMAN)	1
N21 (JOHNSON)	1
N21 (LAMB)	1
N21 (SOKOLOWSKI)	1
N21 (STOREY)	1
N22	1
N22 (HANEY)	1
N22 (MURPHY)	1
N22 (SCARAMOZZI)	1
N23	1
N23 (BARTHOLOW)	1
N23A (HERRON)	1
N23 (MCCONNELL)	10
N23 (HEBERLEIN)	1
N24	1
N25	1
N25 (BUCKLER)	1

	Copies
N73	1
N73 (LITTLE)	1
N74 (GIDEP)	1
N81	1
N81 (GEORGE)	1
N84	1
N84 (CULLEN)	1

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** March 1992	**3. REPORT TYPE AND DATES COVERED** Final March 1992

4. TITLE AND SUBTITLE AEGIS Measures Definition	**5. FUNDING NUMBERS**
6. AUTHOR(S) Edward J. Dudash David E. McConnell	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Surface Warfare Center, Dahlgren Division (Code N21), Dahlgren, Virginia 22448-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES	
12a. DISTRIBUTION/AVAILABILITY Authorized for public release; distribution is unlimited.	**12b. DISTRIBUTION CODE**

13. ABSTRACT *(Maximum 200 words)*

As part of an effort to improve the development and maintenance processes for AEGIS computer programs at the Naval Surface Warfare Center Dahlgren Division, a software measurement program has been initiated. The measurement program is based on a structure representing the process employed, the products involved, the resources required, and the management controls specified. A standard set of attributes is defined across the software engineering entities as a means of obtaining complete measurement coverage. A life-cycle process model is specified to standardize the collection of required data. Then a set of measures that spans the entities and attributes and a method for validation of the defined measures is described. The defined measures are intended to be employed as part of a mechanism to provide the objective management necessary to support continuous improvement of AEGIS lifetime processes.

14. SUBJECT TERMS Life-Cycle Phases, Measurement Definition, Measurement Validation	**15. NUMBER OF PAGES** 42
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR