Software Quality Measurement: Validation of a Foundational Approach (Final Report, Year Two)

James D. Arthur, Richard E. Nance, and Edward V. Dorsey

TR 92-19

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

April 23, 1992

Technical Report SRC-92-003*

SOFTWARE QUALITY MEASUREMENT: VALIDATION OF A FOUNDATIONAL APPROACH

(Final Report, Year Two)

James D. Arthur
Richard E. Nance
Edward V. Dorsey

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

23 April 1992

---

* Cross-referenced with CS-TR-92-19, Department of Computer Science, Virginia Tech.

Work supported by the Joint Logistics Commanders through the Systems Research Center under contract number N60921-89-D-A239 B004.

A2-9

ABSTRACT

This report discusses the second year findings of a proposed four year investigation effort that focuses on the assessment and prediction of software quality. The research exploits fundamental linkages among software engineering Objectives, Principles, and Attributes (the OPA framework). Process and documentation quality indicators (DQI) are presented relative to the OPA framework with an elaboration on their individual role in assessing and predicting software quality. The development of a document quality analyzer is discussed, with a particular emphasis being placed on the selected subset of DQI measures that it provides.

Key Words and Phrases: Simulation quality assessment and prediction; software engineering objectives, principles and attributes; software quality indicators, process indicators, documentation quality indicators, automated document analysis.

Table of Contents

1. Introduction and Project Overview 4

2. Task Statement: Year Two 4

3. Refinement of Product Quality Indicators 5

   3.1 Code Indicators 5
   3.2 Documentation Quality Indicators 6
   3.3 Process Indicators 15

4. Project Installation, Instrumentation, Data Collection and Refinement 18
   4.1 Interaction Between VTSRC and E-Systems Personnel 18
   4.2 Installation and Instrumentation 19
   4.3 Code Walkthroughs 26
   4.4 Data Collection and Refinement 27

5. Activities Related to Deployment 27

6. Summarization and Future Plans 28

   6.1 Year Two: Completed Work 28
   6.2 Year Three: Current Research 29
   6.3 Year Four: Future Plans 29

7. References 31

8. Appendix A - Process Indicators 32

9. Appendix B - Implemented Documentation Quality Indicators 35

1. Introduction and Project Overview

This report describes activities and findings associated with year two of a proposed four year effort focusing on the validation of a procedure for predicting and assessing software quality. The procedure utilizes an underlying foundation provided by the Objectives/Principles/Attributes (OPA) framework [ARTH90]. The OPA framework recognizes the relationships between project-level goals and desirable software engineering objectives. It defines sets of linkages that relate the achievement of such objectives to principles that, when employed in the development process, introduce desirable attributes in the product. The proposed software quality prediction and assessment procedure employs the identification and quantification of process and product properties that undeniably indicate the presence (or absence) of desirable attributes, which in turn, indicate use of desirable software engineering principles in the development process and the achievement of project-related software engineering objectives.

From a year-by year perspective, major steps in the validation effort are:

Year One: The identification of code and documentation indicators, the refinement of code indicators and the development of a code analyzer (status: completed),

Year Two: The refinement of code and documentation indicators, the identification of prospective process indicators, the development of a documentation analyzer, the selection of a validation site and the initiation of on-site activities (status: completed),

Year Three: Continued refinement of code, documentation and process indicators through data collection, experimentation and result assessment, and the design and implementation of deployment activities related to the collection of validation data (status: in progress), and

Year Four: The collection of deployment data and hypothesis validation through statistical analysis (status: future work).

2. The Task Statement : Year Two

The set of subtasks presented below outline the separate research directions for year two of the Software Quality Measurement project. They also reflect, in order, the subjects addressed in the following major sections.

Subtask 1: Continued Refinement of Product Quality Indicators: Review and continue to refine code, documentation and process indicators with an emphasis being placed on the latter two; concurrent with the refinement process, develop a prototype document analyzer that exploits measurable document quality indicators (DQIs).

Subtask 2: Project Installation and Instrumentation, Data Refinement and Collection.: Examine the selected program's software development process to determine the optimum integration strategy and data collection points; use both manual and automated approaches to collect and analyze program data; deploy on-site personnel to install code and document analyzer and to support data collection activities.

Subtask 3: Post-Deployment Activities: Investigate post-deployment environment; determine methods, tools and techniques needed to support the continued collection of indicator data after the selected system is operational.

3. Refinement of Product Quality Indicators

Product quality indicator definition and refinement continues along three paths. Code indicators are being examined with the intention to integrate additional metrics reflecting characteristics of more conventional languages. Documentation quality indicators are being examined relative to the OPA framework and their effective computability. Process indicator refinement continues with an emphasis being placed on identifying those applicable to E-System's development process.

3.1 Code Indicators

The Objectives/Principles/Attributes Framework provides a formal basis for defining a software quality assessment procedure. Currently, 66 automatable indicators are identified: eight are based on data type information, 12 exploit properties of statement level structures, and 46 reflect characteristic assessments of unit level constructs like packages, subprograms and so forth.

Because our initial research direction has focused on Ada-based indicators, several potential indicators applicable to other languages, and possibly to Ada, have been ignored. We are currently investigating a complementary set of indicators whose code properties are easily detectable and whose applicability extends beyond the programs written in Ada. Consider for example, "fan-out." Fan-out is defined as the number of procedures invoked within a given procedure. Stevens [STEV81] states that multiple calls from a module indicate a tendency toward too much control within a single module. Modules with an excessive number of calls have a detrimental impact on ease of change, complexity and cohesion. He suggests that function and control in such modules should be divided to reflect singular intents. Card, Church and Agresti [CARD86] state that no module should have more than seven (7) calls to other routines. Other indicators being considered include the use of recursive calls and the use of negative compound boolean expressions, both of which have a negative impact on code complexity.

A second item requiring further investigation is the tendency of some modules to exhibit indicator measures that cluster around 5 (implying no definitive basis for judgement). While this phenomenon is an exception rather than the rule, it does signify that additional refinement of selected indicators might be appropriate.

3.2 Documentation Quality Indicators

The Software Quality Assurance (SQA) project conducted by the Systems Research Center at Virginia Tech assesses software product quality from three perspectives:

- software development process,
- software specification (documentation), and
- software implementation (code).

The following discussion provides an overview of the documentation quality assessment phase. Topics outlined include:

(a) The Indicator Concept,
(b) The Integration of Documentation Quality Indicators (DQIs) into the OPA Framework, and
(c) The Developmental Status of the DQIs.

3.2.1 The Indicator Concept

To assess the quality of a product, important characteristics of the product must be measured. A well-defined assessment framework provides structure and meaning to the measurement of product characteristics, defining the relationship between product characteristics and product quality. Arthur and Nance [ARTH86] describe a framework applied to the assessment of software development methodologies which relates *objectives* of software development to software engineering *principles*, the use of which is evidenced by the presence of *attributes* in the resulting product. This objectives/principles/attributes (OPA) framework is just as applicable to the assessment of software documentation quality as it is to source code. In particular,

- the development of software documentation is motivated by certain software engineering *objectives*,
- to achieve these objectives, certain documentation *principles* must be applied in the software documentation development process, and
- the application of these principles results in the presence of definable documentation *attributes*, determined through the direct measurement of documentation properties.

Using the OPA attribute-property pair as a model [ARTH87], an *indicator* of software documentation quality is described as:

A *software quality indicator* is a variable whose value can be determined through direct analysis of product characteristics and whose evidential relationship to one or more attributes is undeniable.

For example, a property of software documentation is the support of the requirements specifications by the design (i.e., does the design fulfill the specified requirements). This property is indicative of the *accuracy* of the documentation, or, how well the documentation represents the current status of the software product. Through the application of the principle of *continuous refinement*, accuracy is promoted during documentation. Continuous refinement contributes to the documentation objective *correctness*. Thus, the following is identified as one *documentation quality indicator* (DQI):

Correctness as induced through the application of **continuous refinement** resulting in **accuracy** of the documentation, as evidenced by the presence of **requirements supported by design.**

The application of the OPA approach provides a documentation quality assessment procedure having both objectivity and consistency. Many of the OPA linkages for documentation are supported in the literature, reducing the subjectivity of the DQIs and enhancing the the objective validity of the procedure. The assessments made using this procedure are consistent because the measurements are related to the linkages the same way each time. Hence, the OPA approach applied to code and documentation supports a uniform assessment process for the entire software product.

3.2.2 Integration of DQIs into the OPA Framework

The original OPA framework is oriented towards software quality assessment in general. Due to characteristics specific to documentation, some adaptations are made to form a modified OPA framework incorporating the assessment of documentation quality. While the objectives remain basically the same as in the original framework, a refinement of principles and the introduction of additional attributes are evident. The OPA framework as it relates to documentation quality assessment is shown in Figure 1 and is described below.

3.2.2.1 Objectives of Documentation

Examining documentation use as a foundation to identify what constitutes quality documents, the following set of software engineering objectives are determined to be consequences of software product documentation:

- Maintainability,
- Correctness,
- Portability,
- Reusability,
- Testability, and
- Adaptability.

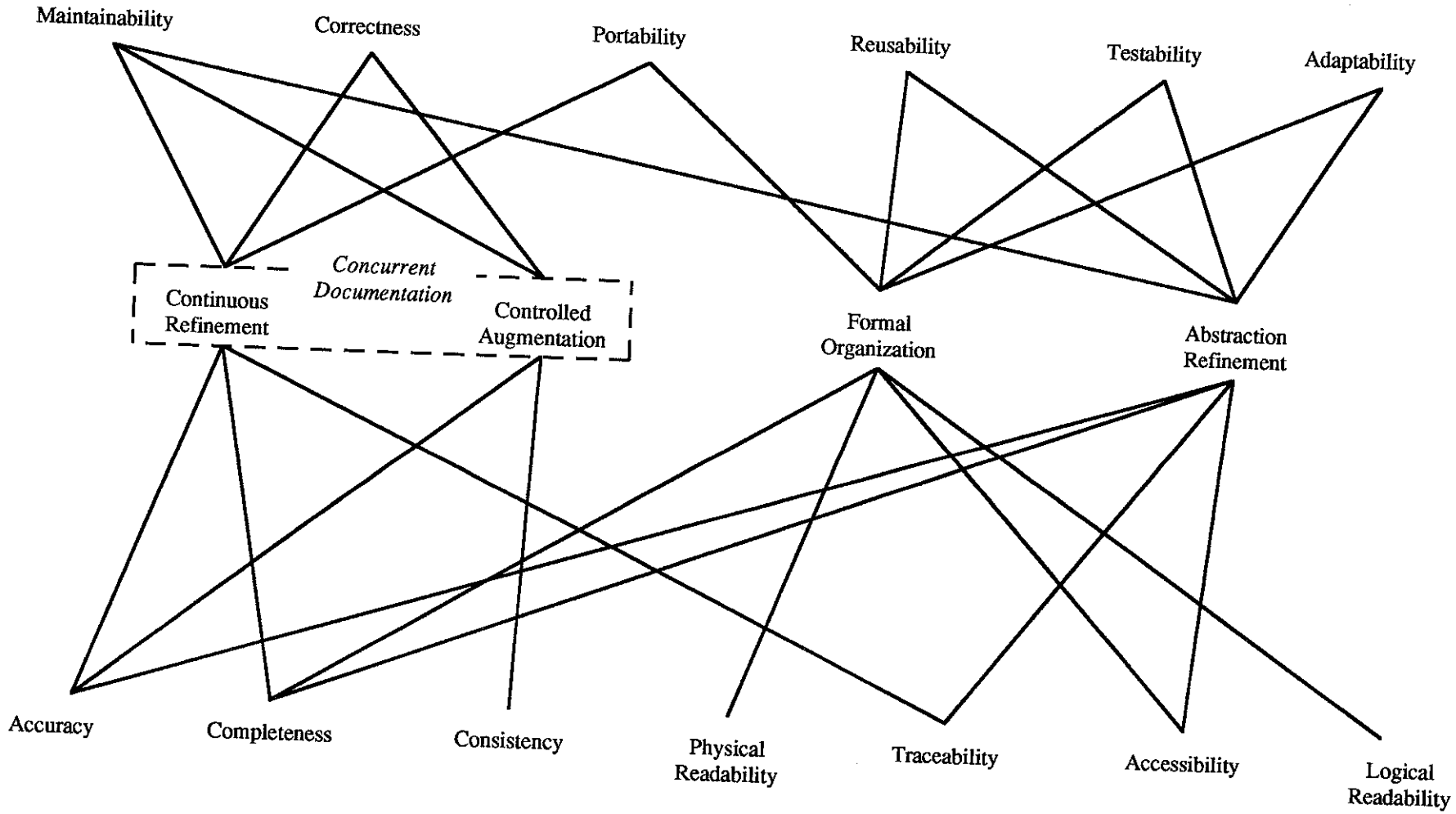

Figure 1

The OPA Framework Related to Documentation Quality Assessment

Maintainability

With respect to software products in general, maintainability is "the ease with which corrections can be made to respond to recognized inadequacies," [DAND87, p. 30]. From the perspective of documentation quality assessment, documentation achieves the objective of maintainability to the extent that it enhances the maintainability of the overall product. Note that since documentation is part of the product, its maintainability is also an issue of consideration. Thus, with respect to documentation assessment:

> *maintainability* is the degree to which the documentation facilitates corrections/augmentations to the software product.

Correctness

Correctness is an objective of any software product - if the resulting program is not correct, then what good is it? This prompts the question, "if the documentation is not correct, can the code be?" As discussed previously, the run-time version of a software product is entailed by the requirements and design specifications preceding it. Thus, correctness is an objective of documentation. With respect to documentation assessment:

> *correctness* is the specification of requirements and design in a manner which enables accurate interpretation and realization of product function.

Portability

Portability is the ease with which a software system can be transported to a different run-time environment (e.g., operating system) and retain its original functionality. When altering system characteristics to enable the transporting of a software system to a different environment, the system documentation is consulted to determine where changes are needed and the nature of the changes required. With respect to documentation assessment:

> *portability* is the facilitation of rendering the equivalent software functionality in a different run-time environment by the product documentation.

Reusability

A recently emerging goal of software engineering is the promotion of software component reuse. By not "reinventing the wheel", software development productivity is increased, and as a code component is reused many times, the reliability of the code increases as errors are discovered and corrected over time. The design of a software product plays an important role in its reusability, as components must be designed with reuse in mind if reusability is to be realized in the product. Documentation influences the reusability of software, because when a software component is reused, the documentation must accompany the code to facilitate maintenance and general understanding of the component in question. With respect to documentation assessment:

> *reusability* is the tempering of product functionality description with consideration to the granularity of system components to allow adaptations facilitating component reuse.

Testability

Before a software product is delivered, it must be tested. In response to the call to "push activities back in the software development life cycle," test cases are often developed well before a code implementation exists. Therefore, the documentation must assist in the generation of test cases as early in the software life cycle as possible. With respect to documentation assessment:

> *testability* is the specification of requirements so that test criteria can be readily identified and adherence to criteria decidable.

Boehm cites testability as one of the basic criteria for software documentation verification and validation [BOEH84].

Adaptability

From a software perspective, adaptability is the ease with which software can accommodate to change [ARTH86]. In order to make changes to a software product, the documentation should provide sufficient information in appropriate forms to facilitate adaptation(s). With respect to documentation assessment:

> *adaptability* is the provision of appropriate information in a form which facilitates adaptation of the software product.

3.2.2.2 Documentation Principles

Of the original set of software engineering principles espoused by the OPA framework, two are primarily applicable to documentation quality: (1) concurrent documentation, and (2) abstraction refinement. For the purposes of documentation assessment, however, concurrent documentation requires additional refinement. We therefore refer to it in terms of the two subprinciples, continuous refinement and controlled modification, when considering documentation quality assessment. In addition, the principle of applying formal organization to document production is applied in the assessment of software documentation.

Concurrent Documentation

Arthur and Nance identify concurrent documentation as a principle of software product development [ARTH86]. This principle is also applicable to the development of the documentation of a software product, and is therefore included in the OPA framework specific to documentation assessment. Concurrent documentation is considered the management of documentation throughout the software development life cycle, so that at any time during product development, the documentation provides a faithful representation of the current product status.

The definition of concurrent documentation given above is broad - too broad, in fact, for precise assessment of documentation quality. For this purpose the principle of concurrent documentation is decomposed into two separate principles, continuous refinement, and controlled modification. By considering concurrent documentation as two distinct principles, an assessment process embracing these principles yields more specific (and thus more useful) results.

Continuous refinement as a development principle refers to the updating of documentation to insure that changes incurred during product development are reflected in the documentation *as they occur*. Processes such as the logging of software changes, resolution of postponed specifications, and notation of traceability references are involved in the application of this principle.

A procedure crucial to a successful software quality assurance program is "that documentation is controlled and cannot be modified without proper controls" [DHIL87, p.170]. Controlled modification as a documentation principle refers to the exercise of strict control over document modification. The implementation of formal change request reviews is crucial to this principle, as such reviews maintain tight control over the state of the documentation as well as the executable product. Personnel are granted specific access to documentation, regulating who can change which documents.

Formal Organization

Standards are developed by many organizations to control the form and content of the documentation produced by in-house development teams as well as contractors enlisted by the organization. A standard is "an approved, documented, and available set of criteria used to determine the adequacy of an action (process standard) or object (product standard)," [DORF90, p.1]. Another tool used to influence the shape assume by documentation is a guideline. A guideline is "a well-defined and documented set of standards and definitions that guide an activity or task. Guidelines are usually not rules or procedures, but allow for judgement and flexibility," [DORF90, p.1]. An example of a guideline for documentation development is DoD-STD-2167A. The application of a standard or guideline to the development of documentation is referred to in the OPA framework as the principle of formal organization.

Stepwise Refinement

The specification of a software product can be viewed as a series of refinements, the initial requirements specifications defining the product in a much more abstract manner than the design, which in turn is successively refined to yield executable code. The refinement of specifications by adding detail through successive levels of specification is called stepwise refinement. Put another way, stepwise refinement is the structuring of the documentation so that specifications refine the abstractions present in the preceding level of specification.

3.2.2.3 Documentation Attributes

Obviously, the attributes of a requirements specification document employed as indicators of quality are different from those of a code component; they are different embodiments of product specification. For this reason, the attributes of documentation employed in the OPA framework differ from the original set of attributes applied to code assessment. The documentation attributes are:

- Accessibility,
- Accuracy,
- Completeness,
- Consistency,
- Logical Readability,
- Physical Readability, and
- Traceability.

Accessibility

Related to the need for references to other areas within the documentation is the need for the reader to be able to access specific information quickly in a non-sequential manner. Tools such as indices and tables of contents are examples of ways in which such access is achieved. With respect to the OPA assessment procedure:

> *accessibility* is the ability to determine the location of specific information within the documentation through the use of access mechanisms such as indices, tables of contents, and references.

Accuracy

Documentation is dynamic in nature. Updates, refinements, and other changes occur to the documentation of a software product throughout the life cycle, including the maintenance phase. At any given time, it is important that the specifications of a software product reflect the current state of that product, to avoid erroneous actions made on the premises supplied by outdated information. With respect to the OPA assessment procedure:

> *accuracy* is the faithful representation of the current status of the software product by the documentation.

Completeness

Completeness is "one of the four basic V&V (verification and validation) criteria for requirements and design specifications" [BOEH84, p. 76], and is an attribute of software requirements specifications which "often [is] the difference between success and failure of a large software product" [ROYC75, p.57]. When proceeding to the next level of refinement in the development of a software product, it is necessary to refer to the previous level(s) of specification for guidance. For example, in creating the low-level design for a software component, pertinent high-

level designs (and possibly the requirements specifications) must be consulted. If these documents are incomplete, then assumptions regarding the specifications are necessitated, resulting in an increased chance of error at the current level of specification. Unresolved references and missing document sections contribute to incompleteness, and must be eliminated expediently. With respect to the OPA assessment procedure:

> *completeness* is the inclusion of all required document sections, and the proper resolution of references throughout the documentation wherever possible.

Consistency

Boehm [BOEH84] and Royce [ROYC75] both identify consistency as a vital attribute of software documentation. Among the purposes of software documentation is the definition of objects associated with the product and the corresponding ranges of values such objects may assume. The specification of such value ranges occurs at many places throughout the documentation, sometimes in several different forms. It is vital to the correctness of the final product that objects are given consistent meanings and values throughout the documentation. Otherwise, different documents may supply conflicting descriptions of the same object, resulting in errors in the product. With respect to the OPA assessment procedure:

> *consistency* is the invariant application of definitions and concepts within the documentation.

Logical Readability

The attribute physical readability pertains to the physical format of the documentation, but more important is the content and comprehensibility of the documentation. If personnel cannot comprehend what they read in a document, due to poor structure, poorly defined terms, and cryptic use of acronyms and abbreviations, then the documentation might as well be nonexistent. Within the OPA framework, this facet of readability is referred to as logical readability. With respect to the OPA assessment procedure:

> *logical readability* is the ability to easily comprehend the information contained in the documentation through the use of standards, definitions, and terminological usage.

Physical Readability

A moderately large software project produces reams of documents, containing requirements specifications, design specifications, testing procedures and results, and various other information. While a single person rarely (if ever) reads the *entire* set of documents, many personnel must consult large quantities of the documentation to perform specific task(s). If a document is not presented in a manner facilitating reading, personnel are hindered by eye fatigue and possibly confusion. Thus, the physical layout of the documentation is important to its quality. With respect to the OPA assessment procedure:

physical readability is the presence of physical characteristics which facilitate documentation reading.

Traceability

When personnel use documentation, parts of several documents must often be employed to garner the needed information on a single subject. A topic may be elaborated upon in a different paragraph, section, or even a totally different document. For this reason, documentation must be traceable; sections must be referenced where appropriate so that the reader is aware of the presence of further information in other areas. With respect to the OPA assessment procedure:

traceability is the presence of explicit references within the documentation which clearly denote the the relationships between document sections at various levels of specification.

3.2.3 Developmental Status of DQIs

Currently, 37 property-attribute pairs are proposed as indicators of documentation quality within the OPA framework. The development of a documentation property for use in the OPA framework involves several steps:

(1) Identification and Definition of the DQI. Individual characteristics of documentation that contribute to documentation quality are identified and related to specific DQIs. The objectives, principles, and attributes affected by the property are identified and noted.

(2) Determination of Rationale for Inclusion. A justification statement is developed, outlining specific reason(s) why the DQI is important to documentation quality, and thereby warranting further consideration.

(3) Identification of Measurement Approach. The approach to measuring the relationship between a DQI and documentation quality is identified along with the data items needed to perform the measurement.

(4) Definition of Appropriate Metrics. Using data elements specified by the measurement approach, a numerically-valued expression, or *metric*, is defined, reflecting an assessment based on the rationale for inclusion.

The developmental status of the DQIs relative to the above stages is displayed in Table 1 and provides the following information on each DQI:

- *Documentation Attribute*: The attribute relative to the OPA framework in which the DQI is applicable.
- *Documentation Property*: The actual property of the documentation which is being measured and associated with a corresponding attribute.
- *Developmental Status*: This is a continuum, upon which the development of a DQI is measured. The levels of developmental completeness on this scale are:

  Undefined: a formal definition has yet to be established,

Measure Defined: the property to be measured is formally defined,
Rationale Complete: the applicability of this property as an indicator of documentation quality is established,
Measurement Approach in Progress: abstract concepts related to how the property might be measured are being investigated,
Measurement Approach Complete: a method for measuring the property is established,
Metric Development in Progress: several metric formulations for quantifying the property measure are under investigation,
Metric Complete: the formal quantification of the property is complete,

3.3 Process Indicators

On-site investigation of development process activities at E-Systems, Melpar Division in Fairfax, has enabled further understanding of the relationship between process and product. Early work involved review of major Melpar documents related to the development process, e.g., the software engineering manual, the software development plan and the software test plan for AN/SLQ-50 (XN-2) BGPHES. The software development plan remained in draft form as of 17 October 1991, but has since been reviewed and delivered to the Navy.

Attendance at meetings of the Software Development Research Group, which produces the Systems Issues Report, is providing the current status of problems or difficulties spanning all aspects of the BGPHES project.

The original set of process properties considered as potential sources for indicators include:

- design review completeness
- involvement of unacclimated personnel
- detection of requirements defects after software specification review
- addition/deletion of requirements following software specification review
- creation of software development folders
- the updating of software development folders
- quality objectives in the software development plan

To these potential indicator properties have been added:

- employment of a project-independent software quality assurance organization, and
- software quality assessment coverage.

Both are described in detail in Appendix 1.

Challenging issues still remain, among them the two most prominent are:

(1) how to assess the results of a design/code walkthrough or inspection, and
(2) how to include language specific characteristics for design assessment.

Table 1a

Developmental Status of Documentation Quality Indicators

Documentation Attribute	Documentation Property	Undefined	Measure Defined	Rationale Complete	Measurement Approach in Progress	Measurement Approach Complete	Metric Development in Progress	Metric Complete		Automation Feasibility (# automatable data items of # necessary data items)
Accuracy	Requirements Supported by Design									0 of 2
	Design Supported by Code									0 of 2
	Design Utility									0 of 2
	Code Utilization									0 of 2
Completeness	Documentation Coverage									
	TBD / TBS									1 of 1
	Missing / Incorrect References									2 of 2
	Appropriateness of References									2 of 2
Consistency	Factual Consistency									0 of 4
	Invarience of Concept									0 of 2
Physical Readability	Adequacy of Print									0 of 2
	Format Appropriateness									
	Format Consistency									
	Module Appropriateness									
Traceability	Bottom-Up Traceability									
	Top-Down Traceability									

Table 1b

Developmental Status of Documentation Quality Indicators

Documentation Attribute	Documentation Property	Undefined	Measure Defined	Rationale Complete	Measurement Approach in Progress	Measurement Approach Complete	Metric Development in Progress	Metric Complete		Automation Feasibility (# automatable data items of # necessary data items)
Accessibility	TBD / TBS							■		1 of 1
	Missing / Incorrect References							■		2 of 2
	Appropriateness of References							■		2 of 2
	Completeness of TOC							■		2 of 2
	Correctness of TOC							■		3 of 3
	Format of TOC							■		3 of 4
	Sufficiency of Index							■		1 of 2
	Locational Accuracy of Index							■		1 of 1
	Index Order							■		1 of 1
	Glossary Completeness							■		3 of 3
	Order of Glossary							■		1 of 1
Logical Readability	Text Conciseness							■		3 of 3
	Textual Clarity							■		1 of 1
	Term Usage Consistency							■		1 of 2
	Glossary Completeness							■		3 of 3
	Order of Glossary							■		1 of 1
	Adherence to Standards							■		1 of 1
	Simplicity / Modularity							■		1 of 2
	Keyword Context Consistency							■		2 of 2
	Acronym Usage							■		3 of 3
	Redundancy Appropriateness									0 of 1

The basis for assessment is the primary problem in the evaluation of design/code walkthroughs. Should the evaluation be based on:

(1) The number of action items generated?

a) High could be indicative of good process, marginal product.
b) Low could be indicative of good product, marginal process.
c) Variation with other CSUs seems more promising, but must this:

i) Be more comparative and retrospective (cannot say anything about early CSUs under review)?
ii) Be sensitive to attendees of review – some key person(s) could have strong effect?

(2) Reaction to action item report, e.g., Sandy von Kuekelgen gives the software custodian five (5) days to respond

a) Could take the number of items eliminated compared to the total.
b) Could try to employ a severity level weighting.

(3) Length of time versus number of CSUs or some size measure (LOC or design pages)?

With regard to language specific influences on design, the primary example is the use of Ada packages for defining both data types and instantiations, as described in a report from the Software Engineering Institute [HUMP87].

During the past six months, partially caused by the experience with the BGPHES project, the project team has reviewed its characterization of Visibility of Behavior (VoB) as an attribute in the OPA framework. This review and questions and decisions generated from it are being captured in an SRC Memorandum, which is not intended for general distribution. Access to SRC Memoranda can be requested by project sponsors, who also can direct the sharing of these documents with others. Please note that the intent of the SRC Memoranda series is to capture timely, pertinent data in a less than formal write-up so that important problems and issues are not lost.

4.0 Project Installation, Instrumentation, Data Collection and Refinement

This section describes major on-site activities related to the validation effort. These activities are divided into three categories and discussed separately. The first subsection outlines activities where the VTSRC project team has had substantial contact with E-Systems personnel. The second subsection addresses installation and instrumentation. Finally, we discuss the activities associated with data collection and refinement.

4.1 Interaction Between VTSRC and E-Systems Personnel

While interaction with E-Systems began early in the summer of 1991, location in the on-site offices did not occur until 21 October. A presentation to all BGPHES personnel on 22 October was intended to explain the purpose of

our study and to allay any suspicions or concerns with respect to our presence. A similar presentation was made to the ERA division of E-Systems in Fredericksburg on 10 December. The primary point of contact at E-Systems is John Finch of the Software Quality Assurance Group.

Between the time period 23 July 1991 and 15 February 1992 VTSRC has attended several meetings related to project assessment and project activities. In each case, VTSRC personnel maintained an observer status with the primary intent to gain an understanding of the project related goals and issues. Project assessment meetings include interface design reviews (23 July, 18 September, 9 October and 19 October), the System Terminal Advisory Group meeting on interface issues, and the Critical Design Review (CDR) on 19 November. (The CDR has been the major event during the period of on-site investigation and was preceded by a "dry-run" conducted on 7 November in Falls Church.) Background material from those presentations has provided data on software components relative to expected size and growth. Project status meetings attended by VTSRC personnel include an SDRG meeting, test plan development and several code walkthroughs (e.g. ICEX, ICSI, ICTA, POWI, POPSY, POFL, PORE, POGR, POUT, and POPI).

Shortly after 1 January 1992, the project team installed a Mac II system (processor, monitor, hard disk, and printer) in the STAG (Surface Terminal Advisory Group) room at the E-Systems Melpar site. This system is intended to be used for documentation analysis, report preparation, and possibly extended code analysis. This latter issue depends on the approval of transfer of basic data files from the secure environment in Area 10 of E-Systems in Fairfax.

4.2 Installation and Instrumentation

Since the arrival of the VTSRC team at E-Systems, each activity in which they have participated has been approached with the underlying goal of process instrumentation and data collection. This section outlines those activities intended to support the installation of software for data collection and process instrumentation.

4.2.1 The Ada Code Analyzer

The Ada code analyzer (Adalyze) has been successfully ported to E-System's SunADA environment. Because we can only place software on the machines at E-Systems (and not later remove the software), newer versions of Adalyze are continuously replacing older ones. Currently, the newer versions represent implementations where accounting errors have been corrected. Eventually, revisions will also include enhancements like those described in Section 3.1

In porting Adalyze to the SunADA environment at E-Systems, one significant problem has surfaced: the SunADA compiler at E-Systems has an optimization error. Several visits to E-Systems, consultations with the designers of ALEX and AYACC (at the University of California at Irvine) and significant amounts of test time were needed to

resolve a work around. If necessary, Adalyze can now be modified and recompiled within the SunADA environment at E-Systems.

4.2.2 The Documentation Analyzer

The research focus for year two has concentrated on the development of a documentation analyzer. Installation is to be completed by March, 1993. As described below, the development effort addresses the automatic collection of data to support the computation of a selected (tractable) subset of the documentation quality indicators.

4.2.2.1 Criteria for Inclusion of a DQI into Implementation Set

Inclusion of a DQI into the set of indicators automated by the prototype document analysis tool (DAT) is determined using three criteria:

- Tractability of automation,
- Verifiability through manual procedures, and
- Coverage of the OPA framework.

In considering the first of these issues, tractability, the concern is more precisely, "can a programmer familiar with the OPA framework and its applicability to documentation quality assessment implement this measure using Ada?", as the project requirements include that the analysis tool be constructed using Ada. It should be noted that, the use of several state-of-the-art tools and techniques from other disciplines (e.g., thesauri from information retrieval) could benefit future versions of the prototype, but project constraints currently prohibit the incorporation of many tools and techniques, as most such tools are implemented in the C programming language, or AI-based functional languages (e.g., Prolog).

The second consideration, manual assessability, exists because the technique employed to determine the accuracy of the measures implemented involves comparison of the automated results to the results obtained by a panel of objective software engineering experts. Another reason for this criteria is the need for manual tracing and checking of measures during analyzer production. Without the ability to manually determine the values of the implemented DQIs, there is no defensible way to examine the correctness of the prototype analyzer.

The last criteria, coverage of the OPA framework, helps to ensure that the maximal portion of the existing framework is scrutinized for validity. Figure 2 displays those linkages covered by the validation set of DQIs in bold type.

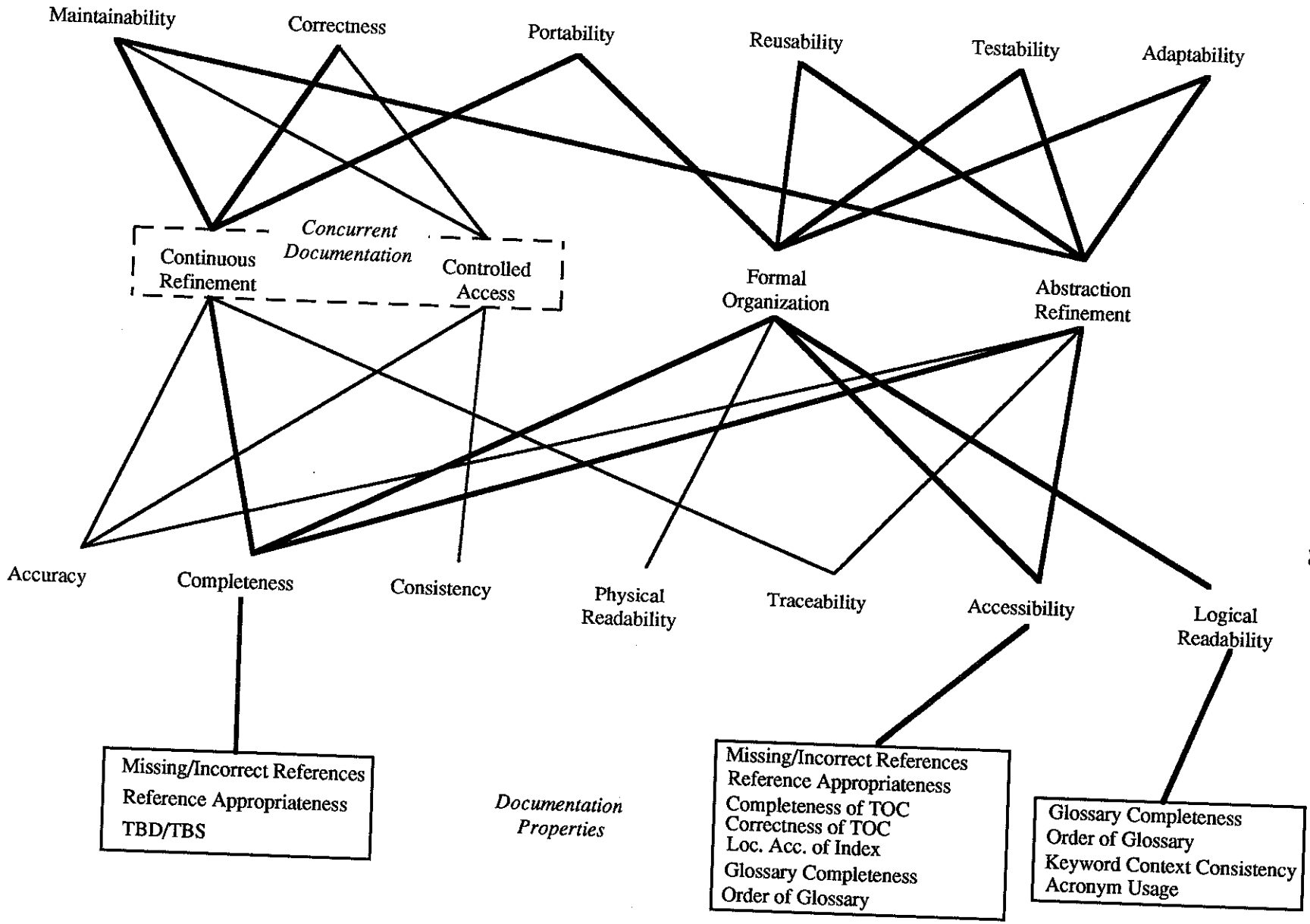

Figure 2

Documentation Quality Indicators: Linkages Affected in Validation Project

4.2.2.2 Implementation Subset

Of the full set of DQIs, ten are implemented in the prototype Document Analyzer Tool:

- Locational Accuracy of Index: the accuracy with which the index cites location of terms,
- Order of Glossary: the use of an alphabetical ordering format in the glossary,
- Glossary Completeness: the degree to which the glossary contains entries for terms which should be in the glossary,
- Missing/Incorrect References: The portion of missing references to the total number of existing references in the documentation,
- Appropriateness of References: the context similarity of sections between which references exists,
- Correctness of Table of Contents: the accuracy with which the TOC cites the section locations and titles,
- Completeness of Table of Contents: the degree to which all important topics contained in the documentation are included in the TOC,
- TBD/TBS (To be defined/To be specified): the frequency of references to later points in the development epoch for items in project development that occur in the documentation,
- Acronym Usage: the degree to which acronyms are used in a well-defined and consistent manner, and
- Keyword Context Consistency: the degree to which keywords are used consistently across all portions of a document set.

Each of these DQIs is described in a detailed manner in Appendix B of this report.

4.2.2.3 The DAT Prototype

Currently, a prototype DAT exists. As shown in Figure 3, it is comprised of five major elements:

1. document preprocessor,
2. document parser,
3. intermediate representations of document,
4. document analyzer, and
5. report generator.

The prototype analyzer is capable of successfully analyzing documents in several differing forms, with minimal formatting constraints. If a document has previously unseen features, DAT is adaptable, and can be modified to accommodate new formats, within reason. Each of these components is described in the following sections.

The Document Preprocessor

The key to the adaptability of DAT is the preprocessor. This component of DAT is actually composed of several smaller components, each performing specific tasks. Tasks performed by the preprocessor include: (1) the creation of information repositories for the index and glossary of the documentation (if applicable), (2) the creation of a keyword information repository, and (3) the preprocessing of the document body so that it can be effectively parsed.

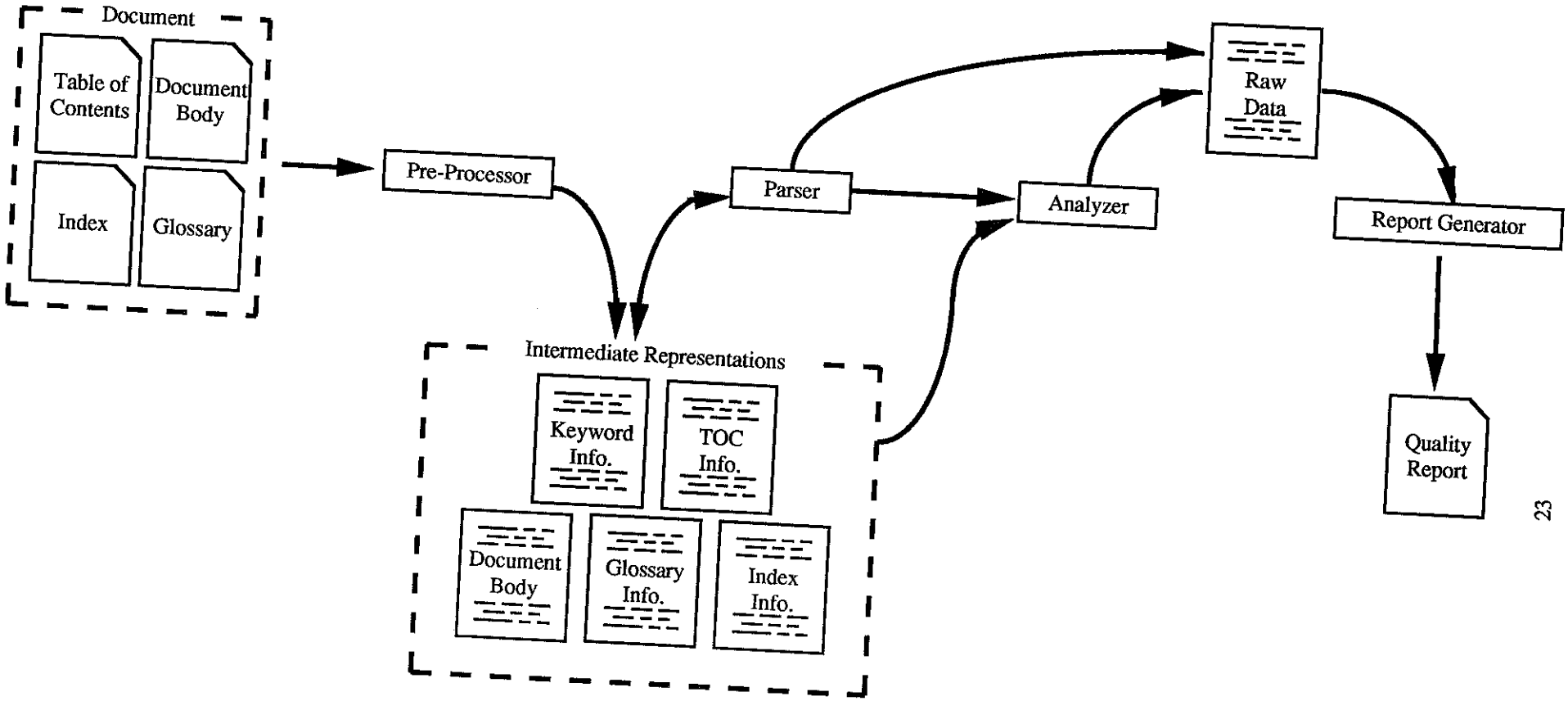

Figure 3

Documentation Analysis Tool (DAT) Structure

The most vital function relative to prototype adaptability is (3) above. By substituting certain idiosyncratic formatting symbology with unambiguous parser-recognizable symbols, the preprocessor enables DAT to be applied to a potentially wide variety of document formats.

The Document Parser

The portion of the analyzer that reads the actual text representation of the document to be analyzed, the parser, is generated by a lexical analyzer/parser generator application pair, *aflex* and *ayacc*. The reader familiar with UNIX utilities will note a similarity in the naming of these two applications to the *lex* and *yacc* utilities provided under the UNIX operating system. Indeed, aflex and ayacc are identical in function to their more familiar namesakes, with the exception that the code in which the lexical analyzer and parser are generated is Ada, as opposed to C.

The main purpose of the lexical analyzer/parser front end analyzer is to read the preprocessed representation of the document to be analyzed and pass it to the parser. As a document is parsed, important characteristics are captured and used to augment the existing information repositories created by the preprocessor. Other data are communicated to the analyzer component of DAT for "on-the-fly" computations. The table of contents (TOC) information repository is created by this component of DAT.

Because the parser is generated using an automated parser generator, changes can be made relatively easily to accommodate format variety, if necessary, by simply changing the specification files provided to the generation software. As long as changes in format and content are nominal, adjustments can be easily made.

One disadvantage of using a lexical analyzer/parser generation facility (i.e., aflex/ayacc) is efficiency. The code generated by such facilities is generally less efficient than a customized analyzer/parser. If a software producer uses DAT on a frequent basis with format-identical documents, a customized front end may be advisable; however, DAT is currently a prototype, and the need for adaptability far outweighs the need for efficiency.

Intermediate Representations

In formulating the measurement of documentation properties, certain data must be acquired before the document is parsed; an example is the set of document keywords. A term is considered a keyword if its frequency relative to other terms, or the role it serves within the document (e.g., is a term an acronym), is indicative of considerable importance to the document. The keyword repository produced by the preprocessor contains information on the frequency of occurrence of each keyword, and is augmented by the parser with information pertaining to the context in which the keyword occurs within the document. This data, as well as other information, are employed in the calculation of documentation properties.

The table of contents (TOC) of a document is examined for completeness and accuracy by DAT. To accomplish this activity, the following data must be gathered:

- the section number of each TOC entry,
- the section title of each TOC entry,
- the starting page of each TOC entry, and
- the number of entries in the TOC.

These data are gathered by the parser, and are stored in the TOC repository. As each new section is encountered by the parser in the document body, on-the-fly comparisons are made between the document body and the representation of the TOC in the TOC repository to determine the completeness and accuracy of the table of contents.

The glossary and index repositories are generated by the preprocessor and contain information on the terms present as entries within each reference aid, including frequency of occurrence of each entry. These entries are used to evaluate the completeness and format (order) of the index and glossary. The locational accuracy of the index is also assessed using location data contained within the index repository.

Document Analyzer

The analyzer component of DAT is primarily responsible for the computation of various documentation properties, using data collected and compiled in the information repositories by the preprocessor and parser. Using the metrics associated with each of the DQIs in Appendix B, the analyzer derives quantitative assessments of the document quality. These data are then stored in a format required by the report generator.

Report Generator

The report generator is to read the data stored by DAT, as well as data stored by Adalyze, an Ada code analyzer which applies OPA-based quality measures to Ada programs. The report generator compiles this information and produces a report on the overall quality of the product analyzed, suitable for managerial perusal. Integration of the DAT output with that of Adalyze is not yet complete. Efforts are currently underway to complete the integration.

4.2.2.4 DAT Inadequacies and Proposed Solution Approaches

Since DAT is a prototype, it is not the ideal automated documentation quality analyzer. There are a few areas in which DAT is inadequate for practical application.

Adaptability

Initially, the adaptability of DAT with respect to document format was to be achieved through the modification of the aflex/ayacc generated parser. Recently the impracticality of this approach has been discovered, resulting in the

decision to shift the responsibility for analyzer adaptability to the preprocessor components of the analyzer. At the time of this report, such preprocessing is not completed. Thus, DAT is currently less adaptable than previously anticipated.

In order to improve the adaptability of DAT with respect to input format, a preprocessor is being constructed that will map a large variety of document formats into a "universal" document format. A "standard" parser will then be generated which accepts documents in this universal format, hopefully eliminating the need for complicated alterations to the parser grammar.

Cross-Document Measures

Currently, DAT can only analyze one document at a time. This means that any cross-document properties, such as requirements traceability, cannot currently be derived.

The current prototype analyzer is not designed to implement cross-document measures. Future analyzer versions may use intermediate document representations to accomplish cross-document analysis.

Integration to Report Generator

As DAT is refined, changes in measurements occur due to the discovery of unanticipated documentation characteristics. This dynamic nature of the prototype development has prevented the integration of the metrics derived by DAT with the existing report generator. The metrics are derived and displayed, however, in a temporary form by using an extension of the analyzer component of DAT. While all measures generated are currently displayed, DAT should integrate its documentation assessments with the code assessments rendered by the existing OPA-oriented code analyzer, Adalyze.

Since DAT currently reports the metrics it derives in a comprehensible format, the integration of DAT and the existing report generator is not urgent. Before integration can occur, decisions relative to the theoretical integration of code and documentation measures espoused by the OPA framework must be made.

4.3 Code Walkthroughs

By far the most important activity has been the observation of the code walkthroughs for several computer software components (CSCs) developed at Melpar. The project team is provided a walkthrough package prior to the walkthrough and the review is done along the same lines as that by other attendees. The project participation attempts to determine if

- problems noted in the preliminary walkthrough package are addressed during the walkthrough,
- action items are recorded with responsibilities assigned, and
- the process is carried out according to the E-Systems software engineering manual.

Project participants will raise a question if an answer is needed to clarify matters or identify a problem not noted during the walkthrough discussion. Code walkthroughs attended during this period include:

ICEX, ICSI, ICTA 4 December 1991
POWI 5 December 1991
POGR 8 January 1992
POUT 5-6 February 1992.

When unable to attend other walkthroughs (e.g. ICLO on 20 February 1992) the project team has reviewed the walkthrough package and the completed notes of John Finch in SQA to assess the results of the walkthrough. Walkthrough results are the basis for the software development folders (SDF) examinations slated to begin later in February 1992.

4.4 Data collection and Refinement

To date, data collection efforts have focused on supporting installation efforts, i.e. verifying that the correct data is being collected, and the identification of collection points and data sampling to support indicator computation. In particular, the set of Ada support routines common to the POGR CSC have been used to verify the operational accuracy of Adalyze. Currently, one minor problem has been identified and is in the process of being corrected. As mentioned above, code walkthroughs and the generation of attendant software development folders (SDFs) have been identified as a significant source of data for several process indicators. We are currently examining the SDFs to verify the correctness and applicability of included data items.

5.0 Activities Related to Deployment

Subtask three in the current Statement of Work addresses those activities related to the collection of post-deployment data necessary to support hypothesis validation. In particular, educational activities, identification of management data access points and the (semi-)automatic collection of such data are outlined. Because BGPHES is not scheduled for deployment until 1993, these activities have been necessarily delayed; current investigative efforts are being directed toward on-site installation, instrumentation, data collection and refinement.

6.0 Summary and Future Plans

6.1 Year Two: Completed Work

The research investigation for year two has proceeded in two major directions: (1) the development of a prototype document quality analyzer and (2) the initiation of on-site activities to establish a firm understanding of the software development process at E-Systems. A third direction, though not as substantial as the two mentioned above, has been a continual refinement of the Ada code analyzer.

- Establishing an automated process for document quality assessment has necessitated a further refinement of the documentation quality indicator (DQI) concept, the selection of a subset of DQIs amenable to automated assessment, and the implementation of the document analyzer. Currently 36 of the 37 proposed DQIs are completely defined, including supporting metrics. Of those 36, 10 have been identified as automatable. Selection is based on the ability to automatically collect associated data items, their contribution to document quality assessment and their applicability to E-Systems documentation (although the latter criterion is a minor one). A document analyzer tool (DAT) has been developed that collects and computes the measures associated with the 10 selected DQIs. On-site installation of the DAT is in progress and is to execute on the VTSRC MAC II at E-Systems. Remaining to be developed, however, are the technical manual and users manual for DAT. Installation of the DAT and the completion of a users manual is expected within the first quarter of this current task year.

- VTSRC has established on-site presence at E-Systems and is actively pursuing an examination of the software development process at E-Systems. Activities relating to the critical design review (CDR), the STAG meeting on interface issues and the numerous walkthrough have provided significant insights into (1) the software development process, and (2) the identification of potential sources for data items pertinent to process indicator computation. In support of the validation effort, VTSRC has also installed a MAC II on-site to facilitate document examination and the analysis of data produced by the code analyzer. Because the first BGPHES build is not to occur until late March, no real prediction/assessment data has been collected.

- The Ada code analyzer has been installed at E-Systems and currently executes in the SunADA environment hosted by E-Systems. Problems with the existing code optimizer have been encountered, but work-arounds have been determined and implemented. Routines common to the POWI CSC are being used to substantiate data collection accuracy of the analyzer.

6.2 Year Three: Current Research Directions

Research directions for the current year include continued refinement of code and documentation indicators, the installation and testing of the document analyzer and the following new initiatives:

- *Refinement of Product Quality Indicators*: While refinement of the code and documentation indicators will continue, a major emphasis is being placed on the identification and formulation of process indicators that support the prediction of software quality. Examination of development process activities and project data is expected to play and integral role in the identification, formulation and refinement of process indicators. We anticipate that this activity will continue through the third quarter of this task year.

- *Installation and Integration of the Document Analyzer*: Currently, the VTSRC project team has plans to install the document analyzer on-site within the first three months of this task year. It is to execute on the MAC II currently residing at E-Systems. Following installation, the DAT is to be integrated with the current report generation system.

- *Instrumentation, Data Collection and Refinement*: Clearly, the collection of data to support indicator refinement and hypothesis validation will be a major focus of this task year's effort. Integral to this activity is the identification and collection of data items pertinent to indicator computation. Where possible automated approaches will be pursued to collect data. Where automation in not possible, e.g. activities associated with design reviews, the VTSRC staff will provide the manual effort necessary to effect data collection. Although the identification and instrumentation of critical data collection points are to be completed by the third quarter of this year, data collection will continue into the fourth year.

- *Deployment Process Design*: Anticipating deployment in the second quarter of the fourth year, VTSRC personnel will investigate (a) deployment transition requirements that support the collection of data items needed for prediction and assessment, (b) deployment site characteristics and processes that can be used to collect validation data, and (c) educational activities necessary to effect a smooth transition of the validation project to the deployment and maintenance site. We anticipate that this activity will begin during the third quarter of the current task year.

6.3 Year Four: Future Plans

Primary efforts during year four will focus on post-deployment activities. In particular, process design supporting deployment activities will continue to evolve. Deployment training and installation of appropriate data collection

methods and techniques will provide the necessary basis for validation data extraction. Finally, statistical analysis supporting (or refuting) the predictive/assessment capabilities of the evaluation procedure will be performed.

References

[ARTH86] Arthur, J., Nance, R. and Henry, S., "A Procedural Approach to Evaluating Software Development Methodologies: The Foundation," Technical Report SRC-86-008, Systems Research Center, Virginia Tech, 1986.

[ARTH87] Arthur, J. and Nance, R., "Developing an Automated Procedure for Evaluating Software Development Methodologies and Associated Products," Technical Report SRC-87-007, Systems Research Center, Virginia Tech, 1987.

[ARTH90] Arthur, J. and Nance, R., "A Framework for Assessing the Adequacy and Effectiveness of Software Development Methodologies," *Proceedings of the Fifteenth Annual Software Workshop*, NASA Goddard Space Flight Center, Greenbelt MD, 1990,Session 2: Process Improvement.

[BOEH84] Boehm, B., "Verifying and Validating Software Requirements and Design Specifications," *IEEE Software*, January 1984, pp. 75-88.

[CARD86] Card, D., Church, V., and Agresti, W., "An Empirical Study of Software Design Practices," *IEEE Transactions on Software Engineering*, Vol. 12, No. 2, pp. 115-139.

[DAND87] Dandekar, A., "Procedural Approach to the Evaluation of Software Development Methodologies," Masters Thesis, Virginia Tech, 1987.

[DHIL87] Dhillon, B., *Reliability in Computer System Design*, Ablex Pub. Co., 1987, pp. 167-190.

[DORF90] Dorfman, M. and R. Thayer, *Standards, Guidelines, and Examples on System and Software Requirements Engineering*, IEEE Computer Society Press, 1990.

[HUMP87] Humphrey, w., et al, "A Method for Assessing the Software Engineering Capability of Contractors," Technical Report CMU/SEI-87-TR-23, Software Engineering Institute, Carnegie Mellon Institute, Pittsburgh, PA.

[ROYC75] Royce, W. "Software Requirements Analysis: Sizing and Costing," from *Practical Strategies for Developing Large Software System*, E. Horowitz, ed., Addison-Wesley Pub. Co., 1975, pp. 57-70.

[STEV81] Stevens, W., *Using Structured Design*, John Wiley, New York, 1981.

Appendix A

Additional Proposed Process Indicators

Process Property: Employment of a project-independent software quality assurance (SQA) organization.

Impact of the Property: A SQA organization, not in the reporting line to the project manager but responsible to a higher level in the organization, can exercise the independent authority in monitoring the development process to assure conformance with stated quality procedures and derive and organize feedback for process improvement.

OPA Entity Affected: Visibility of Behavior (Attribute)

Justification: An SQA organization prescribes the process characteristics that assure a high quality product and defines the points where conformance with these characteristics is assured. The existence of an SQA organization, beyond the project organization, induces a more consistent process, one that conforms to the perceived requisites for product quality to the extent that SQA influences execution and continual revision of the process.

Measurement Approach: From a base predicated on the existence of a SQA organization, independent of the project management, add values reflecting the degrees of involvement in and influence on the process.

Process Property: Software quality assessment coverage of the development process activities.

Impact of the Property: The SQA function should be consistently and thoroughly applied throughout the development process to assure that attention to quality exists through all phases and that process improvement can be realized as the major responsibility of the SQA organization.

OPA Entity Affected: Visibility of Behavior (Attribute)

Justification: A process that mandates SQA audit from review of requirements until completion of system test for *all* software components is more likely to produce a high quality product than a process that permits deviations, waivers, and exclusions.

Measurement Approach: Check each code component to see if SQA has been involved at each prescribed point from the System Requirements Review until the System Test (or System and Integration Test if treated together). The criterion for SQA involvement is that written signoff by an SQA representative be mandatory for each review/inspection. The computation is carried out for each component (i : = 1, 2, ..., n), and the factor (10/n) scales the result in the interval [0, 10].

Appendix B

Implemented Documentation Quality Indicators

Property: Locational Accuracy of Index

What is measured: The accuracy with which the index cites locations of terms.

Rationale: A good index enables the user to selectively read sections or segments of a document that address specific topics of interest. If an index misleads a user with respect to the locations of a term within the text of a document, the user suffers losses of both time and confidence in the documentation.

Measure: This property is an assessment of the accuracy of the index entries (i.e., if the index says **missile warning system** appears on page 22, does it?). Deductions are made for erroneous entries, perhaps accounting for the degree of error committed in each case.

An additional facet of correctness is the degree to which a term is represented at the location(s) specified by the index (i.e., is the term actually discussed, or does it just make a cameo appearance?). This might be assessed in the future by determining whether the object term is used as the subject of a sentence on the page noted within the index.

Property: Order of Glossary

What is measured: The use of an alphabetical ordering format in the glossary.

Rationale: A good glossary enables the user to find definitions to terms specific to the project quickly and easily. This enables a reader to read selectively without having to worry about missing the definition of a term/acronym at its first usage. The order of the glossary should be alphabetical, as this is the universally accepted format, and due to the fact that an alphabetical format greatly enhances the search process by paralleling the intuition of users.

Measure: The glossary is examined to determine if alphabetical ordering is employed. Deductions are incurred for each unordered glossary entry relative to the total number of entries in the glossary.

Property: Glossary Completeness

What is measured: The degree to which the glossary contains entries for terms which should be in the glossary.

Rationale: A useful glossary enables the user to find definitions of terms specific to the project quickly and easily. This enables selective reading without having to worry about missing the definition of a term/acronym at its first usage. A useful glossary contains all terms for which target users of a document may need explicit definition (project specific terms, acronyms, abbreviations).

Measure: Completeness of the glossary is assessed by checking whether all project-specific terms and acronyms are defined in the glossary. Deductions are made for each omitted entry, augmented by a term frequency weighting scheme.

Property: Missing/Incorrect References

What is measured: The proportion of missing and incorrect references to the total number of existing references in the documentation. A "missing" reference is defined as a reference to a section that doesn't exist. An "incorrect" reference is a reference to a section that is not germane to the topic at hand.

Rationale: Documentation which does not explicitly link related sections together for the user is not only less useful, but less complete than documentation which is linked textually as well as organizationally. Erroneous references mislead the user, and waste time and effort.

Measure: For each inter-sectional reference, the context of the reference is compared to the content of the section referenced, yielding a quantified similarity measure. Those references whose similarity to the section referenced fall below a predefined threshold are deemed incorrect. If a referenced section is not present in the document, then the reference is deemed missing. A ratio of missing and incorrect references is calculated.

Property: Appropriateness of References

What is Measured: The context similarity of sections between which a reference exists.

Rationale: Knowing the number of inter-sectional references made within a document, and the percentage of these references that are valid relative to the context of the sections involved is useful information; it is more useful, however, to have knowledge of the *degree* to which the context of those sections having a reference between them are similar.

Measure: The context similarity of two sections between which a reference exists is calculated. If this similarity measure is above a pre-determined threshold used to determine the "appropriateness" of a reference, then the similarity measure is added to the existing sum of "appropriate" reference similarity measures. An average is the taken over all such "appropriate" references.

Property: Correctness of Table of Contents (TOC)

What is measured: The accuracy with which the TOC cites the section locations and titles.

Rationale: If a table of contents is to be of assistance to the user of a document, it must correctly guide the user to the location of the desired section. Inaccuracies of this nature cause wasted time and effort, and thereby diminish the effectiveness of the TOC as a reference tool.

Measure: The following criteria for TOC correctness are evaluated:

- Does the section begin at the location specified by the TOC?
- Does the section heading concur with the title cited in the TOC?

These criteria are assessed by comparing the entries of the TOC with the document body for accuracy.

Property: Completeness of Table of Contents

What is measured: The degree to which all important topics contained in the documentation are included in the table of contents.

Rationale: If a table of contents is to be of assistance to the user of a document, it must be complete. This means that the table of contents should include all topics that are of sufficient importance that a section of the documentation has been dedicated to them. If important topics are not included, then the table of contents loses effectiveness as a reference aid.

Measure: Completeness is measured by evaluating the extent to which all sections are contained within the table of contents. The reasoning behind this is that if a topic is important enough to warrant a separate section within the documentation, then an entry in the table of contents is warranted as well.

The sections can be counted by parsing the document set and counting each section heading, as these headings should have a distinct format.

Property: TBD/TBS (To Be Defined/To Be Specified)

What is measured: The frequency of references to a later point in the development epoch for items in project development (e.g. "to be defined ...", "to be specified ...") that occur within the documentation.

Rationale: The existence of references of the above form inhibits a user's garnering of needed information from the documentation. These references leave gaps in the documents where they exist.

Measure: In order to assess this quantifier, phrases of the "TBD/TBS" form must be counted. These phrases can be counted by searching for the literal phrases denoting such references (e.g. "to be defined ...", "to be specified ...", etc.)

TBD/TBS statements are appropriate at many times during the development of software documentation, but those statements that remain unresolved over time detract from the accuracy, completeness, and usability of the documentation. The *persistence* of TBD/TBS phrases from one development epoch to the next is an indicator of the inability to resolve the (underlying issues to eliminate) (issues underlying the elimination of) these phrases.

Property: Acronym Usage

What is measured: The degree to which acronyms are used in a well-defined and consistent manner.

Rationale: Acronym usage is helpful to the user of a document only if the acronyms are defined in a readily accessible manner and the acronyms are used consistently.

Measure: Each acronym used in a document is checked for the following:

- definition at initial usage OR
- definition in the glossary or acronym list accompanying the documentation, with reference to such a list on initial use
- each usage of the acronym is within a similar context (referred to below as a "proper use")

The meaning of an acronym in this case is simply its verbose form. For example, SRC "means" Systems Research Center. Once it has been used in this context, SRC should not appear anywhere in the same document(s) meaning State Regulatory Commission. This would be an instance of inconsistent usage of an acronym.

Property: Keyword Context Consistency

What is measured: The degree to which keywords are used consistently across *all* portions of a document set. A keyword is any symbol string deemed to the project (acronyms and abbreviations excluded).

Rationale: The use of keywords in a documentation set should be consistent with respect to meaning throughout the documentation. Usability is greatly diminished if this is not the case, because the user may easily become confused by conflicting use of a keyword within the documentation.

Measure: A list of keywords is constructed. Associated with each keyword is a collection of the contexts in which the keyword is used throughout the documentation text. Each context is compared with the other contexts in which the keyword is used for consistency of meaning.

These comparisons are effected through the use of a table, as shown below. This table is analyzed for consistent usage of keywords, by comparing each instance of a keyword with every other instance. Consistency of keyword context is assessed relative to the most frequently occurring context for a keyword.

keyword_1	context_1	context_2	...	context_i
keyword_2	context_1	context_2	...	context_j
keyword_3	context_1	context_2	...	context_k
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
keyword_n	context_1	context_2	...	context_l

Keyword Consistency Table

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION: Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): Systems Research Center SRC-92-003	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: Systems Research Center; 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION: Naval Surface Warfare Center
6c. ADDRESS (City, State, and ZIP Code): 320 Femoyer Hall, Virginia Tech, Blacksburg, Virginia 24061-0251	7b. ADDRESS (City, State, and ZIP Code): Dahlgren, Virginia 22448-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: JLC/CSM; 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code): Dr. Raghu Singh, Chair, Space and Naval Warfare Systems Command, Mail Code 31F1 Washington, DC 20363-5100	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification): Software Quality Measurement: Validation of a Foundational Approach (Final Report, Year Two)

12. PERSONAL AUTHOR(S): James D. Arthur, Richard E. Nance, and Edward V. Dorsey

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final - Year Two	FROM 1/1/91 TO 12/31/91	23 April 1992	45

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number): Simulation quality assessment and prediction; software engineering objectives, principles and attributes; software quality indicators, process indicators, documentation quality indicators, automated document analysis

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

This report discusses the second year findings of a proposed four year investigation effort that focuses on the assessment and prediction of software quality. The research exploits fundamental linkages among software engineering Objectives, Principles, and Attributes (the OPA framework). Process and documentation quality indicators (DQI) are presented relative to the OPA framework with an elaboration on their individual role in assessing and predicting software quality. The development of a document quality analyzer is discussed, with a particular emphasis being placed on the selected subset of DQI measures that it provides.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

DD Form 1473, JUN 86 *Previous editions are obsolete.* SECURITY CLASSIFICATION OF THIS PAGE