Rome Laboratory Research in Software Measurement

A DACS Custom Bibliographic Search

22 November 1993

Prepared for:

Rome Laboratory
RL/C3CB
525 Brooks Road
Griffiss AFB, NY 13441-4505

Prepared by:

Kaman Sciences Corporation
P.O. Box 120
Utica, NY 13503-0120

The Data & Analysis Center for Software (DACS) is a Department of Defense (DoD) Information Analysis Center (IAC), administratively managed by the Defense Technical Information Center (DTIC) under the DoD IAC Program. The DACS is technically managed by Rome Laboratory (RL). Kaman Sciences Corporation manages and operates the DACS, serving as a source for current, readily available data and information concerning software engineering and software technology.
Software engineering annotated bibliographies are available from the DACS, of which eight volumes comprise our general bibliography. In addition, annotated bibliographies on specific topics are made available from time to time. This report contains citations and abstracts for 102 documents, the result of a search of the DACS Software Engineering Bibliographic Database (SEBD) for documents produced under sponsorship of Rome Laboratory (formerly Rome Air Development Center) in the software measurement area. The SEBD is composed of texts, technical reports, theses, journal articles, proceedings, and other documents relating to all aspects of software engineering. A citation, index terms, and abstracts are provided for each document in the bibliography. Keyword-in-context (KWIC), keyword, and author indices are provided.
**GENERAL INSTRUCTIONS FOR COMPLETING SF 298**

The report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to stay within the lines to meet optical scanning requirements.

| Block 1. Agency Use Only (leave blank). |
| Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year. |
| Block 3. Type of report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87-30 Jun 88). |
| Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses. |
| Block 5. Funding Numbers. To include contract and grant numbers: may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels: |
| C - Contract | PR - Project |
| G - Grant | TA - Task |
| PE - Program | WU - Work Unit |
| Element | Accession No. |
| Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s). |
| Block 7. Performing Organization Name(s) and Address(es). Self-explanatory. |
| Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report. |
| Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory. |
| Block 10. Sponsoring/Monitoring Agency Report Number. (if known) |
| Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with ...; Trans. of ...; To be published in ... When a report is revised, include a statement whether the new report supersedes or supplements the older report. |
| Block 12a. Distribution/Availability Statement. Denotes public availability or limitation. Cite any availability to the public. Enter additional limitations or special markings in all capital letters (e.g. NOFORN, REL. ITAR). |
| DOD - See DoDD 5230.24, ‘Distribution Statements on Technical Documents.’ |
| DOE - See authorities. |
| NASA - See Handbook NHB 2200.2 |
| NTIS - Leave blank |
| Block 12b. Distribution Code. |
| DOD - Leave blank |
| DOE - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports. |
| NASA - Leave blank |
| NTIS - Leave blank |
| Block 13. Abstract. Include a brief (Maximum 200 words) factual summary of the most significant information contained in the report. |
| Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report. |
| Block 15. Number of Pages. Enter the total number of pages. |
| Block 16. Price Code. Enter appropriate price code (NTIS only). |
| Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited. |
PREFACE

Although software engineering encompasses several rapidly growing technologies, it is not yet fully recognized as a true engineering discipline. One factor hindering the development of software engineering as a discipline is lack of access to research that is useful for developing and refining the principles and methods of these software technologies. The Data & Analysis Center for Software (DACS) was established to serve both as a center for the analysis of software engineering data and as a focal point for Scientific and Technical Information (STI) on software technology.

The Data & Analysis Center for Software (DACS) is a Department of Defense (DoD) Information Analysis Center (IAC). The DACS is administratively managed by the Defense Technical Information Center (DTIC) as a component of the DTIC IAC Program. The DACS is technically managed by Rome Laboratory (RL) and operated by Kaman Sciences Corporation.

The DACS supports the development, testing, validation, and transitioning of software engineering technology to the defense community, industry, and academia. DACS' subject areas encompass the entire software life cycle and include software standards, and acquisition management. Also included are programming environments and language techniques such as Ada and Object Oriented Design (OOD), software failures, test methodologies, software quality metrics and measurements, software reliability, software safety, cost estimation and modeling, standards and guides for software development and maintenance, and software technology for research, development, and training.

Additional information on the DTIC IAC Program or the products and services offered by the DACS can be obtained from the following contacts:

Mr. Dennis J. Wesolowski, DACS Program Manager
Data & Analysis Center for Software
PO Box 120
Utica, NY 13503-0120
(315) 734-3644; FAX (315) 734-3699
Internet: dennisw@utica.kaman.com

Mr. Paul M. Engelhart, RL Project Manager
Rome Laboratory
RL/C3CB
525 Brooks Road
Griffiss AFB, NY 13441-4505
(315) 330-4477; DSN 587-4477
FAX (315) 330-7989; DSN 587-7989
Internet: paulen@lonex.rl.af.mil

Dr. Forrest Frank, DoD IAC Program Manager
Defense Technical Information Center
DTIC-AI, Building 5
Cameron Station
Alexandria, VA 22304-6145
(703) 274-7633; FAX (703) 274-0980
Internet: ffrank@dtic.dla.mil
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2. BIBLIOGRAPHIC CITATIONS</td>
<td>2</td>
</tr>
<tr>
<td>3. TITLE KEYWORDS IN CONTEXT</td>
<td>31</td>
</tr>
<tr>
<td>4. KEYWORD INDEX</td>
<td>48</td>
</tr>
<tr>
<td>5. AUTHOR INDEX</td>
<td>55</td>
</tr>
</tbody>
</table>
1. INTRODUCTION

The DACS Software Engineering Bibliographic (SEB) database was established to provide a readily accessible source of comprehensive information on the state-of-the-art in software engineering and to channel that information to those able to use it in developing, maintaining, and managing software. The bibliographic collection is composed of texts, technical reports, theses, journal articles, proceedings, and other documents relating to software engineering, reliability, costs, and quality factors, maintainability and other appropriate topics. These documents are obtained from the Defense Technical Information Center (DTIC), the National Technical Information Service (NTIS), professional societies (e.g. IEEE and ACM), and publishers. Each reference in the SEB includes a bibliographic citation, availability information (source and order number), index terms (keywords), and an abstract. Historically, the DACS has offered two methods of accessing the information in the SEB: through custom bibliographic searches, such as this one, and through published DACS Software Engineering Bibliographies.

This bibliography contains citations and abstracts for 102 documents, the result of a search of the SEB for documents produced under sponsorship of Rome Laboratory (and Rome Air Development Center) in the software measurement area. Abstracts provided with the original document were edited as necessary to reflect the document’s content; when an author-written version was unavailable, the abstract was written by a DACS staff member.

The bibliography is composed of five sections. Section 1 contains this introduction. Section 2 is the listing of the document citations, index terms, and abstracts. Each citation references a document accession number (DAN). The DAN is printed at the beginning of each citation. Section 2 is ordered by the DAN, while references in Section 3, 4 and 5 are identified by their DAN. If the complete document is desired, ordering information is found in the document citation.

Section 3 contains the keyword-in-context (KWIC) index, or rotated title display index. Documents are listed by each significant word in the title; smaller or very common words (e.g. the, a, an, software) are screened out from the index terms process by a stop list. The title is then rotated so that a complete set of entries for the document SOFTWARE QUALITY METRICS ENHANCEMENTS looks like this:

QUALITY METRICS ENHANCEMENTS *** SOFTWARE METRICS ENHANCEMENTS *** SOFTWARE QUALITY ENHANCEMENTS *** SOFTWARE QUALITY METRICS

The lines or entries are then organized in alphabetical order with the rest of the document entries so that an entry

QUALITY. PRELIMINARY HANDBOOK ON SOFTWARE QUALITY FOR AN ACQUISITION MANAGER *** FACTORS IN SOFTWARE

appears after the first entry in the example group above. Thus, all documents about quality are brought together by their complete titles. A document has as many entries as there are significant words in its title. The DAN also appears with each title and can be used to locate the citation in Section 2.

Section 4 is an index of the document keywords. Each keyword is followed by a list of DANs for the associated citations. Section 5 contains the author index. Each author is listed alphabetically with the document title and its accession number. After locating a particular author and a title of interest, refer to Section 2 for complete information on the document.

---

This document was produced as the result of a search run on 19 November 1993 with the following search strategy:

1. contract/grant org. in ( "3756", "6052", "6055", "7251") and keyword matches "MEASUREMENT"
2. contract/grant org. in ( "3855", "6055") and keyword matches "METRIC"
3. report# matches "RADC" and keyword matches "MEASUREMENT"
4. report# matches "RADC" and keyword matches "METRIC"
5. report# matches "RL*" and keyword matches "MEASUREMENT"
6. report# matches "RL*" and keyword matches "METRIC"
2. BIBLIOGRAPHIC CITATIONS


This report presents several measures of computer program complexity, in the sense of comprehensibility or intellectual manageability. The measures consider the program as an abstract process, and so are independent of programming language or implementation details.


An hierarchical definition of factors affecting software quality was compiled after an extensive literature search. The definition covers the complete range of software development and is broken down into non-oriented and software-oriented characteristics. For the lowest level of the software-oriented factors, metrics were developed that would be independent of the programming language. These measurable criteria were collected and validated using actual Air Force data bases. A handbook was generated that will be useful to Air Force acquisition managers for specifying the overall quality of a software system.


An hierarchical definition of factors affecting software quality was compiled after an extensive literature search. The definition covers the complete range of software development and is broken down into non-oriented and software-oriented characteristics. For the lowest level of the software-oriented factors, metrics were developed that would be independent of the programming language. These measurable criteria were collected and validated using actual Air Force data bases. A handbook was generated that will be useful to Air Force acquisition managers for specifying the overall quality of a software system.


An hierarchical definition of factors affecting software quality was compiled after an extensive literature search. The definition covers the complete range of software development and is broken down into non-oriented and software-oriented characteristics. For the lowest level of the software-oriented factors,
Metrics were developed that would be independent of the programming language. These measurable criteria were collected and validated using actual Air Force data bases. A handbook was generated that will be useful to Air Force acquisition managers for specifying the overall quality of a software system.
22,000 lines of source code are provided. The statistical results of this effort may directly support the activities of FORTRAN language study and standardization efforts which address language and compiler design, optimization, and subsetting.


Key words: Complexity Measurement, Maintainability, Modification Procedures, Performance Evaluation, Program Maintenance

This report analyzes the possible ripple effect of software modifications during the maintenance phase on the performance of the system, and leads to the development of a maintenance technique for predicting which performance requirements in the system may be affected by a proposed modification. A formal description of the mechanisms for propagation of performance changes is included, as well as a discussion of critical sections of a program. Also covered are dependency relationships between virtual performance attributes and performance attributes.


Key words: Behavior Modeling, Fault Correction, Maintainability, Maintenance Tools and Techniques, Modification Procedures, Modification, Performance, Program Testing, Regression Testing, Reliability Measurement

This report documents research performed by Northwestern University in the area of developing effective techniques for large-scale software maintenance, including those for the design, implementation, validation, and evaluation of reliable and maintainable software systems with a high degree of automation. During the contract period, research in the areas of ripple effect analysis, testing during software maintenance, specification for program modifications, quality factors for software maintainability, and dynamic monitoring of program behavior was conducted.


Key words: Development Methodologies, Chief Programmer Team, Complexity Measurement, Productivity Models, Data Analysis, Error Analysis, Fault Data Analysis, Management Tools and Techniques, Quality Attributes, Software Experience Data, Software Science, Structured Programming, Top-Down Development, Cost Models

This report is the first of two volumes describing the performance of the Launch Support Database (LSDB) project implemented under Advanced Systematic Techniques for Reliable Software (ASTROS). This volume presents a condensed report of the results of this study which has been prepared for a non-scientific and/or managerial audience. Readers wanting a detailed account of the theories and analyses underlying these results are referred to Volume II (DAN 2041). Where material has been substantially condensed in this volume, a reference will be provided to the corresponding section in Volume II where further elaboration of the material is available.


Key words: Development Methodologies, Chief Programmer Team, Complexity Measurement, Productivity Models, Data Analysis, Error Analysis, Fault Data Analysis, Management Tools and Techniques, Quality Attributes, Software Experience Data, Software Science, Structured Programming, Top-Down Development, Cost Models

This report is the second of two volumes describing the performance of the Launch Support Data Base (LSDB) project implemented under Advanced Systematic Techniques for Reliable Software (ASTROS). This volume presents a detailed report of the results of this study. Readers wanting a condensed account of the theories and analyses underlying these results are referred to Volume I (DAN 2040).


The Communications Processor Operating System (CPOS) effort is one program of a multiple program effort whose purpose is the development of a Unified Digital Switch (USD) for strategic communications. This switch will have the capability to perform circuit, packet and store-and-forward message switching in an integrated communication complex. The Communications Processor System (CPS) will control the switching node and will be supported by an operating system called the Communications Processor Operating System. In particular, multilevel communications security conforming to DoD requirements represents a difficult problem for the CPOS and requires solutions which are on the fringe of the current technology. In addition, the need for high reliability is a cause of concern because of the inexact science of software technology. These concerns have resulted in heavy emphasis being given to Tasks 2, 3, 6 and 7. A specification has been prepared as a stand-alone document suitable for the next stage of contractual or in-house development suitable for the next stage of the CPS.


Key words: Correctness Proofs, Efficiency, Flexibility, Integrity, Interoperability, Maintainability, Portability, Quality Metrics, Reliability, Reusability, Testability, Usability

Software metrics (or measurements) which predict software quality have been refined and enhanced. Metrics were classified as anomaly-detecting metrics which identify deficiencies in documentation or source code, predictive metrics which measure the logic of the design and implementation, and acceptance metrics which are applied to the end product to assess compliance with requirements.


Key words: Correctness Proofs, Efficiency, Flexibility, Integrity, Interoperability, Maintainability, Portability, Quality Metrics, Reliability, Reusability, Testability, Usability
The purpose of this manual is to present a complete set of procedures and guidelines for introducing and utilizing current software quality measurement techniques in a quality assurance program associated with large scale software system developments. These procedures and guidelines will identify: (1) How to identify and specify software quality requirements (Setting Quality Goals), (2) How and when to apply software metrics (Applying Metrics), and (3) How to interpret the information obtained from the application of the metrics (Making a Quality Assessment).


This report summarizes research performed in the area of complexity measures, test models and techniques, methods for developing programs with low error content, software reliability models and software management models. Significant results are highlighted with their interrelation and potential.


Key words: Complexity Measurement, Control Structures, Modules, Programming Aids, Software Engineering Tools and Techniques.

A new measure of software complexity is introduced, one which describes a flowchart by a polynomial. This measure takes both the elements of a flowchart and its structure into account. Rules are given for obtaining the polynomials for various types of flowcharts. The polynomial complexity measure can be used to compare alternate software designs and to divide a computer program into modules to obtain a minimal overall complexity. A comparison is made of this measure with several known complexity measures.


Key words: Failures, Geometric-Poisson Model, Jelinski-Moranda Model, Musa’s Model, Quality Assurance, Reliability Models, Reliability, Schick-Wolverton Model, Shooman Model, Testing.

This report presents a summary and evaluation of most of the available models for software reliability assessment. A discussion of software quality, software testing and software reliability is included. Brief descriptions of times between failures and failure count software reliability models are provided along with fault seeding and input domain models. Assumptions, limitations and applicability of these models are discussed. Included is a step by step procedure for developing a software reliability model.


Key words: Availability, Data Analysis, Error Categories, Failure Data, Failure Rates, Fault Detection, Geometric-Poisson Model, Goel-Okumoto Model, Jelinski-Moranda Model, Musa’s Model, Performance Evaluation, Reliability Models, Reliability Prediction, Schick-Wolverton Model, Shooman Model, System Reliability.

This report presents the results of a software reliability modelling and estimation research. Two new models of very general applicability are introduced and the necessary mathematical and practical details
are developed in this report. A new methodology for determining when to stop testing and start using software is described and developed. Finally, a new model for analyzing the operational performance of a combined hardware-software system is reported, even though it was not a part of the original research plan. (author)


Key words: Algorithm Analysis, Computation Structures, Efficiency, Execution Time, Flexibility, Integrity, Interoperability, Maintainability, Portability, Quality Metrics, Reusability, Testability, Usability

This is the first of two volumes constituting the final technical report for a study entitled "Algorithmic Complexity." This volume comprises a general introduction to the entire series and a research plan. First, the report begins with a description of the goals of the overall contract effort. This is followed by a discussion of previous RADC work on software quality metrics, emphasizing measures concerned with the time and storage efficiency of programs. Next, an overview of the field of algorithm analysis and computational complexity is given. A final section contains an introduction to the particular research investigations pursued in the other portions of the study. Next, the author presents a research plan for advancing the state-of-the-art in the area of algorithm performance. While recommending continuing work of an applied nature in algorithm analysis and computational complexity, new and unresolved issues concerning the relationships between programming languages, computer architecture, and the performance of algorithms on computer systems are also identified. Problem application areas considered include computational algebra; sorting, searching, and database systems; pattern matching in strings; combinatorial optimization problems; and computational geometry. An appendix contains an outline of the major topics and issues addressed in the area of algorithm analysis and computational complexity, together with an annotated select bibliography.


Key words: Access-Control Mechanisms, Algorithm Analysis, Database Management Systems, Efficiency, Quality Metrics

The objective of this study was to conduct applied research directed toward understanding the relationship between the complexity or efficiency of algorithms and the overall quality of computer software. The final report is presented in a two volume series consisting of a total of eight parts. This volume, containing Parts 3 through 8, describes the results of several technical investigations which were conducted. Part 3 is a tutorial on computational algebra, illustrating the nature of research in the area of algorithm analysis and computational complexity. Part 4 develops a systematic approach to the analysis of algorithms. Part 5 is an experimental analysis of a fast, new sorting method called DPS (distributive partitioning sorting). Part 6 applies order statistics to investigate the expected quality of several approximation algorithms for the Euclidean traveling salesman problem, known to be NP-complete. Part 7 presents a survey of data base access methods for both univariate and multivariate range queries. Part 8 describes an experimental evaluation of the frame memory model of a data base structure.


Key words: Interoperability, Quality Metrics, Quality, Reusability
Software metrics (or measurements) which predict software quality were extended from previous research to include two additional quality factors: interoperability and reusability. Aspects of requirements, design, and source language programs which could affect these two quality factors were identified and metrics to measure them were defined. These aspects were identified by theoretical analysis, literature search, interviews with project managers and software engineers, and personal experience. A Guidebook for Software Quality Measurement was produced to assist in setting quality goals, applying metrics and making quality assessments. This document describes the technical effort accomplished under contract F30602-80-C-0265. (author)


Key words: Interoperability, Quality Metrics, Quality, Reusability

Software metrics (or measurements) which predict software quality were extended from previous research to include two additional quality factors: interoperability and reusability. Aspects of requirements, design, and source language programs which could affect these two quality factors were identified and metrics to measure them were defined. These aspects were identified by theoretical analysis, literature search, interviews with project managers and software engineers, and personal experience. A guidebook for Software Quality Measurement was produced to assist in setting quality goals, applying metrics and making quality assessments. This document is a guidebook for using the software quality measurement framework. (author)


Key words: Interoperability, Quality Metrics, Quality, Reusability, Software Life Cycle

This document is Volume II of three of the Final Technical Report (CDRL A003) for the Quality Metrics for Distributed Systems contract, Number F30602-80-C-0330. The contract was performed for Rome Air Development Center (RADC) to provide methodology and technical guidance on software quality metrics to Air Force software acquisition managers. This volume describes the application of quality metrics to distributed systems and provides guidance for AF acquisition managers. The guidebook provides guidance for specifying and measuring the desired level of quality in a software product. (author)


Key words: Quality Metrics, Quality, Reusability, Software Life Cycle

This document is the final technical report (CDRL A003) for the Quality Metrics for Distributed Systems contract, number F30602-80-C-0330. The contract was performed for Rome Air Development Center (RADC) to provide methodology and technical guidance on software quality metrics to Air Force Software acquisitions managers. The objective of this contract was to conduct exploratory development of techniques to measure system quality with a perspective on both software and hardware from a life cycle viewpoint. The effort was expected to develop and validate metrics for software quality on networked computers and distributed systems; i.e., systems whose functions may be tightly distributed over microprocessors or specialized devices such as data base machines. At the same time, the effects hardware has on software was to be studied, as well as the trade-offs between hardware.
firmware, and software. The results of this research are reported in this volume. (author)


Key words: Data Structures, Maintenance Tools and Techniques, Maintenance, Quality Metrics

This report documents the research performed under RADC Contract No. F30602-80-C-0139 by Northwestern University for developing effective methodologies for software maintenance. This contract is a follow-on to Contract No. F30602-76-0397 and focuses on refining, expanding and automating software maintenance concepts and techniques developed under the previous contract. Topics addressed in the report include techniques for specifying and realizing software modification proposals, logical ripple effect analysis and module revalidation after modification. These techniques and the performance ripple effect analysis technique developed during the previous contract period were demonstrated using a DEC VAX 11/780 computer. In addition, the development of a number of software metrics related to modifiability, such as measures for logical and performance stability, module strength and coupling, are described. (author)


Key words: Availability, Command, Control, and Communications, Data Analysis, Design Methodologies, Distributed Processing, Errors, Failures, Fault Detection, Fault Tolerance, Reliability Measurement, Testability

This report discusses fault tolerance, reliability, and testability in distributed systems which support Command, Control and Communications and Intelligence (C3I) activities. Taxonomies of fault tolerance and distributed systems are developed and typical Air Force C3I needs in both fault tolerant and distributed computer systems are characterized. Reliability and availability experience for ten typical computer systems is reported in a consistent format, and the data are analyzed from the perspective of a distributed system user. Previous work on the identification of problems in distributed systems and design methods for their solutions is discussed. Key issues in the design of fault tolerant distributed systems are identified. Fault location techniques for specific computer configurations found in C3I applications are described in detail.


Key words: Interoperability, Quality Metrics, Quality, Reusability, Software Life Cycle

This document is Volume III of the final technical report (CDRL A003) for the Quality Metrics for Distributed Systems contract, number F30602-80-C-0330. The contract was performed for Rome Air Development Center (RADC) to provide methodology and technical guidance on software quality metrics to Air Force software acquisitions managers. This volume describes a qualitative study of distributed system characteristics, reasons for selection, design strategies, topologies, scenarios, and trade-offs. These analyses led to the changes in the Framework shown in Volume I, and to the validation of models. (author)

This report (volume I) contains the results of a study to determine the use and applicability to Air Force software acquisition managers of six quantitative software reliability models to a major command, control, communications, and intelligence (C3I) system. The scope of the study included the collection of software error data from an ongoing C3I project, fitting six software reliability models to the data, analyzing the predictions provided by the models, and developing conclusions, recommendations, and guidelines for software acquisition managers pertaining to the use and applicability of the models. (author)


Key words: Software Science, Development Methodologies, Ada, Automatic Data Collection, Complexity Measurement, Models, Program Complexity, Program Design Language, Quality Metrics, Software Engineering Tools and Techniques

The Rome Air Development Center has developed the Software Quality Framework as a means to specify software quality goals and measure software quality. Much of the work to date has focused on metrics applicable to software code. This report describes an effort undertaken to measure the quality of software products earlier in the software development life cycle, during the design phase, and to automate the capture of metric data from design media. Metrics of software quality, primarily those related to the criterion simplicity (or conversely, complexity), were reviewed. This review includes those metrics previously developed in the Software Quality Framework. Two metrics, Halstead's Software Science and McCabe's Cyclomatic Complexity were chosen for their amenability to measurement during design and their potential for automation. Two design media were used: Design Aids for Real-Time Systems (DARTS), an experimental automated design tool developed at the Charles Stark Draper Laboratory; and Ada as a program design language. (PDL) (author)


Key words: Acquisition Management, Quality Attributes, Quality Metrics

This report, Volume I (of three), describes the results and presents recommendations for integrating the Rome Air Development Center developed software quality metrics technology into the Air Force software acquisition management process and for changing Air Force acquisition documentation. In
addition, changes to the baseline software quality framework are presented and features of a proposed specification methodology are summarized. (author)


Key words: Command, Control, and Communications, Quality Attributes, Quality Metrics, Software Engineering Standards

This paper is Volume II (of three) and describes how a software acquisition manager specifies software quality requirements, consistent with needs. Factor interrelationships, tradeoff among factor quality levels in terms of relative costs, and an example for a command and control application are described. Procedures for assessing compliance with the specified requirements based on an analysis of data collected using procedures described in volume III are included. (author)


Key words: Data Collection, Quality Attributes, Quality Metrics, Software Life Cycle

This paper is volume III (of three) and describes procedures and techniques for evaluating achieved quality levels. Worksheets for use in metric data collection by software life cycle phases and scoresheets for scoring each factor are provided. (author)


Key words: Data Collection, Quality Metrics, Reliability Estimation, Reliability Measurement, Reliability Prediction, Software Experience Data, System Reliability

This report is Volume 1 of two volumes which document the results of a research and development effort to develop a methodology for predicting and estimating software reliability throughout the life cycle of a software project. Volume 1 contains the results and findings of the research effort. Volume 2 contains a Methodology in the form of a Guidebook for predicting and estimating software reliability based on the findings in Volume 1. In Volume 1 a description of the framework which spans the life cycle of a software system is found. The framework acknowledges the contributions of past Rome Air Development Center (RADC) research in metrics and models as techniques to aid in the prediction and estimation of reliability during the software development process. Within the framework, the specific data needed to measure software reliability and the utility of the measurements to help make sound software engineering decisions is addressed. Included is the results of a comprehensive data collection effort to investigate software reliability. This extensive database supported the preparation of the Guidebook described in Volume 2.


Key words: Data Collection, Quality Metrics, Reliability Estimation, Reliability Measurement, Reliability Prediction, Software Experience Data, System Reliability
This report is Volume 2 of two volumes which document the results of a research and development effort to develop a methodology for predicting and estimating software reliability throughout the life cycle of a software project. Volume 1 contains the results and findings of the research effort. Volume 2 contains a Methodology in the form of a Guidebook for predicting and estimating software reliability based on the findings in Volume 1. Volume 2 is a Guidebook which provides procedures for the preparation of software reliability predictions and estimations for embedded and separately procured computer systems. These procedures include those for developing a software reliability program plan, for performing software reliability prediction, and for performing software reliability estimation. Data collection procedures which describe what data must be collected to use the software reliability prediction and estimation procedures are provided. Included in the Guidebook are metric data collection worksheets.


Key words: Ada, Error Analysis, Fortran, Measurement, Reliability

This report presents the results of an experiment investigating the effect of FORTRAN and Ada languages on program reliability. The experimental design employed was a 2 by 2 full factorial design, for example, a design in two variables, each at two levels. The problems used in the experiment was the Launch Interceptor Program (LIP), a simple but realistic anti-missile system. Reliability comparisons between Ada and FORTRAN programs were based on the total number of errors as well as on errors found during various testing phases. Some comparisons were also based on error density, the number of errors per 100 non-comment lines of code. It was found that on the average, the Ada programs had about seventy percent less errors than the FORTRAN ones. If errors during unit testing were excluded, the Ada programs had about seventy-eight percent less errors. Similar differences were found for data based on error causes and error types. (author)


Key words: Acquisition Management, Quality Attributes, Quality Metrics

The purpose of this project was to assess the feasibility and utility of transferring the software quality specification and evaluation methodology from the Specification of Software Quality Attributes Guidebooks (3 Volumes, RADC-TR-85-37, February 1985) to the acquisition environment. Two decision aid developments were used as the test programs. Specific recommendations have been made to improve on the methodology, thereby enhancing its acceptance by and utility to software acquisition managers. (author)


Key words: Quality, Methodology, Traceability, Quality Metrics

The four (4) tasks and their results performed under this study are: 1. Perform a critical analysis of the RADC Software Quality Measurement Methodology. Methodology is complex. The specification process should be modified. 2. Determine if AFSC Management and Quality Indicators can be integrated into the RADC methodology. Three (3) of the Quality Indicators can be integrated. 3. Review completeness of Methodology Traceability through software life cycle phases. In general, traceability is present. 4. Create a cross reference guide between Methodology life cycle phase metrics and applicable DOD-STD-2167A DIDs paragraphs. A cross reference matrix was developed; however, many problems exist and recommendations are made to alleviate the problems.
Volume I presents the features of the Quality Evaluation System (QUES). QUES is a tool which automates the process of establishing, maintaining, and applying a software quality evaluation framework. Features are illustrated with examples and reports generated by the tool. Main features include framework creation, framework tailoring, project creation and data collection. QUES will support analysis of Fortran and Ada code on a DEC VAX and analysis of Ada code on a SUN SparcStation. Volume II summarizes the Rome Laboratory Software Quality Framework (RLSQF) as automated with the QUES tool. This volume upgrades the original RADC Software Quality Framework documented in 1985. The primary changes were improved objectivity of questions, improved automatability of questions, improved applicability to Ada, and consistency with DOD-STD-2167A lifecycle phases and terminology.


The purpose of this project was to assess the feasibility and utility of transitioning the software quality specification and evaluation methodology found in the Specification of Software Quality Attributes Guidebooks (3 vols., RADC-TR-85-37, Feb 85) to the software acquisition environment. Two decision aid developments were used as the test programs. Specific recommendations have been made to improve on the methodology, thereby enhancing its acceptance by and utility to software acquisition managers.


The data processing and analysis operations performed on the channel data recordings of Volume II are described, and representative numerical results are discussed. Some of these results are based on defining error bursts in terms of error gaps, instead of error densities (or as the absence of an error-free interval). This basis for defining bursts was found to be advantageous. One important result was the finding that when bursts defined in this way contain only short gaps, the bursts have only limited lengths but are quite closely spaced. Another result is that for the channel data tapes processed, the signal quality parameters are relatively unreliable indicators of demodulator output errors. One result of the evaluation of coding techniques for turbulent channels indicates that multiple-interleaved cascade F-multi-codes are advantageous. Another one of these results is that in the present state of the art, these multicode should be based on cyclic, shortened cyclic, or extended cyclic subcodes, and the inner subsystem should use a bit error detecting F-singlecode with a bit EDAC augmented digital decoder. If the error control performance requirements are sufficiently high, and if the channel bit rate is sufficiently low (e.g., less than 9600 bps), then a hybrid hardware/software decoder implementation may be advantageous if otherwise permissible. In all other cases, a completely hardware implementation is preferable. (Author)
The document proposes a methodology to be employed in the testing of data management systems and submits some recommendations for the continued development of a DMS Test Methodology. The intent of the document was first to characterize a data management system by identifying the various attributes that should comprise a DMS and summarize the techniques that can be employed in implementing these capabilities. Secondly, the standard test techniques that can be used to measure the capabilities of the aforementioned attributes were examined and a DMS Test Methodology was proposed. The report illustrates how the methodology would be employed in the solution of some typical DMS measurement problems. Finally, it was concluded that analysis, benchmark programs, and software monitors were the most useful test techniques available and warrant additional development. (Author)


Key words: Measurement, Data Processing, Test Methods, Computer Programming, Monitors, Management Planning

The document describes the techniques used and the results obtained from actually measuring the performance of two data management systems, MADAPS and AIDS. The intent of the document was to validate a DMS Test Methodology described under separate cover by actually employing the recommended measurement techniques. Additionally, it was expected that the undertaking of such an exercise would illustrate the strengths and weaknesses of the selected techniques and, perhaps, further refine the developed test methodology. The techniques selected included analysis, numerical scoring, benchmark programs, and software monitors; however, the latter technique could not be employed to measure the AIDS system because of inadequate documentation regarding AIDS. The techniques evaluated were found to be valid when employed in a manner consonant with their aims. Software monitoring is found to be worthy of additional development. (Author)


Key words: Measurement, Artificial Intelligence, Computer Program Reliability, Computer Programs, Quality Assurance, Requirements, Specifications, Standardization, Test Methods, Validation, Verification

Rule based systems are being applied to tasks of increasing responsibility. This report focuses on techniques for the verification and validation of these systems. Conventional software quality assurance depends on the availability of requirements and specification documents. For rule systems, there are generally none because the capabilities of these systems evolve through a development process that is partly experimental in nature. Conventional testing techniques are considered; however, such techniques do not carry over absence of errors. Methods for proving the consistency of rule systems are examined. These methods require that the rules be viewed declaratively, which may be too much of a simplification. A semantics for rule systems based on term rewriting is developed. Standard tests for confluence of term rewriting systems cannot be converted to rule systems, however, because the firing of rules can depend on the absence, as well as the presence, of elements in working memory. Finally, we consider deductive methods for the validation of rule systems.


Key words: Measurement, Computer Program Verification, Knowledge Based Systems, Acquisition, Methodology, Quality, Quality Assurance, Requirements, Specifications

The ASQS, a knowledge base software tool, assists software acquisition managers which specify software quality requirements. In the consultation mode, questions are presented to the acquisition manager in terms related to the mission area of interest. Answers are transformed by ASQS into tailored quantitative quality goals based on the thirteen software quality factors of the RADC software quality framework. The current version supports the Intelligence and Satellite mission areas. ASQS serves the purpose of transitioning the Software quality specification methodology found in the RADC...
Key words: Measurement, Computer Program Reliability, Quality Assurance, Computer Programs, Data Bases, Quality, Tools

A long range approach for integrating software quality information with knowledge based engineering technology was developed. A Software Quality Assurance Expert System framework was proposed to plan software quality assurance activities, evaluate systems designs, balance mutually conflicting quality factors, and make design refinement suggestions. This effort determined the basic system architecture and interaction among system components. The proposed expert system framework would include data objects of an object-oriented data base, a rule set, meta rules, and a dependency-based truth maintenance system. To help illustrate how such a system could be used, examples were provided to show how the expert system could assist Software Quality Assurance activities for software reliability. The DOD community will benefit from the results of this work; particularly, anyone attempting to use expert systems to improve the quality of their software.

Key words: Measurement, Ada Programming Language, Coding, Computer Applications, Computer Programs, Experimental Data, Life Cycles, Maintainability, Reliability, Test and Evaluation, Tools

Software test and verification are critical software development problems facing the Air Force. Because software is major element of military systems, substantial amounts of time and money are currently spent by the Air Force to correct and certify software systems. In spite of the enormous investment, it is generally agreed that the results of testing are unsatisfactory. Systems thought to be correct, and purchased as such by the Air Force, may suddenly produce incorrect results or behave erratically. Many errors occur in software released for operational use because of inadequate software testing, resulting primarily from the lack of effective test tool capabilities. In many cases, the developer cannot accurately measure the effectiveness of a particular test and/or determine whether the set of test data has thoroughly exercised the software system. The objective of the ATVS effort is to develop an integrated set of computer based tools to provide computer program test and verification support for the Ada programming language (MIL-STD-1815A), with the goal of improving the reliability and maintainability of Ada software systems. The tools are intended to assist programmers, test personnel and test managers during the coding, testing and maintenance phases of the software life cycle.

Key words: Measurement, Accounting, Ada Programming Language, Archives, Knowledge Based Systems, Planning, Quality, Resources, Tools

This project was part of the Knowledge Based Software Assistant (KBSA) program, whose principal motivation and premises can be summarized as follows. Software development, whether in the large or in the small, is knowledge-intensive. Much of this knowledge resides in people's heads, is never made explicit, and cannot be completely reconstructed from the products produced. This in turn limits tool support to passive product archives that provide only very limited help with the software production process itself. The knowledge-based paradigm states that the explication, formalization, and machine-manipulable representation of software process knowledge will lead to technology that will significantly raise the level of software process integrity and efficiency. Viewed from a paradigm requires and explicit representation of the software process that allows machine-reasoning about the process. The
consequent capabilities include traceability among products and accounting of resources at a known level of precision; assistance in assessing the quality of project plans relative to their goals; support for each participant's particular perspective, role, and responsibilities; automation of the coordination and communication between participants.


Key words: Measurement, Fault Tolerant Computing, Systems Engineering, Prototypes, Acceptance Tests, Computer Programs, Faults, High Reliability, Modification, Reliability

Software Fault Tolerant (SWFT) techniques have been discussed for many years as a means to achieve high reliability in critical systems. However, SWFT techniques are infrequently used in existing systems. This is, in part, due to a lack of a formal development methodology that results in a system design that explicitly identifies the nature of SWFT technique inclusions in the system to be developed in order for that system to meet its reliability requirements, and yet provide the most cost-effective SWFT design alternative possible. Current implementations of SWFT systems are typically the result of costly postdevelopment modifications to the system after the discovery that the initial product proves to be unreliable. In this report, emphasis is placed on the establishment of a SWFT development methodology via a Software Fault Tolerant Design System (SWFTDS) Computer-Aided Software Engineering (CASE) tool used by software system designers to assist in the production of SWFT designs. The rationale and theory behind the SWFTDS prototype developed under this effort are described.


Key words: Measurement, Data Bases, Information Centers, Data Acquisition, Requirements, Costs, Reliability, Quality, Maintainability, Technology Transfer

The Data & Analysis Center for Software (DACS) was established to serve as a central source for data and information on software technology. The DACS was operated on a pilot basis for a 36-month period, then entered a 15-month transition period. This final report presents a detailed summary of the activities carried out, products produced, and services provided during this transition period. Details are also provided on enhancements to the STINFO databases, the Software Life Cycle Empirical Database (SLED), and the implementation and results of the DACS cost recovery program. A list of users of the DACS during this transition period is provided in an Appendix. (Author)


Key words: Measurement, Computer Program Documentation, Life Cycles, Benefits, Computations, Computer Applications, Data Acquisition, Data Reduction, Error Correction Codes, Maintainability, Programming Manuals, Productivity, Programmers, Computer Program Reliability, Survivability

The purpose of this project was to develop a tool to automate the method for evaluating software quality. The Automated Measurement System (AMS), a computer-based software tool, provides the capabilities to monitor the overall quality and resource expenditure of software under development. The AMS collects, stores and analyzes software measurement data for use by software acquisition and software project personnel. It provides managers with a means to quantitatively specify quality goals and track progress toward those goals during all phases of the software life cycle. The underlying philosophy of the AMS is based on a framework consisting of a set of 13 software factors (i.e., reliability, maintainability, reusability, portability, interoperability, usability, integrity, flexibility, expandability, verifiability, correctness, survivability, and efficiency) which are associated with high level concerns of software quality. Quality/productivity benefits include: consistency of collected data via the AMS automated data collection mechanisms; fast and easy to use interactive forms manager for manual data collection; automated data reduction which provides error-free calculations and removes software development personnel from the tedious and error-prone task; timely information is made
available and subsequent quality analysis can be made virtually on the spot; potential software quality
problems identified via the AMS report subsystem provide managers the choice of correcting the
deficiency immediately or delaying until later in the software development life cycle. (JHD)

15372 Dean Jeffrey S.; McCune Brian P.: ADVANCED TOOLS FOR SOFTWARE MAINTENANCE. Report
from DTIC. Order No. D-A126 146.
Key words: Measurement, Computer Program Documentation, Maintenance, Modification, Life Cycles,
Interfaces, Computer Programming, Computer Operators, Personnel Management
This is the final report on a project entitled Software Maintenance Techniques. The purpose of this
project was to study and design advanced software maintenance tools and techniques for the future
ADA programming environment. Current maintenance practices for Air Force C31 (Command Control
Communications and Intelligence) software were studied. Three out of the four major problems
identified were attributed to the difficulty of comprehending software. Nine tools have been proposed to
help solve these and other problems, including a tool to help coordinate the programming process (the
Programming Manager), a tool to aid in the collection and use of documentation (the Documentations
Assistant), and an editor that is knowledgeable about what it is editing (the Intelligent Editor). The nine
tools are based on the computer science technologies of artificial intelligence (particularly knowledge-
Based and expert systems), automatic programming, intelligent user interfaces, formal verification,
software engineering, programming environments, software metrics and computer-assisted instruction.

15373 McCall, James A.; Markham, David: AUTOMATION OF QUALITY MEASUREMENT. 163p.
Sponsored by RADC. Grant/Contract No. F30602-79-C-0267. Avail. from DTIC. Order No. D-A121
360.
Key words: Measurement, Computer Program Reliability, Quality Assurance, Data Acquisition, Data
Storage Systems, Procurement, Prototypes, Data Management, Data Bases, Test and Evaluation
A prototype software system has been developed which allows manual input and provides for automated
collection of software metric data, stores the data, and provides processing and reporting to facilitate
use of the metric information to monitor and control the quality of a software product. The software
system, called the Automated Measurement Tool, processes COBOL source code. (Author)

15374 Lamagna, Edmund A.; Bass, Leonard J.; Anderson, Lyle A.; Bunker, Ralph E.; Janus, Philip J.;
ALGORITHMIC COMPLEXITY. VOLUME II. Report No. 81-161-VOL-2. 353p. Sponsored by
Key words: Measurement, Algorithms, Computer Program Reliability, Computer Programs,
Computations, State of the Art, Multivariate Analysis, Machine Translation, Loops, Subroutines,
Semantics, Sorting, Order Statistics, Data Bases
The objective of this study was to conduct applied research directed toward understanding the
relationship between the complexity or efficiency of algorithms and the overall quality of computer
software. The final report is presented in a two volume series consisting of a total of eight parts. This
volume, containing parts 3 through 8 describes the results of several technical investigations which are
conducted. Part 3 is a tutorial on computational algebra, illustrating the nature of research in the area of
algorithm analysis and computational complexity. Part 4 develops a systematic approach to the analysis
of algorithms. Part 5 is an experimental analysis of a fast, new sorting method called DPS (distributional
partitioning sorting). Part 6 applies order statistics to investigate the expected quality of several
approximation algorithms for the Euclidean traveling salesman problem, known to be NP-complete.
Part 7 presents a survey of data base access methods for both univariate and multivariate range queries.
Part 8 describes an experimental evaluation of the frame memory model of a data base structure.

15375 Lamagna, Edmund A.; Bass, Leonard J.; Anderson, Lyle A.; Bunker, Ralph E.; Janus, Philip J.;
ALGORITHMIC COMPLEXITY. VOLUME I. Report No. 81-161-VOL-1. 129p. Sponsored by RADC.
Key words: Measurement, Algorithms, Computer Program Reliability, State of the Art, Computer
Programs, Computations, Quality, Metric System, Interactions

17
The objective of this study was to conduct applied research directed toward understanding the relationship between the complexity or efficiency of algorithms and the overall quality of computer software. The final report is presented in a two volume series consisting of a total of eight parts. This volume, containing Parts 1 and 2, comprises a general introduction to the entire series and a research plan. Part 1 begins with a description of the goals of the overall contract effort. This is followed by a discussion of previous RADC work on software quality metrics, emphasizing measures concerned with the time and storage efficiency of programs. Next, an overview of the field of algorithm analysis and computational complexity is given. A final section contains an introduction to the particular research investigations pursued in the other portions of this study. Part 2 presents a research plan for advancing the state-of-the-art in the area of algorithm performance. An Appendix contains an outline of the major topics and issues addressed in the area of algorithm analysis and computational complexity, together with an annotated select bibliography.


Key words: Measurement, Computer Program Verification, Computer Programs, Reliability, Systems Analysis, Tools, Cost Analysis, Data Bases, Contracts, Requirements, Validation, Management Planning and Control

Sneak analysis is an engineering tool which can used to identify designed-in conditions that can inhibit desired system functions resulting in lower operational reliability. The objective of this study was to develop guidelines for applying sneak analysis to hardware and software contracts in a cost effective manner. Specific guidelines for establishment of sneak analysis requirements, monitoring of sneak analysis efforts, determining program costs, evaluation criteria and determining candidate equipments have been defined. Criteria for tailoring the guidelines to the program needs have also been established. A measure of sneak analysis effectiveness was determined which shows excellent results in early program development phases including improved system reliability. (Author)


Key words: Measurement, Computer Program Verification, Test and Evaluation, Computer Program Reliability, Computer Program Documentation, Cost Analysis

This report examines the current state of development of automated software testing techniques. The report identifies and describes techniques that are useful for detecting errors in software. It also examines techniques for proving the correctness of programs, for debugging (locating and correcting errors), and for producing documentation automatically.


Key words: Measurement, Computer Programs, Data Management, Data Acquisition, Life Cycles, Productivity, Quality, State of the Art, Systems Engineering, Command and Control Systems, Telecommunications

Software tools which support the measurement and assessment of product quality and life cycle productivity are just beginning to emerge. However, their effectiveness is currently hampered by the lack of appropriate data necessary to drive the measurement and assessments. The state-of-the-art of data collection is limited to a few measurement tools which provide only a small subset of possible data collection is limited to a few measurement tools which provide only a small subset of possible data collection functions. Data collection can be performed environment. The primary objective of this research was to specify and develop a high-level design of automated data collection mechanisms, within the context of a life cycle environment. The resulting technology described in this report includes: (1) identification of data that could be collected automatically; (2) identification of necessary instrumentation mechanisms; (3) description of a prototype, and (4) guidelines for instrumenting tools and environments.

Key words: Measurement, Management Planning and Control, Quality Assurance, Requirements, Documents, Regulations, Systems Engineering, Quality Control, Methodology

This effort was co-ordinated among Rome Air Development Center (RADC), Defense Logistics Agency Headquarters (DLA Hq), Air Force Contracts Management Division Headquarters (AFCMD Hq), and Electronic Systems Division (ESD). System Architects, Inc. (SAI) performed this effort and has examined, analyzed, and evaluated the current software acquisition and contract administration management documents, software quality assurance tools, techniques and communication methods and has developed a series of recommendations for improved methods for assuring quality software. These improved methods encompass the entire software development life cycle which consists of five phases: (1) Requirement Analysis, (2) Design, (3) Code and Checkout, (4) Test and Integration, and (5) Operation and maintenance. SAI examined relevant documentation, conducted interviews and compiled the results from a comprehensive questionnaire as the basis for the analysis, evaluation and recommendations which can be found herein. SAI's recommendations for improved methods of assuring quality software are classified in four groups: (1) Establish clear, unambiguous Government Software Quality Assurance Guidance Documents, (2) Includes Software Quality Assurance Functions in all phases of the Software Development Life Cycle, (3) Improve communication methods and model documents primarily by mutual agreement regarding allocation of functional responsibilities between CAO's and Program Offices, and (4) Provide up-to-date training and people skilled in software to government SQA organizations.


Key words: Measurement, Computer Program Reliability, Computers, Computer Programs, Reliability, Interactions, Markov Processes

A general methodology is developed for combining hardware and software reliability. Based on this general methodology, a baseline combined HW/SW reliability was developed incorporating and unifying the SW reliability theory of Jelinski-Moranda, Goel-Okumoto with traditional HW reliability theory. The baseline model is computerized and includes various HW/SW failure and repair characteristics, allowance for imperfect SW debugs and modes of HW/SW interaction. Finally a HW/SW tradeoff procedure is developed using a combined HW/SW availability measure. Examples are provided to illustrate the general theory and tradeoff procedure. (Author)


Key words: Measurement, Communications Networks, Computer Communications, Integrated Systems, Information Exchange, Quality Assurance, Computer Architecture

This final technical report documents the following specific activities of the subject contract: (1) Review of the NSW project organizational structure, including roles, responsibilities and contractor assignments, (2) Description of the NSW development process, (3) Survey of the NSW system architecture and user functionality, (4) Establishment of the current operational status of NSW including the chronology of major events, and finally (5) Serve as an introductory NSW tutorial for new users. (Author)


Key words: Measurement, Computer Programming, Methodology, Communications Networks, Computer Communications, Resource Management, Maintenance, Tools

The objective of this effort was to develop and apply guidelines, standards, and procedures for the selection, installation, and maintenance of software tools on the National Software Works (NSW), a network of computers which is a subset of the DoD-wide ARPANET system of computers. This was the
first step in the establishment of the function of a Tool Manager role within the NSW community. A 'Tool Configuration and Quality Management Plan' was developed to provide a framework for the classification, selection, installation, maintenance, and control of tools on the NSW. (Author)


Key words: Measurement, Computer Programming, Programming Languages, Fortran, Computerized Simulation, Performance Engineering, Performance Tests, Test and Evaluation, Analysis of Variance, Syntax, User Needs, Computer Programs

The purpose of this study was to obtain quantitative measures of the relative performance of two very different programming languages. One language was Fortran and the other was the Klerer-May Two Dimensional Language. The report describes the experiments conducted, presents the experimental results and provides a discussion of the results obtained. (Author)


Key words: Measurement, Computer Program Verification, Test and Evaluation, Maintainability, Data Bases, Feasibility Studies, Microcomputers

ATVS is a system for testing, maintaining, and documenting Ada programs. A 12-month effort to specify ATVS has resulted in two interim technical reports, a Functional Description, a Database Specification, and a System/Subsystem Specification. This final report summarizes those reports and describes the research undertaken to generate the functional description and design. The specified ATVS will provide static and dynamic tasking analysis; static data and structural flow analysis; dynamic branch, assertion, and timing analysis; collection of software metrics; automated generation of reports suitable for incorporation into DOD-STD-2167 Data Items; an interactive source walkthrough feature; extraction and insertion of descriptive prologues in Ada programs; and Ada programming standards checking. Particular attention was paid to specifying and designing a user interface for ATVS that is both menu and command-driven. The user interface will support both keyboard and mouse input. ATVS display processing will take advantage of multiple-window support by VMS 4.0 on DEC VAX computers, to support ATVS's multi-faceted program analysis approach.


Key words: Measurement, Computer Program Verification, Computer Programming, Automation, Specifications, User Needs, Computer Program Documentation, Quality Control

The COBOL language, and automated software testing tools, have been studied in order to design an Automated Verification System for COBOL. The proposed functions and design of the system are summarized in this report. Details of the system are presented in the CAVS Functional Description and CAVS System/Subsystem Specification. To provide a perspective for the capabilities of the proposed system, this report contains a critique of the COBOL language, a description of methods of software testing, and a characterization of errors in COBOL. Considerations for future capabilities of the system are also outlined. (Author)


Key words: Measurement, Computer Program Reliability, Mathematical Models, Quality Control, Computer Program Verification, Low Costs, Life Cycle Costs

Discrete (binomial and Poisson) software reliability models, previously developed by IBM, are presented. They were examined for their validity using historical data provided by RADC from seven
development projects. Simulated error data that closely agreed with the model assumptions were also used for validation. The results indicated that the models provide reasonable fits to the historical error data.


Data were collected over the development cycle of the PAVE PAWS software development project. This project was designed to be a technology demonstration of modern programming practices. The practices studied on this project included: (1) Top-down design and implementation, (2) Structured coding and precompilers, (3) Program support library, (4) Program design language and HIPO charts, (5) Chief programmer teams, (6) Structured walkthroughs, (7) Independent test and quality assurance groups. The data available for evaluating this project included personhours, trouble reports, compiler summaries, code progression and durability charts from the library, management summaries, and personnel profiles. It was concluded from these data and our previous study on the LSDB project that modern programming practices are not miraculous productivity aids. Rather, these practices represent sound management principles which make software development more manageable and the prediction of project outcomes more accurate. Recommendations are also made for methods of collecting software life cycle data in future studies.


Key words: Measurement, Computer Programming, Military Applications, Air Force Research, Systems Management, Systems Engineering, Quality Assurance, Technology Transfer, Fortran, State of the Art, Management Planning and Control

This study represents the first formal, rigorous approach to modernizing the DMA programming environment. This study produced a plan for upgrading the current software production practices at all DMA installations using modern software engineering tools and procedures. (Author)


Key words: Measurement, Computer Programs, Computer Programming, Error Analysis, Models, Test Methods, Computer Program Reliability, Management, Mathematical Programming

This interim report summarizes the research performed by Polytechnic Institute of New York for Rome Air Development Center. The principal topics covered are (1) software test models and implementation of Automated test drivers to force-execute every program path, (2) development of new measures of program complexity based upon information theory, (3) models of software management and organizational structure, and (4) statistical measures relating the probability of finding a program error to the testing of that program.


Key words: Measurement, Computer Programs, Computer Program Reliability, Mathematical Models, Time Dependence, Markov Processes

In this report, the concept of software reliability is discussed. Various software reliability models along with the basic philosophy of their development, their characteristics, and derived figures of merit are examined. These models' strengths and weaknesses are evaluated. Attempts for validating these models are discussed. (Author)
FAVS, for FORTRAN Automated Verification System, is a tool for analyzing source programs written in FORTRAN or DMATRAN. It is essentially a software system to be used as an aid in improving, documenting, and validating the quality of software and software testing by providing for: syntax and structural analysis of the user’s source program, static analysis to detect inconsistencies in program structure or in the use of variables, automated documentation, instrumentation of the source code, analysis of testing coverage, and retesting guidance. A separate function that FAVS can perform is the transformation of an unstructured FORTRAN program into a logically equivalent DMATRAN program. This manual describes how to use FAVS from the beginning of the software development cycle to its completion.

JAVS, for JOVIAL Automated Verification System, provides measurements of testing thoroughness, retesting assistance, and automated software documentation for JOVIAL J3 programs. This report describes the design, implementation and testing of a new JAVS syntax analyzer. Background information regarding all JAVS contracts is provided in this report, as are procedures for installing the complete JAVS software package. Familiarity with the JOVIAL language and with software verification terminology is assumed. (Author)

In this report a model for the operational phase of a software system is developed which incorporates the uncertainty of error removal and the time spent in correcting errors. Expressions for various measures of software system availability are derived. Numerical examples are used to illustrate these results. (Author)

Key words: Measurement, Computer Program Reliability, Computer Programming, Bayes Theorem, Markov Processes, Model Theory, Mathematical Models

In this report a stochastic model for software failure phenomena is developed for the case when the errors are not corrected with certainty. Expressions for several quantities of interest are derived to establish quantitative measures for software performance assessment. Approximations for large-scale software systems using a gamma distribution are also discussed. Numerical examples are used to illustrate the computations and usefulness of various quantities. (Author)


Key words: Measurement, Computer Programs, Mathematical Models, Systems Analysis, Reliability, Error Analysis, Statistical Tests, Markov Processes, Microprocessors, Microprogramming, Cost Effectiveness, Command and Control Systems

This report documents research performed under RADC contract by Polytechnic Institute of New York in the area of software modeling. Research in the areas of software error, reliability, and availability models (such as Markov availability models, bug tagging estimates of initial error counts and error models incorporating error generation), test models and techniques (such as determination of the number of the tests necessary to execute all program paths and statistical test models), and complexity models (such as component measures like testedness and natural language theory measures) that was described in previous progress and technical reports is summarized and unfinished research (in areas such as acceptance test models and automatic programming techniques) not previously reported is described. The significant results are highlighted along with their interrelations and potential.


Key words: Measurement, Computer Programs, Information Centers, Data Bases, Input Output Processing, Data Management, Pilot Studies

The purpose of the Software Data Repository is to upgrade the software development process through collection, analysis, and dissemination of software development experience. A functional definition of the repository including a discussion of the inputs, processes, and outputs is presented in this report. The input processing and the requirements of an information system for storing and processing the data is discussed, along with a presentation of the recommendations for the repository including the development and operation of a pilot facility and the expansion of this facility into a fully operational center. (Author)


Key words: Measurement, Computer Programming, Quality, Specifications

An hierarchical definition of factors affecting software quality was compiled after an extensive literature search. The definition covers the complete range of software development and is broken down into non-oriented and software-oriented characteristics. For the lowest level of the software-oriented factors, metrics were developed that would be independent of the programming language. These measurable criteria were collected and validated using actual Air Force data bases. A handbook was generated that will be useful to Air Force acquisition managers for specifying the overall quality of a software system. (Author)

Key words: Measurement, Pattern Recognition, Computer Programs, Automatic, Computer Programming, Comprehension

The On-Line Pattern Analysis and Recognition System (OLPARS) was demonstrated as a tool for evaluating program characteristics which contribute to program readability, freedom from errors, and development time. Structural features were extracted automatically from a data base of 155 PL/I programs and used as inputs to OLPARS. As expected, program length had a dominant effect on the time required to understand programs; additional features affecting understandability included GOTO and RETURN statements. Significant contributions of this study were methods for estimating program error rates from archival data, and reliable techniques for estimating understandability of programs. An interesting by-product was the ability to identify individual programmer styles. The principal outcome of the study was the demonstration that OLPARS could provide a facility for evaluating factors contributing to software quality. (Author)


Key words: Measurement, Programming Languages, Computer Programming, Computer Program Documentation, Machine Coding, Costs, Savings, Macroprogramming, Computer Programs, Symbols

Aspects of the development of the ECL Programming System are presented. ECL is built around the extensible programming language EL1. Specifically, the material deals with debugging facility, measurement aids, and optimization mechanisms in ECL, symbolic evaluation of EL1 programs, and the design and development of a subset of the full EL1 language particularly suited to systems programming. (Author)


Key words: Measurement, Computer Program Documentation, Computer Programming, Programmers

The overall objective of the Evaluation of Standard Programming Practices study is to assist RADC in the assessment of the effects of Modern Programming Practices (MPPs) on the development of large systems produced for the Department of Defense (DoD). The objective is met here by performing case studies on two projects. First a broad band evaluation is made of a variety of techniques applied to a high technology space surveillance system (COBRA DANE). Second, the results obtained in applying a very powerful development and management tool (the Air Force Satellite Control Facility's COMPOOL-Sensitive System) to a very complex and difficult software integration task are described. These are two entirely different levels of detail in assessing MPPs but both serve to illustrate the close relationship between applied software development technology and proper software project management. Both are essential if goals of increased programming productivity and improved software quality are to be achieved.


Key words: Measurement, Computer Program Reliability, Computer Program Verification, Computer Program Documentation, Computer Programs, Test and Evaluation, Test Methods

As part of its program to improve software quality and reliability through the implementation of advanced technology, Rome Air Development Center has contracted with General Research Corporation for assistance in using GRC's JOVIAL Automated Verification System (JAVS) to structurally test a large and complex operational program not designed for automated testing. While performing the systematic test, JAVS was evaluated and enhanced, and an applicable testing methodology was developed. This report describes the results of the systematic test, evaluation of JAVS, testing methodology current configuration of JAVS, and proposed capabilities of future tools. JAVS is
operational on the HIS 6180 at RADC operating under GCOS, the HIS 6080 at Offutt AFB, Omaha, Nebraska under WWMCCS, and the CDC 6400 at GRC in Santa Barbara, California under the GOLETA Operating System. (Author)


Key words: Measurement, Computer Programming, Computer Program Reliability, Error Analysis, Linear Regression Analyses, Computer Programs, Mathematical Prediction, Command and Control Systems, Low Costs, Programmers, Computer Program Documentation

This report presents and discusses the results obtained for statistical predictions of programming errors using multiple linear regression analysis. Programming errors were predicted from linear combinations of program characteristics and programmer variables. Each of the program characteristics variables were considered to be measures of the program's complexity and structure. Two distinct data samples comprising 783 programs with approximately 297,000 source instructions written for command and control software applications were analyzed. Background data on both samples is provided which includes discussions related to each sample's software development environment, testing conditions, predictor variables, definition of programming errors, and general data characteristics. Equations obtained and a discussion of the predictability of errors and error rate in each sample. Conclusion of the study and recommendations for further research are also provided. (Author)


Key words: Measurement, Telecommunication, Communication Equipment, Communications Networks, Performance, Assessment

The SYPAC report is the result of nine years of research on system performance assessment and control, buttressed by field measurements, operational evaluations, and special studies. The resultant output is a System Performance Assessment and Control (SYPAC) concept suitable for implementation in the Defense Communications System (DCS). Specific examples of the problems faced in a wide flung communication systems such as the DCS are given along with some of the procedures to identify, isolate, and quantize the difficulty. The use of Automated Tech Control (ATEC) hardware is demonstrated to portray the capability offered by an automated sensing assembly and selected printouts are provided to clarify these examples. The rationale is given for the approach selected and the impact is presented on the operational, organizational, analysis, software, hardware, and order-wire activities of the Air Force and DOD. The SYPAC report (and this Executive Summary) is written in non-scientific terms so that high level managers as well as working level programmers and staff agencies can quickly grasp the approach and intent.


Key words: Measurement, Computer Programming, Standards, Computer Program Documentation, Computer Program Reliability, Algorithms, Costs, Quality Control, Cost Effectiveness, Digital Computers, Computer Programs, Subroutines

This document presents the results of an evaluation of the impact of modern programming practices (MPP) applied to TRW's Ballistic Missile Defense (BMD) Systems Technology Program (STP) Software development. The report provides an overview of STP including a reconstructed chronology of significant events and a description of the STP software development environment. It also presents detailed results of: identification, selection and definition of appropriate STP practices to be evaluated; investigation of the impact of selected MPP on software development cost, schedule and quality; as well as definition and preliminary application of techniques and tools comprising methodology for comparing TRW practices with those used by other contractors on other projects. (Author)
This technical report and report summary show that for the surveyed program developments: (a) Top-down program development reduced cost by 15%. (b) Regardless of the method for development used labor was generally distributed as follows: (1) 10% for analysis, (2) 30% for design, (3) 35% for coding and debugging, (4) 25% for testing. (c) Program testing was automated using real-time on-line simulation, scenario control, data recording and reduction. (d) Software quality assurance and configuration management were formal disciplines. (e) Through system design, operating system software provided real-time on-line recovery of system functions from hardware failures. (f) Through system design, operating system software allowed for on-line maintenance testing of hardware equipments while continuing real-time operation. (g) Software tools found effective on the surveyed programs were described and evaluated.

This report presents the progress of Polytechnic Institute of New York research in the area of software reliability. Subjects of continuing investigation are as follows: Work in progress includes development of a micro reliability model, incorporating representative features of the internal program structure, involving path (module) traversed frequencies and times and path failure probabilities parameters; extension of software physics formulas to define a measure of complexity; planning and execution of small scale tests for gathering of parameters to verify modeling techniques; continuation of work on automatic and modular techniques for constructing low-cost, low-error content application programs; measurement of program length and complexity based on the application statistical natural language theory - (Zipf's Laws on work probabilities); algorithms for the enumeration of feasible program test paths; and development of driver programs for testing every program path at least once.

This report deals with an effort to develop an interactive computer model for analysis and design of digital communications systems. In this report concepts are formulated and described for a Model for Interactive design and Analysis of Communication Systems (MIDACS). Methods of approach for concept formulation are described, examples of previous modeling approaches are presented, characteristics of the MIDACS concept are documented, and a representative computer model is described. Various interactive approaches are shown along with general hardware and software requirements. The formulated interactive concept includes the computer, display devices, and other input/output devices. Mathematical modeling approaches are described including standard Monte Carlo techniques, deterministic methods, and analytical models. The recommended modeling structure includes a modified Monte Carlo technique to minimize computer time for certain communication system performance measures. A multi-level analysis structure is recommended which includes a hierarchy for the physical model combined with a hierarchy of analysis levels which permits greatest computer use flexibility. The recommended programming structure is described including the development chain, software composition, and approach to software development. Function and system library elements are identified and MIDACS hardware requirements are stated.

Key words: Measurement, Computer Program Documentation, Computer Programs, Programming Languages, Subroutines, Algorithms, Models, Errors, Flow Charting

This report presents the progress of this research. Subjects of investigation were error generation and seeding/tagging models, measures for the evaluation of software, analytical data selection methods for program testing, modular programming techniques, methods for finding feasible program paths, statistical program testing and proving, and methods for automatically testing every program path. Work has been completed on development of error generation/manpower deployment models to describe the error correction process in terms of error generation and correction rates as well as the number of man-months spent in debugging. Work has also been completed in the development of seeding/tagging techniques to estimate the number of software errors and related statistical quantities; measures for comparing programs, such as accessibility, testability, and testedness, based on defining a program as a set of executable modules; and a method for selecting test data sets for a program based on determining the interrelationships among program variables. Work still in progress includes development of techniques to interactively write programs using stored library modules and/or user supplied code, investigation of a satisfactory algorithm to estimate the number of feasible paths in a program, development of a statistical theory for program testing and proving based on using a strategy of both testing and journal proving using Black's model, and implementation of a PL/1 driver to automatically test every possible path and catch errors of any PL/1 program, subject to some minor constraints.


Key words: Measurement, Computer Program Documentation, Computer Programs, Data Bases, Automation, Quality Control, Configurations

The current state of software data collection leaves much to be desired. There is no comprehensive base of software development information, and the data that are gathered lack reliability, accuracy and validity. Data gathered from diverse sources and projects are not comparable due to a lack of standardization in meaning, measuring techniques and collection procedures. One of the chief obstacles to gathering more reliable data are project resistance to managerial control and reluctance to release data that might reflect on project efficiency or reveal proprietary technology. This report reviews the literature dealing with software data collection problems, evaluates current military data collection practices, and surveys briefly some project monitor systems that could aid in the collection of more valid and reliable data. The alternatives available in managing a software data collection operation facility are also examined. Actions leading to objective, reliable and standard measures and for removing some of the subjectivity of measures are considered. No easy or fine solutions are seen, but improvements are attainable to technological and managerial changes and through economic inducements to provide better data. (Author)


Key words: Measurement, Computer Program Documentation, Computer Programs, Computer Program Reliability, Data Bases, Computer Programming, Data Processing, Programming Languages, Costs, Batch Processing, Time Sharing, Life Cycles

Ever increasing costs of software development without a parallel increase in software quality has generated much attention on software reliability, project productivity and the overall problems inherent in the software development process. One of the objectives of the Data Collection Study is to recommend a set of parameters to be collected for the RADC Software Data Repository that will form an historical data base to support research and analyses requirements within RADC. Using past and present data collection systems and the results of a survey of the literature as guides, data parameters from contract award to software system installation are discussed. A classification of data parameters
results from the examination of significant software development factors. The first class consists of project environment factors including contract and customer relations data, organizational and personnel characteristics, hardware and support facilities parameters, and overall attributes of the software product itself, such as size, complexity, etc. The second class of data is project performance information reflecting the amount and quality of work performed for the duration of the project period. Class three data consists of automatically generated product measurements, which demonstrate the structure and behavior of the product through the application of analysis tools and testing procedures. (author)


Key words: Measurement, Communication Equipment, Computerized Simulation, Interactions, Computer Graphics, Time Domain, Models, Computer Programming

This interim report deals with the initial activities under an effort to develop an interactive computer model for analysis and design of digital communications systems. In this report concepts are formulated and described for a model for Interactive Design and Analysis of Communication Systems (MIDACS). Methods of approach for concept formulation are described, examples of previous modeling approaches are presented, and characteristics of the MIDACS concept are documented. Various interactive approaches are shown along with general hardware and software requirements. The formulated interactive concept includes the computer, display devices, and other input/output devices. Mathematical modeling approaches are described including standard Monte Carlo techniques, deterministic methods, and analytical models. The recommended modeling structure includes a modified Monte Carlo technique to minimize computer time for certain communication system performance measures. A multi-level analysis structure is proposed which includes a hierarchy for the physical model combined with a hierarchy of analysis levels which permits greatest computer use flexibility. (Author)


Key words: Measurement, Computer Programming, Reliability, Mathematical Models, Tests

During the period of time of 1 Jul 75 to 31 Dec 75, Polytechnic Institute of New York conducted research in the area of software reliability. This report presents the progress of this research. Subjects of investigation were prediction models for the number of errors in computer software, theoretical models for software testing, automatic programming, measures for the evaluation of software, probabilistic models of program testing, and estimation of the initial number of program errors.


Key words: Measurement, Computer Program Documentation, Computer Programs, Reliability, Test Methods, Specifications, Flow Charting, Data Bases

This report describes the methodology which underlies and is supported by the Automated Verification System (AVS) which is scheduled for delivery to the Air Force in June 1, 1975. The methodology is tailored to be largely independent of implementation and language. The AVS is intended to reduce the cost of assuring that software systems written in the JOVIAL J3 dialect, are comprehensively tested. The methodology is intended to engineer workable and practical first-level solutions to automating the measurement of computer program testing effectiveness, assistance in manual testcase design and selection, and increased mechanization of certain aspects of software system maintenance. (Author)


Key words: Measurement, Computer Programming, Errors, Classification
This report presents preliminary results of a study in the area of error classification. A general method of error classification is described which is designed to serve as a guideline for experiment-specific application. A survey of error classification and analysis work, both in the general literature and at MITRE, as well as a study of error experiment design considerations, are reflected in the discussion and conclusions.


Key words: Measurement, Computer Programming, Reliability, Mathematical Logic, Algorithms

This paper explores the benefits of structured programming via two approaches. The first approach uses case studies of small problems to illustrate, in a qualitative way, the benefits as well as some problems associated with the application of these principles. The second approach presented in the Appendix, proposes a method for quantitatively measuring the effect of structured programming through controlled experiments.


Key words: Measurement, Programming Languages, Computer Programming, Reliability

This report is concerned with the enormous productivity that has been an observed concomitant of LISP programming. Section I suggests a possible course of study directed towards defining the significance and bounds of applicability of a number of salient features of this language. Section II is a sample in-depth study, within the scheme of Section I, of a particular LISP feature, recursion.


Key words: Measurement, Computer Programming, Microprogramming, Management, Interfaces

This report defines the problem addressed by software-first, and gives a definition of software-first as a design methodology for dealing with the interfaces among components in the design of complex systems. Hardware and software engineering technology which could support software-first design activities is reviewed. A final section addresses the role a software-first machine might play in Air Force systems development.


Key words: Measurement, Computer Programming, Management, Quality Control

This paper examines, from a managerial viewpoint, the effects of programming group organization on both the people in the group and on the software thus produced. In particular, Weinberg's suggestions of teams are discussed.


Key words: Measurement, Computer Programming, Reliability, Data Management, Validation, Costs

The purpose of this effort was to investigate and develop techniques to reduce the procurement cost and increase the reliability and timeliness of complex computer software. Specific goals of the program were to study and define a Software-First Machine (SOFM), determine to what extent the technique of Structured Programming is beneficial in the testing of software and contributes to the reliability of software, and provide the Air Force with initial guidance in the area of software reliability. This report
presents an overview of the technological background common to the six tasks of project 522A, a part of MITRE Project 5220. Besides discussing general background, this volume provides an introduction to each of the other seven volumes of the entire report.


Key words: Measurement, Computer Programming, Compilers, Computer Graphics

This report describes a system developed to edit cartographic data in a raster format. The Raster Edit System consists of interfaced PDP-9/PDP-15 computers, disc storage, and 2 CRT devices. Cartographic manuscripts in digital raster form may be displayed on one CRT for selection of an area to be shown at increased scale on the other. The data displayed may then be examined and altered. The Raster Edit System can be used to great advantage to edit data in raster plotter format and to examine raster scanner data for quality. In the near future it is expected to be an important factor in the efficient use of the raster to lineal conversion process. (Author)
3. TITLE KEYWORDS IN CONTEXT

216 ACQUISITION MANAGER *** FACTORS IN SOFTWARE QUALITY. PRELIMINARY HANDBOOK ON SOFTWARE QUALITY FOR AN

15401 ACQUISITION MANAGER *** FACTORS IN SOFTWARE QUALITY. VOLUME-III. PRELIMINARY HANDBOOK ON SOFTWARE QUALITY FOR AN

15357 ADA SYSTEMS *** KNOWLEDGE-BASED PROJECT MANAGEMENT ASSISTANT FOR
15384 ADA TEST AND VERIFICATION SYSTEM (ATVS) ***
15356 ADA TEST AND VERIFICATION SYSTEM (ATVS) IMPLEMENTATION ***
15406 AIR FORCE PROJECTS *** AN INVESTIGATION OF PROGRAMMING PRACTICES IN SELECTED

4233 ALGORITHMIC COMPLEXITY ***
4234 ALGORITHMIC COMPLEXITY ***
15375 ALGORITHMIC COMPLEXITY. VOLUME I ***
15374 ALGORITHMIC COMPLEXITY. VOLUME II ***
15371 AMS). *** AUTOMATED MEASUREMENT SYSTEM ()

15343 ANALYSIS AND EVALUATION OF CODING TECHNIQUES. *** CODING FOR TURBULENT CHANNELS. VOLUME III. TURBULENT CHANNEL DATA
15376 ANALYSIS APPLICATION GUIDELINES *** SNEAK
15370 ANALYSIS CENTER FOR SOFTWARE: AN IAC IN TRANSITION *** DATA AND
1496 ANALYSIS FOR LARGE-SCALE SOFTWARE MAINTENANCE *** PERFORMANCE RIPPLE EFFECT
15388 ANALYSIS OF DISCRETE SOFTWARE RELIABILITY MODELS ***
15415 ANALYSIS OF SOFTWARE DATA COLLECTION PROBLEMS AND CURRENT CAPABILITIES *** SOFTWARE DATA COLLECTION STUDY. VOLUME II. AN
15424 ANALYSIS OF THE LISP PROGRAMMING LANGUAGE) *** ENGINEERING OF QUALITY SOFTWARE SYSTEMS (TOWARDS AN

1430 ANALYZER *** STATIC FORTRAN
15376 APPLICATION GUIDELINES *** SNEAK ANALYSIS
15352 ARTIFICIAL INTELLIGENCE SOFTWARE *** ENGINEERING FOR
15353 ASQS) OPERATIONAL CONCEPT DOCUMENT. VOLUME 1 *** THE ASSISTANT FOR SPECIFYING THE QUALITY SOFTWARE ()
3982 ASSESSMENT *** A GUIDEBOOK FOR SOFTWARE RELIABILITY
15409 ASSESSMENT AND CONTROL CONCEPT *** SYSTEM PERFORMANCE
15357 ASSISTANT FOR ADA SYSTEMS *** KNOWLEDGE-BASED PROJECT MANAGEMENT
15353 ASSISTANT FOR SPECIFYING THE QUALITY SOFTWARE (ASQS) OPERATIONAL CONCEPT DOCUMENT. VOLUME 1 *** THE
15379 ASSURANCE METHODS *** IMPROVING SOFTWARE QUALITY
15354 ASSURANCE TOOLS *** KNOWLEDGE BASED QUALITY
2040 ASTROS PLAN *** A MATCHED PROJECT EVALUATION OF MODERN PROGRAMMING PRACTICES. MANAGEMENT REPORT ON THE
ASTROS PLAN *** A MATCHED PROJECT EVALUATION OF MODERN PROGRAMMING PRACTICES. SCIENTIFIC REPORT ON THE

ATTRIBUTES *** SPECIFICATION OF SOFTWARE QUALITY

ATTRIBUTES SOFTWARE QUALITY EVALUATION GUIDEBOOK *** SPECIFICATION OF SOFTWARE QUALITY

ATTRIBUTES SOFTWARE QUALITY SPECIFICATION GUIDEBOOK *** SPECIFICATION OF SOFTWARE QUALITY

ATVS) *** ADA TEST AND VERIFICATION SYSTEM (**

ATVS) IMPLEMENTATION *** ADA TEST AND VERIFICATION SYSTEM (**

AUTOMATED MEASUREMENT SYSTEM (AMS). ***

AUTOMATED VERIFICATION SYSTEM *** JOVIAL

AUTOMATED VERIFICATION SYSTEM (FAVS). VOLUME I *** FORTRAN

AUTOMATED VERIFICATION SYSTEM (FAVS). VOLUME II. USER'S MANUAL *** FORTRAN

AUTOMATED VERIFICATION SYSTEM: STUDY PHASE *** COBOL

AUTOMATING SOFTWARE DESIGN METRICS ***

AUTOMATION OF QUALITY MEASUREMENT ***

BAYESIAN SOFTWARE PREDICTION MODELS. VOLUME I. AN IMPERFECT DEBUGGING MODEL FOR RELIABILITY AND OTHER QUANTITATIVE MEASURES OF SOFTWARE SYSTEMS ***

BAYESIAN SOFTWARE PREDICTION MODELS. VOLUME III ***

CAPABILITIES *** SOFTWARE DATA COLLECTION STUDY. VOLUME II. AN ANALYSIS OF SOFTWARE DATA COLLECTION PROBLEMS AND CURRENT

CASE STUDIES IN STRUCTURED PROGRAMMING) *** ENGINEERING OF QUALITY SOFTWARE SYSTEMS (SOME

CENTER FOR SOFTWARE: AN IAC IN TRANSITION *** DATA AND ANALYSIS

CHANNEL DATA ANALYSIS AND EVALUATION OF CODING TECHNIQUES. *** CODING FOR TURBULENT CHANNELS. VOLUME III. TURBULENT

CHANNELS. VOLUME III. TURBULENT CHANNEL DATA ANALYSIS AND EVALUATION OF CODING TECHNIQUES. *** CODING FOR TURBULENT

CLASSIFICATION METHODOLOGY) *** ENGINEERING OF QUALITY SOFTWARE SYSTEMS (A SOFTWARE ERROR

COBOL AUTOMATED VERIFICATION SYSTEM: STUDY PHASE ***

CODING FOR TURBULENT CHANNELS. VOLUME III. TURBULENT CHANNEL DATA ANALYSIS AND EVALUATION OF CODING TECHNIQUES. ***

CODING TECHNIQUES. *** CODING FOR TURBULENT CHANNELS. VOLUME III. TURBULENT CHANNEL DATA ANALYSIS AND EVALUATION OF

COLLECTION AND VALIDATION *** FACTORS IN SOFTWARE QUALITY. METRIC DATA

COLLECTION PROBLEMS AND CURRENT CAPABILITIES *** SOFTWARE DATA COLLECTION STUDY. VOLUME II. AN ANALYSIS OF SOFTWARE DATA

COLLECTION STUDY. VOLUME II. AN ANALYSIS OF SOFTWARE DATA COLLECTION PROBLEMS AND CURRENT CAPABILITIES *** SOFTWARE DATA
15416 COLLECTION STUDY. VOLUME III. DATA REQUIREMENTS FOR PRODUCTIVITY AND RELIABILITY STUDIES *** SOFTWARE DATA
15380 COMBINED HW/SW RELIABILITY MODELS ***
15413 COMMUNICATION SYSTEMS MODELING STUDY *** INTERACTIVE
15417 COMMUNICATION SYSTEMS MODELING STUDY *** INTERACTIVE
2052 COMMUNICATIONS PROCESSOR OPERATING SYSTEM TASK 2 - RELIABILITY CONSIDERATIONS ***
4233 COMPLEXITY *** ALGORITHMIC
4234 COMPLEXITY *** ALGORITHMIC
3850 COMPLEXITY *** SOFTWARE MODELING STUDIES - THE POLYNOMIAL MEASURE OF
160 COMPLEXITY OF COMPUTER SOFTWARE) *** ENGINEERING OF QUALITY SOFTWARE SYSTEMS (MEASURING THE
15375 COMPLEXITY. VOLUME I *** ALGORITHMIC
15374 COMPLEXITY. VOLUME II *** ALGORITHMIC
15419 COMPREHENSIVE SOFTWARE TESTING *** METHODOLOGY FOR
160 COMPUTER SOFTWARE) *** ENGINEERING OF QUALITY SOFTWARE SYSTEMS (MEASURING THE COMPLEXITY OF
5417 COMPUTING SYSTEMS: IMPACT ON SOFTWARE QUALITY *** SOFTWARE QUALITY MEASUREMENT FOR DISTRIBUTED SYSTEMS - DISTRIBUTED
15409 CONCEPT *** SYSTEM PERFORMANCE ASSESSMENT AND CONTROL
15353 CONCEPT DOCUMENT. VOLUME I *** THE ASSISTANT FOR SPECIFYING THE QUALITY SOFTWARE (ASQS) OPERATIONAL
15425 CONCEPTS) *** ENGINEERING OF QUALITY SOFTWARE SYSTEMS (SOFTWARE FIRST
214 CONCEPTS AND DEFINITIONS OF SOFTWARE QUALITY *** FACTORS IN SOFTWARE QUALITY,
2052 CONSIDERATIONS *** COMMUNICATIONS PROCESSOR OPERATING SYSTEM TASK 2 - RELIABILITY
15409 CONTROL CONCEPT *** SYSTEM PERFORMANCE ASSESSMENT AND
15415 CURRENT CAPABILITIES *** SOFTWARE DATA COLLECTION STUDY. VOLUME II. AN ANALYSIS OF SOFTWARE DATA COLLECTION PROBLEMS AND
15390 CYCLE DATA FROM THE PAVE PAWS PROJECT. *** EVALUATION OF SOFTWARE LIFE
15343 DATA ANALYSIS AND EVALUATION OF CODING TECHNIQUES. *** CODING FOR TURBULENT CHANNELS. VOLUME III. TURBULENT CHANNEL
15370 DATA AND ANALYSIS CENTER FOR SOFTWARE: AN IAC IN TRANSITION ***
215 DATA COLLECTION AND VALIDATION *** FACTORS IN SOFTWARE QUALITY. METRIC
15415 DATA COLLECTION PROBLEMS AND CURRENT CAPABILITIES *** SOFTWARE DATA COLLECTION STUDY. VOLUME II. AN ANALYSIS OF SOFTWARE
15415 DATA COLLECTION STUDY. VOLUME II. AN ANALYSIS OF SOFTWARE DATA COLLECTION PROBLEMS AND CURRENT CAPABILITIES *** SOFTWARE
15416 DATA COLLECTION STUDY. VOLUME III. DATA REQUIREMENTS FOR PRODUCTIVITY AND RELIABILITY STUDIES *** SOFTWARE

33
DATA FROM THE PAVE PAWS PROJECT *** EVALUATION OF SOFTWARE LIFE CYCLE
DATA MANAGEMENT SYSTEM TEST METHODOLOGY DEVELOPMENT ***
DATA MANAGEMENT SYSTEM TESTING AND METHODOLOGY VALIDATION ***
DATA REPOSITORY STUDY *** SOFTWARE
DATA REQUIREMENTS FOR PRODUCTIVITY AND RELIABILITY STUDIES *** SOFTWARE DATA COLLECTION STUDY. VOLUME III.
DEBUGGING MODEL FOR RELIABILITY AND OTHER QUANTITATIVE MEASURES OF SOFTWARE SYSTEMS *** BAYESIAN SOFTWARE PREDICTION MODELS. VOLUME I. AN IMPERFECT
DEFINITIONS OF SOFTWARE QUALITY *** FACTORS IN SOFTWARE QUALITY. CONCEPTS AND
DEMONSTRATION PROJECT I *** SOFTWARE QUALITY MEASUREMENT
DEMONSTRATION PROJECT II *** SOFTWARE QUALITY MEASUREMENT
DEMONSTRATION STUDY. VOLUME 1 *** RELIABILITY MODEL
DEMONSTRATION STUDY. VOLUME 2 *** RELIABILITY MODEL
DESIGN METRICS *** AUTOMATING SOFTWARE
DESIGN SYSTEM *** SOFTWARE FAULT TOLERANCE
DETERMINING SOFTWARE QUALITY *** PATTERN RECOGNITION METHODS FOR
DEVELOPMENT *** DATA MANAGEMENT SYSTEM TEST METHODOLOGY
DEVELOPMENT *** IMPACT OF MPP ON SYSTEM
DEVELOPMENTS IN THE ECL PROGRAMMING SYSTEM *** RECENT
DIMENSIONAL LANGUAGE VERSUS FORTRAN FOR FIRST-COURSE PROGRAMMERS *** SOFTWARE MODELING STUDIES EXPERIMENTAL STUDY OF A TWO
DISCRETE SOFTWARE RELIABILITY MODELS *** ANALYSIS OF
DISTRIBUTED COMPUTING SYSTEMS: IMPACT ON SOFTWARE QUALITY *** SOFTWARE QUALITY MEASUREMENT FOR DISTRIBUTED SYSTEMS -
DISTRIBUTED SYSTEMS *** FAULT TOLERANCE, RELIABILITY AND TESTABILITY FOR
DISTRIBUTED SYSTEMS *** SOFTWARE QUALITY MEASUREMENT FOR
DISTRIBUTED SYSTEMS - DISTRIBUTED COMPUTING SYSTEMS: IMPACT ON SOFTWARE QUALITY *** SOFTWARE QUALITY MEASUREMENT FOR
DISTRIBUTED SYSTEMS GUIDEBOOK FOR SOFTWARE QUALITY *** SOFTWARE QUALITY MEASUREMENT FOR
DMA MODERN PROGRAMMING ENVIRONMENT STUDY ***
DOCUMENT. VOLUME 1 *** THE ASSISTANT FOR SPECIFYING THE QUALITY SOFTWARE (ASQS) OPERATIONAL CONCEPT
ECL PROGRAMMING SYSTEM *** RECENT DEVELOPMENTS IN THE
EDITING SOFTWARE *** RASTER-LINEAL
EFFECTS OF MANAGEMENT PHILOSOPHY ON SOFTWARE PRODUCTION) *** ENGINEERING OF QUALITY SOFTWARE SYSTEMS (ENGINEERING FOR ARTIFICIAL INTELLIGENCE SOFTWARE ***

34
| 15427 | ENGINEERING OF QUALITY SOFTWARE SYSTEMS *** |
| 15421 | ENGINEERING OF QUALITY SOFTWARE SYSTEMS (A SOFTWARE ERROR CLASSIFICATION METHODOLOGY) *** |
| 15426 | ENGINEERING OF QUALITY SOFTWARE SYSTEMS (EFFECTS OF MANAGEMENT PHILOSOPHY ON SOFTWARE PRODUCTION) *** |
| 160  | ENGINEERING OF QUALITY SOFTWARE SYSTEMS (MEASURING THE COMPLEXITY OF COMPUTER SOFTWARE) *** |
| 15425 | ENGINEERING OF QUALITY SOFTWARE SYSTEMS (SOFTWARE FIRST CONCEPTS) *** |
| 965  | ENGINEERING OF QUALITY SOFTWARE SYSTEMS (SOFTWARE RELIABILITY MODELING AND MEASUREMENT TECHNIQUES) *** |
| 15422 | ENGINEERING OF QUALITY SOFTWARE SYSTEMS (SOME CASE STUDIES IN STRUCTURED PROGRAMMING) *** |
| 15424 | ENGINEERING OF QUALITY SOFTWARE SYSTEMS (TOWARDS AN ANALYSIS OF THE LISP PROGRAMMING LANGUAGE) *** |
| 2192 | ENHANCEMENTS *** SOFTWARE QUALITY METRICS |
| 12815 | ENHANCEMENTS STUDY RESULTS *** SOFTWARE QUALITY MEASUREMENT METHODOLOGY |
| 15378 | ENVIRONMENT MEASUREMENT INSTRUMENTATION *** C3I |
| 15391 | ENVIRONMENT STUDY *** DMA MODERN PROGRAMMING |
| 15421 | ERROR CLASSIFICATION METHODOLOGY) *** ENGINEERING OF QUALITY SOFTWARE SYSTEMS (A SOFTWARE |
| 15408 | ERRORS *** STATISTICAL PREDICTION OF PROGRAMMING |
| 4187  | ESTIMATION TECHNIQUES *** SOFTWARE RELIABILITY MODELING AND |
| 6796  | EVALUATION GUIDEBOOK *** SPECIFICATION OF SOFTWARE QUALITY ATTRIBUTES SOFTWARE QUALITY |
| 15343 | EVALUATION OF CODING TECHNIQUES. *** CODING FOR TURBULENT CHANNELS. VOLUME III. TURBULENT CHANNEL DATA ANALYSIS AND |
| 2040  | EVALUATION OF MODERN PROGRAMMING PRACTICES. MANAGEMENT REPORT ON THE ASTROS PLAN *** A MATCHED PROJECT |
| 2041  | EVALUATION OF MODERN PROGRAMMING PRACTICES. SCIENTIFIC REPORT ON THE ASTROS PLAN *** A MATCHED PROJECT |
| 15390 | EVALUATION OF SOFTWARE LIFE CYCLE DATA FROM THE PAVE PAWS PROJECT. *** |
| 13923 | EVALUATION SYSTEM (QUES). VOLUME 1. *** QUALITY |
| 3849  | EXECUTIVE SUMMARY *** SOFTWARE MODELING STUDIES. |
| 8544  | EXPERIMENTAL INVESTIGATION INTO SOFTWARE RELIABILITY *** AN |
| 15383 | EXPERIMENTAL STUDY OF A TWO DIMENSIONAL LANGUAGE VERSUS FORTRAN FOR FIRST-COURSE PROGRAMMERS *** SOFTWARE MODELING STUDIES |
| 214   | FACTORS IN SOFTWARE QUALITY. CONCEPTS AND DEFINITIONS OF SOFTWARE QUALITY *** |
| 215   | FACTORS IN SOFTWARE QUALITY. METRIC DATA COLLECTION AND VALIDATION *** |
| 216   | FACTORS IN SOFTWARE QUALITY. PRELIMINARY HANDBOOK ON SOFTWARE QUALITY FOR AN ACQUISITION MANAGER *** |
| 15401 | FACTORS IN SOFTWARE QUALITY. VOLUME-III. PRELIMINARY HANDBOOK ON SOFTWARE QUALITY FOR AN ACQUISITION MANAGER *** |
15365 FAULT TOLERANCE DESIGN SYSTEM *** SOFTWARE
15085 FAULT TOLERANCE, RELIABILITY AND TESTABILITY FOR DISTRIBUTED SYSTEMS ***
15395 FAVS). VOLUME I *** FORTRAN AUTOMATED VERIFICATION SYSTEM (FAVS). VOLUME I
15394 FAVS). VOLUME II. USER'S MANUAL *** FORTRAN AUTOMATED VERIFICATION SYSTEM (FAVS). VOLUME II
15425 FIRST CONCEPTS *** ENGINEERING OF QUALITY SOFTWARE SYSTEMS (SOFTWARE FIRST-CONCEPTS)
15385 FORTRAN FOR FIRST-COURSE PROGRAMMERS *** SOFTWARE MODELING STUDIES EXPERIMENTAL STUDY OF A TWO DIMENSIONAL LANGUAGE VERSUS FORTRAN FOR
15406 FORCE PROJECTS *** AN INVESTIGATION OF PROGRAMMING PRACTICES IN SELECTED AIR
1430 FORTRAN ANALYZER *** STATIC
15395 FORTRAN AUTOMATED VERIFICATION SYSTEM (FAVS). VOLUME I ***
15394 FORTRAN AUTOMATED VERIFICATION SYSTEM (FAVS). VOLUME II USER'S MANUAL ***
15385 FORTRAN FOR FIRST-COURSE PROGRAMMERS *** SOFTWARE MODELING STUDIES EXPERIMENTAL STUDY OF A TWO DIMENSIONAL LANGUAGE VERSUS FORTRAN FOR
6796 GUIDEBOOK *** SPECIFICATION OF SOFTWARE QUALITY ATTRIBUTES SOFTWARE QUALITY EVALUATION
6795 GUIDEBOOK *** SPECIFICATION OF SOFTWARE QUALITY ATTRIBUTES SOFTWARE QUALITY SPECIFICATION
4394 GUIDEBOOK FOR SOFTWARE QUALITY *** SOFTWARE QUALITY MEASUREMENT FOR DISTRIBUTED SYSTEMS
3982 GUIDEBOOK FOR SOFTWARE RELIABILITY ASSESSMENT *** A
15376 GUIDELINES *** SNEAK ANALYSIS APPLICATION
216 HANDBOOK ON SOFTWARE QUALITY FOR AN ACQUISITION MANAGER *** FACTORS IN SOFTWARE QUALITY. PRELIMINARY
15401 HANDBOOK ON SOFTWARE QUALITY FOR AN ACQUISITION MANAGER *** FACTORS IN SOFTWARE QUALITY. VOLUME-III. PRELIMINARY
15380 HW/SW RELIABILITY MODELS *** COMBINED
15370 IAC IN TRANSITION *** DATA AND ANALYSIS CENTER FOR SOFTWARE: AN
15410 IMPACT OF MPP ON SYSTEM DEVELOPMENT ***
5417 IMPACT ON SOFTWARE QUALITY *** SOFTWARE QUALITY MEASUREMENT FOR DISTRIBUTED SYSTEMS - DISTRIBUTED COMPUTING SYSTEMS:
15398 IMPERFECT DEBUGGING MODEL FOR RELIABILITY AND OTHER QUANTITATIVE MEASURES OF SOFTWARE SYSTEMS *** BAYESIAN SOFTWARE PREDICTION MODELS. VOLUME I. AN
15356 IMPLEMENTATION *** ADA TEST AND VERIFICATION SYSTEM (ATVS)
15382 IMPLEMENTATION *** NATIONAL SOFTWARE WORKS TOOL
15379 IMPROVING SOFTWARE QUALITY ASSURANCE METHODS ***
15378 INSTRUMENTATION *** C3I ENVIRONMENT MEASUREMENT
15381 INTEGRATION STUDIES *** NATIONAL SOFTWARE WORKS TOOL

36
15352 INTELLIGENCE SOFTWARE *** ENGINEERING FOR ARTIFICIAL

15413 INTERACTIVE COMMUNICATION SYSTEMS MODELING STUDY ***

15417 INTERACTIVE COMMUNICATION SYSTEMS MODELING STUDY ***

4383 INTEROPERABILITY AND REUSABILITY *** SOFTWARE

4384 INTEROPERABILITY AND REUSABILITY *** SOFTWARE

8544 INVESTIGATION INTO SOFTWARE RELIABILITY *** AN EXPERIMENTAL

15406 INVESTIGATION OF PROGRAMMING PRACTICES IN SELECTED AIR FORCE PROJECTS *** AN

969 INVESTIGATION OF SOFTWARE MODELS *** SUMMARY OF TECHNICAL PROGRESS.

15392 INVESTIGATION OF SOFTWARE MODELS *** SUMMARY OF TECHNICAL PROGRESS.

15407 JAVS FINAL REPORT ***

15396 JOVIAL AUTOMATED VERIFICATION SYSTEM ***

15354 KNOWLEDGE BASED QUALITY ASSURANCE TOOLS ***

15357 KNOWLEDGE-BASED PROJECT MANAGEMENT ASSISTANT FOR ADA SYSTEMS ***

15424 LANGUAGE) *** ENGINEERING OF QUALITY SOFTWARE SYSTEMS (TOWARDS AN

ANALYSIS OF THE LISP PROGRAMMING

15383 LANGUAGE VERSUS FORTRAN FOR FIRST-COURSE PROGRAMMERS *** SOFTWARE

MODELING STUDIES EXPERIMENTAL STUDY OF A TWO DIMENSIONAL

1496 LARGE-SCALE SOFTWARE MAINTENANCE *** PERFORMANCE RIPPLE EFFECT ANALYSIS FOR

15390 LIFE CYCLE DATA FROM THE PAVE PAWS PROJECT. *** EVALUATION OF SOFTWARE

15424 LISP PROGRAMMING LANGUAGE) *** ENGINEERING OF QUALITY SOFTWARE SYSTEMS

(TOWARDS AN ANALYSIS OF THE

15372 MAINTENANCE *** ADVANCED TOOLS FOR SOFTWARE

4519 MAINTENANCE *** METHODOLOGY FOR SOFTWARE

1496 MAINTENANCE *** PERFORMANCE RIPPLE EFFECT ANALYSIS FOR LARGE-SCALE

SOFTWARE

15357 MANAGEMENT ASSISTANT FOR ADA SYSTEMS *** KNOWLEDGE-BASED PROJECT

15426 MANAGEMENT PHILOSOPHY ON SOFTWARE PRODUCTION) *** ENGINEERING OF

QUALITY SOFTWARE SYSTEMS (EFFECTS OF

2040 MANAGEMENT REPORT ON THE ASTROS PLAN *** A MATCHED PROJECT EVALUATION

OF MODERN PROGRAMMING PRACTICES.

15344 MANAGEMENT SYSTEM TEST METHODOLOGY DEVELOPMENT *** DATA

15345 MANAGEMENT SYSTEM TESTING AND METHODOLOGY VALIDATION *** DATA

216 MANAGER *** FACTORS IN SOFTWARE QUALITY. PRELIMINARY HANDBOOK ON

SOFTWARE QUALITY FOR AN ACQUISITION

15401 MANAGER *** FACTORS IN SOFTWARE QUALITY. VOLUME-III. PRELIMINARY

HANDBOOK ON SOFTWARE QUALITY FOR AN ACQUISITION

15394 MANUAL *** FORTRAN AUTOMATED VERIFICATION SYSTEM (FAVS). VOLUME II. USER'S

2193 MANUAL *** SOFTWARE QUALITY MEASUREMENT

37
MATCHED PROJECT EVALUATION OF MODERN PROGRAMMING PRACTICES. MANAGEMENT REPORT ON THE ASTROS PLAN *** A

MATCHED PROJECT EVALUATION OF MODERN PROGRAMMING PRACTICES. SCIENTIFIC REPORT ON THE ASTROS PLAN *** A

MEASURE OF COMPLEXITY *** SOFTWARE MODELING STUDIES - THE POLYNOMIAL

MEASUREMENT *** AUTOMATION OF QUALITY

MEASUREMENT DEMONSTRATION PROJECT (I) *** SOFTWARE QUALITY

MEASUREMENT DEMONSTRATION PROJECT II *** SOFTWARE QUALITY

MEASUREMENT FOR DISTRIBUTED SYSTEMS *** SOFTWARE QUALITY

MEASUREMENT FOR DISTRIBUTED SYSTEMS - DISTRIBUTED COMPUTING SYSTEMS: IMPACT ON SOFTWARE QUALITY *** SOFTWARE QUALITY

MEASUREMENT FOR DISTRIBUTED SYSTEMS GUIDEBOOK FOR SOFTWARE QUALITY *** SOFTWARE QUALITY

MEASUREMENT INSTRUMENTATION *** C3I ENVIRONMENT

MEASUREMENT MANUAL *** SOFTWARE QUALITY

MEASUREMENT METHODOLOGY ENHANCEMENTS STUDY RESULTS *** SOFTWARE QUALITY

MEASUREMENT SYSTEM (AMS). *** AUTOMATED

MEASUREMENT TECHNIQUES) *** ENGINEERING OF QUALITY SOFTWARE SYSTEMS (SOFTWARE RELIABILITY MODELING AND

MEASURES *** SOFTWARE TESTING

MEASURES OF SOFTWARE SYSTEMS *** BAYESIAN SOFTWARE PREDICTION MODELS. VOLUME I. AN IMPERFECT DEBUGGING MODEL FOR RELIABILITY AND OTHER QUANTITATIVE

MEASURING THE COMPLEXITY OF COMPUTER SOFTWARE) *** ENGINEERING OF QUALITY SOFTWARE SYSTEMS ( SOFTWARE RELIABILITY MODELING AND

METHODOLOGY) *** ENGINEERING OF QUALITY SOFTWARE SYSTEMS (A SOFTWARE ERROR CLASSIFICATION

METHODOLOGY DEVELOPMENT *** DATA MANAGEMENT SYSTEM TEST

METHODOLOGY ENHANCEMENTS STUDY RESULTS *** SOFTWARE QUALITY MEASUREMENT

METHODOLOGY FOR COMPREHENSIVE SOFTWARE TESTING ***

METHODOLOGY FOR SOFTWARE MAINTENANCE ***

METHODOLOGY FOR SOFTWARE RELIABILITY PREDICTION ***

METHODOLOGY FOR SOFTWARE RELIABILITY PREDICTION ***

METHODOLOGY VALIDATION *** DATA MANAGEMENT SYSTEM TESTING AND

METHODS *** IMPROVING SOFTWARE QUALITY ASSURANCE

METHODS FOR DETERMINING SOFTWARE QUALITY *** PATTERN RECOGNITION

METRIC DATA COLLECTION AND VALIDATION *** FACTORS IN SOFTWARE QUALITY.

METRICS *** AUTOMATING SOFTWARE DESIGN

METRICS ENHANCEMENTS *** SOFTWARE QUALITY

MODEL DEMONSTRATION STUDY. VOLUME I *** RELIABILITY
PAWS PROJECT. *** EVALUATION OF SOFTWARE LIFE CYCLE DATA FROM THE PAVE
PERFORMANCE ASSESSMENT AND CONTROL CONCEPT *** SYSTEM
PERFORMANCE RIPPLE EFFECT ANALYSIS FOR LARGE-SCALE SOFTWARE MAINTENANCE ***
PHASE *** COBOL AUTOMATED VERIFICATION SYSTEM: STUDY
PHILOSOPHY ON SOFTWARE PRODUCTION) *** ENGINEERING OF QUALITY SOFTWARE SYSTEMS (EFFECTS OF MANAGEMENT
PLAN *** A MATCHED PROJECT EVALUATION OF MODERN PROGRAMMING PRACTICES. MANAGEMENT REPORT ON THE ASTROS
PLAN *** A MATCHED PROJECT EVALUATION OF MODERN PROGRAMMING PRACTICES. SCIENTIFIC REPORT ON THE ASTROS
POLYNOMIAL MEASURE OF COMPLEXITY *** SOFTWARE MODELING STUDIES - THE
PRACTICES IN SELECTED AIR FORCE PROJECTS *** AN INVESTIGATION OF PROGRAMMING
PRACTICES, MANAGEMENT REPORT ON THE ASTROS PLAN *** A MATCHED PROJECT EVALUATION OF MODERN PROGRAMMING
PRACTICES, SCIENTIFIC REPORT ON THE ASTROS PLAN *** A MATCHED PROJECT EVALUATION OF MODERN PROGRAMMING
PRACTICES STUDY REPORT *** MODERN PROGRAMMING
PREDICTION *** METHODOLOGY FOR SOFTWARE RELIABILITY
PREDICTION *** METHODOLOGY FOR SOFTWARE RELIABILITY
PREDICTION MODELS. VOLUME I. AN IMPERFECT DEBUGGING MODEL FOR RELIABILITY AND OTHER QUANTITATIVE MEASURES OF SOFTWARE SYSTEMS *** BAYESIAN SOFTWARE
PREDICTION MODELS. VOLUME III *** BAYESIAN SOFTWARE
PREDICTION OF PROGRAMMING ERRORS *** STATISTICAL
PRELIMINARY HANDBOOK ON SOFTWARE QUALITY FOR AN ACQUISITION MANAGER *** FACTORS IN SOFTWARE QUALITY.
PRELIMINARY HANDBOOK ON SOFTWARE QUALITY FOR AN ACQUISITION MANAGER *** FACTORS IN SOFTWARE QUALITY. VOLUME-III.
PROBLEMS AND CURRENT CAPABILITIES *** SOFTWARE DATA COLLECTION STUDY. VOLUME II. AN ANALYSIS OF SOFTWARE DATA COLLECTION
PROCESSOR OPERATING SYSTEM TASK 2 - RELIABILITY CONSIDERATIONS *** COMMUNICATIONS
PRODUCTION) *** ENGINEERING OF QUALITY SOFTWARE SYSTEMS (EFFECTS OF MANAGEMENT PHILOSOPHY ON SOFTWARE
PRODUCTIVITY AND RELIABILITY STUDIES *** SOFTWARE DATA COLLECTION STUDY. VOLUME III. DATA REQUIREMENTS FOR
PROGRAMMERS *** SOFTWARE MODELING STUDIES EXPERIMENTAL STUDY OF A TWO DIMENSIONAL LANGUAGE VERSUS FORTRAN FOR FIRST-COURSE
PROGRAMMING) *** ENGINEERING OF QUALITY SOFTWARE SYSTEMS (SOME CASE STUDIES IN STRUCTURED
PROGRAMMING ENVIRONMENT STUDY *** DMA MODERN
PROGRAMMING ERRORS *** STATISTICAL PREDICTION OF
PROGRAMMING LANGUAGE) *** ENGINEERING OF QUALITY SOFTWARE SYSTEMS (TOWARDS AN ANALYSIS OF THE LISP
PROGRAMMING PRACTICES IN SELECTED AIR FORCE PROJECTS *** AN INVESTIGATION OF
PROGRAMMING PRACTICES. MANAGEMENT REPORT ON THE ASTROS PLAN *** A MATCHED PROJECT EVALUATION OF MODERN
PROGRAMMING PRACTICES. SCIENTIFIC REPORT ON THE ASTROS PLAN *** A MATCHED PROJECT EVALUATION OF MODERN
PROGRAMMING PRACTICES STUDY REPORT *** MODERN
PROGRAMMING SYSTEM *** RECENT DEVELOPMENTS IN THE ECL
PROGRESS *** SELF-METRIC SOFTWARE. SUMMARY OF TECHNICAL
PROGRESS *** SOFTWARE MODELING STUDIES. VOLUME I. SUMMARY OF TECHNICAL
PROGRESS - SOFTWARE MODELING STUDIES *** SUMMARY OF TECHNICAL
PROGRESS, INVESTIGATION OF SOFTWARE MODELS *** SUMMARY OF TECHNICAL
PROGRESS, INVESTIGATION OF SOFTWARE MODELS *** SUMMARY OF TECHNICAL
PROGRESS, SOFTWARE MODELING STUDIES *** SUMMARY OF TECHNICAL
PROGRESS, SOFTWARE MODELING STUDIES *** SUMMARY OF TECHNICAL
PROGRESS SOFTWARE MODELING STUDIES *** SUMMARY OF TECHNICAL
PROJECT. *** EVALUATION OF SOFTWARE LIFE CYCLE DATA FROM THE PAVE PAWS
PROJECT EVALUATION OF MODERN PROGRAMMING PRACTICES. MANAGEMENT REPORT ON THE ASTROS PLAN *** A MATCHED
PROJECT EVALUATION OF MODERN PROGRAMMING PRACTICES. SCIENTIFIC REPORT ON THE ASTROS PLAN *** A MATCHED
PROJECT (I). *** SOFTWARE QUALITY MEASUREMENT DEMONSTRATION
PROJECT II *** SOFTWARE QUALITY MEASUREMENT DEMONSTRATION
PROJECT MANAGEMENT ASSISTANT FOR ADA SYSTEMS *** KNOWLEDGE-BASED
PROJECTS *** AN INVESTIGATION OF PROGRAMMING PRACTICES IN SELECTED AIR FORCE
QUALITY *** FACTORS IN SOFTWARE QUALITY. CONCEPTS AND DEFINITIONS OF SOFTWARE
QUALITY *** PATTERN RECOGNITION METHODS FOR DETERMINING SOFTWARE
QUALITY *** SOFTWARE QUALITY MEASUREMENT FOR DISTRIBUTED SYSTEMS - DISTRIBUTED COMPUTING SYSTEMS: IMPACT ON SOFTWARE
QUALITY *** SOFTWARE QUALITY MEASUREMENT FOR DISTRIBUTED SYSTEMS GUIDEBOOK FOR SOFTWARE
QUALITY ASSURANCE METHODS *** IMPROVING SOFTWARE
QUALITY ASSURANCE TOOLS *** KNOWLEDGE BASED
QUALITY ATTRIBUTES *** SPECIFICATION OF SOFTWARE
QUALITY ATTRIBUTES SOFTWARE QUALITY EVALUATION GUIDEBOOK *** SPECIFICATION OF SOFTWARE
QUALITY ATTRIBUTES SOFTWARE QUALITY SPECIFICATION GUIDEBOOK *** SPECIFICATION OF SOFTWARE
QUALITY, CONCEPTS AND DEFINITIONS OF SOFTWARE QUALITY *** FACTORS IN SOFTWARE

QUALITY EVALUATION GUIDEBOOK *** SPECIFICATION OF SOFTWARE QUALITY ATTRIBUTES SOFTWARE

QUALITY EVALUATION SYSTEM (QUES). VOLUME I. ***

QUALITY FOR AN ACQUISITION MANAGER *** FACTORS IN SOFTWARE QUALITY. PRELIMINARY HANDBOOK ON SOFTWARE

QUALITY FOR AN ACQUISITION MANAGER *** FACTORS IN SOFTWARE QUALITY. VOLUME-III. PRELIMINARY HANDBOOK ON SOFTWARE

QUALITY MEASUREMENT *** AUTOMATION OF

QUALITY MEASUREMENT DEMONSTRATION PROJECT (I). *** SOFTWARE

QUALITY MEASUREMENT DEMONSTRATION PROJECT II *** SOFTWARE

QUALITY MEASUREMENT FOR DISTRIBUTED SYSTEMS *** SOFTWARE

QUALITY MEASUREMENT FOR DISTRIBUTED SYSTEMS - DISTRIBUTED COMPUTING SYSTEMS: IMPACT ON SOFTWARE QUALITY *** SOFTWARE

QUALITY MEASUREMENT FOR DISTRIBUTED SYSTEMS GUIDEBOOK FOR SOFTWARE QUALITY *** SOFTWARE

QUALITY MEASUREMENT MANUAL *** SOFTWARE

QUALITY MEASUREMENT METHODOLOGY ENHANCEMENTS STUDY RESULTS *** SOFTWARE

QUALITY. METRIC DATA COLLECTION AND VALIDATION *** FACTORS IN SOFTWARE

QUALITY METRICS ENHANCEMENTS *** SOFTWARE

QUALITY. PRELIMINARY HANDBOOK ON SOFTWARE QUALITY FOR AN ACQUISITION MANAGER *** FACTORS IN SOFTWARE

QUALITY SOFTWARE (ASQS) OPERATIONAL CONCEPT DOCUMENT. VOLUME 1 *** THE ASSISTANT FOR SPECIFYING THE

QUALITY SOFTWARE SYSTEMS *** ENGINEERING OF

QUALITY SOFTWARE SYSTEMS (A SOFTWARE ERROR CLASSIFICATION METHODOLOGY) *** ENGINEERING OF

QUALITY SOFTWARE SYSTEMS (EFFECTS OF MANAGEMENT PHILOSOPHY ON SOFTWARE PRODUCTION) *** ENGINEERING OF

QUALITY SOFTWARE SYSTEMS (MEASURING THE COMPLEXITY OF COMPUTER SOFTWARE) *** ENGINEERING OF

QUALITY SOFTWARE SYSTEMS (SOFTWARE FIRST CONCEPTS) *** ENGINEERING OF

QUALITY SOFTWARE SYSTEMS (SOFTWARE RELIABILITY MODELING AND MEASUREMENT TECHNIQUES) *** ENGINEERING OF

QUALITY SOFTWARE SYSTEMS (SOME CASE STUDIES IN STRUCTURED PROGRAMMING) *** ENGINEERING OF

QUALITY SOFTWARE SYSTEMS (TOWARDS AN ANALYSIS OF THE LISP PROGRAMMING LANGUAGE) *** ENGINEERING OF

QUALITY SPECIFICATION GUIDEBOOK *** SPECIFICATION OF SOFTWARE QUALITY ATTRIBUTES SOFTWARE

QUALITY. VOLUME-III. PRELIMINARY HANDBOOK ON SOFTWARE QUALITY FOR AN ACQUISITION MANAGER *** FACTORS IN SOFTWARE

QUANTITATIVE MEASURES OF SOFTWARE SYSTEMS *** BAYESIAN SOFTWARE PREDICTION MODELS. VOLUME I. AN IMPERFECT DEBUGGING MODEL FOR RELIABILITY
AND OTHER

13923  QUESTIONS. VOLUME I. QUALITY EVALUATION SYSTEM (RASTER-LINEAL EDITING SOFTWARE

15428  RECENT DEVELOPMENTS IN THE ECL PROGRAMMING SYSTEM

15405  RECOGNITION METHODS FOR DETERMINING SOFTWARE QUALITY

15404  RELIABILITY *** AN EXPERIMENTAL INVESTIGATION INTO SOFTWARE

8544  RELIABILITY AND OTHER QUANTITATIVE MEASURES OF SOFTWARE SYSTEMS

15398  BAYESIAN SOFTWARE PREDICTION MODELS. VOLUME I. AN IMPERFECT DEBUGGING MODEL FOR

5098  RELIABILITY AND TESTABILITY FOR DISTRIBUTED SYSTEMS FAULT TOLERANCE,

3982  RELIABILITY ASSESSMENT A GUIDEBOOK FOR SOFTWARE

2052  RELIABILITY CONSIDERATIONS COMMUNICATIONS PROCESSOR OPERATING SYSTEM TASK 2

5844  RELIABILITY MODEL DEMONSTRATION STUDY. VOLUME 1

5845  RELIABILITY MODEL DEMONSTRATION STUDY. VOLUME 2

15393  RELIABILITY MODELING ON SOFTWARE

4187  RELIABILITY MODELING AND ESTIMATION TECHNIQUES SOFTWARE

965  RELIABILITY MODELING AND MEASUREMENT TECHNIQUES ENGINEERING OF QUALITY SOFTWARE SYSTEMS

15388  RELIABILITY MODELS *** ANALYSIS OF DISCRETE SOFTWARE

15380  RELIABILITY MODELS *** COMBINED HW/SW

7868  RELIABILITY PREDICTION *** METHODOLOGY FOR SOFTWARE

7869  RELIABILITY PREDICTION *** METHODOLOGY FOR SOFTWARE

15416  RELIABILITY STUDIES *** SOFTWARE DATA COLLECTION STUDY. VOLUME III. DATA REQUIREMENTS FOR PRODUCTIVITY AND

15400  REPOSITORY STUDY SOFTWARE DATA

15416  REQUIREMENTS FOR PRODUCTIVITY AND RELIABILITY STUDIES SOFTWARE DATA COLLECTION STUDY. VOLUME III. DATA

12815  RESULTS SOFTWARE QUALITY MEASUREMENT METHODOLOGY ENHANCEMENTS STUDY

4383  REUSABILITY SOFTWARE INTEROPERABILITY AND

4384  REUSABILITY SOFTWARE INTEROPERABILITY AND

1496  RIPPLE EFFECT ANALYSIS FOR LARGE-SCALE SOFTWARE MAINTENANCE PERFORMANCE

2041  SCIENTIFIC REPORT ON THE ASTROS PLAN A MATCHED PROJECT EVALUATION OF MODERN PROGRAMMING PRACTICES.

15406  SELECTED AIR FORCE PROJECTS AN INVESTIGATION OF PROGRAMMING PRACTICES

1497  SELF-METRIC SOFTWARE. SUMMARY OF TECHNICAL PROGRESS

15376  SNEAK ANALYSIS APPLICATION GUIDELINES
SUMMARY *** SOFTWARE MODELING STUDIES, EXECUTIVE
SUMMARY OF TECHNICAL PROGRESS *** SELF-METRIC SOFTWARE.
SUMMARY OF TECHNICAL PROGRESS *** SOFTWARE MODELING STUDIES. VOLUME I.
SUMMARY OF TECHNICAL PROGRESS - SOFTWARE MODELING STUDIES ***
SUMMARY OF TECHNICAL PROGRESS, INVESTIGATION OF SOFTWARE MODELS ***
SUMMARY OF TECHNICAL PROGRESS, INVESTIGATION OF SOFTWARE MODELS ***
SUMMARY OF TECHNICAL PROGRESS, SOFTWARE MODELING STUDIES ***
SUMMARY OF TECHNICAL PROGRESS, SOFTWARE MODELING STUDIES ***
SUMMARY OF TECHNICAL PROGRESS SOFTWARE MODELING STUDIES ***
SUMMARY *** SOFTWARE RELIABILITY MODELS *** COMBINED HW/
SYSTEM *** JOVIAL AUTOMATED VERIFICATION
SYSTEM *** RECENT DEVELOPMENTS IN THE ECL PROGRAMMING
SYSTEM *** SOFTWARE FAULT TOLERANCE DESIGN
SYSTEM (AMS). *** AUTOMATED MEASUREMENT
SYSTEM (ATVS) *** ADA TEST AND VERIFICATION
SYSTEM (ATVS) IMPLEMENTATION *** ADA TEST AND VERIFICATION
SYSTEM DEVELOPMENT *** IMPACT OF MPP ON
SYSTEM (FAVS). VOLUME I *** FORTRAN AUTOMATED VERIFICATION
SYSTEM (FAVS). VOLUME II. USER'S MANUAL *** FORTRAN AUTOMATED VERIFICATION
SYSTEM PERFORMANCE ASSESSMENT AND CONTROL CONCEPT ***
SYSTEM (QUES). VOLUME I. *** QUALITY EVALUATION
SYSTEM; STUDY PHASE *** COBOL AUTOMATED VERIFICATION
SYSTEM TASK 2 - RELIABILITY CONSIDERATIONS *** COMMUNICATIONS PROCESSOR OPERATING
SYSTEM TEST METHODOLOGY DEVELOPMENT *** DATA MANAGEMENT
SYSTEM TESTING AND METHODOLOGY VALIDATION *** DATA MANAGEMENT
SYSTEMS *** BAYESIAN SOFTWARE PREDICTION MODELS, VOLUME I. AN IMPERFECT DEBUGGING MODEL FOR RELIABILITY AND OTHER QUANTITATIVE MEASURES OF SOFTWARE
SYSTEMS *** ENGINEERING OF QUALITY SOFTWARE
SYSTEMS *** FAULT TOLERANCE, RELIABILITY AND TESTABILITY FOR DISTRIBUTED
SYSTEMS *** KNOWLEDGE-BASED PROJECT MANAGEMENT ASSISTANT FOR ADA
SYSTEMS *** SOFTWARE QUALITY MEASUREMENT FOR DISTRIBUTED
SYSTEMS - DISTRIBUTED COMPUTING SYSTEMS: IMPACT ON SOFTWARE QUALITY *** SOFTWARE QUALITY MEASUREMENT FOR DISTRIBUTED
SYSTEMS (A SOFTWARE ERROR CLASSIFICATION METHODOLOGY) *** ENGINEERING OF QUALITY SOFTWARE
SYSTEMS (EFFECTS OF MANAGEMENT PHILOSOPHY ON SOFTWARE PRODUCTION) *** ENGINEERING OF QUALITY SOFTWARE
SYSTEMS GUIDEBOOK FOR SOFTWARE QUALITY *** SOFTWARE QUALITY MEASUREMENT FOR DISTRIBUTED
5417 SYSTEMS: IMPACT ON SOFTWARE QUALITY *** SOFTWARE QUALITY MEASUREMENT FOR DISTRIBUTED SYSTEMS - DISTRIBUTED COMPUTING

160 SYSTEMS (MEASURING THE COMPLEXITY OF COMPUTER SOFTWARE) *** ENGINEERING OF QUALITY SOFTWARE

15413 SYSTEMS MODELING STUDY *** INTERACTIVE COMMUNICATION

15417 SYSTEMS MODELING STUDY *** INTERACTIVE COMMUNICATION

15425 SYSTEMS (SOFTWARE FIRST CONCEPTS) *** ENGINEERING OF QUALITY SOFTWARE

965 SYSTEMS (SOFTWARE RELIABILITY MODELING AND MEASUREMENT TECHNIQUES) *** ENGINEERING OF QUALITY SOFTWARE

15422 SYSTEMS (SOME CASE STUDIES IN STRUCTURED PROGRAMMING) *** ENGINEERING OF QUALITY SOFTWARE

15424 SYSTEMS (TOWARDS AN ANALYSIS OF THE LISP PROGRAMMING LANGUAGE) *** ENGINEERING OF QUALITY SOFTWARE

2052 TASK 2 - RELIABILITY CONSIDERATIONS *** COMMUNICATIONS PROCESSOR OPERATING SYSTEM

1497 TECHNICAL PROGRESS *** SELF-METRIC SOFTWARE, SUMMARY OF

15399 TECHNICAL PROGRESS *** SOFTWARE MODELING STUDIES. VOLUME I. SUMMARY OF

818 TECHNICAL PROGRESS - SOFTWARE MODELING STUDIES *** SUMMARY OF

969 TECHNICAL PROGRESS, INVESTIGATION OF SOFTWARE MODELS *** SUMMARY OF

15392 TECHNICAL PROGRESS, INVESTIGATION OF SOFTWARE MODELS *** SUMMARY OF

15412 TECHNICAL PROGRESS, SOFTWARE MODELING STUDIES *** SUMMARY OF

15414 TECHNICAL PROGRESS, SOFTWARE MODELING STUDIES *** SUMMARY OF

15418 TECHNICAL PROGRESS SOFTWARE MODELING STUDIES *** SUMMARY OF

15343 TECHNIQUES, *** CODING FOR TURBULENT CHANNELS. VOLUME III. TURBULENT CHANNEL DATA ANALYSIS AND EVALUATION OF CODING

965 TECHNIQUES) *** ENGINEERING OF QUALITY SOFTWARE SYSTEMS (SOFTWARE RELIABILITY MODELING AND MEASUREMENT

4187 TECHNIQUES *** SOFTWARE RELIABILITY MODELING AND ESTIMATION

15384 TEST AND VERIFICATION SYSTEM (ATVS) *** ADA

15356 TEST AND VERIFICATION SYSTEM (ATVS) IMPLEMENTATION *** ADA

15344 TEST METHODOLOGY DEVELOPMENT *** DATA MANAGEMENT SYSTEM

5098 TESTABILITY FOR DISTRIBUTED SYSTEMS *** FAULT TOLERANCE, RELIABILITY AND

15419 TESTING *** METHODOLOGY FOR COMPREHENSIVE SOFTWARE

15345 TESTING AND METHODOLOGY VALIDATION *** DATA MANAGEMENT SYSTEM

15377 TESTING MEASURES *** SOFTWARE

15363 TOLERANCE DESIGN SYSTEM. *** SOFTWARE FAULT

5098 TOLERANCE, RELIABILITY AND TESTABILITY FOR DISTRIBUTED SYSTEMS *** FAULT

15382 TOOL IMPLEMENTATION *** NATIONAL SOFTWARE WORKS

15381 TOOL INTEGRATION STUDIES *** NATIONAL SOFTWARE WORKS

15354 TOOLS *** KNOWLEDGE BASED QUALITY ASSURANCE
<table>
<thead>
<tr>
<th>15372</th>
<th>TOOLS FOR SOFTWARE MAINTENANCE *** ADVANCED</th>
</tr>
</thead>
<tbody>
<tr>
<td>15424</td>
<td>TOWARDS AN ANALYSIS OF THE LISP PROGRAMMING LANGUAGE) *** ENGINEERING OF QUALITY SOFTWARE SYSTEMS (</td>
</tr>
<tr>
<td>15370</td>
<td>TRANSITION *** DATA AND ANALYSIS CENTER FOR SOFTWARE: AN IAC IN</td>
</tr>
<tr>
<td>15343</td>
<td>TURBULENT CHANNEL DATA ANALYSIS AND EVALUATION OF CODING TECHNIQUES. *** CODING FOR TURBULENT CHANNELS. VOLUME III.</td>
</tr>
<tr>
<td>15343</td>
<td>TURBULENT CHANNELS. VOLUME III. TURBULENT CHANNEL DATA ANALYSIS AND EVALUATION OF CODING TECHNIQUES. *** CODING FOR</td>
</tr>
<tr>
<td>15394</td>
<td>USER'S MANUAL *** FORTRAN AUTOMATED VERIFICATION SYSTEM (FAVS). VOLUME II.</td>
</tr>
<tr>
<td>15345</td>
<td>VALIDATION *** DATA MANAGEMENT SYSTEM TESTING AND METHODOLOGY</td>
</tr>
<tr>
<td>215</td>
<td>VALIDATION *** FACTORS IN SOFTWARE QUALITY. METRIC DATA COLLECTION AND</td>
</tr>
<tr>
<td>15396</td>
<td>VERIFICATION SYSTEM *** JOVIAL AUTOMATED</td>
</tr>
<tr>
<td>15384</td>
<td>VERIFICATION SYSTEM (ATVS) *** ADA TEST AND</td>
</tr>
<tr>
<td>15356</td>
<td>VERIFICATION SYSTEM (ATVS) IMPLEMENTATION *** ADA TEST AND</td>
</tr>
<tr>
<td>15395</td>
<td>VERIFICATION SYSTEM (FAVS). VOLUME I *** FORTRAN AUTOMATED</td>
</tr>
<tr>
<td>15394</td>
<td>VERIFICATION SYSTEM (FAVS). VOLUME II. USER'S MANUAL *** FORTRAN AUTOMATED</td>
</tr>
<tr>
<td>15387</td>
<td>VERIFICATION SYSTEM: STUDY PHASE *** COBOL AUTOMATED</td>
</tr>
<tr>
<td>15383</td>
<td>VERSUS FORTRAN FOR FIRST-COURSE PROGRAMMERS *** SOFTWARE MODELING STUDIES EXPERIMENTAL STUDY OF A TWO DIMENSIONAL LANGUAGE</td>
</tr>
<tr>
<td>15401</td>
<td>VOLUME-III. PRELIMINARY HANDBOOK ON SOFTWARE QUALITY FOR AN ACQUISITON MANAGER *** FACTORS IN SOFTWARE QUALITY.</td>
</tr>
<tr>
<td>15382</td>
<td>WORKS TOOL IMPLEMENTATION *** NATIONAL SOFTWARE</td>
</tr>
<tr>
<td>15381</td>
<td>WORKS TOOL INTEGRATION STUDIES *** NATIONAL SOFTWARE</td>
</tr>
</tbody>
</table>
4. KEYWORD INDEX

Acceptance Tests: 15363
Access-Control Mechanisms: 2052, 4234
Accounting: 15357
Acquisition Management: 215, 216, 6794, 8697
Acquisition: 15353
Ada Programming Language: 13923, 15356, 15357
Ada: 6458, 8544
Air Force Research: 15391
Algorithm Analysis: 4233, 4234
Algorithms: 15374, 15375, 15410, 15414, 15422
Analysis of Variance: 15383
Analyzers: 15396
Archives: 15357
Artificial Intelligence: 15352
Assessment: 15409
Automatic Data Collection: 6458, 818
Automatic Programming: 818
Automatic: 15394, 15404
Automation: 15387, 15395, 15415
Availability: 4187, 5098
Bandwidth: 15413
Batch Processing: 15416
Bayes Theorem: 15397, 15398
Behavior Modeling: 1497
Benefits: 15371
Chief Programmer Team: 2040, 2041
Classification: 15421
Coding: 15344, 15356, 15411
Collection: 13923
Command and Control Systems: 15378, 15399, 15408
Command, Control, and Communications: 5098, 6795
Communication Equipment: 15409, 15417
Communications Networks: 15381, 15382, 15409
Compilers: 1430, 15390, 15428
Complexity Measurement: 1496, 160, 2040, 2041, 3849, 3850, 6458, 818, 969
Complexity: 214
Comprehension: 15404
Computation Structures: 4233
Computations: 15371, 15374, 15375
Computer Aided Instruction: 15396
Computer Applications: 15356, 15371
Computer Architecture: 15381
Computer Communications: 15381, 15382
Computer Graphics: 15417, 15428
Computer Logic: 15394
Computer Operators: 15372
Computer Program Documentation: 15371, 15372, 15377, 15387, 15394, 15395, 15405, 15406, 15407, 15408, 15410, 15412, 15414, 15415, 15416, 15419
Computer Program Reliability: 15352, 15354, 15371, 15373, 15374, 15375, 15377, 15380, 15388, 15390, 15392, 15393, 15397, 15398, 15407, 15408, 15410, 15412, 15416
Computer Program Verification: 15353, 15376, 15377, 15384, 15387, 15388, 15394, 15395, 15396, 15407
Computer Programming: 15343, 15345, 15372, 15382, 15383, 15387, 15390, 15391, 15392, 15397, 15398, 15401, 15404, 15405, 15406, 15408, 15410, 15411, 15412, 15416, 15417, 15418, 15421, 15422, 15424, 15425, 15426, 15427, 15428
Computer Programs: 13923, 13933, 15352, 15354, 15356, 15363, 15374, 15375, 15376, 15378, 15380, 15383, 15392, 15393, 15394, 15395, 15396, 15399, 15400, 15404, 15405, 15407, 15408, 15410, 15412, 15413, 15414, 15415, 15416, 15419
Computerized Simulation: 15383, 15417
Computers: 15380, 15396
Conciseness: 2052
Configuration Management: 215, 216
Configurations: 15415
Consistency: 13923
Contracts: 15376
Control Structures: 3850
Conversions: 1430
Correctness Proofs: 214, 2192, 2193
Cost Analysis: 15376, 15377
Cost Effectiveness: 15399, 15410
Cost Models: 2040, 2041, 3849
Cost-Benefit Analysis: 216
Costs: 15370, 15405, 15410, 15411, 15416, 15427
Data Acquisition: 13923, 15370, 15371, 15373, 15378
Data Analysis: 2040, 2041, 215, 4187, 5098
Data Bases: 13923, 15354, 15370, 15373, 15374, 15375, 15376, 15384, 15390, 15400, 15404, 15405, 15407, 15408, 15410, 15415, 15416, 15419
Data Collection: 215, 6796, 7868, 7869
Data Domains: 2052
Data Management: 15373, 15378, 15400, 15427
Data Processing Equipment: 15390
Data Processing: 15344, 15345, 15416
Data Reduction: 15371, 15411
Data Storage Systems: 15373
Data Structures: 4519
Data Transmission Systems: 15343
Database Management Systems: 4234
Decision Making: 13933

49
Decoding: 15343
Demodulators: 15343
Design Methodologies: 5098
Development Cycle: 215
Development Management: 215
Development Methodologies: 2040, 2041, 6458
Development Process: 2052
Development Support Librarian: 2052
Digital Computers: 15410, 15411, 15413
Digital Systems: 15343, 15413
Distributed Processing: 5098
Documents: 15379
Efficiency: 2052, 214, 2192, 2193, 4233, 4234
Error Analysis: 15392, 15399, 15408, 2040, 2041, 2052, 8544
Error Categories: 2052, 4187
Error Correction Codes: 15371
Error Detection Codes: 15395
Error Models: 818, 969
Errors: 15414, 15421, 5098
Execution Time: 4233
Experimental Data: 15356
Failure Data: 4187
Failure Rates: 2052, 3849, 4187
Failures: 3982, 5098
Fault Correction: 1497
Fault Data Analysis: 2040, 2041
Fault Data: 818
Fault Detection: 4187, 5098, 969
Fault Tolerance: 2052, 5098
Fault Tolerant Computing: 15363
Faults: 15363
Feasibility Studies: 15384
Firmware: 2052
Flexibility: 214, 2192, 2193, 4233
Flow Charting: 15395, 15414, 15419
Fortran: 13923, 1430, 15383, 15391, 15394, 8544
Geometric-Poisson Model: 3982, 4187, 5844, 5845
Goel-Okumoto Model: 4187, 5844, 5845
Hardware/Software Tradeoffs: 2052
Hazard Function: 818
High Reliability: 15363
Imperfect Debugging: 818
Indicators: 13923
Information Centers: 15370, 15400
Information Exchange: 15381
Information Retrieval: 15344
Input Output Devices: 15344
Input Output Processing: 15400
Integrated Systems: 15381
Integrity: 2192, 2193, 4233
Interactions: 15375, 15380, 15417
Interactive Graphics: 15413
Interfaces: 15372, 15425
Interoperability: 2192, 2193, 4233, 4383, 4384, 4394, 5417
Jelinski-Moranda Model: 3982, 4187, 965
JOVIAL: 2052
Knowledge Based Systems: 15353, 15357
Laboratories: 13923
Language Design: 1430
Language Evaluation: 1430
Libraries: 15390
Life Cycle Costs: 15388
Life Cycle Testing: 15390
Life Cycles: 15356, 15371, 15372, 15378, 15416
Linear Regression Analyses: 15408
Loops: 15374
Low Costs: 15388, 15408, 15412
Machine Coding: 15390, 15405
Machine Translation: 15374
Macroprogramming: 15405
Maintainability: 1496, 1497, 15356, 15370, 15371, 15384, 214, 2192, 2193, 4233
Maintenance Tools and Techniques: 1497, 4519
Maintenance: 15372, 15382, 4519
Management Planning and Control: 15376, 15379, 15391, 15411
Management Planning: 15344, 15345
Management Tools and Techniques: 2040, 2041, 215, 216
Management: 15392, 15425, 15426, 3849
Markov Processes: 15380, 15393, 15398, 15399
Mathematical Logic: 15344, 15422
Mathematical Methodologies: 965
Mathematical Models: 15388, 15393, 15397, 15398, 15399, 15418
Mathematical Prediction: 15397, 15408
Mathematical Programming: 15392
Measurement: 13933, 15343, 15344, 15345, 15352, 15353, 15354, 15356, 15357, 15363, 15370, 15371, 15372, 15373, 15374, 15375, 15376, 15377, 15378, 15379, 15380, 15381, 15382, 15383, 15384, 15387, 15388, 15390, 15391, 15392, 15393, 15394, 15395, 15396, 15397, 15398, 15399, 15400, 15401, 15404, 15405, 15406, 15407, 15408, 15409, 15410, 15411, 15412, 15413, 15414, 15415, 15416, 15417, 15418, 15419,

51
15421, 15422, 15424, 15425, 15426, 15427, 15428, 8544
Memory Devices: 15344
Methodology: 12815, 15353, 15379, 15382
Metric System: 13923, 15375
Microcomputers: 15384
Microprocessors: 15399
Microprogramming: 15399, 15412, 15425
Military Applications: 15391
Model Theory: 15398
Models: 15392, 15412, 15414, 15417, 6458, 818, 969
Modification Procedures: 1496, 1497
Modification: 1497, 15363, 15372
Modular Construction: 15412
Modular Programming: 818
Modularity: 214
Module Testing: 1430
Modules: 3850
Monitors: 15345
Monte Carlo Method: 15413
Multivariate Analysis: 15374
Musa's Model: 3982, 4187
Numerical Analysis: 15397
Operating Systems: 2052
Order Statistics: 15374
Pattern Recognition: 15404
Performance Engineering: 15383
Performance Evaluation: 1496, 4187
Performance Tests: 15383
Performance: 1497, 15409
Personnel Management: 15372
Pilot Studies: 15400
Planning: 15357
Portability: 214, 2192, 2193, 4233
Probability: 15397
Procurement: 15373
Productivity Models: 2040, 2041, 3849
Productivity: 15371, 15378
Program Analysis: 1430
Program Complexity: 160, 6458, 969
Program Design Language: 6458
Program Maintenance: 1496
Program Testing: 1497
Programmers: 15371, 15406, 15408

52
Programming Aids: 3850
Programming Language Complexity: 3849
Programming Languages: 15385, 15396, 15405, 15412, 15414, 15416, 15424
Programming Manuals: 15371
Prototypes: 15363, 15373, 15390
Quality Assurance: 13923, 15352, 15353, 15354, 15373, 15379, 15381, 15391, 15411, 3982
Quality Attributes: 160, 2040, 2041, 214, 215, 216, 6794, 6795, 6796, 8697
Quality Control: 15379, 15387, 15388, 15410, 15415, 15426
Quality Metrics: 12815, 1430, 214, 215, 216, 2192, 2193, 4233, 4234, 4383, 4384, 4394, 4395, 4519, 5417, 6458, 6794, 6795, 6796, 7868, 7869, 818, 8697
Quality: 12815, 13923, 13933, 15353, 15354, 15370, 15375, 15378, 15401, 4383, 4384, 4394, 4395, 5417
Real Time: 15411
Regression Testing: 1497
Regulations: 15379
Reliability Estimation: 7868, 7869, 965
Reliability Measurement: 1497, 2052, 5098, 5844, 5845, 7868, 7869, 965
Reliability Models: 2052, 3849, 3982, 4187, 5844, 5845, 818, 965
Reliability Prediction: 3849, 4187, 5844, 5845, 7868, 7869, 965
Reliability: 15356, 15363, 15370, 15376, 15380, 15399, 15418, 15419, 15422, 15424, 15427, 214, 2192, 2193, 3982, 818, 8544
Requirements: 15352, 15353, 15370, 15376, 15379
Resource Management: 15382, 15390
Resources: 15357
Reusability: 214, 2192, 2193, 4233, 4383, 4384, 4394, 4395, 5417
Savings: 15405
Scheduling: 15411
Schick-Wolverton Model: 3982, 4187
Semantics: 15374
Shooman Model: 3982, 4187, 965
Signal To Noise Ratio: 15413
Simulation: 15343
Software Engineering Project Management: 3849
Software Engineering Standards: 6795
Software Engineering Tools and Techniques: 3850, 6458
Software Engineering: 214
Software Experience Data: 2040, 2041, 215, 5845, 7868, 7869
Software Library Management System: 2052
Software Life Cycle: 4394, 4395, 5417, 6796
Software Science: 2040, 2041, 6458
Sorting: 15374
Specifications: 13933, 15352, 15353, 15387, 15401, 15419, 216
Standardization: 15352
Standards: 15410
State of the Art: 15374, 15375, 15378, 15391
Statistical Tests: 15399
Structured Programming: 2040, 2041, 2052
Structuredness: 2052
Subroutines: 15374, 15410, 15414
Sun: 13923
Survivability: 15371
Symbols: 15405
Syntax: 15344, 15383, 15394
System Reliability: 2052, 4187, 7868, 7869
Systems Analysis: 15376, 15399
Systems Engineering: 15363, 15378, 15379, 15391, 15413
Systems Management: 15391
Technology Transfer: 13923, 15370, 15391
Telecommunication: 15409
Telecommunications: 15378
Test and Evaluation: 15356, 15373, 15377, 15383, 15384, 15407
Test Data Generation: 818, 969
Test Design: 969
Test Driver Tools: 969
Test Methodologies: 969
Test Methods: 15345, 15352, 15392, 15396, 15407, 15412, 15419
Testability: 214, 2192, 2193, 4233, 5098
Testing Tools: 1430
Testing: 3982
Tests: 15418
Time Dependence: 15393
Time Domain: 15417
Time Sharing: 15416
Tools: 13923, 15354, 15356, 15357, 15376, 15382, 215
Top-Down Development: 2040, 2041
Traceability: 12815
Usability: 2192, 2193, 4233
User Needs: 15383, 15387, 15396
Validation: 15352, 15376, 15427, 215
Verification: 13923, 15352, 215
Volume: 13923
<table>
<thead>
<tr>
<th>Author Name</th>
<th>Page Numbers</th>
<th>Titles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amory, W.</td>
<td>15421</td>
<td>Engineering of Quality Software Systems (A Software Error Classification Methodology)</td>
</tr>
<tr>
<td>Anderson, Lyle A.</td>
<td>15374</td>
<td>Algorithmic Complexity, Volume II</td>
</tr>
<tr>
<td>Andrews, Dorothy M.</td>
<td>15375</td>
<td>Algorithmic Complexity, Volume I</td>
</tr>
<tr>
<td>Babst, Tom A.</td>
<td>15395</td>
<td>Fortran Automated Verification System (FAVS). Volume I</td>
</tr>
<tr>
<td>Bass, Leonard J.</td>
<td>15404</td>
<td>Pattern Recognition Methods for Determining Software Quality</td>
</tr>
<tr>
<td>Bennett, David A.</td>
<td>15377</td>
<td>Automated Measurement System (AMS).</td>
</tr>
<tr>
<td>Benson, J.</td>
<td>15380</td>
<td>Combined HW/SW Reliability Models</td>
</tr>
<tr>
<td>Boaz, Wade</td>
<td>15381</td>
<td>Reliability Model Demonstration Study. Volume I</td>
</tr>
<tr>
<td>Bowen, C.</td>
<td>15382</td>
<td>Reliability Model Demonstration Study. Volume 2</td>
</tr>
<tr>
<td>Bowen, J. B.</td>
<td>15383</td>
<td>Combined HW/SW Reliability Models</td>
</tr>
<tr>
<td>Bowen, Thomas P.</td>
<td>15384</td>
<td>Reliability Model Demonstration Study. Volume I</td>
</tr>
<tr>
<td></td>
<td>15385</td>
<td>Reliability Model Demonstration Study. Volume 2</td>
</tr>
<tr>
<td></td>
<td>4383</td>
<td>Software Interoperability and Reusability</td>
</tr>
<tr>
<td></td>
<td>4384</td>
<td>Software Interoperability and Reusability</td>
</tr>
<tr>
<td></td>
<td>4394</td>
<td>Software Quality Measurement for Distributed Systems Guidebook for Software Quality</td>
</tr>
<tr>
<td></td>
<td>4395</td>
<td>Software Quality Measurement for Distributed Systems</td>
</tr>
<tr>
<td></td>
<td>5417</td>
<td>Software Quality Measurement for Distributed Systems - Distributed Computing Systems: Impact on Software Quality</td>
</tr>
<tr>
<td></td>
<td>6794</td>
<td>Specification of Software Quality Attributes</td>
</tr>
<tr>
<td></td>
<td>6795</td>
<td>Specification of Software Quality Attributes Software Quality Specification Guidebook</td>
</tr>
<tr>
<td></td>
<td>6796</td>
<td>Specification of Software Quality Attributes Software Quality Evaluation Guidebook</td>
</tr>
<tr>
<td>Name</td>
<td>Title</td>
<td></td>
</tr>
<tr>
<td>---------------------------</td>
<td>----------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Branning, W. E.</td>
<td>MODERN PROGRAMMING PRACTICES STUDY REPORT</td>
<td></td>
</tr>
<tr>
<td>Brooks, N. B.</td>
<td>JAVS FINAL REPORT</td>
<td></td>
</tr>
<tr>
<td>Brooks, W. D.</td>
<td>ANALYSIS OF DISCRETE SOFTWARE RELIABILITY MODELS</td>
<td></td>
</tr>
<tr>
<td>Brown, John R.</td>
<td>DMA MODERN PROGRAMMING ENVIRONMENT STUDY</td>
<td></td>
</tr>
<tr>
<td>Bullen, Richard H., Jr</td>
<td>IMPACT OF MPP ON SYSTEM DEVELOPMENT</td>
<td></td>
</tr>
<tr>
<td>Bunker, Ralph E.</td>
<td>ENGINEERING OF QUALITY SOFTWARE SYSTEMS (SOFTWARE FIRST CONCEPTS)</td>
<td></td>
</tr>
<tr>
<td>Buratti, Davey L.</td>
<td>ALGORITHMIC COMPLEXITY. VOLUME II</td>
<td></td>
</tr>
<tr>
<td>Carroll, Clifford E.</td>
<td>ALGORITHMIC COMPLEXITY</td>
<td></td>
</tr>
<tr>
<td>Cheatham, Thomas E., Jr</td>
<td>SNEAK ANALYSIS APPLICATION GUIDELINES</td>
<td></td>
</tr>
<tr>
<td>Cheng, L. L.</td>
<td>NATIONAL SOFTWARE WORKS TOOL IMPLEMENTATION</td>
<td></td>
</tr>
<tr>
<td>Clapp, Judith A.</td>
<td>RECENT DEVELOPMENTS IN THE ECL PROGRAMMING SYSTEM</td>
<td></td>
</tr>
<tr>
<td>Colley, L. E.</td>
<td>ENGINEERING OF QUALITY SOFTWARE SYSTEMS (SOME CASE STUDIES IN STRUCTURED PROGRAMMING)</td>
<td></td>
</tr>
<tr>
<td>Collofello, James S.</td>
<td>ENGINEERING OF QUALITY SOFTWARE SYSTEMS (A SOFTWARE ERROR CLASSIFICATION METHODOLOGY)</td>
<td></td>
</tr>
<tr>
<td>Curtis, Bill</td>
<td>ENGINEERING OF QUALITY SOFTWARE SYSTEMS CODING FOR TURBULENT CHANNELS, VOLUME III TURBULENT CHANNEL DATA ANALYSIS AND EVALUATION OF CODING TECHNIQUES. PERFORMANCE RIPPLE EFFECT ANALYSIS FOR LARGE-SCALE SOFTWARE MAINTENANCE EVALUATION OF SOFTWARE LIFE CYCLE DATA FROM THE PAVE PAWS PROJECT.</td>
<td></td>
</tr>
<tr>
<td>Daum, Marilyn</td>
<td>KNOWLEDGE-BASED PROJECT MANAGEMENT ASSISTANT FOR ADA SYSTEMS</td>
<td></td>
</tr>
<tr>
<td>Davlin, James J.</td>
<td>IMPROVING SOFTWARE QUALITY ASSURANCE METHODS</td>
<td></td>
</tr>
<tr>
<td>Dean, Jeffrey S.</td>
<td>ADVANCED TOOLS FOR SOFTWARE MAINTENANCE</td>
<td></td>
</tr>
<tr>
<td>Derenbacher, Richard L.</td>
<td>RASTER-LINEAL EDITING SOFTWARE</td>
<td></td>
</tr>
<tr>
<td>Dewolf, J.B.</td>
<td>AUTOMATING SOFTWARE DESIGN METRICS</td>
<td></td>
</tr>
<tr>
<td>Dobmeier, John J.</td>
<td>NATIONAL SOFTWARE WORKS TOOL IMPLEMENTATION</td>
<td></td>
</tr>
<tr>
<td>Domitz, Robert</td>
<td>COMMUNICATIONS PROCESSOR OPERATING SYSTEM TASK 2 - RELIABILITY CONSIDERATIONS</td>
<td></td>
</tr>
<tr>
<td>Donaldson, Ernest E.</td>
<td>INTERACTIVE COMMUNICATION SYSTEMS MODELING STUDY</td>
<td></td>
</tr>
<tr>
<td>Duvall, Lorraine M.</td>
<td>SOFTWARE DATA REPOSITORY STUDY</td>
<td></td>
</tr>
<tr>
<td>Dyson, Karen A.</td>
<td>QUALITY EVALUATION SYSTEM (QUES).</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Hecht, M.</td>
<td>Testability for Distributed Systems</td>
<td>7868</td>
</tr>
<tr>
<td>Heidler, W.</td>
<td>Methodology for Software Reliability Prediction</td>
<td>7869</td>
</tr>
<tr>
<td>Hersh, Harry M.</td>
<td>FAULT TOLERANCE, RELIABILITY AND TESTABILITY FOR DISTRIBUTED SYSTEMS</td>
<td>5098</td>
</tr>
<tr>
<td>Hobin, C. W.</td>
<td>Methodology for Software Reliability Prediction</td>
<td>7868</td>
</tr>
<tr>
<td>Huang, Gwo-Long</td>
<td>Methodology for Software Reliability Prediction</td>
<td>7869</td>
</tr>
<tr>
<td>James, L. E.</td>
<td>Software Testing Measures</td>
<td>15377</td>
</tr>
<tr>
<td>Janus, Philip J.</td>
<td>Combined HW/SW Reliability Models</td>
<td>15380</td>
</tr>
<tr>
<td>Juellig, Richard</td>
<td>Algorithmic Complexity. Volume II</td>
<td>15374</td>
</tr>
<tr>
<td>Kahn, Larry</td>
<td>Algorithmic Complexity. Volume I</td>
<td>15375</td>
</tr>
<tr>
<td>Kaminsky, Alan R.</td>
<td>Knowledge-Based Quality Assurance Tools</td>
<td>12815</td>
</tr>
<tr>
<td>Keller, Steve</td>
<td>The Assistant for Specifying the Quality Software (ASQS) Operational Concept Document. Volume 1</td>
<td>15353</td>
</tr>
<tr>
<td>Kerbel, A.</td>
<td>Software Testing Measures</td>
<td>15377</td>
</tr>
<tr>
<td>Kim, K.H.</td>
<td>Fault Tolerance, Reliability and Testability for Distributed Systems</td>
<td>5098</td>
</tr>
<tr>
<td>Klerer, Melvin</td>
<td>SOFTWARE MODELING STUDIES EXPERIMENTAL STUDY OF A TWO DIMENSIONAL LANGUAGE VERSUS FORTRAN FOR FIRST-COURSE PROGRAMMERS</td>
<td>15383</td>
</tr>
<tr>
<td>Kruesi, Elizabeth</td>
<td>Evaluation of Software Life Cycle Data From the Pave Paws Project</td>
<td>15390</td>
</tr>
<tr>
<td>LaPadula, Leonard J.</td>
<td>Engineering of Quality Software Systems</td>
<td>15427</td>
</tr>
<tr>
<td>Lamagna, Edmund A.</td>
<td>Algorithmic Complexity. Volume II</td>
<td>15374</td>
</tr>
<tr>
<td>Landauer, Christopher</td>
<td>Pattern Recognition Methods for Determining Software Quality</td>
<td>15404</td>
</tr>
<tr>
<td>Lapadula, Leonard J.</td>
<td>Engineering of Quality Software Systems</td>
<td>965</td>
</tr>
<tr>
<td>Lasky, Jeffrey A.</td>
<td>SOFTWARE QUALITY MEASUREMENT METHODOLOGY ENHANCEMENTS STUDY RESULTS</td>
<td>12815</td>
</tr>
<tr>
<td>Name</td>
<td>Code</td>
<td>Title</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Lazzara, Allen V.</td>
<td>15428</td>
<td>RASTER-LINEAL EDITING SOFTWARE</td>
</tr>
<tr>
<td>Lee, Jinshuan</td>
<td>15354</td>
<td>KNOWLEDGE BASED QUALITY ASSURANCE TOOLS</td>
</tr>
<tr>
<td>Like, K. L.</td>
<td>15378</td>
<td>C3I ENVIRONMENT MEASUREMENT INSTRUMENTATION</td>
</tr>
<tr>
<td>Lubbes, R. K.</td>
<td>15428</td>
<td>RASTER-LINEAL EDITING SOFTWARE</td>
</tr>
<tr>
<td>MacGregor, Terry E.</td>
<td>15393</td>
<td>ON SOFTWARE RELIABILITY MODELING</td>
</tr>
<tr>
<td>Markham, David</td>
<td>15373</td>
<td>AUTOMATION OF QUALITY MEASUREMENT</td>
</tr>
<tr>
<td>Matsumoto, Mike</td>
<td>2192</td>
<td>SOFTWARE QUALITY METRICS ENHANCEMENTS</td>
</tr>
<tr>
<td>McCall, James A.</td>
<td>15373</td>
<td>AUTOMATION OF QUALITY MEASUREMENT</td>
</tr>
<tr>
<td>McCune, Brian P.</td>
<td>15372</td>
<td>ADVANCED TOOLS FOR SOFTWARE MAINTENANCE</td>
</tr>
<tr>
<td>McDaniel, J.</td>
<td>15380</td>
<td>COMBINED HW/SW RELIABILITY MODELS</td>
</tr>
<tr>
<td>McGibbon, Thomas L.</td>
<td>15404</td>
<td>PATTERN RECOGNITION METHODS FOR DETERMINING SOFTWARE QUALITY</td>
</tr>
<tr>
<td>McGoldrick, Francis M.</td>
<td>15344</td>
<td>DATA MANAGEMENT SYSTEM TEST METHODOLOGY DEVELOPMENT</td>
</tr>
<tr>
<td>Mckelvey, N.</td>
<td>15345</td>
<td>DATA MANAGEMENT SYSTEM TESTING AND METHODOLOGY VALIDATION</td>
</tr>
<tr>
<td>Meeson, R.</td>
<td>7868</td>
<td>METHODOLOGY FOR SOFTWARE RELIABILITY PREDICTION</td>
</tr>
<tr>
<td>Melton, Richard</td>
<td>7869</td>
<td>METHODOLOGY FOR SOFTWARE RELIABILITY PREDICTION</td>
</tr>
<tr>
<td>Miller, E. F., Jr.</td>
<td>15377</td>
<td>SOFTWARE TESTING MEASURES</td>
</tr>
<tr>
<td>Milliman, Phil</td>
<td>15387</td>
<td>COBOL AUTOMATED VERIFICATION SYSTEM: STUDY PHASE</td>
</tr>
<tr>
<td></td>
<td>15394</td>
<td>FORTRAN AUTOMATED VERIFICATION SYSTEM (FAVS). VOLUME II. USER'S MANUAL</td>
</tr>
<tr>
<td></td>
<td>15395</td>
<td>FORTRAN AUTOMATED VERIFICATION SYSTEM (FAVS). VOLUME I</td>
</tr>
<tr>
<td></td>
<td>15419</td>
<td>METHODOLOGY FOR COMPREHENSIVE SOFTWARE TESTING</td>
</tr>
<tr>
<td></td>
<td>2040</td>
<td>A MATCHED PROJECT EVALUATION OF MODERN PROGRAMMING PRACTICES. MANAGEMENT REPORT ON THE ASTROS PLAN</td>
</tr>
<tr>
<td></td>
<td>2041</td>
<td>A MATCHED PROJECT EVALUATION OF MODERN PROGRAMMING PRACTICES. SCIENTIFIC REPORT ON THE ASTROS PLAN</td>
</tr>
<tr>
<td>Mitchell, M. E.</td>
<td>15343</td>
<td>CODING FOR TURBULENT CHANNELS. VOLUME III. TURBULENT CHANNEL DATA ANALYSIS AND EVALUATION OF CODING TECHNIQUES.</td>
</tr>
<tr>
<td>Montes, P. N.</td>
<td>15356</td>
<td>ADA TEST AND VERIFICATION SYSTEM (ATVS) IMPLEMENTATION</td>
</tr>
<tr>
<td>Morris, John</td>
<td>15404</td>
<td>PATTERN RECOGNITION METHODS FOR DETERMINING SOFTWARE QUALITY</td>
</tr>
<tr>
<td>Moss, Richard W.</td>
<td>7868</td>
<td>METHODOLOGY FOR SOFTWARE RELIABILITY PREDICTION</td>
</tr>
<tr>
<td></td>
<td>7869</td>
<td>METHODOLOGY FOR SOFTWARE RELIABILITY PREDICTION</td>
</tr>
<tr>
<td>Motley, R. W.</td>
<td>15413</td>
<td>INTERACTIVE COMMUNICATION SYSTEMS MODELING STUDY</td>
</tr>
<tr>
<td></td>
<td>15417</td>
<td>INTERACTIVE COMMUNICATION SYSTEMS MODELING STUDY</td>
</tr>
<tr>
<td>Okumoto, K.</td>
<td>15388</td>
<td>ANALYSIS OF DISCRETE SOFTWARE RELIABILITY MODELS</td>
</tr>
<tr>
<td></td>
<td>15408</td>
<td>STATISTICAL PREDICTION OF PROGRAMMING ERRORS</td>
</tr>
<tr>
<td>Perry, Grover H.</td>
<td>15397</td>
<td>BAYESIAN SOFTWARE PREDICTION MODELS, VOLUME III</td>
</tr>
<tr>
<td>Pierce, Patricia</td>
<td>15398</td>
<td>BAYESIAN SOFTWARE PREDICTION MODELS, VOLUME I: AN IMPERFECT DEBUGGING MODEL FOR RELIABILITY AND OTHER QUANTITATIVE MEASURES OF SOFTWARE SYSTEMS</td>
</tr>
<tr>
<td>Post, Jonathan V.</td>
<td>8697</td>
<td>AN INVESTIGATION OF PROGRAMMING PRACTICES IN SELECTED AIR FORCE PROJECTS</td>
</tr>
<tr>
<td></td>
<td>4383</td>
<td>SOFTWARE INTEROPERABILITY AND REUSABILITY</td>
</tr>
<tr>
<td></td>
<td>4384</td>
<td>SOFTWARE INTEROPERABILITY AND REUSABILITY</td>
</tr>
<tr>
<td></td>
<td>4394</td>
<td>SOFTWARE QUALITY MEASUREMENT FOR DISTRIBUTED SYSTEMS GUIDEBOOK FOR SOFTWARE QUALITY</td>
</tr>
<tr>
<td></td>
<td>4395</td>
<td>SOFTWARE QUALITY MEASUREMENT FOR DISTRIBUTED SYSTEMS</td>
</tr>
<tr>
<td></td>
<td>5417</td>
<td>SOFTWARE QUALITY MEASUREMENT FOR DISTRIBUTED SYSTEMS - DISTRIBUTED COMPUTING SYSTEMS: IMPACT ON SOFTWARE QUALITY</td>
</tr>
<tr>
<td>Presson, Edward P.</td>
<td>4383</td>
<td>SOFTWARE INTEROPERABILITY AND REUSABILITY</td>
</tr>
<tr>
<td></td>
<td>4384</td>
<td>SOFTWARE INTEROPERABILITY AND REUSABILITY</td>
</tr>
<tr>
<td></td>
<td>4394</td>
<td>SOFTWARE QUALITY MEASUREMENT FOR DISTRIBUTED SYSTEMS GUIDEBOOK FOR SOFTWARE QUALITY</td>
</tr>
<tr>
<td></td>
<td>4395</td>
<td>SOFTWARE QUALITY MEASUREMENT FOR DISTRIBUTED SYSTEMS</td>
</tr>
<tr>
<td></td>
<td>5417</td>
<td>SOFTWARE QUALITY MEASUREMENT FOR DISTRIBUTED SYSTEMS - DISTRIBUTED COMPUTING SYSTEMS: IMPACT ON SOFTWARE QUALITY</td>
</tr>
<tr>
<td>Pyster, A.</td>
<td>15377</td>
<td>SOFTWARE TESTING MEASURES</td>
</tr>
<tr>
<td>Randall, W.</td>
<td>7868</td>
<td>METHODOLOGY FOR SOFTWARE RELIABILITY PREDICTION</td>
</tr>
<tr>
<td></td>
<td>7869</td>
<td>METHODOLOGY FOR SOFTWARE RELIABILITY PREDICTION</td>
</tr>
<tr>
<td>Rasmussen, Normand</td>
<td>15381</td>
<td>NATIONAL SOFTWARE WORKS TOOL</td>
</tr>
<tr>
<td>Author</td>
<td>Page Number</td>
<td>Title</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Rice, Robert W.</td>
<td>15413</td>
<td>Integration Studies: Interactive Communication Systems Modeling Study</td>
</tr>
<tr>
<td>Richards, Paul K.</td>
<td>15401</td>
<td>Factors in Software Quality, Volume III. Preliminary Handbook on Software Quality for an Acquisition Manager</td>
</tr>
<tr>
<td></td>
<td>214</td>
<td>Factors in Software Quality, Concepts and Definitions of Software Quality</td>
</tr>
<tr>
<td></td>
<td>215</td>
<td>Factors in Software Quality, Metric Data Collection and Validation</td>
</tr>
<tr>
<td></td>
<td>216</td>
<td>Factors in Software Quality, Preliminary Handbook on Software Quality for an Acquisition Manager</td>
</tr>
<tr>
<td>Rosenfeld, Pilar</td>
<td>15384</td>
<td>Ada Test and Verification System (ATVS)</td>
</tr>
<tr>
<td>Rosner, Andrew J.</td>
<td>6458</td>
<td>Automating Software Design Metrics Engineering for Artificial Intelligence Software</td>
</tr>
<tr>
<td>Rushby, John</td>
<td>15352</td>
<td>Software Quality Measurement for Distributed Systems Guidebook for Software Quality</td>
</tr>
<tr>
<td>Ruston, Henry</td>
<td>15392</td>
<td>Summary of Technical Progress, Investigation of Software Models</td>
</tr>
<tr>
<td></td>
<td>15399</td>
<td>Software Modeling Studies, Volume I. Summary of Technical Progress</td>
</tr>
<tr>
<td></td>
<td>15412</td>
<td>Summary of Technical Progress, Software Modeling Studies</td>
</tr>
<tr>
<td></td>
<td>15414</td>
<td>Summary of Technical Progress, Software Modeling Studies</td>
</tr>
<tr>
<td></td>
<td>3849</td>
<td>Software Modeling Studies, Executive Summary</td>
</tr>
<tr>
<td></td>
<td>3850</td>
<td>Software Modeling Studies - The Polynomial Measure of Complexity</td>
</tr>
<tr>
<td></td>
<td>818</td>
<td>Summary of Technical Progress - Software Modeling Studies</td>
</tr>
<tr>
<td></td>
<td>969</td>
<td>Summary of Technical Progress, Investigation of Software Models</td>
</tr>
<tr>
<td>Ruston, Henry</td>
<td>15418</td>
<td>Summary of Technical Progress Software Modeling Studies</td>
</tr>
<tr>
<td>Schaezner, J. P.</td>
<td>15411</td>
<td>Modern Programming Practices Study Report</td>
</tr>
<tr>
<td>Schmidt, Robert L.</td>
<td>4383</td>
<td>Software Interoperability and Reusability</td>
</tr>
<tr>
<td></td>
<td>4384</td>
<td>Software Interoperability and Reusability</td>
</tr>
<tr>
<td></td>
<td>4394</td>
<td>Software Quality Measurement for Distributed Systems Guidebook for Software Quality</td>
</tr>
<tr>
<td></td>
<td>4395</td>
<td>Software Quality Measurement for Distributed Systems</td>
</tr>
<tr>
<td></td>
<td>5417</td>
<td>Software Quality Measurement for Distributed Systems - Distributed Computing Systems: Impact on Software Quality</td>
</tr>
<tr>
<td>Senn, Roy</td>
<td>7868</td>
<td>Methodology for Software Reliability Prediction</td>
</tr>
<tr>
<td></td>
<td>7869</td>
<td>Methodology for Software Reliability Prediction</td>
</tr>
<tr>
<td>Sentz, Donald R.</td>
<td>15413</td>
<td>Interactive Communication Systems Modeling Study</td>
</tr>
<tr>
<td>Sharp, Michael</td>
<td>15387</td>
<td>Cobol Automated Verification System: Study Phase</td>
</tr>
<tr>
<td>Author</td>
<td>Page No.</td>
<td>Title</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------</td>
<td>---------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Sheppard, Sylvia B.</td>
<td>15390</td>
<td>EVALUATION OF SOFTWARE LIFE CYCLE DATA FROM THE PAVE PAWS PROJECT.</td>
</tr>
<tr>
<td>Shooman, Martin L.</td>
<td>15392</td>
<td>SUMMARY OF TECHNICAL PROGRESS, INVESTIGATION OF SOFTWARE MODELS</td>
</tr>
<tr>
<td>Siefert, Paul T.</td>
<td>15399</td>
<td>SOFTWARE MODELING STUDIES, VOLUME I.</td>
</tr>
<tr>
<td>Slavinski, Richard T.</td>
<td>15412</td>
<td>SUMMARY OF TECHNICAL PROGRESS, SOFTWARE MODELING STUDIES</td>
</tr>
<tr>
<td>Smith, Rita M.</td>
<td>15414</td>
<td>SUMMARY OF TECHNICAL PROGRESS, SOFTWARE MODELING STUDIES</td>
</tr>
<tr>
<td>Sodano, Nancy M.</td>
<td>15418</td>
<td>SUMMARY OF TECHNICAL PROGRESS, SOFTWARE MODELING STUDIES</td>
</tr>
<tr>
<td>Stickel, Mark E.</td>
<td>3849</td>
<td>SOFTWARE MODELING STUDIES, EXECUTIVE SUMMARY</td>
</tr>
<tr>
<td>Tsai, J.E.</td>
<td>818</td>
<td>SUMMARY OF TECHNICAL PROGRESS - SOFTWARE MODELING STUDIES</td>
</tr>
<tr>
<td>Siefert, Paul T.</td>
<td>818</td>
<td>SOFTWARE MODELING STUDIES, EXECUTIVE SUMMARY</td>
</tr>
<tr>
<td>Slavinski, Richard T.</td>
<td>969</td>
<td>SUMMARY OF TECHNICAL PROGRESS - INVESTIGATION OF SOFTWARE MODELS</td>
</tr>
<tr>
<td>Smith, Rita M.</td>
<td>15371</td>
<td>AUTOMATED MEASUREMENT SYSTEM (AMS).</td>
</tr>
<tr>
<td>Sodano, Nancy M.</td>
<td>15378</td>
<td>C3I ENVIRONMENT MEASUREMENT INSTRUMENTATION</td>
</tr>
<tr>
<td>Stickel, Mark E.</td>
<td>1430</td>
<td>STATIC FORTRAN ANALYZER</td>
</tr>
<tr>
<td>Strelitch, Thomas P.</td>
<td>15363</td>
<td>SOFTWARE FAULT TOLERANCE DESIGN SYSTEM.</td>
</tr>
<tr>
<td>Stucki, Leon</td>
<td>6458</td>
<td>AUTOMATING SOFTWARE DESIGN METRICS</td>
</tr>
<tr>
<td>Sullivan, J.E.</td>
<td>6458</td>
<td>ENGINEERING FOR ARTIFICIAL INTELLIGENCE SOFTWARE</td>
</tr>
<tr>
<td>Szulewski, Paul A.</td>
<td>15384</td>
<td>ADA TEST AND VERIFICATION SYSTEM (ATVS)</td>
</tr>
<tr>
<td>Templeton, M. P.</td>
<td>15391</td>
<td>DMA MODERN PROGRAMMING ENVIRONMENT STUDY</td>
</tr>
<tr>
<td>Tsai, Jay T.</td>
<td>62</td>
<td>ENGINEERING OF QUALITY SOFTWARE SYSTEMS (MEASURING THE COMPLEXITY OF COMPUTER SOFTWARE)</td>
</tr>
<tr>
<td>Tsai, Juijen</td>
<td>6794</td>
<td>SPECIFICATION OF SOFTWARE QUALITY ATTRIBUTES</td>
</tr>
<tr>
<td></td>
<td>6795</td>
<td>SPECIFICATION OF SOFTWARE QUALITY ATTRIBUTES SOFTWARE QUALITY SPECIFICATION GUIDEBOOK</td>
</tr>
<tr>
<td></td>
<td>6796</td>
<td>SPECIFICATION OF SOFTWARE QUALITY ATTRIBUTES SOFTWARE QUALITY EVALUATION GUIDEBOOK</td>
</tr>
<tr>
<td>Tsai, Juijen</td>
<td>4383</td>
<td>SOFTWARE INTEROPERABILITY AND REUSABILITY</td>
</tr>
<tr>
<td></td>
<td>4384</td>
<td>SOFTWARE INTEROPERABILITY AND REUSABILITY</td>
</tr>
<tr>
<td></td>
<td>4394</td>
<td>SOFTWARE QUALITY MEASUREMENT FOR DISTRIBUTED SYSTEMS GUIDEBOOK FOR SOFTWARE QUALITY</td>
</tr>
<tr>
<td></td>
<td>4395</td>
<td>SOFTWARE QUALITY MEASUREMENT FOR DISTRIBUTED SYSTEMS</td>
</tr>
<tr>
<td></td>
<td>5417</td>
<td>SOFTWARE QUALITY MEASUREMENT FOR DISTRIBUTED SYSTEMS - DISTRIBUTED COMPUTING SYSTEMS: IMPACT ON SOFTWARE QUALITY</td>
</tr>
<tr>
<td>Author(s)</td>
<td>ISBN</td>
<td>Title</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Vandenberg, S.J.</td>
<td>5844</td>
<td>RELIABILITY MODEL DEMONSTRATION STUDY. VOLUME 1</td>
</tr>
<tr>
<td></td>
<td>5845</td>
<td>RELIABILITY MODEL DEMONSTRATION STUDY. VOLUME 2</td>
</tr>
<tr>
<td>Vick, Charles R.</td>
<td>15363</td>
<td>SOFTWARE FAULT TOLERANCE DESIGN SYSTEM.</td>
</tr>
<tr>
<td>Vienneau, Robert</td>
<td>7868</td>
<td>METHODOLOGY FOR SOFTWARE RELIABILITY PREDICTION</td>
</tr>
<tr>
<td></td>
<td>7869</td>
<td>METHODOLOGY FOR SOFTWARE RELIABILITY PREDICTION</td>
</tr>
<tr>
<td>Virklcr, Gary W.</td>
<td>15428</td>
<td>RASTER-LINEAL EDITING SOFTWARE</td>
</tr>
<tr>
<td>Waldinger, Richard J.</td>
<td>15352</td>
<td>ENGINEERING FOR ARTIFICIAL INTELLIGENCE SOFTWARE</td>
</tr>
<tr>
<td>Walters, Gene F.</td>
<td>15401</td>
<td>FACTORS IN SOFTWARE QUALITY. VOLUME-III. PRELIMINARY HANDBOOK ON SOFTWARE QUALITY FOR AN ACQUISITION MANAGER</td>
</tr>
<tr>
<td></td>
<td>214</td>
<td>FACTORS IN SOFTWARE QUALITY. CONCEPTS AND DEFINITIONS OF SOFTWARE QUALITY</td>
</tr>
<tr>
<td></td>
<td>215</td>
<td>FACTORS IN SOFTWARE QUALITY. METRIC DATA COLLECTION AND VALIDATION</td>
</tr>
<tr>
<td></td>
<td>216</td>
<td>FACTORS IN SOFTWARE QUALITY. PRELIMINARY HANDBOOK ON SOFTWARE QUALITY FOR AN ACQUISITION MANAGER</td>
</tr>
<tr>
<td>Wang, Yeou-Wei</td>
<td>15354</td>
<td>KNOWLEDGE BASED QUALITY ASSURANCE TOOLS</td>
</tr>
<tr>
<td>Warthman, James L.</td>
<td>13933</td>
<td>SOFTWARE QUALITY MEASUREMENT DEMONSTRATION PROJECT (I.)</td>
</tr>
<tr>
<td>Weedon, Margaret</td>
<td>15379</td>
<td>IMPROVING SOFTWARE QUALITY ASSURANCE METHODS</td>
</tr>
<tr>
<td>Welch, Howard O.</td>
<td>15363</td>
<td>SOFTWARE FAULT TOLERANCE DESIGN SYSTEM.</td>
</tr>
<tr>
<td>Wigle, Gary B.</td>
<td>6794</td>
<td>SPECIFICATION OF SOFTWARE QUALITY ATTRIBUTES</td>
</tr>
<tr>
<td></td>
<td>6795</td>
<td>SPECIFICATION OF SOFTWARE QUALITY ATTRIBUTES SOFTWARE QUALITY GUIDEBOOK</td>
</tr>
<tr>
<td></td>
<td>6796</td>
<td>SPECIFICATION OF SOFTWARE QUALITY ATTRIBUTES SOFTWARE QUALITY EVALUATION GUIDEBOOK</td>
</tr>
<tr>
<td>Willmorth, Norman E.</td>
<td>15406</td>
<td>AN INVESTIGATION OF PROGRAMMING PRACTICES IN SELECTED AIR FORCE PROJECTS</td>
</tr>
<tr>
<td></td>
<td>15415</td>
<td>AN ANALYSIS OF SOFTWARE DATA COLLECTION PROBLEMS AND CURRENT CAPABILITIES</td>
</tr>
<tr>
<td>Willson, D. M.</td>
<td>15411</td>
<td>MODERN PROGRAMMING PRACTICES STUDY REPORT</td>
</tr>
<tr>
<td>Wisehart, William R.</td>
<td>15384</td>
<td>ADA TEST AND VERIFICATION SYSTEM (ATVS)</td>
</tr>
<tr>
<td></td>
<td>15407</td>
<td>JAVS FINAL REPORT</td>
</tr>
<tr>
<td>Worrells, Suellen</td>
<td>8697</td>
<td>SOFTWARE QUALITY MEASUREMENT DEMONSTRATION PROJECT II</td>
</tr>
<tr>
<td>Yates, P.</td>
<td>7868</td>
<td>METHODOLOGY FOR SOFTWARE RELIABILITY PREDICTION</td>
</tr>
<tr>
<td></td>
<td>7869</td>
<td>METHODOLOGY FOR SOFTWARE RELIABILITY PREDICTION</td>
</tr>
<tr>
<td>Yau, Stephen S.</td>
<td>1496</td>
<td>PERFORMANCE RIPPLE EFFECT ANALYSIS FOR LARGE-SCALE SOFTWARE MAINTENANCE</td>
</tr>
<tr>
<td></td>
<td>1497</td>
<td>SELF-METRIC SOFTWARE. SUMMARY OF TECHNICAL PROGRESS</td>
</tr>
<tr>
<td></td>
<td>15354</td>
<td>KNOWLEDGE BASED QUALITY ASSURANCE</td>
</tr>
</tbody>
</table>
TOOLS
15393 ON SOFTWARE RELIABILITY MODELING
4519 METHODOLOGY FOR SOFTWARE MAINTENANCE
The Data & Analysis Center for Software (DACS) is a Department of Defense (DoD) Information Analysis Center (IAC), administratively managed by the Defense Technical Information Center (DTIC) under the DoD IAC Program. The DACS is technically managed by Rome Laboratory (RL). Kaman Sciences Corporation manages and operates the DACS, serving as a source for current, readily available data and information concerning software engineering and software technology.