

RL-TR-91-407, Vol I (of two)
Final Technical Report
December 1991

AD-A252 679



QUALITY EVALUATION SYSTEM (QUES)

Software Productivity Solutions, Inc.

Karen A. Dyson



APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

92-17044



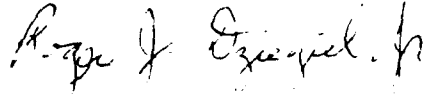
92 6 29 037

Rome Laboratory
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

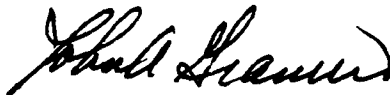
RL-TR-91-407, Volume I (of two) has been reviewed and is approved for publication.

APPROVED:



ROGER J. DZIEGIEL, JR.
Project Engineer

FOR THE COMMANDER:



JOHN A. GRANIERO
Chief Scientist
Command, Control, and Communications

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify RL(3CB) Griffiss AFB, NY 13441-5700. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Washington Headquarters Service, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave Blank)		2. REPORT DATE December 1991		3. REPORT TYPE AND DATES COVERED Final Dec 87 - Sep 91	
4. TITLE AND SUBTITLE QUALITY EVALUATION SYSTEM (QUES)				5. FUNDING NUMBERS C - F30602-88-C-0019 PE - 63728F PR - 2527 TA - 03 WU - 15	
6. AUTHOR(S) Karen A. Dyson					
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Software Productivity Solutions, Inc. 122 N. 4th Avenue Indialantic FL 32903				8. PERFORMING ORGANIZATION REPORT NUMBER A015	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Rome Laboratory (C3CB) Griffiss AFB NY 13441-5700				10. SPONSORING/MONITORING AGENCY REPORT NUMBER RL-TR-91-407, Vol I (of two)	
11. SUPPLEMENTARY NOTES Rome Laboratory Project Engineer: Roger J. Dziegiel, Jr/C3CB/(315) 330-4476					
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.				12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) 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.					
14. SUBJECT TERMS Software Quality, Software Evaluation, Metrics, Software Quality Indicators				15. NUMBER OF PAGES 108	
				16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL		

Table of Contents

List of Figures.....	vi
List of Tables	viii
1.0 Introduction	1
1.1 Contents of this Technical Report	1
2.0 Software Quality Evaluation Frameworks.....	4
2.1 Background	4
2.2 Rome Laboratory Software Quality Framework Overview.....	5
2.2.1 The Quality Model.....	7
2.2.2 Metric Elements	9
2.2.3 Scoring Methodology	9
2.3 Indicators Framework Overview.....	12
3.0 The QUES Tool.....	14
3.1 The QUES Database.....	14
3.2 Predefined Database Components	18
3.3 Tool Interfaces.....	20
3.4 QUES Implementation of RSQF	20
3.4.1 Framework Structure.....	22
3.4.2 Project Architecture Levels.....	24
3.4.3 Scoring	24
3.5 QUES Implementation of SQMI	26
3.5.1 Framework Structure.....	26
3.5.2 Scoring	26
3.6 QUES Operational Scenario.....	27
3.6.1 Framework Manager.....	27
3.6.2 Acquisition Manager	27
3.6.3 Project Manager.....	29
3.6.4 Engineer.....	29

4.0	Framework Definition.....	30
4.1	Establishing a Framework	30
4.1.1	Creating a New Framework.....	32
4.1.1.1	Defining the hierarchical structure	32
4.1.1.2	Adding framework elements.....	34
4.1.1.3	Defining the scoring equations.....	40
4.1.1.4	Setting up automated data collection	44
4.1.1.5	Creating reports	44
4.1.2	Using a predefined framework component	52
4.2	Tailoring a Predefined Framework	52
4.2.1	Importing the predefined RSQF	52
4.2.2	Reducing the Number of Factors	54
4.2.3	Other Framework Modifications.....	56
4.3	Validating the Framework	57
5.0	Tailoring and Goal Setting.....	58
5.1	Tailoring the Applicability of a Framework Member.....	59
5.2	Setting Goals	60
6.0	Project Creation	61
6.1	Associating a Framework with the Project.....	61
6.2	Defining Project Users and Roles.....	61
6.3	Establishing the Project Static Structure	62
6.4	Collecting Project Data	64
6.5	Computing Quality Scores.....	66
6.6	Evaluation of Project Quality with Reports.....	67
6.6.1	Identifying a Problem.....	67
6.6.2	Evaluating Compliance.....	69
6.6.3	Comparing Project Components	69
6.6.4	Trend Reports	69

Quality Evaluation System

7.0 Data Collection	74
7.1 Data Collection Forms	74
7.2 Automated Data Collection Tools.....	76
8.0 Conclusions and Recommendations	80
List of References.....	81
Acronyms List	82
Appendix A -- Analysis of Questions in the RSQF	83
Appendix B -- Cross Reference of Questions in the RSQF to Automated Data Collection Tools.....	89



Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

List of Figures

Figure 2.2-1.	The RL Software Quality Model.....	6
Figure 2.2.1-1.	RSQF Factors and Associated Criteria.	8
Figure 3.1-1.	Components of the QUES Database.	15
Figure 3.1-2.	Possible Framework and Project Level Definitions.....	17
Figure 3.1-3.	Example of an Unlimited Number of Project Levels.....	17
Figure 3.3-1.	QUES External Interfaces.....	21
Figure 3.6-1.	Tasks Performed by QUES Users.	28
Figure 4.1-1.	Options for Establishing a Framework.....	31
Figure 4.1.1-1.	Requirements of the Code Size Metrics Framework.	33
Figure 4.1.1.2-1.	CSM Framework Structure Table and Diagram.	35
Figure 4.1.1.2-2.	CSM Framework as Displayed in QUES.	37
Figure 4.1.1.2-3.	Abstract and Specific Definition Windows.	39
Figure 4.1.1.3-1.	Equation Definition Window.....	41
Figure 4.1.1.5-1.	CSM Reports: Framework Members and Equations.....	47
Figure 4.1.1.5-2.	CSM Report: Framework Question Summary.....	48
Figure 4.1.1.5-3.	CSM Report: Framework Automated Questions.	49
Figure 4.1.1.5-4.	CSM Report: Data Collection Form.....	50
Figure 4.2.1-1.	The Structure of Phase B of the RSQF Framework.	53
Figure 4.2.2-1.	RSQF Phase B after Deletion of Unrelated Factors and Criteria	55
Figure 6.2-1.	User Role Definition Window.....	63
Figure 6.4-1.	Three Examples of Annotation of Answers.....	65
Figure 6.6.1-1.	Factor Score Report.....	68
Figure 6.6.1-2.	Comparison of Criteria Scores for a Single Factor.	68
Figure 6.6.2-1.	Compliance Report.....	70
Figure 6.6.3-1.	Comparison of Factor Scores for Three CSCI's.....	71
Figure 6.6.4-1.	Comparison of Factor Scores Over Three Phases.....	72
Figure 6.6.4-2.	Trend Plot of a Factor Score in a Single Phase.	73

List of Figures (Continued)

Figure 7.1-1.	An Example DCF constrained on the Attribute "constant".	75
Figure 7.2-1.	Tool Association Summary Report.	77
Figure 7.2-2.	Answer Summary Report.....	79
Figure A-1.	Number of Questions at Each Project Level.	81
Figure A-2.	Number of Questions for Each Phase.	81
Figure A-3.	Number of Questions in Each Phase by Project Level.	82
Figure A-4.	Number of Questions by Criterion.	83

List of Tables

Table 2.2.2-a.	RSQF Data Collection Forms.	10
Table 2.2.2-b.	Levels Questions are Asked in the RSQF Framework.....	10
Table 2.3-a.	Management and Quality Indicators.	13
Table 3.1.-a.	Example Project Level Partitioning with Unlimited Number of Levels.	16
Table 3.1-b.	Comparison of Framework Abstract and Specific Members.	19
Table 3.2-a.	Predefined QUES Database Components.....	19
Table 3.4.1-a.	RSQF Framework Level Definition.....	23
Table 3.4.1-b.	RSQF Phase Framework Names in QUES.....	23
Table 3.4.2-a.	RSQF Project Level Definition.	25
Table 3.5.1-a.	SMQI Framework Level Definition.....	26
Table 4.1.1-a.	Level Definitions for CSM Framework.	33
Table 4.1.1.2-a.	Specific Members of CSM Framework.	37
Table 4.1.1.2-b.	Summary of CSM Framework Equations.....	39
Table 4.1.1.3-a.	Summary of Scoring Equations for CSM Framework.....	43
Table 4.2.2-a.	Criteria and Metric Elements Related to Factors Reliability and Maintainability.	55

1.0 Introduction

This technical report showcases the features of the QUality Evaluation System (QUES). The QUES tool, which will be referred to as "QUES", is an automated software quality evaluation framework management and application tool. QUES makes software metrics practical by automating the process of applying a software quality evaluation framework to a software development project. A framework may be applied at any point in the life cycle of an on-going development project, or may be used to evaluate the quality of an existing product.

1.1 Contents of this Technical Report

This technical report begins with two overview sections:

- **Software Quality Evaluation Frameworks**
- **The QUES Tool**

which provide background information about the software quality evaluation frameworks that QUES was designed to automate and about the QUES tool itself. These overview sections provide a foundation for the discussion of the QUES features which are the main topic of this report.

The next four sections discuss QUES features in four main categories which correspond to steps in a typical operational scenario:

- **Framework Definition**
- **Tailoring and Goal Setting**
- **Project Creation**
- **Data Collection**

Two frameworks are presented in the **Software Quality Evaluation Frameworks** section:

- Rome Laboratory Software Quality Framework (RSQF) [BOW85]
- Air Force Systems Command Software Management and Quality Indicators (SMQI) framework [AFS86] and [AFS87]

These frameworks are by no means the only frameworks that the QUES tool will support. In fact, QUES frameworks are completely user-defined. The two frameworks listed above are predefined for QUES for the user's convenience and may be customized.

The **QUES Tool** section consists of six sub-sections:

- The QUES Database
- QUES Database Components
- Tool Interfaces
- Implementation of RSQF
- Implementation of SMQI
- QUES Operational Scenario

The first sub-section introduces QUES terminology, the characteristics of a QUES framework, and the overall method by which a framework is established and applied to a software development project. More detail about the QUES features is presented in the later sections of this report.

The **Database Components** sub-section describes the break-down of the QUES database into its building blocks. An important feature of QUES is that database components may be exported from and imported to the database so that they may be shared. The predefined frameworks (RSQF and SMQI) are delivered in the form of a framework database component.

The **Tool Interfaces** sub-section, describes the external tools with which QUES interfaces. These external tools are:

- Rome Laboratory Software Life Cycle Support Environment (SLCSE)
- NASA Goddard's Static Analyzer Program (SAP)
- Virginia Tech's Ada Metric Analyzer

The next two sub-sections (**QUES Implementation of RSQF** and **QUES Implementation of SMQI**) describe how the two predefined frameworks are

implemented in QUES. Such information as the framework structure, hierarchy levels and scoring are presented.

The final sub-section of The QUES Tool section is the **Operational Scenario**. This sub-section describes the types of QUES users and their tasks, in the order in which they are likely to be performed. The operational scenario forms the basis for presenting the features of QUES later in the report.

The remainder of this report consists of four sections which correspond to major QUES functions: framework definition, tailoring and goal setting, project creation, and data collection. An example of an application of QUES to a software project from start to finish illustrates how QUES is used. Actual QUES window displays and reports are presented.

Volume II of this report contains complete documentation of the RSQF framework implemented in the QUES tool. Upgrades were made to the original framework as documented in [BOW85], including:

- improved objectivity of questions
- improved automatability of questions
- improved applicability to Ada
- consistency with DoD-STD-2167A life cycle phases and terminology

2.0 Software Quality Evaluation Frameworks

This section presents the background of the creation of the QUES tool and the software quality frameworks that the tool automates. This framework overview section is followed by a description of the QUES tool.

2.1 Background

Traditionally, software quality evaluation has been an expensive and labor intensive process. This situation has worsened as software systems grow increasingly large and complex. To combat this situation the Rome Laboratory (RL) [(formerly known as Rome Air Development Center (RADC))] has been engaged in research over the last 15 years that has resulted in a formalized process of evaluating software quality. The formalized process is documented in the form of an evaluation framework, the RL Software Quality Framework (RSQF). Parallel effort by the Air Force Systems Command (AFSC) has resulted in the definition of management and quality indicators for software. These indicators have become the AFSC Software Management and Quality Indicators (SMQI) framework. Together these technologies provide frameworks for evaluating the software development process and products. The impetus behind QUES was to automate the application of these frameworks.

QUES assists in both the management and the application of an evaluation framework. The framework management aspects of QUES allow for refinement and adaptation of the formalized process of evaluating the quality of software systems. Software quality measurement experts will use the framework management capabilities to improve the evaluation framework as new technology dictates.

QUES supports the application of the evaluation framework to software development projects by providing interactive quality engineering support to personnel who are actively involved in the development of mission critical computer resource software. QUES allows its users to specify, assess, and consequently achieve their quality goals.

Mechanisms are provided to track and then present quality information about the development process and products over the course of the development life cycle. It is this feedback loop that allows specific projects to realize the immediate benefit of the quality evaluation process. Due to this feedback, the use of QUES will ultimately improve the quality of software products and the productivity of the software development organizations. The result is higher quality software at reduced cost.

2.2 Rome Laboratory Software Quality Framework Overview

The software quality evaluation framework developed by RL is known as the Rome Laboratory Software Quality Framework (RSQF). The basis of the framework is a quality model which establishes a hierarchical relationship between a user-oriented quality *factor* at the top level, and software-oriented attributes at the second and third levels (*criteria* and *metrics*, respectively). The model is shown in Figure 2.2-1. Software quality is predicted and measured by the presence, absence, or degree of identifiable software attributes.

The next three sub-sections provide an overview of the RSQF. The quality model sub-section shows the relationship between factors and criteria which is the essence of the quality model. The metric element sub-section describes how metric elements are organized. And the scoring methodology sub-section describes how quality scores are computed from the elements of the framework.

Volume II of this report contains a complete description of the RSQF, including the Data Collection Forms and scoring equations. Appendix A contains a summary of the questions grouped by life cycle phase, by architecture level, and by criterion. This appendix is helpful when estimating the cost to apply the RSQF. Appendix B contains a cross reference of the RSQF questions and the metrics collected by the automated data collection tools that interface with QUES. Reference is made to Appendix B in section 7 of this report which discusses data collection.

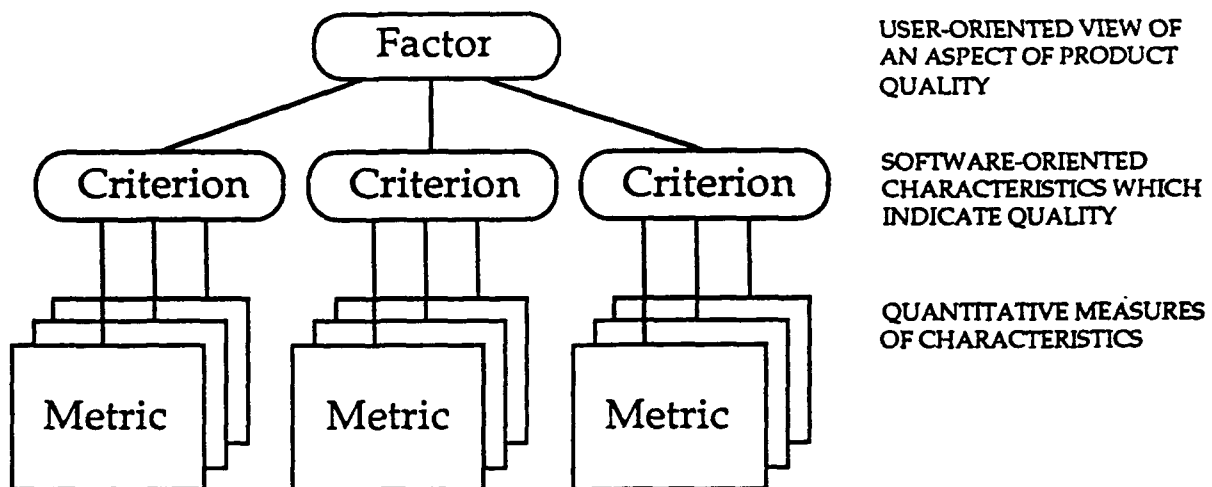


Figure 2.2-1. The RL Software Quality Model.

2.2.1 The Quality Model

The RSQF contains a total of 13 software quality factors, 29 criteria, and 74 metrics. Refer to Volume II for a complete list of and descriptions of these components of the framework. The essence of the RSQF quality model is the relationship that describes how criteria combine to form factors. Figure 2.2.1-1 illustrates the factor/criteria relationship.

Notice that some criteria are related to more than one factor. For example, the criterion Anomaly Management is used to compute the factors Reliability and Survivability. The ability of a system to provide continuity of operations under anomalous conditions (Anomaly Management) is related to both the frequency of failures (Reliability) and the ability to recover from a failure of a portion of the system (Survivability). In contrast, the three Effectiveness criteria are used only by the factor Efficiency.

The number of criteria related to a factor varies from one, in the case of Integrity, to nine in the case of Reusability. In fact, two factors, Flexibility and Portability, are both subsets of the factor Reusability. Shared criteria indicate a complementary relationship between the factors. A high Flexibility rating is an indicator of high Reusability. The factor/criteria chart does not express, however, that some factors have an inverse relationship. For example, high Efficiency may lead to low Portability. Any set of factors with no shared criteria may have an inverse relationship.

Quality Evaluation System

FACTOR		E F F I C I E N C Y	I N T E G R I T Y	R E L I A B I L I T Y	S U R V I V A B I L I T Y	U S A B I L I T Y	C O R R E C T N E S S	M A I N T A I N A B I L I T Y	V E R I F I A B I L I T Y	E X P A N D A B I L I T Y	F L E X I B I L I T Y	I N T E R O P E R A B I L I T Y	P O R T A B I L I T Y	R E U S A B I L I T Y	
Criterion	Acronym														
Accuracy	AC			x											
Anomaly Management	AM			x	x										
Autonomy	AU				x										
Distributedness	DI				x										
Effectiveness - Communication	EC	x													
Effectiveness - Processing	EP	x													
Effectiveness - Storage	ES	x													
Operability	OP					x									
Reconfigurability	RE				x										
System Accessibility	SS		x												
Training	TN					x									
Completeness	CP						x								
Consistency	CS						x	x							
Traceability	TC						x								
Visibility	VS							x	x						
Application Independence	AP														x
Augmentability	AT									x					
Commonality	CL											x			
Document Accessibility	DO														x
Functional Overlap	FO											x			
Functional Scope	FS														x
Generality	GE									x	x				x
Independence	ID											x	x		x
System Clarity	ST														x
Virtuality	VR									x					
System Compatibility	SY											x			
Modularity	MO				x			x	x	x	x	x	x	x	x
Self-Descriptiveness	SD							x	x	x	x			x	x
Simplicity	SI			x				x	x	x	x				x

Figure 2.2.1-1. RSQF Factors and Associated Criteria.

2.2.2 Metric Elements

Metrics are comprised of *metric elements* which are specific questions applied to a project under development to assess and predict quality. The metric elements of the RSQF are grouped into a Data Collection Form (DCF). All DCF's are included in Volume II. Each DCF is organized to correspond to a phase of the development process. These phases and the associated DCF's are given in the Table 2.2.2-a.

Metric elements gather information at various levels of the software implementation architecture. Information can be collected based on system-wide questions, on questions applicable to each Computer Software Configuration Item (CSCI), on questions applicable to each Computer Software Component (CSC), and on questions applicable to each Computer Software Unit (CSU). Table 2.2.2-b illustrates how the levels of the questions asked for each DCF vary with the stage of the development life cycle.

2.2.3 Scoring Methodology

Scoring the RSQF is the process of combining metric element results into increasingly higher levels of abstraction. Rather than looking individually at the over 1400 metric element data items applied repeatedly for the various software phases and components, the scoring process allows the user to abstract and combine these results into meaningful calculations. These scoring results support differing views of the system under analysis. The scoring methodologies are presented in detail in the Scoring section of Volume II.

Two scoring methodologies are defined for the RSQF:

1. Adherence scoring
2. Aggregation scoring

In adherence scoring, the score corresponds to a ratio of compliances to opportunities for a particular metric element. Only "adherence-oriented" metrics are scored this way. An example of such a score is 2,500/5,000. With this method, the ratio is NOT reduced to 1/2.

System Requirements Analysis/Design	DCF A
Software Requirements Analysis	DCF B
Preliminary Design	DCF C
Detailed Design	DCF D
Coding and CSU Testing	DCF E
CSC Integration and Test	DCF F
CSCI Testing	DCF G
System Testing	DCF H
Operational Test and Evaluation	DCF I

Table 2.2.2-a. RSQF Data Collection Forms

DCF	Level of Questions:			
	System	CSCI	CSC	CSU
A	x			
B	x	x		
C		x	x	
D	x	x	x	x
E	x	x	x	x
F	x	x	x	x
G	x	x		
H	x			
I	x			

Table 2.2.2-b. Levels Questions are Asked in the RSQF Framework.

In aggregation scoring, aggregations of metric scores are made across the following levels:

- scoring across the elements of the framework (metric elements, metrics, criteria, and factors)
- scoring across the architecture of the system (CSU, CSC, CSCI, and the system itself)
- scoring across the functionality of the system in evaluating quality results for the system-level capabilities

Aggregation scores are decimal numbers in the range of 0.0 to 1.0 where 1.0 represents a perfect score. Metric elements are either a yes/no question or a ratio of two values. An example of an aggregation score is 0.75.

Framework element scores are "aggregated" across levels with a weighted average. Aggregation across architecture levels occurs only for the metric scoring. For example, *CSC-level* metric scores are the weighted average of the *CSU-level* metric scores of the subordinate CSU components of that CSC. Aggregation across framework elements occurs for every framework level score. For example, *factor* scores are the weighted average of the applicable *criterion* scores, at each architecture level. A metric scoring equation can combine aggregation across framework elements as well as aggregation across architecture levels. For example, the *CSCI-level metric* score is the weighted average of the *CSCI-level metric elements* plus the weighted average of the *CSC-level metric* scores of the subordinate CSC's. More examples are presented in the section on the QUES implementation of the framework.

2.3 Indicators Framework Overview

The AFSC Indicators (SMQI) framework is the product of two AFSC pamphlets [800-14 and 800-43] which define a set of management and quality indicators. Refer to the pamphlets for details concerning the indicators framework. The intent of the pamphlets is to provide a common sense approach to gaining insight into the software development process. The management indicators are designed to reflect the status of software development in an acquisition program. The quality indicators provide insight into the quality, reliability, and maintainability of the software products being developed.

The framework consists of two sets of indicators: the 6 management indicators, and the 7 quality indicators. Table 2.3-a. lists the indicators and a brief description of each. An indicator is comprised of one or more metrics. Data is gathered and presented at the CSCI level. Most indicators are best represented graphically, showing progress over time.

Indicator Set	Indicator	Description
Management	Computer Resource Utilization	degree of memory, CPU, and I/O utilization
	Software Development Manpower	contractor and program office staffing (plan vs. actual)
	Requirements Definition and Stability	counts untestable and untraceable requirements; tracks open ECP's and action items
	Software Progress--Development and Test	tracks completion of design, unit test, and integration; test progress and open problem reports
	Cost/Schedule Deviations	tracks cost and estimates cost at completion
	Software Development Tools	tracks availability of development tools
Quality	Completeness	assesses adequacy of specifications and design
	Design Structure	assesses simplicity of design
	Defect Density	tracks defects discovered and corrected during design and code
	Fault Density	tracks faults discovered and corrected during test
	Test Coverage	measures completeness of test progress
	Test Sufficiency	assesses sufficiency of integration and system test
	Documentation	subjective measure of documentation product adequacy

Table 2.3-a. Management and Quality Indicators.

3.0 The QUES Tool

The QUES tool consists of a database, predefined database components, and a set of interfaces to external data collection tools.

3.1 The QUES Database

A QUES database consists of the following components: personnel list, tools list, frameworks, projects, and General Information Objects (GIO's). Figure 3.1-1 illustrates the components of the QUES database. Reports are part of a framework or a project component and are not considered a separate database component. Frameworks, projects, and GIO's can be imported and exported separately, and so can be used as building blocks for a database.

QUES Framework Criteria

- ☐ Structure is hierarchical from highest level of abstraction to lowest level of abstraction
- ☐ Higher level items are computed from items at a lower level using simple mathematical operators (addition, subtraction, multiplication, division) or one of two aggregation functions (weighted average and sum)
- ☐ Lowest level of abstraction is a question that can be answered Yes or No or by a numerical value (how many, what size)

Thus QUES enforces a structured approach to applying metrics, which is necessary to enable automation. Metrics can be applied in a consistent and unambiguous manner.

QUES also enforces another philosophy: that a project is associated with one unchanging framework. In the QUES database, a project is associated with a particular framework at its inception. Essentially, QUES takes a "snapshot" of the framework and copies it into the project. After that point, the project's framework does not change, regardless of changes to frameworks in the QUES database. In order to view the results of a different framework on that same set of software components, it is necessary to create a new project in the QUES database. This limitation prevents confusion about how metrics were calculated at a certain point in the project life-cycle, and assures that comparisons are valid between scores calculated on different dates.

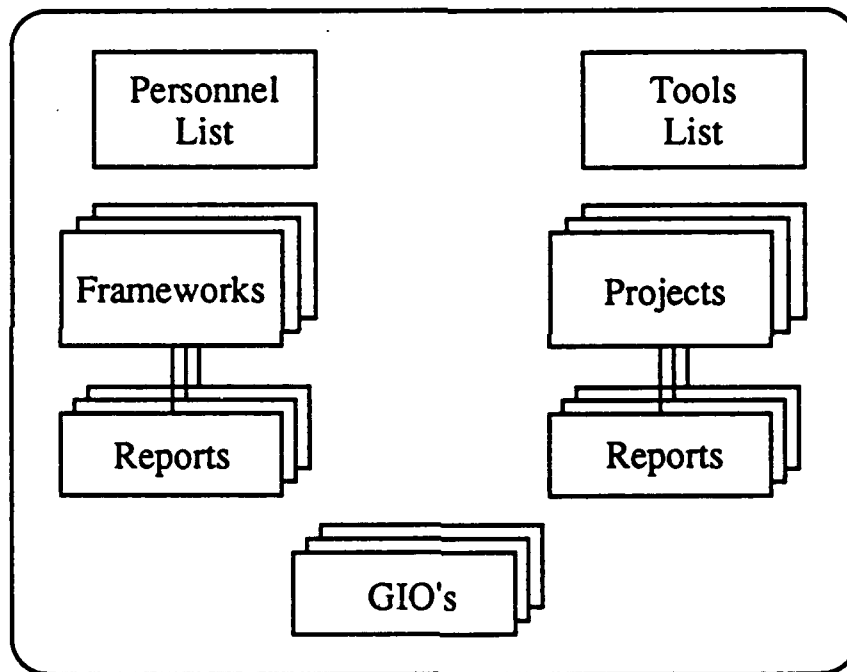


Figure 3.1-1. Components of the QUES Database.

The QUES hierarchical structure applies to both frameworks and projects. The project hierarchy defines the static structure of a software development project. The nature of the hierarchies are defined by *levels* and *partitions*. The levels of the framework and project hierarchies are named from highest level of abstraction down to the lowest level. Each level may then partition into one or more lower levels.

QUES is flexible in that it allows the level definitions to be user-defined. Figure 3.1-2 shows examples of possible framework level and project level definitions. QUES puts no limit on the number of levels. Since a level may partition into itself, it is theoretically possible to allow for an unlimited number of levels. For example, with the project level definition in Table 3.1-a, it is possible to have a project structure as shown in Figure 3.1-3, which could have an endless number of levels of CSC's.

Project level name	Partitions into:
CSCI	CSC CSU
CSC	CSC CSU
CSU	

Table 3.1.-a. Example Project Level Partitioning with Unlimited Number of Levels.

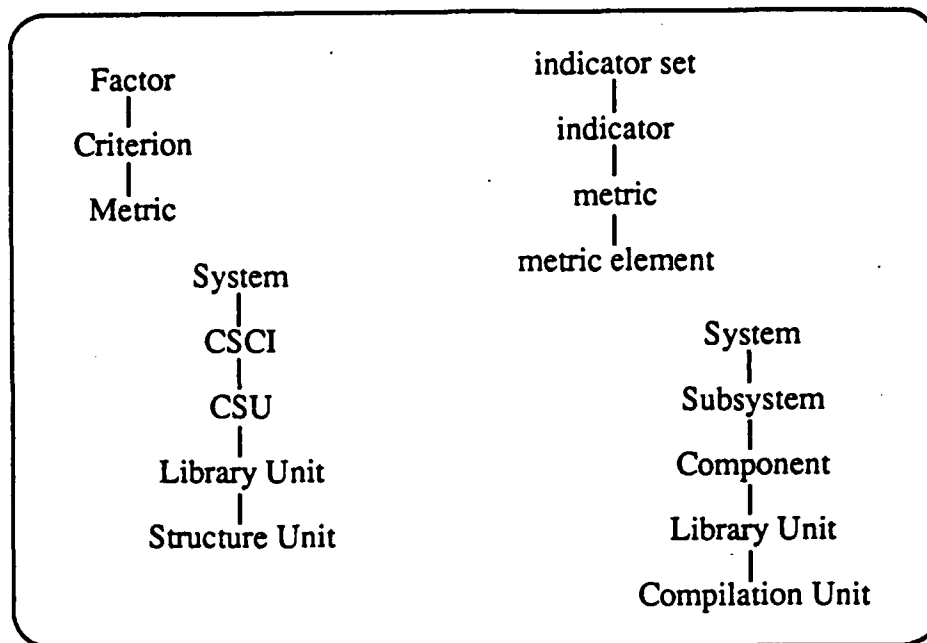


Figure 3.1-2. Possible Framework and Project Level Definitions.

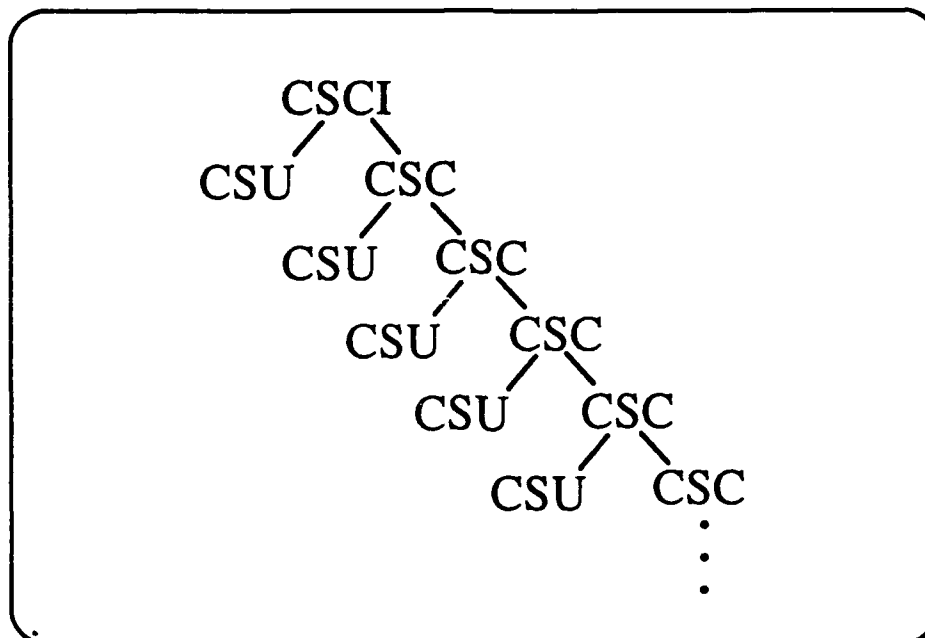


Figure 3.1-3. Example of an Unlimited Number of Project Levels.

The elements of the framework and project hierarchies are designated as either *abstract* or *specific*. The distinction is that abstract elements are subdivided into subordinate elements, whereas specific elements cannot be further subdivided. Framework elements are called *members* and project elements are called *components*.

The QUES abstract versus specific designation has two implications for framework members: the number of project levels associated with the member, and the contents of the member. A specific member is where the data is gathered; it consists of a question and the answers to that question. A specific member is associated with a single project level. An abstract member is computed from the results of subordinate members in the framework hierarchy; it consists of a set of scoring equations and the computed scores. An abstract member is associated with one or more project levels. Each abstract member scoring equation corresponds to one project level. The distinctions between abstract and specific framework members is summarized in Table 3.1-b.

For project components, the abstract versus specific designation mainly affects the static structure of the project. An abstract project component is subdivided into one or more subordinate components. Typically only the lowest level project components (CSU's, for example) are designated as specific components. Because QUES is used throughout the project life cycle, the project may change and grow as new components are defined. Project components should always be designated as abstract if they will later have subordinate components defined. Defining the project static structure allows QUES to aggregate scores automatically (as specified in the scoring equations) for subordinate components.

3.2 Predefined Database Components

QUES includes three predefined database components (shown in Table 3.2-a) which may be imported into any QUES database.

Two predefined framework components are provided with QUES: Rome Laboratory's Software Quality Framework (RSQF), and Air Force Systems Command's Software Management and Quality Indicators Framework (SMQI). Each predefined framework comes with a set of reports. These two predefined frameworks are used as a baseline from which to build tailored frameworks. Also, a predefined GIO is provided as a database component: the Problem Trouble Report (PTR).

Characteristic	Abstract	Specific
structure	can be subdivided	cannot be subdivided
contents	scoring equations scores	question answers
project levels	multiple	single

Table 3.1-b. Comparison of Framework Abstract and Specific Members.

Predefined Database Components: RSQF Framework SMQI Framework PTR Form
--

Table 3.2-a. Predefined QUES Database Components.

3.3 Tool Interfaces

Figure 3.3-1 illustrates the QUES external interfaces. QUES provides interfaces to the following external tools to enable data collection:

NASA Goddard's Fortran Static Analyzer Program (SAP)
Rome Laboratory's Software Life Cycle Support Environment (SLCSE)
Virginia Tech Ada Metric Analyzer.

The QUES SAP Export Tool is an interface between QUES and the SAP Automatic Data Collection Function which extracts metrics from Fortran source code. The QUES SLCSE Interface Tool extracts two types of data from a SLCSE database: source code metrics and project static structure. The QUES Adafilter Tool extracts Ada source code metrics from the Ada Metric Analyzer results file. All three interfaces are unidirectional; QUES imports information from but does not export information to the external tools. For each interface, QUES creates two types of requests for information: return a list of all available metrics, and return a list of metric data for a list of project components. Given a list of available metrics in the QUES database, a tool metric can be associated with a framework question. The SLCSE interface allows an additional request: return the project static structure.

3.4 QUES implementation of RSQF

This section describes how the RSQF is represented as a QUES framework. Implementation information is presented in three sub-sections: framework structure, project level modifications, and scoring. The framework structure sub-section explains how the factor/criteria/metric hierarchical quality model is implemented. The architecture level modifications sub-section describes changes that were made to the level of some metric elements to facilitate automated data collection. The last sub-section, scoring, explains how the two RSQF scoring methodologies are implemented in the QUES framework.

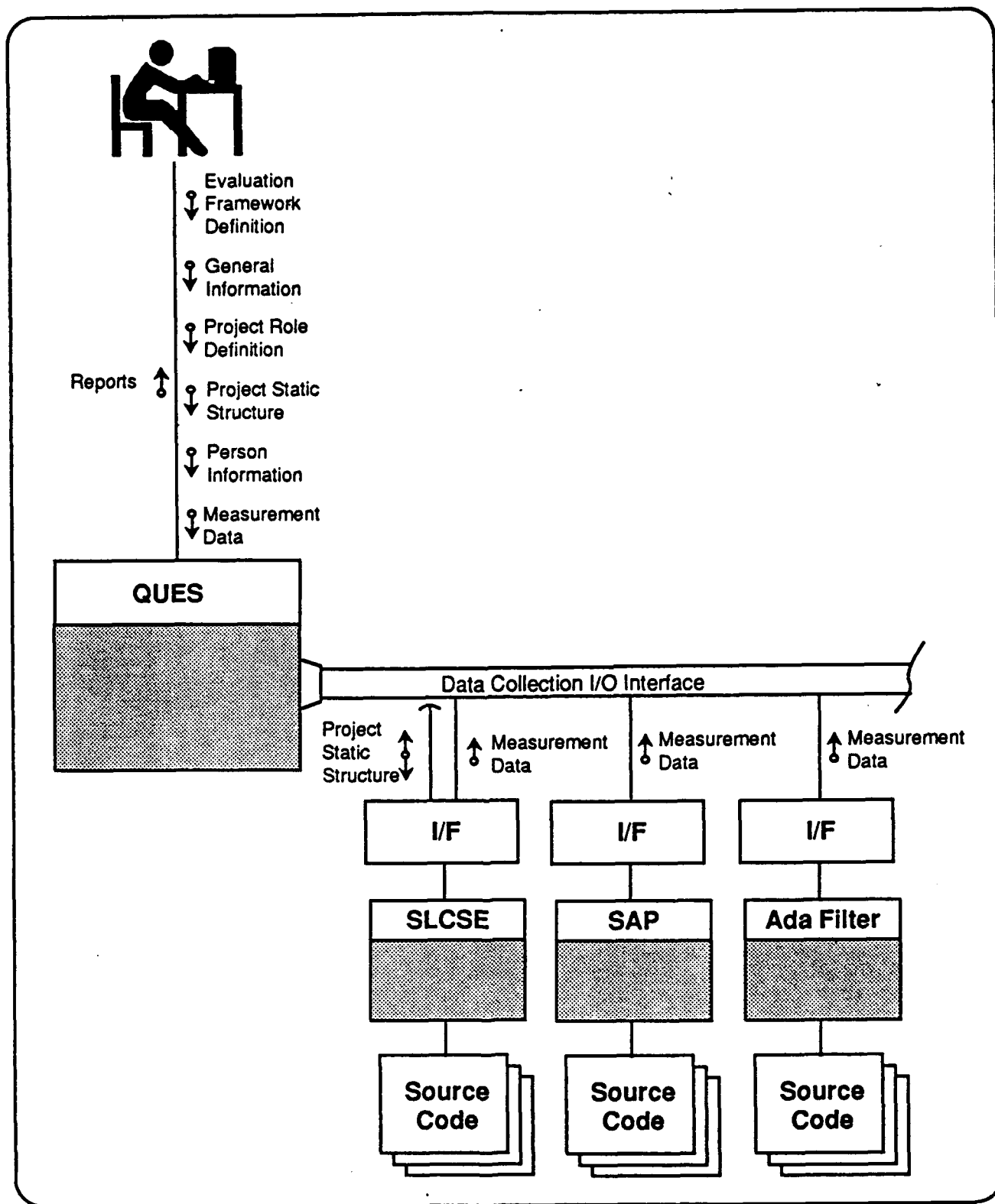


Figure 3.3-1. QUES External Interfaces.

3.4.1 Framework Structure

The RSQF quality model which expresses the factor/criterion/metric hierarchy is implemented as the framework level definition in QUES, shown in Table 3.4.1-a. A lowest level is added, called "question". This lowest level is designated "question" instead of "metric element" because in the QUES implementation, every framework member at this level is a specific member and has a question associated with it. A metric element, then, is equivalent to a question that can be answered Yes or No, or a ratio of the numerical answers to questions such that the ratio is between 0.0 and 1.0. In the QUES implementation, metric elements are expressed in the metric scoring equations.

The RSQF is comprised of a series of 9 DCF's, one for each phase of the software development life cycle. In QUES, each DCF is designated as a QUES *phase framework*, and is named as shown in Table 3.4.1-b. In QUES, each phase framework is independent and may stand alone. No data is shared between phase frameworks.

Each phase framework of the RSQF is structurally similar. All of the criteria are grouped under a factor called "Criteria list", which is not a true user-oriented measure of software quality, of course. Instead this "Criteria list" grouping is for convenience. The relationship of criteria to factors is expressed by the factor scoring equations, rather than in the physical structure of the framework. Note that the list of criteria differs by phase framework, depending on the project level of the questions under the criterion. For example, the criterion "Functional Overlap" does not appear in the phases c-g because the questions are asked at the system level.

The criteria are designated by a unique two-letter code such as "AM" for "Anomaly Management" (see Appendix A for a complete list) and are listed in alphabetical order. Abbreviated names are used for convenience, since the short names make scoring equations easier to read. Metrics and questions are grouped under the appropriate criterion. Metrics take the form "AM.1", "AM.2", and so on. Questions take the form "AM.1.1.a", which is the first question under metric AM.1 in phase a. Therefore each question in the RSQF has a unique name.

Framework Level	Partitions into:
factor	criterion
criterion	metric
metric	question
question	

Table 3.4.1-a. RSQF Framework Level Definition.

QUES Phase Framework	Life Cycle Phase
Phase.a	System Requirements Analysis/Design
Phase.b	Software Requirements Analysis
Phase.c	Preliminary Design
Phase.d	Detailed Design
Phase.e	Coding and CSU Testing
Phase.f	CSC Integration and Test
Phase.g	CSCI Testing
Phase.h	System Testing
Phase.i	Operational Test and Evaluation

Table 3.4.1-b. RSQF Phase Framework Names in QUES

3.4.2 Project Architecture Levels

The project level definition of the RSQF is based on the nature of the questions in the framework. Table 2.2.2-a (in section 2.2.2 about the RSQF framework) illustrates how the project level of the questions varies with the phase of the life cycle. Table 3.4.2-a shows the project level definition of the RSQF. Two additional architecture levels (beyond the DoD-STD-2167A definitions) were added below CSU to facilitate automated data collection.

The Ada Metric Analyzer collects metrics from Ada source at the procedure level; therefore RSQF questions which are automated are asked at the procedure level. In addition, any RSQF questions that appear in a metric element (ratio) with an automated question are also asked at the procedure level. The package level is added simply to aggregate procedures; no questions are asked at the package level.

3.4.3 Scoring

Two scoring methods are used to compute quality scores in the RSQF:

- adherence scoring
- aggregation scoring

Adherence scoring requires the use of a QUES report. Using the report formula function COUNT, the number of occurrences of a Yes answer to a set of questions is computed to define the number of compliances. The total number of questions asked becomes the number of opportunities. The compliances/opportunities ratio may be computed at the metric, criterion, or factor level.

Aggregation scoring is accomplished with the QUES scoring equations. Each abstract member of a framework has a scoring equation associated with each project level of the member. Factor, criterion, and metric scores are defined with scoring equations. The scoring equations for the RSQF framework are presented at the end of Appendix A. All elements of a scoring equation are equally weighted (weight = 1.0) in the predefined RSQF framework. QUES automatically normalizes the resulting score by dividing by the sum of the weights so that the answer is in the range of 0.0 to 1.0. Weights can be modified by editing the scoring equations, and QUES will continue to normalize the resulting scores.

Project Level	Partitions into:
System	CSCI
CSCI	CSC CSU
CSC	CSC CSU
CSU	Package Procedure
Package	Package Procedure
Procedure	

Table 3.4.2-a. RSQF Project Level Definition.

3.5 QUES Implementation of SQMI

3.5.1 Framework Structure

The SQMI framework is a single phase framework in QUES since the measures are used throughout the entire software development life cycle. The framework levels are defined as shown in Table 3.5.1-a. All data is collected at the CSCI level, so only one project level is defined. The indicators are grouped into two sets: management and quality. Each indicator consists of metrics which are calculated from data input. In some cases, the indicator is plotted; in other cases the metrics or data input is plotted. Metrics are frequently used for intermediate calculations in this QUES implementation.

Framework Level	Partitions into:
Indicator Set	Indicator
Indicator	Metric Data input
Metric	Data input
Data input	

Table 3.5.1-a. SMQI Framework Level Definition.

3.5.2 Scoring

The scoring method depends on the indicator. Refer to the AFSC pamphlets for more details about how the various indicators are computed. All calculations in QUES are accomplished with scoring equations. Results are plotted using the predefined QUES reports.

3.6 QUES Operational Scenario

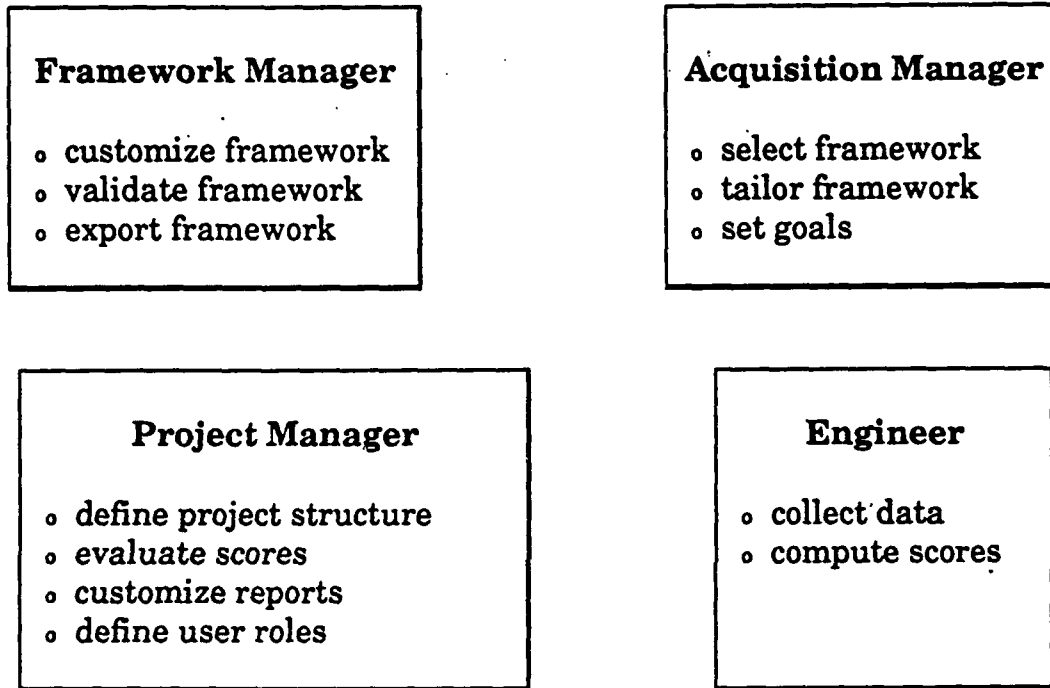
There are four general categories of QUES users: the Framework Manager, the Acquisition Manager, the Project Manager, and the Engineer. Categories are delineated by the tasks performed, which translate into QUES features. A Framework Manager is a framework expert, who is responsible for establishing and disseminating frameworks. An Acquisition Manager selects a framework for a particular product and determines the quality goals that product should meet. The Project Manager is given a framework and must apply it to his software development project and report on results. The Engineer records the detailed project quality information in the database. Figure 3.6-1 summarizes the types of tasks performed by these users.

3.6.1 Framework Manager

With QUES, a Framework Manager can build a complete framework from scratch. It is more likely, however, that he will adopt a predefined framework as a starting point and modify it to suit the quality concerns of a particular product area. QUES provides two predefined frameworks as database components which the Framework Manager can import to build his database. Removing sections of a predefined framework, or adding new framework components, creates a customized framework. The Framework Manager can then use QUES reporting and integrity checking features to validate the customized framework. When satisfied, he can export his framework as a database component to be used by other QUES users.

3.6.2 Acquisition Manager

The acquisition manager decides what framework is appropriate for a particular product based on his quality concerns. Indeed, he probably has asked a Framework Manager to create for him a customized framework. Frameworks are tailored to reduce to the bare minimum the amount of metric data to be collected, since data collection costs money. Even with an automated tool such as QUES to help with data collection, it is still important to remove irrelevant metrics from the framework in tailoring it to a product. Unwanted portions of the framework may be deleted or simply designated "Not Applicable" with the QUES tailoring feature. The acquisition manager then sets quality goals in the framework before giving it to the Project Manager.



QUES Users

Figure 3.6-1 Tasks Performed by QUES Users.

3.6.3 Project Manager

A Project Manager is responsible for the development of a product which meets the quality goals specified by the Acquisition Manager. He is given a tailored framework and then applies it to his software project. As the project grows, the manager adds project components to the static structure stored in the QUES database. Data is collected (by the Engineer) for the project components and quality scores can then be computed at every level of the project hierarchy. Using the QUES reporting feature, the Project Manager views the status of his project and evaluates how well the goals are being met. He can identify problem areas and see that they are addressed.

3.6.4 Engineer

An Engineer is responsible for developing the various parts of the software system. Engineers use QUES to record quality information about their particular parts of a project and to evaluate the quality of those parts. The Engineer enters data about his project components by answering questions in a Data Collection Form, which is a data-entry QUES report. Or the Engineer can enter data by running a data collection tool against his software and then importing the resulting metrics into QUES.

4.0 Framework Definition

The first step in using software metrics is to decide upon a framework. Choosing a framework is possibly the most difficult and certainly the most crucial step in the process. It is difficult because there are many types of frameworks and even different versions of frameworks within a type. As technology advances, existing frameworks are refined, new metrics are invented, and validation of quantitative approaches to quality measurement becomes available. That is why this first step is performed by the framework expert we call the Framework Manager. That is also why QUES allows user-defined frameworks instead of imposing one particular framework.

Framework selection is crucial because data collection is expensive and labor-intensive. The selected framework should minimize the need for manual data collection, and should not include irrelevant metrics. The goal is to automate all data collection with tools such as SAP, SLCSE, and the Ada Metric Analyzer. There are, however, many useful metrics that are not yet automated.

4.1 Establishing a Framework

The QUES user has several options (see Figure 4.1-1) for the establishment of a framework: build a framework from scratch, adopt a predefined framework, or modify a predefined framework. Building a new framework from scratch is a simple and straightforward process in QUES. *Designing* a new framework, however, is not as easy as it sounds. Deciding what questions to ask and determining how the answers are to be combined to formulate quality scores is a job for an expert. Most QUES users will instead use a predefined framework database component as the basis for their framework, and tailor that framework to suit their needs. Therefore we will only briefly discuss the QUES features related to creation of a new framework, and concentrate on the modification and tailoring aspects. The section on creation of a new framework is of interest to all QUES users, though, because it illustrates the elements of a framework and how they are implemented in QUES.

QUES Options for Establishing a Framework

- ☐ Build a framework from scratch
- ☐ Adopt a predefined framework
- ☐ Modify a predefined framework

Figure 4.1-1 Options for Establishing a Framework.

4.1.1 Creating a New Framework

For the purposes of illustrating the QUES framework creation features, we will create a small (admittedly trivial) framework from scratch. The contents of this example framework are not meant to be significant in terms of software quality metrics, but are chosen because they are easy to understand and explain.

The objective of the framework will be to track two types of code metrics: size in lines of code and interface variables. We would like to be able to sum up both the lines of code and the number of interface variables for a CSCI given the information for all of the subordinate CSU's. We would also like to be able to calculate the percentage of comments in the lines of code, and the percentage of parameters in the interface variables for each CSU. These metrics will give us a general idea of how descriptive the source code is and how much of the interface is defined as formal parameters (versus global variables or COMMON blocks). These are essentially software oriented metrics; we do not really have a user-oriented factor in mind.

These metric elements could be collected during the Detailed Design phase or the Coding and CSU Testing phase of software development. We will choose the Detailed Design phase, and name the framework "Code Size Metrics" or CSM. The requirements of the new CSM framework are summarized in Figure 4.1.1-1.

4.1.1.1 Defining the hierarchical structure

The first step in defining a new framework is to establish the nature of the hierarchical structure. There are actually two hierarchies to be defined: the framework levels and the project levels. Level definitions determine the number and name of the levels of the framework or project hierarchies and how each level partitions into the next lower level. For our CSM framework, we will use the framework and project level definitions shown in Table 4.1.1.1-a.

CSM Framework Requirements	
measure	lines of code comments number of parameters number of global variables
compute	percentage comments in total lines percentage parameters in total interface variables
aggregate	total lines of code number of interface variables

Figure 4.1.1-1. Requirements of the Code Size Metrics Framework.

Framework level name	Partitions into:
factor	criterion
criterion	metric
metric	question
question	

Project level name	Partitions into:
CSCI	CSC CSU
CSC	CSC CSU
CSU	

Table 4.1.1.1-a. Level Definitions for CSM Framework.

4.1.1.2 Adding framework elements

Elements of a QUES framework, called framework members, must be associated with a framework level and a project level. That is why defining the levels is the first step. A framework is built by adding one member at a time to the framework structure, designating it as either an *abstract* or a *specific* member. Abstract members can be subdivided into subordinate members, and specific members cannot be further subdivided. So the first framework element is always an abstract member. For our framework, it is designated as a factor.

First, we create a phase called "Detailed Design", and then open that phase to begin adding framework members.

The CSM framework will be structured as follows:

We will group all of the questions under a single factor called "Size". Under that factor will be two criteria, "% commented" and "% parameters" which are the two ratios that we want to calculate. Under "% commented" will be a metric "Size in LOC" to aggregate the total number of lines for each unit, which will become the denominator of the ratio. Under "Size in LOC" will be the SLOC question. Another metric "Total comments" will go under "% commented" and this metric will sum the number of lines with in-line comments to the number of lines with just comments to get the numerator of the ratio. Under the "Total comments" metric will go the two comment questions, "In-line comments" and "Comment lines".

The parameter-related items will go under the "% parameters" criterion. One metric "I/F variables" is defined which will sum the declared parameters to the global variables to get the total number of variables in the interface (the denominator of the ratio). Under this metric are the questions "Num parameters" and "Num globals". Questions are asked at the CSU level, the lowest level of the CSM project hierarchy. It is assumed that no code occurs at the CSC level.

Figure 4.1.1.2-1 is a depiction of the CSM framework structure in tabular and diagrammatic form.

Factors	Criteria	Metrics	Questions
Size	% commented	Size in LOC	SLOC
		Total comments	In-line comments Comment lines
	% parameters	I/F variables	Num parameters Num globals

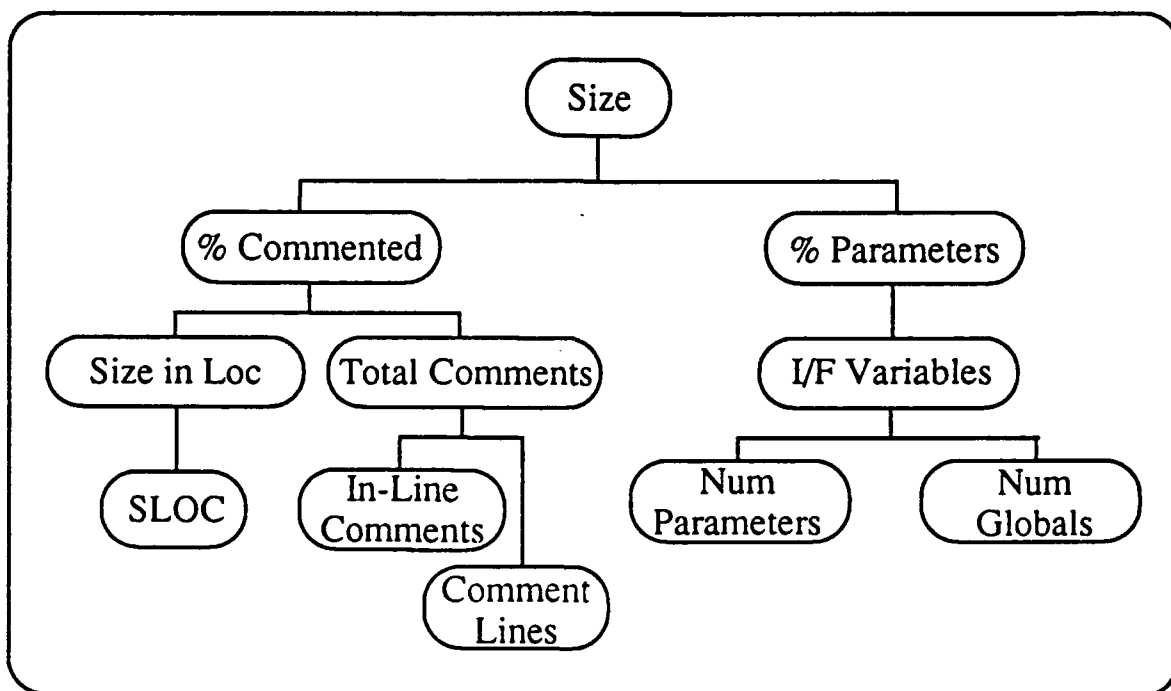


Figure 4.1.1.2-1. CSM Framework Structure Table and Diagram.

QUES displays the framework hierarchy as a series of lists. The first list (from left to right) is the highest level, and successive levels are displayed next to each other. The top item of a list becomes the default contents of the next list displayed. Figure 4.1.1.2-2 illustrates the CSM framework in QUES format. Since specific framework members have no subordinate members, there are no lists to the right of a specific member. To view the partitioning of any item in a list, just select the item with the mouse and the lists are displayed.

The CSM framework has 5 specific members as shown in Table 4.1.1.2-a.

Why do we bother to make the distinction between source lines and comment lines in the lines of code counts? Because that distinction is made in most of the automated data collection tools, and we plan to make use of automated data collection whenever possible. Generally these counts are given separately and must be added to get the total number of lines in a unit.

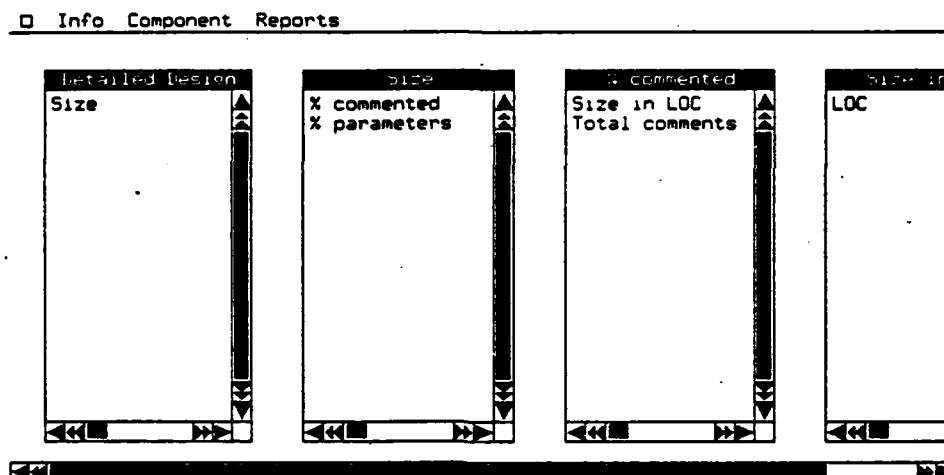


Figure 4.1.1.2-2. CSM Framework as Displayed in QUES.

Specific member name	Description
SLOC	Number of source lines of code (excluding blank lines and comments)
In-line comments	Number of source lines which contain embedded comments
Comment lines	Number of lines which are exclusively comments
Num parameters	Number of declared parameters in the unit's calling sequence
Num globals	Number of external variables referenced by the unit

Table 4.1.1.2-a. Specific Members of CSM Framework.

We could define the criterion scoring equations directly from the questions instead of using the metric scoring equations as intermediate calculations. Why then do we have the metric level at all? In this framework the metric level has two functions: to group together similar questions, and to provide a place-holder for an equation which aggregates the totals (Size in LOC and Total comments) across project levels. The equations in Table 4.1.1.2-b are CSU level equations. Scores calculated at the higher project levels are derived from the CSU level information using special QUES aggregation functions. This will be discussed in more detail in section 4.1.1.3.

Adding each framework member to the structure defines its place in the hierarchy. The next step is to *Open* each member to define its framework level, project level, and contents. The QUES abstract and specific definition windows in Figure 4.1.1.2-3 show the information associated with those member types. Both abstract and specific members have acronyms which are abbreviated names. By default, each abstract member is given the highest framework level, or "factor". Modifying the level is very easy; just select it and hit the spacebar to cycle among the possible framework levels, and stop when you see the one you want.

By default the abstract member has no project level associated with it; the levels must be selected explicitly. In general, each abstract should have all project levels from the highest down to the lowest level of any of its subordinate members. In the CSM framework, the lowest level is CSU, so all abstracts will need the levels CSCI, CSC, and CSU. Specific members have a single project level which is modified in the same way that abstract member framework levels are modified. Specific members also have an area for typing in the question text.

Framework Member	Framework Level	Equation
% commented	criterion	Total comments/Size in LOC
% parameters	criterion	Num parameters/I/F variables
Size in LOC	metric	SLOC + Comment lines
Total comments	metric	Comment lines + In-line comments
I/F variables	metric	Num parameters + Num globals

Table 4.1.1.2-b. Summary of CSM Framework Equations.

Abstract Window

☐ Info Reports

Acronym:

Framework Level:

Selected Levels

csc1
csc
csu

Project Levels

csc1
csc
csu

Specific Window

☐ Info Component Reports

Acronym:

Question:

Answer is type Project level is

Figure 4.1.1.2-3. Abstract and Specific Definition Windows.

4.1.1.3 Defining the scoring equations

A scoring equation is defined for every project level of an abstract member. Equations are built by selecting operands and operators on the equation definition window (shown in Figure 4.1.1.3-1.). This example of an equation definition window shows the CSU-level scoring equation for the Total comments metric (refer to the window title). This is simply the sum of the number of comment lines plus the number of lines with in-line comments. In this equation definition window, both operands appear in the Operands list for this metric.

A scoring equation may contain any operand in the framework at a lower framework level than the member for which the equation is defined. A metric scoring equation may contain questions for operands. A criterion scoring equation may contain metrics or questions. The initial operand list is the list of subordinate members; the Up and Down functions can be used to traverse the framework in order to access new operands for the building of the equation. For example, the Size in LOC metric requires the Comment lines question as an operand, even though this question is part of another metric (Total comments metric).

The left side of the window is a palette of operators which are used to build and equation. The W.AVG and SUM functions compute the weighted average and sum, respectively, of the values of the operand for the subordinate project components. The Weight function is used to apply relative weights to operands of an equation. The weights are automatically normalized to sum to 1.0 during calculation of the scores.

On the right side of the window are the upper and lower goals. By default, the upper goal is 1.0 and the lower goal is 0.0 because equations are usually designed to compute scores in that range. Section 5.2 of this report discusses why a user might modify the default goals.

Ques Info Configuration

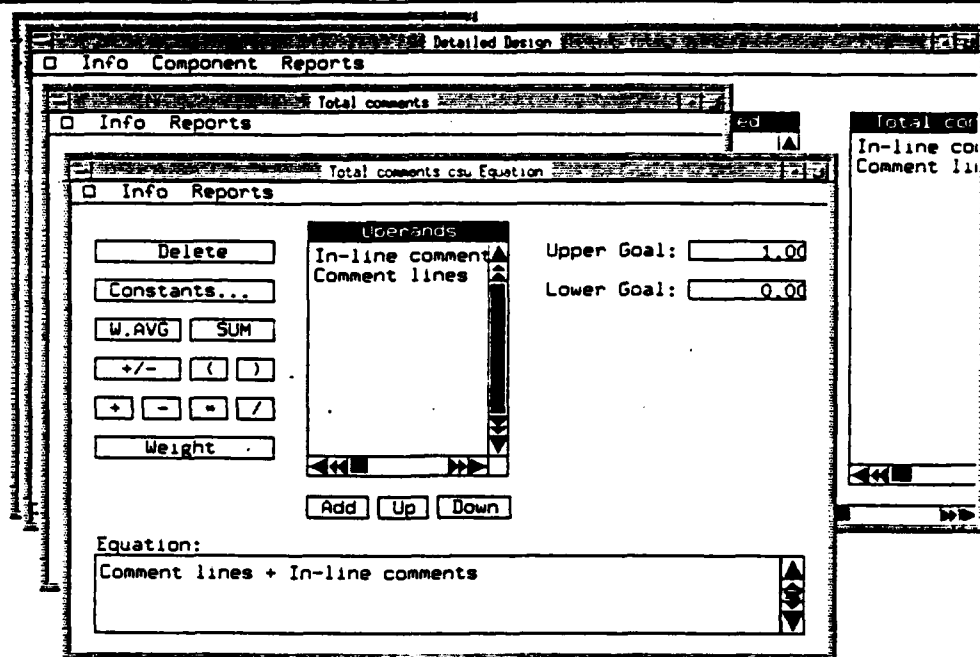


Figure 4.1.1.3-1. Equation Definition Window.

The CSC-level and CSCI-level scoring equations for the metrics are where aggregation across project levels takes place. At the CSU-level, the Total comments metric expresses the total number of lines containing comments for a single CSU. At the CSC-level, the metric will express the total number of lines containing comments summed for all CSU's in the CSC. The CSCI-level metric will express a similar sum of all of the CSC's in the CSCI. The aggregation is accomplished in QUES with the SUM function. This function takes an operand of the same framework level as the equation which contains it. For the Total comments metric, the CSC-level equation is the sum of the CSU-level Total comments metric for all of the CSU's in the CSC. Given the project static structure definition, QUES is able to automatically keep track of which CSU's are subordinate to the CSC for which a score is calculated.

All project level scoring equations for the CSM framework are summarized in Table 4.1.1.3-a.

In the criterion scoring equations at the CSC and CSCI level, a weighted average aggregation across project levels appears instead of a summation. This gives an indicator of the average criterion score over all of the CSU's in a CSC, and all of the CSC's in a CSCI. By default, all project components are equally weighted. Relative weights can be assigned so that some components have more weight in a weighted average score calculation. For example, if Component 1 has twice the lines of code of Component 2, you might want to give Component 1 a relative weight of two times the weight of Component 2. QUES automatically normalizes the resulting score by dividing by the sum of the component weights. Project component weights are assigned independently of the scoring equations (assigning component weights will be discussed in the Project Management section of this report).

Framework Member	Project Level	Scoring Equation
% commented	CSCI CSC CSU	W.AVG(% commented) W.AVG(% commented) (Total comments * 100) / (Size in LOC)
Size in LOC	CSCI CSC CSU	SUM(Size in LOC) SUM(Size in LOC) SLOC + Comment lines
Total Comments	CSCI CSC CSU	SUM(Total comments) SUM(Total comments) Comment lines + In-line comments
% parameters	CSCI CSC CSU	W.AVG(% parameters) W.AVG(% parameters) (Num parameters * 100) / (I/F variables)
I/F variables	CSCI CSC CSU	SUM(I/F variables) SUM(I/F variables) Num parameters + Num globals

Table 4.1.1.3-a. Summary of Scoring Equations for CSM Framework.

4.1.1.4 Setting up automated data collection

To take advantage of automated data collection, a question in a QUES framework must be associated with a metric collected by one of the three external tools (SAP, SLCSE, and the Ada Metric Analyzer). The predefined QUES frameworks come with this association already established, but for a newly created framework, the user must specify the association.

Configuring a tool to a QUES database loads a list of available metrics into the QUES database. A specific metric is selected from the list for each automatable framework question. A framework question may be associated with more than one data collection metric; this allows the framework to be generic enough to handle more than one higher level language. It so happens that all questions in our CSM framework are automatable; in other words, the required data is extracted from the source code by the external tools. In fact, all questions except the Num globals question are collected by all three external tools. Num globals is not available from the SLCSE tool.

In the next section on creating reports, a summary report of the automated questions and their association with automated data collection metrics is presented.

4.1.1.5 Creating reports

QUES reports are a view into the QUES database. During the creation of a new framework, reports are used to check the framework structure and contents. The structure of a QUES report is dependent on the structure of the QUES data, so the creation of a new framework requires defining a set of new reports for that framework. Framework reports are typically presented in a tabular format.

QUES reports are defined on a spreadsheet template. A cell of the spreadsheet may be a text *label* or may be a *field* which extracts information from the QUES database. Because the QUES data structure is hierarchical, it is necessary to traverse the structure from the highest level down to the level of the information to report. Traversal is accomplished by designating a cell as a *macro*, which opens up into another spreadsheet. Therefore a typical report is a series of spreadsheets which descend the QUES data structure, displaying the requested information as requested at each level.

Quality Evaluation System

In addition to fields and macros, spreadsheet cells may also contain:

- *formula* which operates on data from other cells on the spreadsheet
- a horizontal *line*
- a *graph*.

Any cell contents may be resized to extend over multiple cells, or dragged to a new location. Thus arranging the contents of a report is quick and easy, and the results immediately viewed.

In addition, the constraint feature of QUES reports acts as a data filter. Constraints can be used to restrict the scope of information presented, for example by constraining a metric name equal to "Size in LOC". This would prevent the metrics "Total comments" and "I/F variables", and any data pertaining to these metrics, from being displayed. Also, constraints can be used to show a range of data, for example, to list all CSU's whose scores are below a threshold value.

In larger frameworks, the QUES attribute feature may be used to separate types of questions. For example, questions can be designated with the attribute "automated". Then a report, constrained on "automated" questions, could list all questions with that attribute and their answers. This would be useful to scan the automated data collected by an external tool after it has been loaded into the QUES database.

Report fields are, by default, protected from data entry. By setting a field to unprotected status, data can be entered from the view of a report directly into the QUES database. The main use of this feature is to create a Data Collection Form (DCF) report for manual entry of answers to questions (questions that are not automatable).

For our CSM framework, we create a series of framework reports:

- Framework Members report -- lists the framework members in tabular format
- Framework Equations report -- shows all scoring equations in the framework
- Framework Questions report -- lists the text of all questions, the question project level and answer type
- Framework Automated Questions report -- shows the association to automated data collection metrics
- Data Collection Form -- lists the questions and answers for manual data input

All of the framework reports, with the exception of the Data Collection Form, are created by traversing the framework branch of the QUES database structure. The traversal begins at the top with the name of the framework, and then the phase frameworks. For our CSM framework, there is only one phase. Then the phase framework structure is traversed from the factor level down to the question level. In the case of the Framework Members report, the only information displayed about a framework member is its name. For the Framework Equations report, the project levels and scoring equations are also displayed for the abstract members. In the Framework Questions report, the project level, answer type (boolean, integer or float), and the question text are displayed.

The Framework Automated Questions report makes use of the attribute constraint to filter the questions to display only those questions which are "automated". For these questions, the new information displayed is the data collected from the automatic data collection tools. In the CSM framework, all questions happen to be automated.

The Data Collection Form displays information from the project branch of the data structure, because it is listing the answers to questions. Answers are associated with particular project components, and project components fall under the project data structure. This report is created within the CSM framework so that it will go along with that framework when the framework is associated with a project. However, the contents of the Data Collection Form report cannot be displayed until a test project is created in the QUES database. In this case, the CSM framework requires at least one CSU-level project component to exist in order to display the DCF report, since the questions are asked at the CSU level.

These reports are shown on the following pages.

Quality Evaluation System

Framework Members Report

Framework Name: Code Size Metrics

Phase Name: Detailed Design

Factors	Criteria	Metrics	Questions
Size	% commented	Size in LOC	SLOC
		Total comments	In-line comments Comment lines
	% parameters	I/F variables	Num parameters Num globals

Framework Equations Report

Framework Name: Code Size Metrics

Phase Name: Detailed Design

Framework Member	Project Level	Equation
Size	csci csc csu	
% commented	csci csc csu	W.AVG (% commented) W.AVG (% commented) (Total comments * 100.0) / Size in LOC
Size in LOC	csci csc csu	SUM (Size in SLOC) SUM (Size in SLOC) SLOC + Comment lines
Total comments	csci csc csu	SUM (Total comments) SUM (Total comments) Comment lines + In-line comments
% parameters	csci csc csu	W.AVG (% parameters) W.AVG (% parameters) (Num parameters * 100.0) / I/F variables
I/F variables	csci csc csu	SUM (I/F variables) SUM (I/F variables) Num parameters + Num globals

Figure 4.1.1.5-1. CSM Reports: Framework Members and Equations.

Framework Questions Report	
Framework Name:	Code Size Metrics
Phase name:	Detailed Design
Question:	<u>SLOC</u>
Project level:	csu
Answer type:	2
Question text:	
Lines of code, excluding comments and blank lines, in this unit.	
Question:	<u>In-line comments</u>
Project level:	csu
Answer type:	2
Question text:	
Number of lines of code with in-line comments, excluding lines which are exclusively comments, in this unit.	
Question:	<u>Comment lines</u>
Project level:	csu
Answer type:	2
Question text:	
Number of comment lines, excluding in-line comments, in this unit.	
Question:	<u>Num parameters</u>
Project level:	csu
Answer type:	2
Question text:	
Number of declared parameters in this unit's calling sequence.	

Figure 4.1.1.5-2. CSM Report: Framework Question Summary.

Framework Automated Questions		
Framework Name: Code Size Metrics		
Phase name: Detailed Design		
Question	Project Level	
	Type	Tool data
SLOC	csu	
	2	lines_of_code (adafilter) ICTSCD (sap) NUM_ADA_LINES (slcse)
In-line comments	csu	
	2	number_of_embedded_comment_lines (adafilter) ICTSXP (sap) NUM_CODE_COM_LINES (slcse)
Comment lines	csu	
	2	number_of_comment_lines (adafilter) ICTSCM (sap) NUM_COMMENT_LINES (slcse)
Num parameters	csu	
	2	number_of_parameters (adafilter) ICTARG (sap) NUM_DECL_PARM (slcse)
Num globals	csu	
	2	number_of_global_referenced (adafilter) ICTEXT (sap)

Figure 4.1.1.5-3. CSM Report: Framework Automated Questions.

QUality Evaluation System

Data Collection Form		
Line Count Questions		
Project Name: Test_framework		
csu name: csu_la		
Question: LOC		
Project level: csu		
Answer type: 2		
Lines of code, excluding comments and blank lines, in this unit.		
Answers:	Value	Date
	0.00	19910715
Question: In-line comments		
Project level: csu		
Answer type: 2		
Number of lines of code with in-line comments, excluding lines which are exclusively comments, in this unit.		
Answers:	Value	Date
	0.00	19910715
Question: Comment lines		
Project level: csu		
Answer type: 2		
Number of comment lines, excluding in-line comments, in this unit.		
Answers:	Value	Date
	0.00	19910715

Figure 4.1.1.5-4. CSM Report: Data Collection Form.

4.1.2 Using a Predefined Framework Component

Instead of building a framework from scratch, a predefined framework can be imported into the QUES database. Two predefined framework database components are delivered with QUES: the RSQF and the SMQI frameworks. Any QUES framework can be exported and then imported by another QUES user. A customized framework can be reused on new projects. Once a framework is imported into a QUES database it can be modified as illustrated in the following section on tailoring a predefined framework.

4.2 Tailoring a Predefined Framework

A QUES framework can be tailored in several ways. New framework members can be added, existing members can be cut and pasted in new locations, members can have their properties modified, and portions of the framework can be deleted.

This section will illustrate a typical tailoring done to a predefined framework. The predefined RSQF framework will be imported and then modified to reduce the number of factors for which data is gathered. It is easier to start with the complete RSQF framework and delete unused portions than to rebuild a partial RSQF framework from scratch, even if you only need a single factor.

4.2.1 Importing the Predefined RSQF

Importing a framework is simply a matter of selecting the Import option from the framework window pane. Opening a phase framework of the RSQF displays the structure of the framework as shown in Figure 4.2.1-1. Notice that the first factor listed is the "Criteria list" factor which serves as a grouping of criteria. The remaining members in the factor list are the 13 user-oriented quality characteristics. Each factor contains a set of equations which describes the relationship between the factor and its corresponding software-oriented criteria.

The criteria are named with a two-letter abbreviation and are listed in alphabetical order for convenience. The metric and question names follow the same two-letter convention.

Ques Info Configuration

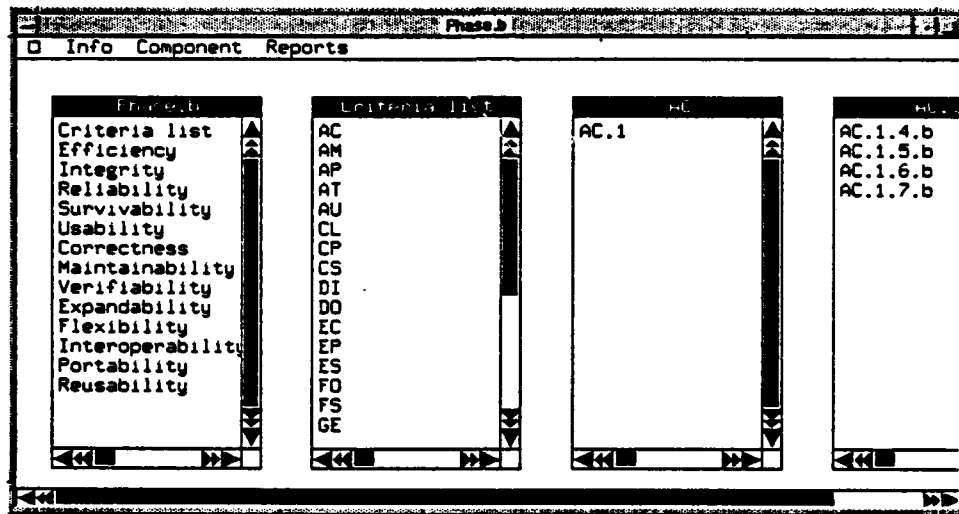


Figure 4.2.1-1. The Structure of Phase B of the RSQF Framework.

4.2.2 Reducing the Number of Factors

We decide to evaluate the factors Reliability and Maintainability for our hypothetical project. In that case, the other 11 factors can be eliminated from the framework. Because the relationship between factors and criteria exists only in the factor scoring equations, the unwanted 11 factors can simply be deleted.

Any criteria that are not used by the Reliability and Maintainability factors can also be deleted, but keep in mind that metric scoring equations sometimes include metric elements from other metrics. When a criterion is deleted from the framework, all subordinate framework members (metrics and questions) are deleted also. Therefore we must determine which, if any, metric elements are required from criteria that are unrelated to Reliability and Maintainability by checking the metric scoring equations in Appendix A of Volume II.

Table 4.2.2-a shows which criteria are related to Reliability and Maintainability and which "unrelated" questions are used by metric scoring equations in the Reliability and Maintainability criteria. For example, question AU.1.2.d is used in the denominator of a metric element of the SI.4 scoring equation for Phase D. All criteria except the related criteria and AP, AU, and CP can be deleted. Since only a few "unrelated" questions must be retained, they can be cut from their current location and pasted into a new criteria called "miscellaneous". Then the remainder of the unrelated criteria (AP, AU and CP) can be deleted. Using cut and paste retains the links to the questions in the metric scoring equations, so it is not necessary to modify any of the equations after the paste operation.

Figure 4.2.2-1 shows the framework after the deletion of the unused factors and unrelated criteria. In Phase B, there is only one question in the "miscellaneous" category: CP.1.3.b.

Factor	Related Criteria	Unrelated Questions Used in Equations
Reliability	AC	none
	AM	none
	SI	CP.1.2.c, AU.1.2.d, CP.1.2.d, AP.3.3.e
Maintainability	CS	AP.5.2.b, CP.1.3.b, AP.5.6.c, AP.5.7.d, AP.5.7.e
	MO	none
	SD	AP.3.3.e
	SI	same as SI above
	VS	none

Table 4.2.2-a. Criteria and Metric Elements Related to Factors Reliability and Maintainability.

Ques Info Configuration

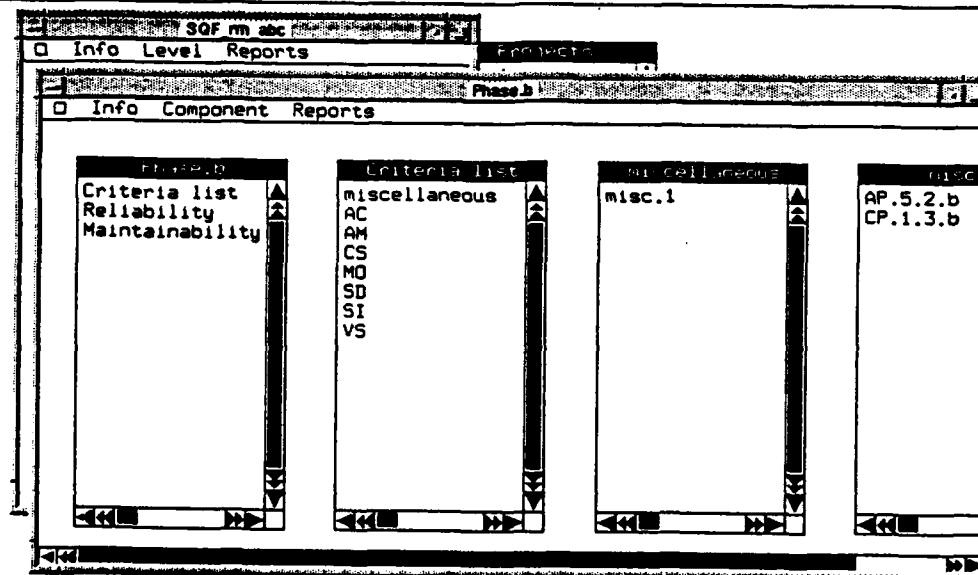


Figure 4.2.2-1. RSQF Phase B after Deletion of Unrelated Factors and Criteria.

4.2.3 Other Framework Modifications

Other options for customizing a framework include modifying the text of questions to clarify them for a particular application or language, or to specify answers that are always the same. If the project will be coded in Ada, for example, certain questions will always be answered Yes, such as EP.2.9.b:

Does the source code language enable variable initialization (at compile time) when the variable is declared?

The QUES feature called *attribute* can be used to sort questions into categories for reporting. For example, the questions can be assigned an attribute from the list:

constant
automated
manual

to indicate how the question is to be answered. "Constant" questions will always have the same answer for a particular project. "Automated" questions will be answered by automated data collection. And "manual" questions will require some human analysis before answering. EP.2.9.b would be given the attribute "constant" for an Ada project. The text of the question could be modified to say:

Does the source code language enable variable initialization (at compile time) when the variable is declared?

(ANSWER = YES for an Ada CSCI)

A data collection form could be created for "constant" questions by constraining the report to list only the questions with the attribute "constant". Another useful report would summarize the answers to "automated" questions after data has been loaded from an automated data collection tool.

4.3 Validating the Framework

When a new framework is built or customization of a predefined framework has been completed, it is necessary to check the validity of the framework structure and the scoring equations. QUES reports are used during framework creation to display the structure and the equations. Also during framework creation, when an equation definition window is closed, the syntax of the equation and the project levels of the operands are checked. When tailoring a predefined framework, however, it is not necessary to open every equation to trigger this check because the equations will be checked at the time of project creation.

The best and final check of framework validity occurs when the framework is used on a sample, or test, project. The test project is created for the sole purpose of checking the framework. When a framework is associated with a project, QUES automatically checks the syntax and project levels of equations. QUES will display a warning message for every equation that is in error. If, for example, you have inadvertently deleted a question that is used in a metric scoring equation, this will trigger a syntax error for every equation that included the deleted question. To check the structure of the framework, it is necessary to create one project component at each project level defined for the framework. This insures that the hierarchical structure is intact for every branch of the tree, for every project level.

If you had created a new framework from scratch, you would want to extend validation to include checking the scoring equations by adding some sample data to the test project. It is possible to omit a scoring equation; QUES does not consider the lack of an equation to be an error. And of course it is possible to define an equation that is syntactically correct but is not what you meant to define. Viewing reports on the framework equations and computing scores on the test project check the correctness of the scoring equations.

If any framework errors are found, they must be repaired and the test project validation repeated. When the test project is successful, the framework is ready to be used on a real project.

5.0 Tailoring and Goal Setting

A framework can be tailored before it is associated with a project, as described in section 4.2. At that point, it is possible to modify any aspect of the framework. Framework members can be deleted or cut and pasted to a new location. Framework equations can be modified, project levels of framework members can be changed, and new framework members can be added.

After QUES associates a framework with a project, it makes a copy of the framework which becomes that project's *framework template*. The framework template then has no link back to the original framework in the QUES database. After project association, any modifications made to the original framework have no effect on the framework template. In fact, the original framework could be deleted from the QUES database.

Tailoring of a framework template is limited in scope and affects only that project. Any project components created after tailoring of a framework template are subject to the tailoring modifications. When a new project component is created, a copy of the framework template containing only those items which have the same project level as the newly created component. This component-specific framework is called a *framework instance*. Tailoring of a framework instance affects only that component.

Even though the tailoring of framework templates and framework instances occurs after the creation of a project, it is relevant to Framework Managers and Acquisition Managers as well as Project Managers. There are only three possible modifications that can be made to tailor a framework template or framework instance:

1. Change a member's applicability.
2. Change goals.
3. Associate a question to a automated data collection metric.

The first two types of modifications are discussed in the following sections. The data collection modification will be discussed in section 7.0.

5.1 Tailoring the Applicability of a Framework Member

By default, all members of a framework are considered to be *Applicable*, that is, their answers or scores are counted when computing scores for other framework members. When a framework member is set to *Not Applicable*, then it has no contribution to any calculated score. If a member is Not Applicable, it has an answer or a score equal to zero, and if weighted, its weight is set to zero also. This means that designating a member as Not Applicable does not have a negative affect on scores which are computed from the member.

Consider the following example. The system-level scoring equation for the metric AT.4 in Phase A is:

$$1.0 (AT.4.1.a) + 1.0 (AT.4.2.a) + 1.0 (AT.4.3.a)$$

If the answers were as follows: AT.4.1.a = True
AT.4.2.a = True
AT.4.3.a = True

the score for AT.4 would be computed as:

$$(1.0(1) + 1.0(1) + 1.0(1)) / (1.0 + 1.0 + 1.0) = 1.0 \text{ [a perfect score]}$$

If the answer to AT.4.1.a were False, the score would be

$$(1.0(0) + 1.0(1) + 1.0(1)) / (1.0 + 1.0 + 1.0) = 0.66$$

And if AT.4.1.a were Not Applicable, the AT.4 score would be still be a perfect score because AT.4.1.a has zero weight and thus no contribution to the denominator of the equation.

$$(0.0(0) + 1.0(1) + 1.0(1)) / (0.0 + 1.0 + 1.0) = 1.0$$

When a question is designated Not Applicable, a negative or non-compliant answer to that question has no negative impact on the quality score. Therefore an Acquisition Manager may want to control which framework members are permitted to be designated as Not Applicable.

5.2 Setting Goals

Each abstract framework member has a set of high and low goals for each scoring equation. Goals can be used to represent several different things:

1. Upper and lower bounds of scores to be reported
2. Specification of minimum required scores and desired scores
3. Range of possible values for a score

A Project Manager might use the first approach to restrict reporting such that only the project components whose score is below the lower goal, or above the upper goal appear in a report. This is a useful way to identify quality problem areas. The second approach, using goals as specifications, would allow an Acquisition to set more stringent requirements for some project components and relax the requirements for others. Or the Acquisition Manager could prioritize quality factors by specifying more stringent requirements for some factors. A Framework Manager would use the third approach is applicable on a custom-designed framework when the scores do not always fall in the range of 0.0 to 1.0. For example, if the maximum achievable score for one framework member is 100.0, then the upper goal for that member could be set to 100.0 to indicate the value of a perfect score.

Tailoring of a framework template by specifying goals allows goals to be customized for a particular project. Tailoring a goal for a framework instance sets a specific goal for an individual project component, and this enables different goals to be set for different components of a project.

6.0 Project Creation

Once a framework has been selected, customized, and validated by the Framework Manager and Acquisition Manager, a QUES project is created. The project contains information about the static structure of a software project, the QUES users which have access to that project, the answers to questions for each project component, and the computed scores for the project. In general, this information will be set up by the Project Manager.

6.1 Associating a Framework with the Project

The first step in creating a new project is to select the framework to associate with the project. A project may have one and only one framework, and this framework does not change throughout the life of the project. The Project Manager may begin by importing a framework if he does not have the framework in his QUES database. After association with a project, the framework component of the QUES database is no longer linked to the project. A copy of that framework is created and stored with the project; it becomes the project's *framework template*.

Now the Project Manager may tailor the framework template as described in section 5.0. He may define detailed goals for the framework criteria which express his idea of how to meet the overall system-level quality goals. He may also define customized reports to help identify how well his goals are being met. It is not necessary to define all project reports before the project components are added because QUES reports can be defined and viewed at any time.

6.2 Defining Project Users and Roles

Many users can have their own passworded login to a QUES database, but the database can only be accessed by one user at a time. The Project Manager will be "superuser" which is the default QUES user who has all privileges. The "superuser" can define additional users and control their access to the QUES database. He can control which frameworks and projects in the database another user may open. The "superuser" can also restrict access within a particular project by defining *user roles*.

User roles affect only project access and modification privileges. If a user is given access to a framework in the database, he has the ability to modify that framework. The Project Manager can define a series of roles which limit the ability of a user to make changes to a project. For example, he could define a role called "reports only" which restricts the user to merely viewing reports

about the project. He could define a role called "data input" which gives the user the ability to answer questions and obtain data from a data collection tool, but not to compute scores or delete previous answers.

Roles are defined by selecting a list of capabilities from a list of all possible capabilities. Each capability corresponds to a item in a menu or the selection of a button on a window. To give the user the ability to view reports, for example, it is necessary to give him the capabilities corresponding to all menu and option selections up to and including report viewing. Figure 6.2-1 shows an example of the project role definition window with the role "reports only" defined.

After establishing the user roles as "superuser", the Project Manager can change his role to each of the new roles and make sure that they are correctly defined. To each user that has access to the project, the Project Manager assigns one or more roles. When that user opens the project for the first time, he is automatically assigned the role that is at the top of his role list.

6.3 Establishing the Project Static Structure

The structure of a QUES project is hierarchical, like the structure of a framework. The hierarchy is defined by the project level definition in the framework associated with the project. Project components, like framework members, are either *abstract* or *specific*. Abstract components are composed of subordinate components, whereas specific components cannot be further decomposed.

The QUES project can grow and change as the actual software project grows and is refined. It is possible to define a portion of the project static structure, answer questions and compute scores about that portion, and then later add to the static structure. It is not necessary to completely define the static structure down to the lowest level in order to start computing quality scores. Thus QUES can be used from beginning to end of the software development life cycle.

Each project component is explicitly assigned a project level by the Project Manager when it is added to the structure. At this time the component's *framework instance* is created. The framework instance consists of all framework members which have the same project level as the component. The data associated with a component (i.e., the answers to questions and computed scores) is stored within the framework instance. To view the overall system-level quality factor scores, the user opens the framework instance of the system-level project component.

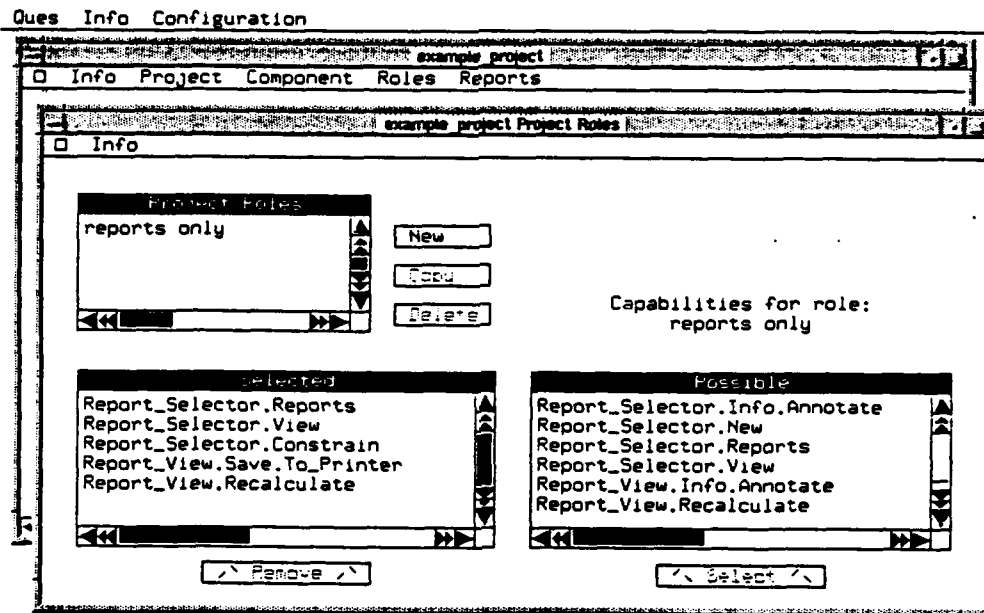


Figure 6.2-1. User Role Definition Window.

Each project component has a component weight which is used in a weighted average computation (wherever the W.AVG function occurs in a scoring equation). By default, all components are equally weighted. Relative weights can be assigned by the Project Manager. Consider this example: the project contains a CSC which is composed of three CSU's, and one CSU has twice as many source lines of code as the other CSU's. In this case the Project Manager may decide to give the larger CSU a relative weight of 2.0 while the other two CSU's have a component weight of 1.0. Component weights, like operand weights in a scoring equation, are normalized during computation so that they sum to 1.0.

6.4 Collecting Project Data

Project data consists of the answers to the questions asked by the framework. There are three ways to input the answers to questions with QUES:

1. Open the question in framework instance of the component
2. Use a data collection form report
3. Use automated data collection

The first method would be a very tedious way to answer a large number of questions for many project components. This method is more suited to spot-checking answers to particular questions or for changing a single answer. The second method is much more efficient. A data collection form is a report that allows data entry directly into the QUES database by designating the answer fields in the report to be editable. (An example of a data collection form is shown in Figure 4.1.1.5-4.) The third method is available only for those questions which can be answered by data collected from the source code by an external tool. This method is discussed in detail in section 7.0 of this report.

The Project Manager will probably assign the task of answering questions about the project to an Engineer user. The Engineer may answer questions any time the data becomes available. Each answer is tagged with a date. If an answer must be modified, it can be input at a later date, and the previous answer will not be lost. When computing scores, the most recent answer will be used unless otherwise specified.

It is possible to annotate an answer with the annotation feature of QUES. Every piece of data in a project, including the project components and the framework members can be assigned one or more notes. Each note consists of a name and a block of text. Three examples of annotation of project data are given in Figure 6.4-1.

□ Info

Annotations

discrepancy
version
description

▲

▲

▲

New

Open

Editor

□ Info

Discrepancy Report #120-1

No reference to documentation of error analysis results.

Editor

□ Info

Version 1.2

Editor

□ Info

Refer to requirement 3.1.1.2 of the SRS.

Figure 6.4-1. Three Examples of Annotation of Answers.

Annotation example #1: Discrepancy reporting.

Every time an answer is "False" indicating non-compliance, the answer is tagged with an annotation called "discrepancy report". The note can be used to explain why the discrepancy exists and to point to a discrepancy report that is filed as part of the software quality assurance activity.

Annotation example #2: Configuration management.

Each answer is tagged with an annotation called "version" which describes the software configuration management version that corresponds to that answer. This is especially useful for answers collected from an automated data collection tool.

Annotation example #3: Descriptive information.

If an answer requires an explanation or description of the analysis used to determine the answer, it is tagged with an annotation called "description". This method is useful for capturing historical information or for documenting analysis when more than one user is answering questions.

6.5 Computing Quality Scores

A score is the result of the computation of a scoring equation. Like the answers to questions, scores are tagged with a date. Scores are computed for the current date, or for a specific date. The latest answer as of the date of computation is used when computing the score.

Project Managers may compute scores on an ad hoc basis, at regular intervals like once a month, or at the end of major milestones. Scores can be tagged with annotations to indicate the occasion with a description such as "after PDR". Scores can be computed for the entire project or for just a single project component. An Engineer who has just answered questions for a group of CSU's may wish to compute the score for the CSC which contains these CSU's to see what effect the new answers have on the CSC score.

Scores can be viewed by opening each framework member in each framework instance or by displaying the scores in a report. Evaluation of scores using reports is discussed next.

6.6 Evaluation of Project Quality with Reports

There are many ways to view the project scores with QUES reports. The Project Manager will probably begin by looking at the overall system results and then continue by viewing progressively greater levels of detail. In this way the Project Manager can isolate problem areas of the project that are contributing to low scores.

6.6.1 Identifying a Problem

For our example framework, we decided to evaluate two quality factors, Reliability and Maintainability. After answering the questions for the System Requirements Analysis/Design phase (phases A), we compute the scores for the entire project. At this point the project consists of a single system-level project component called "the system". We will first compare the factor scores with the goals we have established. Figure 6.6.1-1 shows the factor scores report.

The report indicates that the Reliability factor score does not meet the minimum goal. Next we will look at the criteria scores in an attempt to identify which criteria are causing the low Reliability score. Of course, it could be true that all criteria are equally poor; however, it is usually possible to pinpoint a particular area of concern. Criteria in the RSQF framework represent software-oriented attributes, so it is possible to go beyond saying "the system has a reliability problem" by identifying and correcting a deficiency in the software system design. The next report in Figure 6.6.1-2 compares the criteria scores for all of the criteria which contribute to the factor of Reliability. The QUES report constraint features allows us to filter the data displayed in the report so that only the criteria of interest are shown. Evidently, the low score can be attributed to the criterion " ".

Factor Scores Report			
Project Name:	example_project		
System component:	the system		
Phase name:	Phase.a		
Factor	Low Goal	Scores: Value	Date
Reliability	0.95	0.76	19910925
Maintainability	0.80	0.86	19910925

Figure 6.6.1-1. Factor Score Report.

Criteria Scores Report For Factor = Reliability			
Project Name:	example_project		
System component:	the system		
Phase name:	Phase.a		
Criterion	Low Goal	Scores: Value	Date
AC	0.75	0.33	19910925
AM	0.75	0.94	19910925
SI	0.85	1.00	19910925

Figure 6.6.1-2. Comparison of Criteria Scores for a Single Factor.

6.6.2 Evaluating Compliance

Now that we have identified a particular criterion to investigate, we could do the same thing to the metrics that make up the criterion and try to identify which metrics seem to be problems. However, there are not that many questions in this criterion to evaluate, so we can skip the step of metric evaluation and look right at the questions. One way to identify problem questions is to check for non-compliance, that is, which questions have been answered "False" which is equivalent to a "No". In the RSQF framework, a "No" answer is always bad because it is scored as a zero. With a numeric question, it is difficult to tell if the answer is a problem just by looking at it. For numeric questions, you must consider the scoring equation to determine how the question is used to compute a metric score.

Figure 6.6.2-1 is a compliance report which lists only those questions which have been answered "False". For each question listed in this report we must identify the action to take to improve the system design in order to improve reliability. Once the problems have been corrected, new answers are input to QUES and the system scores are recalculated.

6.6.3 Comparing Project Components

Another technique to identify problem areas is to compare the scores of several project components in the same report. In our example project, we have defined three CSCI's and have answered the questions for the Software Requirements Analysis phase (phase B). The next sample report in Figure 6.6.3-1 compares the factor scores for the three CSCI's. Because we are presenting the scores for two factors, we use a QUES graph to help visualize the data.

6.6.4 Trend Reports

QUES graphs are also useful for showing trends such as comparing the scores across different phases of the life cycle, or for plotting the changes in scores over time. Figure 6.6.4-1 shows a trend plot of factor scores over three phases of the life cycle, and Figure 6.6.4-2 shows a historical plot of a single factor score over time in a single phase.

Compliance Report

project name: example_project

system component: the system

Phase.a

Criterion = AC

<u>Non-compliant Question</u>	<u>Answer</u>	<u>Date</u>
AC.1.1.a	False	19910925
AC.1.2.a	False	19910925
AC.1.5.a	False	19910925
AC.1.6.a	False	19910925

Figure 6.6.2-1. Compliance Report.

Comparison of CSCI Factor Scores

project name: example_project

system component: the system

Phase b

<u>CSCI name</u>	<u>Reliability</u>	<u>Maintainability</u>
csci_1	0.75	0.83
csci_2	0.87	0.56
csci_3	0.92	0.81

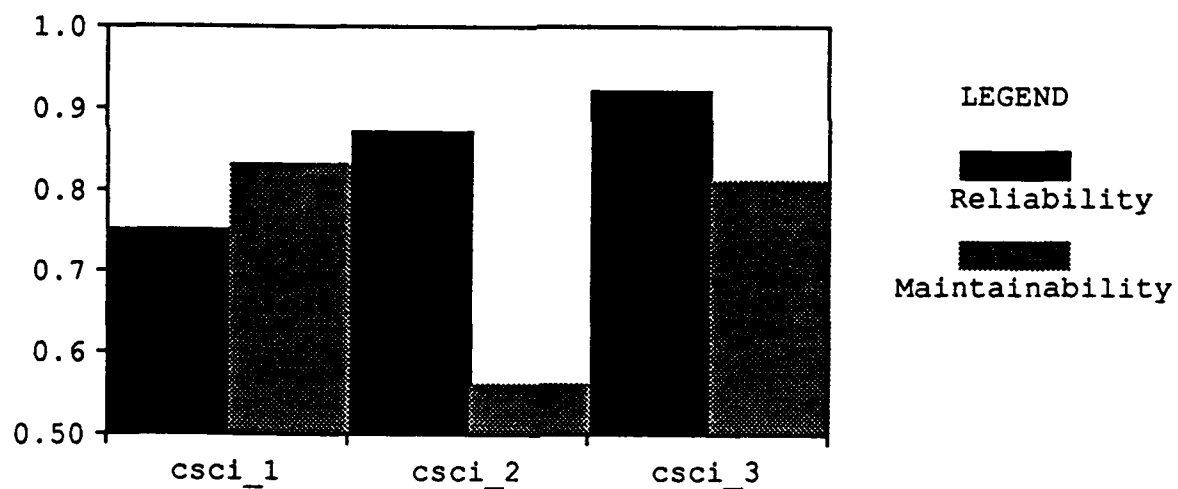


Figure 6.6.3-1. Comparison of Factor Scores for Three CSCI's .

Comparison of Factor Score Across Phases

project name: example_project

system component: the system

Phase	Reliability	Maintainability
Phase.a	0.76	0.86
Phase.b	0.81	0.85
Phase.c	0.85	0.91

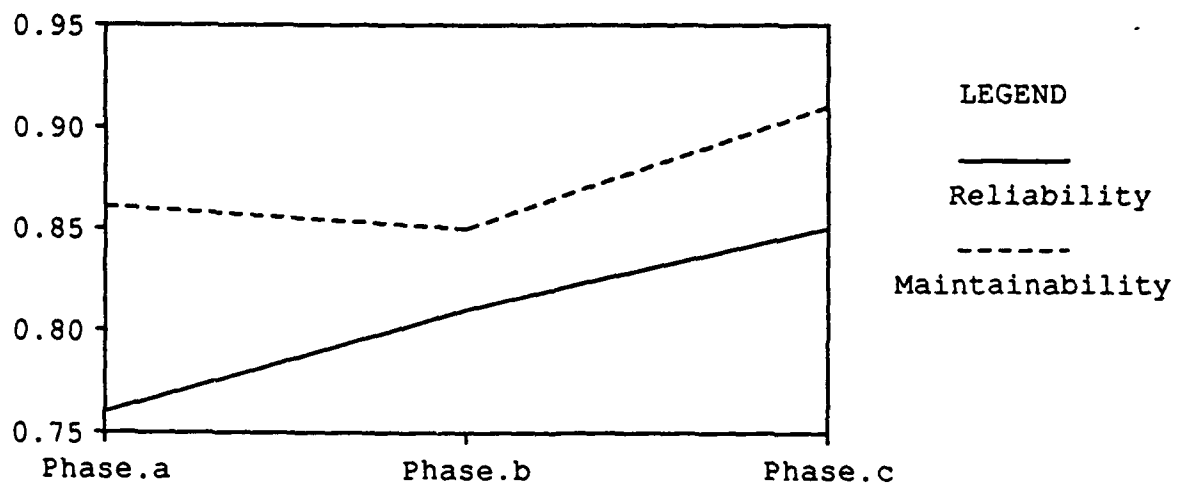


Figure 6.6.4-1. Comparison of Factor Scores Over Three Phases .

Trend Plot of Factor Score

project name: example_project

system component: the system

Phase b

When computed:	Reliability	Maintainability
prelim	0.65	0.50
review a	0.71	0.72
pdr	0.76	0.83
post pdr	0.78	0.85

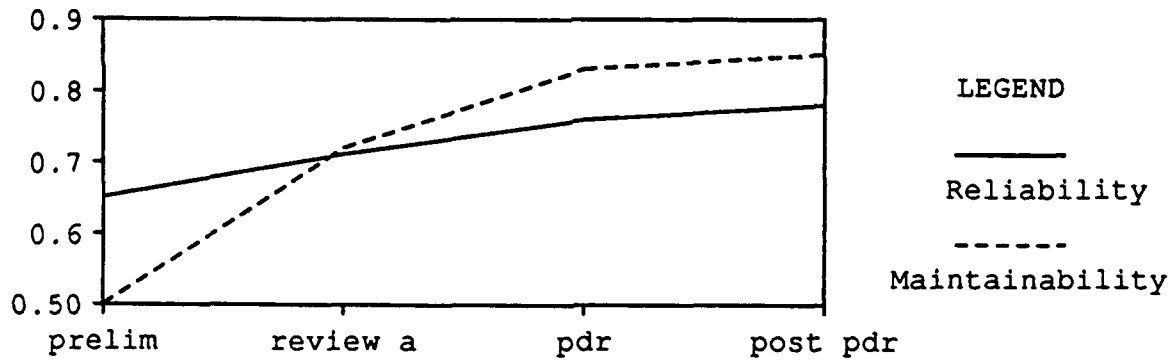


Figure 6.6.4-2. Trend Plot of a Factor Score in a Single Phase.

7.0 Data Collection

Data collection in QUES consists of answering the framework questions for each project component. This would typically be performed by an Engineer user. In general, the answers are input with a Data Collection Form (DCF) report that allows data entry directly into the QUES database. For questions that can be answered by data collected from an external tool, it is not necessary to use a DCF. Automated data is loaded directly into the database. The external tools that interface with QUES currently collect data of use during the Detailed Design (phase D) and Coding and CSU Testing (phase E) phases, and to a very limited extent collect data during phases C and F. Appendix B contains a complete list of the data collected by external tools which can be loaded into the RSQF framework.

7.1 Data Collection Forms

Any QUES report can be set up to accept data entry by designating a field to be *unprotected*. When viewing the report, the user can click on any unprotected field and type in a textual or numeric value. A DCF report is usually set up to display a list of questions for a particular project component, and the answer field is designated as unprotected. Each answer is coupled with a date, but the date field is protected. Typically the data will be entered at the "current date", so it is not necessary to modify the date beside an answer, unless you are attempting to recreate an historical record.

DCF's can be set up to display only a portion of the questions for a particular component by assigning an attribute to each question and then filtering the report of questions by constraining on an attribute. Or a report could be designed to answer the same question for a series of project components. In general, it is easier to think about a single component at a time and answer the questions about it, especially since several questions within a metric may be related.

Figure 7.1-1 gives an example of a DCF which displays a subset of the questions for a particular project component. In this case, we have assigned an attribute "constant" to some questions to indicate that they are to be answered the same way for all project components in the project and the DCF is constrained on the attribute "constant".

Data Collection Form					
Constant Questions					
Project Name: example_project					
Procedure Name: procedure_1.1					
Question: <u>MO.1.8.d</u>					
Is output data passed back to the calling unit? (ANSWER = TRUE for Ada)					
Answers:	<table><thead><tr><th>Value</th><th>Date</th></tr></thead><tbody><tr><td>False</td><td>19910925</td></tr></tbody></table>	Value	Date	False	19910925
Value	Date				
False	19910925				
Question: <u>MO.1.9.d</u>					
Is control always returned to the calling unit when execution is completed? (ANSWER = TRUE for Ada)					
Answers:	<table><thead><tr><th>Value</th><th>Date</th></tr></thead><tbody><tr><td>False</td><td>19910925</td></tr></tbody></table>	Value	Date	False	19910925
Value	Date				
False	19910925				
Question: <u>SI.1.5.d</u>					
How many entrances into the unit? (ANSWER = 1 for Ada)					
Answers:	<table><thead><tr><th>Value</th><th>Date</th></tr></thead><tbody><tr><td>0.0</td><td>19910925</td></tr></tbody></table>	Value	Date	0.0	19910925
Value	Date				
0.0	19910925				
Question: <u>SI.2.1.d</u>					
Is the unit implemented in a structured language or using a preprocessor? (ANSWER = TRUE for Ada)					
Answers:	<table><thead><tr><th>Value</th><th>Date</th></tr></thead><tbody><tr><td>False</td><td>19910925</td></tr></tbody></table>	Value	Date	False	19910925
Value	Date				
False	19910925				

Figure 7.1-1. An Example DCF Constrained on the Attribute "constant".

7.2 Automated Data Collection Tools

Loading data collected from source code by an external tool is actually the final step in a four step process. The process is as follows:

1. Associate Tool Data with a Framework
2. Request the Tool Data
3. Run the External Tool
4. Load the Tool Data

The first step establishes a link between a framework specific member (i.e., a question) and a particular metric data element that is collected by an external tool. Configuring a tool to a QUES database loads a list of all available collected metric data elements. Each question is then linked to one data element from each external tool by using the Select Measurement Data feature. Usually the association is done at framework creation time, so that the association can be used every time the framework is used on a project. However, it is possible to create or modify the tool associations in the project framework template or in the framework instance of a project component. If, for example, some units in a project were coded in Ada, and some in Fortran, the Ada units could be associated with an Ada data collection tool and the Fortran units to a Fortran data collection tool.

Appendix B of this report contains a table which is a cross reference of questions in the RSQF framework to data elements in the three external data collection tools which interface with QUES. The RSQF framework component which is delivered with the QUES tool already has the associations made to the three tools. Figure 7.2-1 shows a report which lists all of the questions in a particular phase of our example framework and the associations to data collection tools.

The next step, requesting the data, is done in a QUES project. Data is requested for the entire project, or for a particular project component. If data is requested for a component, QUES automatically requests data for any subordinate components as well. QUES generates a list of requests which includes the name of the component and the tool data elements that are associated with questions in that component's framework instance. Most questions that are associated with a data collection tool are at the Procedure level. In general, it is important that the name of the project component in QUES matches exactly (case independent) the name of the source code unit that the external tool has analyzed. For an Ada project, it is also important that the QUES Package-level project component name be identical to the actual Ada package that contains the procedure of interest.

Tool Configuration Report		
Tool Name:	adafilter	
Question	Answer Type	Data Element Name
MO.1.5.d	2	number_of_parameters
MO.1.7.d	0	_MO17
MO.1.8.d	0	_MO18
MO.1.9.d	0	_MO19
SI.1.2.d	0	_SI12
SI.1.5.d	2	_SI15
SI.1.6.d	2	_SI16
SI.2.1.d	0	_SI21
SI.3.1.d	2	_SI15
SI.3.2.d	2	_SI16
SI.4.1.d	0	_SI21
SI.4.6.d	2	number_of_FOR
SI.4.7.d	2	_SI47
SI.4.11.d	2	_SI411
SI.4.12.d	2	number_of_local_variables
SI.4.13.d	2	number_of_assignment_statements
SI.4.14.d	2	_SI414
SI.4.15.d	2	_SI415
SI.5.1.d	2	_SI51
SI.5.2.d	2	_SI52
SI.5.3.d	2	number_of_OUT_and_IN_OUT_parameters
SI.6.1.d	2	Halstead_n1
SI.6.2.d	2	Halstead_n2
SI.6.3.d	2	Halstead_N2
VS.1.1.d	2	McCabe
VS.1.3.d	2	number_of_IN_and_IN_OUT_parameters

Figure 7.2-1. Tool Association Summary Report.

The third step, running the external tool, actually occurs outside of QUES. Most source analysis tools require that the source code compile without error. The external tool analyzes the source code and generates a results file which contains the resulting metrics for each source unit. Each time the source undergoes a change, it is necessary to run the analyzing tool again to update the results file. The tool interface is then run which takes as input the tool results file and the request file generated by QUES. The interface extracts the requested information from the results file and creates a QUES-readable output file which is ready for the next step. This file tags each answer with date that the interface tool was run, so that when the data is loaded into QUES, the answers' dates will reflect the date the source analysis was performed, not the date the answers were loaded into the QUES database.

Loading the data into QUES is the final step in the process. Data can be loaded for the entire project or for a particular project component, depending on the menu selection made when requesting the data. If data must be loaded from more than one external tool, it is necessary to repeat steps 2 through 4 for each tool.

Figure 7.2-2. is a report which summarizes the data just loaded from an external tool by constraining the report to display answers to questions with the attribute "automated".

The goal is to extend the external tools set to automate as much as possible of the data collection for the RSQF framework. The automation of data collection provides the greatest leverage in the application of the QUES tool for quality evaluation of software.

Answer Summary Report Automated Questions		
Project Name: example_project		
Procedure Name: procedure_1.1		
Phase.d		
Question	Answer: Value	Date
MO.1.5.d	10.0	19910921
MO.1.7.d	True	19910921
SI.1.2.d	True	19910921
SI.1.6.d	1.0	19910921
SI.3.1.d	12.0	19910921
SI.3.2.d	0.0	19910921
SI.4.1.d	True	19910921
SI.4.6.d	3.0	19910921
SI.4.7.d	0.0	19910921
SI.4.11.d	12.0	19910921
SI.4.12.d	2.0	19910921
SI.4.13.d	3.0	19910921
SI.4.14.d	5.0	19910921
SI.4.15.d	5.0	19910921
SI.5.1.d	3.0	19910921
SI.5.2.d	1.0	19910921
SI.5.3.d	1.0	19910921
SI.6.1.d	6.0	19910921
SI.6.2.d	5.0	19910921
SI.6.3.d	5.0	19910921
VS.1.1.d	23.0	19910921
VS.1.3.d	3.0	19910921

Figure 7.2-2. Answer Summary Report after Data Collection.

8.0 Conclusions and Recommendations

The QUES tool accomplishes the goal of providing improved automated support for the establishment, maintenance, and application of software quality evaluation frameworks. The tool provides interactive framework editing and tailoring capabilities with a state-of-the-practice graphical user interface. The QUES tool interfaces with three external tools to allow automated data collection for Fortran and Ada source code.

The following recommendations are made:

1. *The QUES tool is ready to be applied to pilot projects to assist in transfer of the technology.* The RSQF framework should be applied to actual software development projects to evaluate several quality factors. Projects developed in Ada or Fortran can take advantage of the automated data collection feature. Ideally, the evaluation process should begin at the earliest phase of the development life cycle.
2. *The next step is to focus on increasing the automation of data collection.* Data collection is typically the most labor-intensive part of the process of applying a software quality evaluation framework. Automation of data collection dramatically decreases the cost of applying this technology.

List of References

[BOW85]

Bowen, T.P., Wigle, G.B., and Tsai, J.T., Specification of Software Quality Attributes Final Technical Report, RADC-TR-85-37, Vols 1-3, February 1985.

[AFS86]

Software Management Indicators: Management Insight, AFSCP 800-43, Air Force Systems Command, Andrews AFB, DC, January 13, 1986.

[AFS87]

Software Quality Indicators: Management Quality Insight, AFSCP 800-14, Air Force Systems Command, Andrews AFB, DC, January 20, 1987.

Acronyms List

AFSC	-- Air Force Systems Command
CSC	-- Computer Software Component
CSCI	-- Computer Software Configuration Item
CSM	-- Code Size Metrics
CSU	-- Computer Software Unit
DCF	-- Data Collection Form
GIO	-- General Information Object
LOC	-- Lines of Code
PDR	-- Preliminary Design Review
PTR	-- Problem Trouble Report
QUES	-- Quality Evaluation System
RADC	-- Rome Air Development Center
RL	-- Rome Laboratory
RSQF	-- Rome Laboratory Software Quality Framework
SAP	-- Static Analyzer Program
SLCSE	-- Software Life Cycle Support Environment
SLOC	-- Source Lines of Code
SMQI	-- Software Management and Quality Indicators
W.AVG	-- Weighted Average

Appendix A -- Analysis of Questions in the RSQF

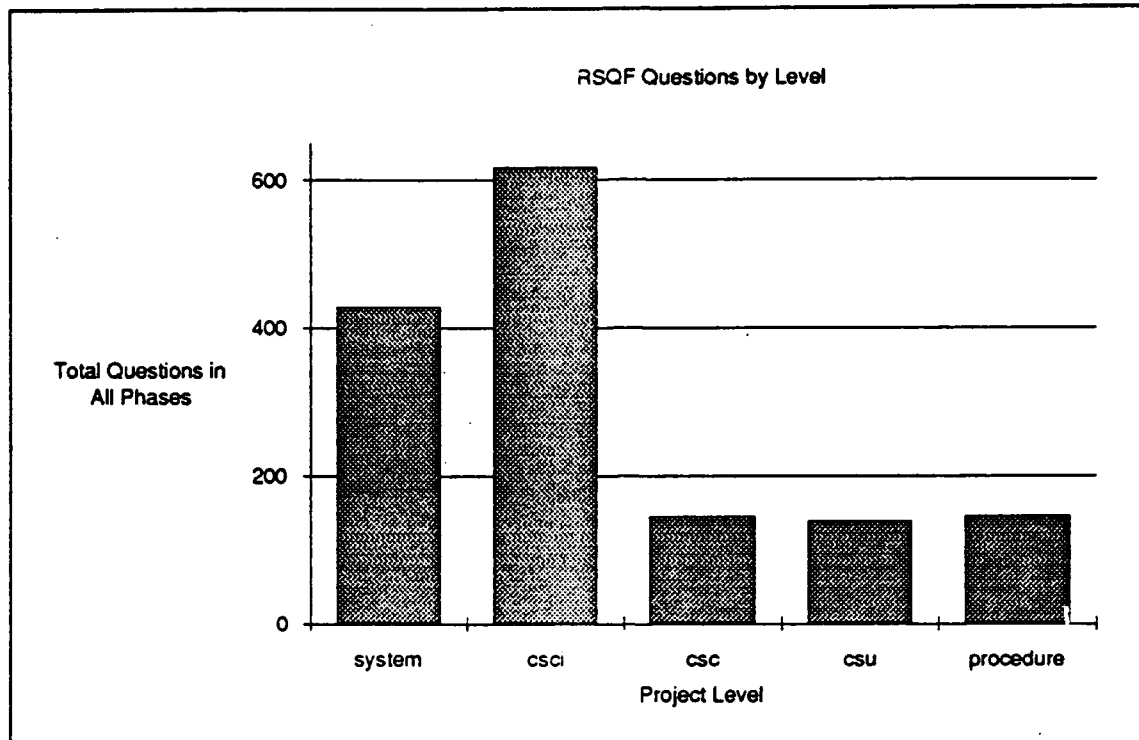


Figure A-1. Number of Questions at Each Project Level.

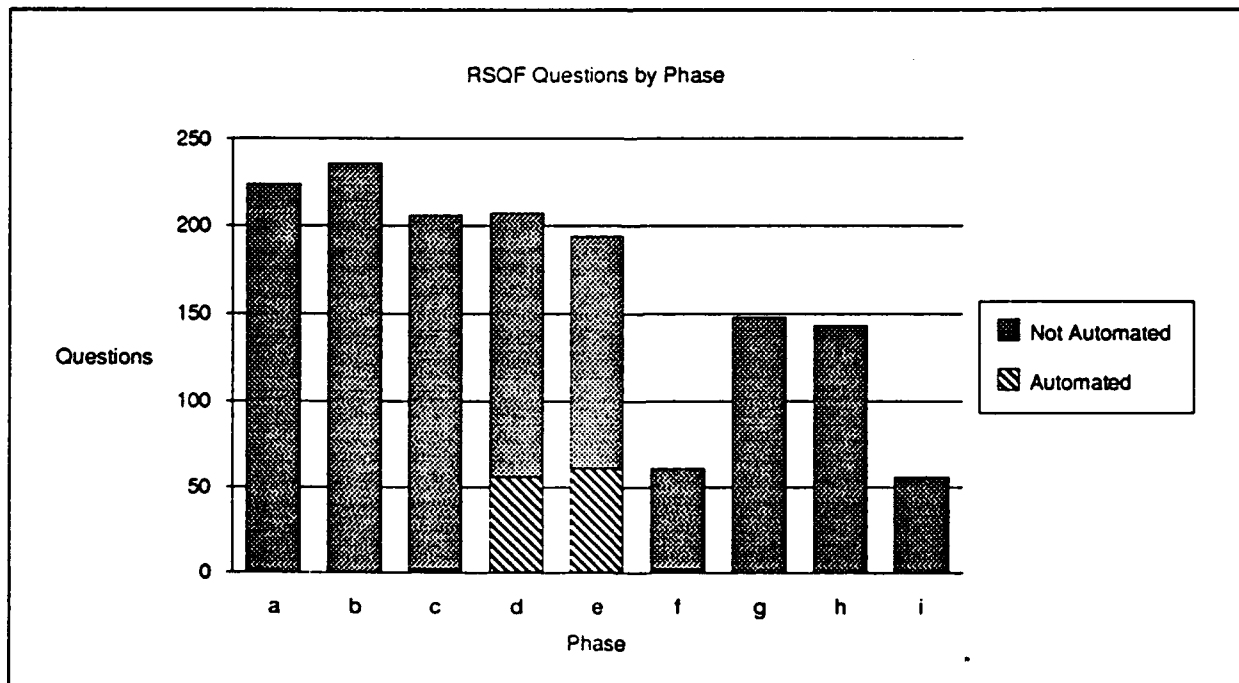


Figure A-2. Number of Questions for Each Phase.

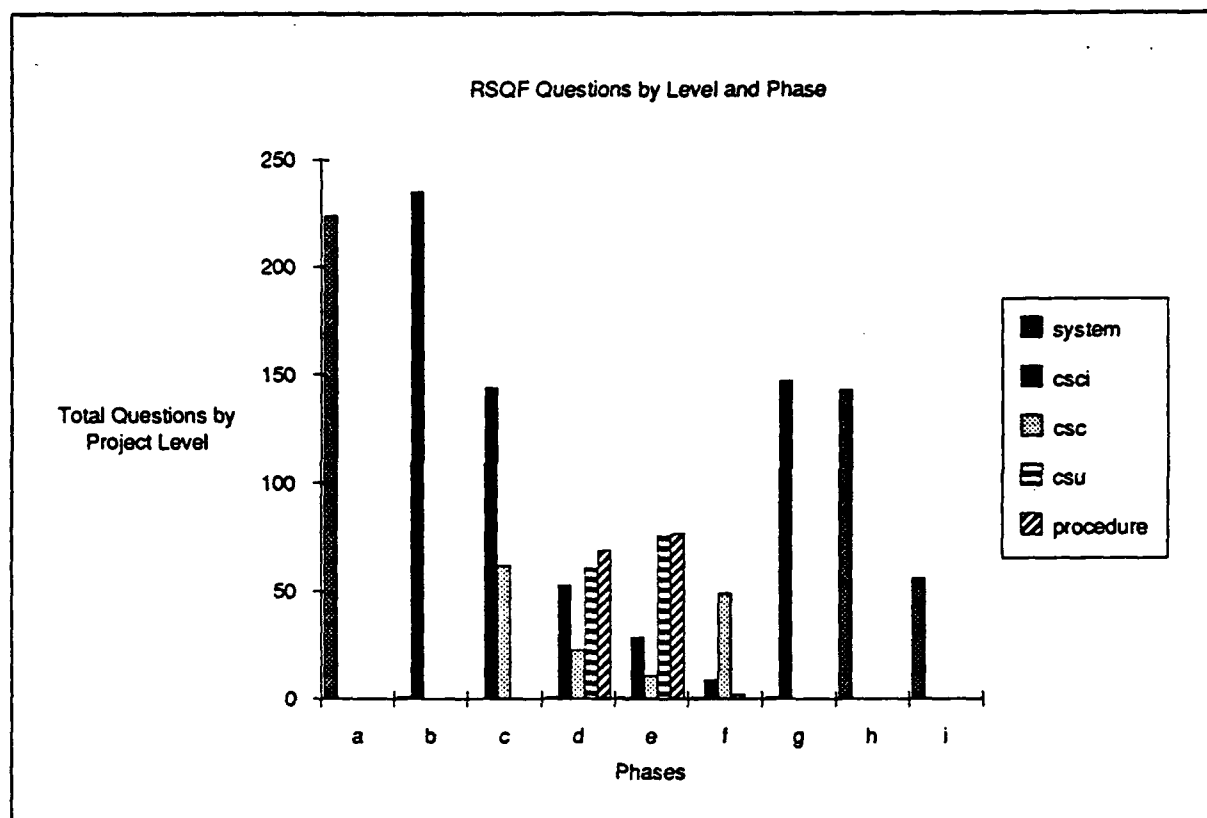


Figure A-3. Number of Questions in Each Phase by Project Level.

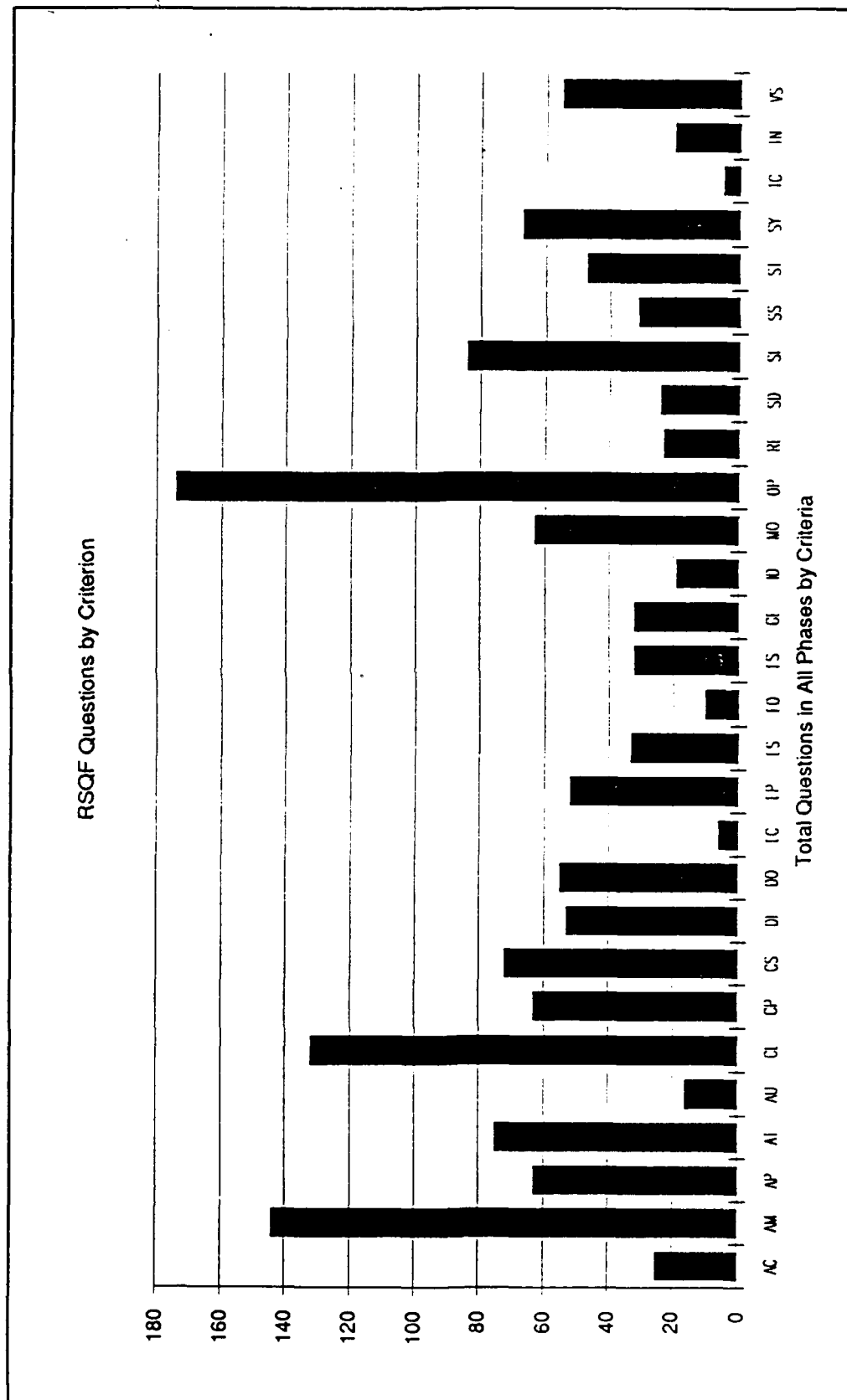


Figure A-4. Number of Questions by Criterion.

**Appendix B -- Cross Reference of Questions in the RSQF to
Automated Data Collection Tools**

Quality Evaluation System

SQF Question	Tool	Type	Project Level	Data Element Name
AP.2.1.d	slcse	real	procedure	NUM_DECL_PARM
AP.2.1.d	sap	real	procedure	ICTARG
AP.2.1.d	adafilter	real	procedure	number_of_parameters
AP.2.1.e	slcse	real	procedure	NUM_DECL_PARM
AP.2.1.e	sap	real	procedure	ICTARG
AP.2.1.e	adafilter	real	procedure	number_of_parameters
AP.2.2.d	sap	real	procedure	ICTEXT
AP.2.2.d	adafilter	real	procedure	number_of_global_referenced
AP.2.2.e	sap	real	procedure	ICTEXT
AP.2.2.e	adafilter	real	procedure	number_of_global_referenced
AP.3.3.e	adafilter	real	procedure	lines_of_code
AP.3.3.e	slcse	real	procedure	NUM_ADA_LINES
AP.3.3.e	sap	real	procedure	ICTSCD
AP.4.1.e	adafilter	boolean	procedure	micro_code
AT.1.3.c	slcse	real	csc	CSC_MEMORY_BUDGET
AT.1.3.d	slcse	real	csc	CSC_MEMORY_BUDGET
AT.1.3.f	slcse	real	csc	CSC_MEMORY_BUDGET
AT.2.4.c	slcse	real	csc	CSC_PROCESS_TIME_ BUDGET
AT.2.4.d	slcse	real	csc	CSC_PROCESS_TIME_ BUDGET
AT.2.4.f	slcse	real	csc	CSC_PROCESS_TIME_ BUDGET
AU.1.2.d	slcse	real	procedure	NUM_ADA_LINES
AU.1.2.d	sap	real	procedure	ICTSCD
AU.1.2.d	adafilter	real	procedure	lines_of_code
AU.1.3.d	sap	real	procedure	CLA07T
AU.1.3.e	sap	real	procedure	CLA07T
CP.1.2.d	sap	real	procedure	ICTREF
CP.1.2.d	adafilter	real	procedure	_CP12
CP.1.2.e	sap	real	procedure	ICTREF
CP.1.2.e	adafilter	real	procedure	_CP12
CP.1.11.d	adafilter	boolean	procedure	_CP111
CP.1.11.e	adafilter	boolean	procedure	_CP111

QQuality Evaluation System

SQF Question	Tool	Type	Project Level	Data Element Name
CP.1.4.d	adafilter	real	procedure	IN_and_IN_OUT_plus_globals
CP.1.4.e	adafilter	real	procedure	IN_and_IN_OUT_plus_globals
CP.1.5.d	sap	real	procedure	ICTCBV
CP.1.5.d	adafilter	real	procedure	_CP15
CP.1.5.e	sap	real	procedure	ICTCBV
CP.1.5.e	adafilter	real	procedure	_CP15
EP.1.5.d	slcse	real	procedure	NUM_LOOP_STMTS
EP.1.5.d	sap	real	procedure	ENDDO
EP.1.5.d	adafilter	real	procedure	_EP15
EP.1.5.e	slcse	real	procedure	NUM_LOOP_STMTS
EP.1.5.e	sap	real	procedure	ENDDO
EP.1.5.e	adafilter	real	procedure	_EP15
EP.1.7.d	slcse	real	procedure	NUM_CMPD_STMTS
EP.1.7.d	adafilter	real	procedure	number_of_compound_
				expressions
EP.1.7.e	slcse	real	procedure	NUM_CMPD_STMTS
EP.1.7.e	adafilter	real	procedure	number_of_compound_
				expressions
EP.1.10.d	adafilter	real	procedure	number_of_PACK
EP.1.10.e	adafilter	real	procedure	number_of_PACK
EP.2.3.d	sap	real	procedure	CLA13T
EP.2.3.d	adafilter	real	procedure	number_of_arithmetic_
				expressions
EP.2.3.e	sap	real	procedure	CLA13T
EP.2.3.e	adafilter	real	procedure	number_of_arithmetic_
				expressions
EP.2.5.d	adafilter	real	procedure	number_of_mixed_type_
				expressions
EP.2.5.e	adafilter	real	procedure	number_of_mixed_type_
				expressions
EP.2.6.e	sap	real	procedure	ICTREF
EP.2.7.d	adafilter	real	procedure	_EP27
EP.2.7.e	adafilter	real	procedure	_EP27
ES.1.8.d	adafilter	boolean	procedure	_ES18
ES.1.8.e	adafilter	boolean	procedure	_ES18
ID.1.5.d	adafilter	boolean	procedure	_ID15
ID.1.5.e	adafilter	boolean	procedure	_ID15
ID.2.3.d	adafilter	boolean	procedure	someio
ID.2.3.e	adafilter	boolean	procedure	someio

QQuality Evaluation System

SQF Question	Tool	Type	Project Level	Data Element Name
ID.2.5.d	adafilter	boolean	procedure	machine_dependent_data
ID.2.5.e	adafilter	boolean	procedure	machine_dependent_data
MO.1.4.e	adafilter	boolean	procedure	_MO14
MO.1.5.d	slcse	real	procedure	NUM_DECL_PARM
MO.1.5.d	sap	real	procedure	ICTARG
MO.1.5.d	adafilter	real	procedure	number_of_parameters
MO.1.5.e	slcse	real	procedure	NUM_DECL_PARM
MO.1.5.e	sap	real	procedure	ICTARG
MO.1.5.e	adafilter	real	procedure	number_of_parameters
MO.1.7.d	adafilter	boolean	procedure	_MO17
MO.1.7.e	adafilter	boolean	procedure	_MO17
MO.1.8.d	adafilter	boolean	procedure	_MO18
MO.1.8.e	adafilter	boolean	procedure	_MO18
MO.1.9.d	adafilter	boolean	procedure	_MO19
MO.1.9.e	adafilter	boolean	procedure	_MO19
MO.2.5.d	slcse	real	csu	CSU_COHESION
MO.2.5.e	slcse	real	csu	CSU_COHESION
SD.1.2.e	slcse	real	procedure	NUM_COMMENT_LINES
SD.1.2.e	sap	real	procedure	ICTSCM
SD.1.2.e	adafilter	real	procedure	number_of_comment_lines
SD.1.3.e	slcse	real	procedure	NUM_CODE_COM_LINES
SD.1.3.e	sap	real	procedure	ICTSXP
SD.1.3.e	adafilter	real	procedure	number_of_embedded_comment_lines
SD.3.1.e	slcse	real	procedure	CSU_STD_PROG_LANG
SD.3.4.e	slcse	real	procedure	NUM_2PLUS_SEMI_PER_LINE
SD.3.5.e	slcse	real	procedure	NUM_0_SEMI_PER_LINE
SI.1.2.d	adafilter	boolean	procedure	_SI12
SI.1.2.e	adafilter	boolean	procedure	_SI12
SI.1.5.d	slcse	real	procedure	NUM_ENTRY_STMTS
SI.1.5.d	sap	real	procedure	ICTENT
SI.1.5.d	adafilter	real	procedure	_SI15
SI.1.5.e	slcse	real	procedure	NUM_ENTRY_STMTS
SI.1.5.e	sap	real	procedure	ICTENT
SI.1.5.e	adafilter	real	procedure	_SI15

Quality Evaluation System

SQF Question	Tool	Type	Project Level	Data Element Name
SI.1.6.d	slcse	real	procedure	NUM_RETURN_STMTS
SI.1.6.d	sap	real	procedure	RETURN
SI.1.6.d	adafilter	real	procedure	_SI16
SI.1.6.e	slcse	real	procedure	NUM_RETURN_STMTS
SI.1.6.e	sap	real	procedure	RETURN
SI.1.6.e	adafilter	real	procedure	_SI16
SI.1.7.d1	sap	real	procedure	ICTCBV
SI.2.1.d	adafilter	boolean	procedure	_SI21
SI.3.1.d	sap	real	procedure	CLA02T
SI.3.1.d	adafilter	real	procedure	_SI31
SI.3.1.d	slcse	real	procedure	NUM_BRANCHES
SI.3.1.e	sap	real	procedure	CLA02T
SI.3.1.e	adafilter	real	procedure	_SI31
SI.3.1.e	slcse	real	procedure	NUM_BRANCHES
SI.3.2.d	adafilter	real	procedure	_SI32
SI.3.2.e	adafilter	real	procedure	_SI32
SI.4.1.d	adafilter	boolean	procedure	_SI41
SI.4.1.e	adafilter	boolean	procedure	_SI41
SI.4.5.d	slcse	real	procedure	NUM_EXIT_STMTS
SI.4.5.e	slcse	real	procedure	NUM_EXIT_STMTS
SI.4.6.d	slcse	real	procedure	NUM_FOR_LOOP_STMTS
SI.4.6.d	sap	real	procedure	DO
SI.4.6.d	adafilter	real	procedure	number_of_FOR
SI.4.6.e	slcse	real	procedure	NUM_FOR_LOOP_STMTS
SI.4.6.e	sap	real	procedure	DO
SI.4.6.e	adafilter	real	procedure	number_of_FOR
SI.4.7.d	adafilter	real	procedure	_SI47
SI.4.7.e	adafilter	real	procedure	_SI47
SI.4.9.e	slcse	real	procedure	NUM_STMT_LABEL
SI.4.9.e	sap	real	procedure	ICTGLB
SI.4.10.d	sap	real	procedure	MIFLEV
SI.4.10.e	sap	real	procedure	MIFLEV
SI.4.11.d	slcse	real	procedure	NUM_BRANCHES
SI.4.11.d	sap	real	procedure	ICTTBR
SI.4.11.d	adafilter	real	procedure	_SI411
SI.4.11.e	slcse	real	procedure	NUM_BRANCHES
SI.4.11.e	sap	real	procedure	ICTTBR
SI.4.11.e	adafilter	real	procedure	_SI411

QUality Evaluation System

SQF Question	Tool	Type	Project Level	Data Element Name
SI.4.12.d	sap	real	procedure	CLA06T
SI.4.12.d	adafilter	real	procedure	number_of_local_variables
SI.4.12.e	sap	real	procedure	CLA06T
SI.4.12.e	adafilter	real	procedure	number_of_local_variables
SI.4.13.d	sap	real	procedure	CLA01T
SI.4.13.d	adafilter	real	procedure	number_of_assignment_ statements
SI.4.13.e	sap	real	procedure	CLA01T
SI.4.13.e	adafilter	real	procedure	number_of_assignment_ statements
SI.4.14.d	sap	real	procedure	ICTREF
SI.4.14.d	adafilter	real	procedure	_SI414
SI.4.14.e	sap	real	procedure	ICTREF
SI.4.14.e	adafilter	real	procedure	_SI414
SI.4.15.d	sap	real	procedure	ICTVAR
SI.4.15.d	adafilter	real	procedure	_SI415
SI.4.15.e	sap	real	procedure	ICTVAR
SI.4.15.e	adafilter	real	procedure	_SI415
SI.5.1.d	adafilter	real	procedure	_SI51
SI.5.1.e	adafilter	real	procedure	_SI51
SI.5.2.d	adafilter	real	procedure	_SI52
SI.5.2.e	adafilter	real	procedure	_SI52
SI.5.3.d	adafilter	real	procedure	number_of_OUT_and_IN_ OUT_parameters
SI.5.3.e	adafilter	real	procedure	number_of_OUT_and_IN_ OUT_parameters
SI.6.1.d	sap	real	procedure	IETA1
SI.6.1.d	adafilter	real	procedure	Halstead_n1
SI.6.1.e	sap	real	procedure	IETA1
SI.6.1.e	adafilter	real	procedure	Halstead_n1
SI.6.2.d	sap	real	procedure	IETA2
SI.6.2.d	adafilter	real	procedure	Halstead_n2
SI.6.2.e	sap	real	procedure	IETA2
SI.6.2.e	adafilter	real	procedure	Halstead_n2
SI.6.3.d	sap	real	procedure	NETA2
SI.6.3.d	adafilter	real	procedure	Halstead_N2
SI.6.3.e	sap	real	procedure	NETA2
SI.6.3.e	adafilter	real	procedure	Halstead_N2

Quality Evaluation System

SQF Question	Tool	Type	Project Level	Data Element Name
ST.1.1.d	adafilter	real	procedure	number_of_global_plus_parameters
ST.1.1.d	slcse	real	procedure	NUM_DECL_PARM
ST.1.1.e	adafilter	real	procedure	number_of_global_plus_parameters
ST.2.1.d	sap	real	procedure	MCCABE
ST.2.1.d	adafilter	real	procedure	McCabe_and_RAISE
ST.2.1.e	sap	real	procedure	MCCABE
ST.2.1.e	adafilter	real	procedure	McCabe_and_RAISE
ST.2.2.d	adafilter	real	procedure	_ST22
ST.2.2.d	slcse	real	procedure	NUM_BRANCHES
ST.2.2.e	adafilter	real	procedure	_ST22
ST.2.2.e	slcse	real	procedure	NUM_BRANCHES
ST.2.3.d	sap	real	procedure	ICTSUB
ST.2.3.d	adafilter	real	procedure	calls
ST.2.3.e	sap	real	procedure	ICTSUB
ST.2.3.e	adafilter	real	procedure	calls
ST.2.4.d	slcse	real	procedure	NUM_FOR_LOOP_STMTS
ST.2.4.d	sap	real	procedure	DO
ST.2.4.d	adafilter	real	procedure	_ST24
ST.2.4.e	slcse	real	procedure	NUM_FOR_LOOP_STMTS
ST.2.4.e	sap	real	procedure	DO
ST.2.4.e	adafilter	real	procedure	_ST24
ST.4.5.d	adafilter	boolean	procedure	_ST45
ST.4.5.e	adafilter	boolean	procedure	_ST45
VS.1.1.d	sap	real	procedure	MCCABE
VS.1.1.d	adafilter	real	procedure	McCabe
VS.1.1.e	sap	real	procedure	MCCABE
VS.1.1.e	adafilter	real	procedure	McCabe
VS.1.3.d	adafilter	real	procedure	number_of_IN_and_IN_OUT_parameters
VS.1.3.e	adafilter	real	procedure	number_of_IN_and_IN_OUT_parameters

**MISSION
OF
ROME LABORATORY**

Rome Laboratory plans and executes an interdisciplinary program in research, development, test, and technology transition in support of Air Force Command, Control, Communications and Intelligence (C³I) activities for all Air Force platforms. It also executes selected acquisition programs in several areas of expertise. Technical and engineering support within areas of competence is provided to ESD Program Offices (POs) and other ESD elements to perform effective acquisition of C³I systems. In addition, Rome Laboratory's technology supports other AFSC Product Divisions, the Air Force user community, and other DOD and non-DOD agencies. Rome Laboratory maintains technical competence and research programs in areas including, but not limited to, communications, command and control, battle management, intelligence information processing, computational sciences and software producibility, wide area surveillance/sensors, signal processing, solid state sciences, photonics, electromagnetic technology, superconductivity, and electronic reliability/maintainability and testability.