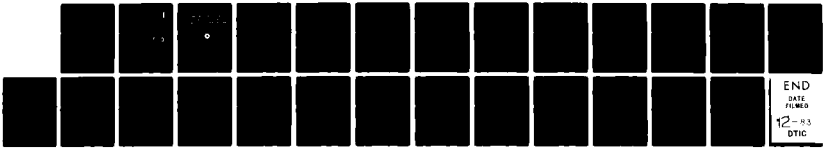
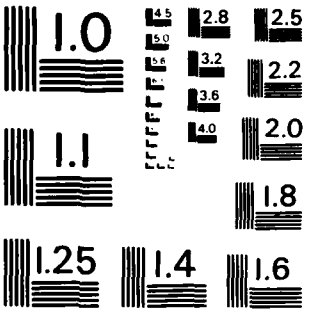
AD-A134 808 SOFTWARE MAINTAINABILITY FACTORS FOR THE ATC SYSTEM COMPUTER REPLACEMENT PROGRAM(U) SYSTEMS AND APPLIED SCIENCES CORP RIVERDALE MD H HECHT APR 80 1/1

UNCLASSIFIED SASC-CD-7260-001 DOT-FA79WAI-081 F/G 9/2 NL



END
DATE FILMED
12-83
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS - 1963 - A

PHOTOGRAPH THIS SHEET

AD-A134 808

DTIC ACCESSION NUMBER

II

LEVEL

INVENTORY

Rpt. No. SASC-CD-7260-001

DOCUMENT IDENTIFICATION Final, Apr. '80

Hecht, Herbert

Contract DOT-FA-79WAI-081

DISTRIBUTION STATEMENT A

Approved for public release;
Distribution Unlimited

DISTRIBUTION STATEMENT

ACCESSION FOR	
NTIS	GRA&I
DTIC	TAB
UNANNOUNCED	
JUSTIFICATION	
BY	
DISTRIBUTION /	
AVAILABILITY CODES	
DIST	AVAIL AND/OR SPECIAL
A-1	

DISTRIBUTION STAMP

DTIC
ELECTE
S NOV 18 1983 D
D

DATE ACCESSIONED

DTIC COPY INSPECTED

83 11 15 177

DATE RECEIVED IN DTIC

PHOTOGRAPH THIS SHEET AND RETURN TO DTIC-DDA-2

DTIC FORM OCT 79 70A

DOCUMENT PROCESSING SHEET

SOFTWARE MAINTAINABILITY FACTORS FOR THE ATC SYSTEM COMPUTER REPLACEMENT PROGRAM

APRIL 1980
FINAL REPORT

Prepared for

U.S. DEPARTMENT OF TRANSPORTATION
FEDERAL AVIATION ADMINISTRATION
Systems Research & Development Service
Washington, D.C. 20590

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.

4. Title and Subtitle	5. Report Date
Software Maintainability Factors for the ATC System Computer Replacement Program	April 1980
	6. Performing Organization Code
7. Author(s) Dr. Herbert Hecht/SoHaR Inc.	8. Performing Organization Report No. SASC-CD-7260-001
9. Performing Organization Name and Address Systems and Applied Sciences Corporation 6811 Kenilworth Avenue Riverdale, MD 20840	10. Work Unit No. (TRAIS)
	11. Contract or Grant No. DOT-FA-79WA1-081
12. Sponsoring Agency Name and Address U.S. Department of Transportation Federal Aviation Administration Systems Research and Development Service Washington D.C. 20591	13. Type of Report and Period Covered Final Report
	14. Sponsoring Agency Code ARD-131

15. Supplementary Notes

16. Abstract

The purpose of the Software Maintainability Factors Study is to identify detail elements of software design and development that enhance the maintainability of the resulting software product. As part of the Software Maintainability Factors Study, the extensive literature on software maintainability has been surveyed and key concepts that deal with maintainability factors have been synopsized. These approaches are then analyzed in the light of specific requirements of the ATC environment.

17. Key Words	18. Distribution Statement
Maintainability factor Software designs Software development	

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	224	

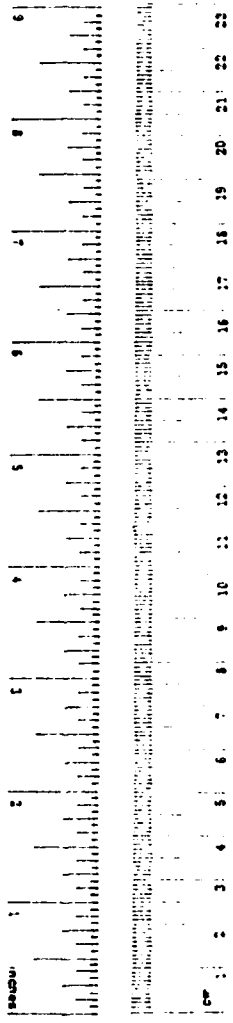
Form DOT F 1700.7 8-72 Reproduction of completed page authorized

METRIC CONVERSION FACTORS

Approximate Conversions to Metric Measures

Symbol	When You Know	Multiply by	To Find	Symbol
		LENGTH		
in	inches	*2.5	centimeters	cm
ft	feet	30	centimeters	cm
yd	yards	0.9	meters	m
mi	miles	1.6	kilometers	km
		AREA		
in^2	square inches	6.5	square centimeters	cm^2
ft^2	square feet	0.09	square meters	m^2
yd^2	square yards	0.8	square meters	m^2
mi^2	square miles	2.6	square kilometers	km^2
	acres	0.4	hectares	ha
		MASS (weight)		
oz	ounces	28	grams	g
lb	pounds	0.45	kilograms	kg
	short tons (2000 lb)	0.9	tonnes	t
		VOLUME		
tsp	teaspoons	5	milliliters	ml
Tbsp	tablespoons	15	milliliters	ml
fl oz	fluid ounces	30	milliliters	ml
c	cups	0.24	liters	l
pt	pints	0.47	liters	l
qt	quarts	0.95	liters	l
gal	gallons	3.8	liters	l
ft^3	cubic feet	0.03	cubic meters	m^3
yd^3	cubic yards	0.76	cubic meters	m^3
		TEMPERATURE (exact)		
°F	Fahrenheit temperature	5/9 (after subtracting 32)	Celsius temperature	°C

*1 in = 2.54 (exactly). For other exact conversions and more detailed tables, see NBS Misc. Publ. 286, Units of Weights and Measures. Price $2.25, SD Catalog No. C13.10:286.



Approximate Conversions from Metric Measures

Symbol	When You Know	Multiply by	To Find	Symbol
		LENGTH		
mm	millimeters	0.04	inches	in
cm	centimeters	0.4	inches	in
m	meters	3.3	feet	ft
m	meters	1.1	yards	yd
km	kilometers	0.6	miles	mi
		AREA		
cm^2	square centimeters	0.16	square inches	in^2
m^2	square meters	1.2	square yards	yd^2
km^2	square kilometers	0.4	square miles	mi^2
ha	hectares (10,000 m^2)	2.5	acres	
		MASS (weight)		
g	grams	0.035	ounces	oz
kg	kilograms	2.7	pounds	lb
t	tonnes (1000 kg)	1.1	short tons	
		VOLUME		
ml	milliliters	0.03	fluid ounces	fl oz
l	liters	2.1	pints	pt
l	liters	1.06	quarts	qt
l	liters	0.26	gallons	gal
m^3	cubic meters	35	cubic feet	ft^3
m^3	cubic meters	1.3	cubic yards	yd^3
		TEMPERATURE (exact)		
°C	Celsius temperature	9/5 (then add 32)	Fahrenheit temperature	°F

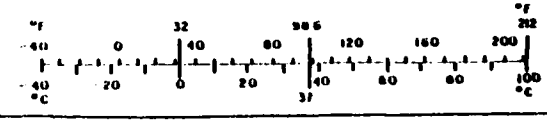


TABLE OF CONTENTS

SECTION 1
INTRODUCTION

This report on the Maintainability Factor Study is the first document generated under Task 10, Software Reliability/ Maintainability Survey, of the Air Traffic Control Computer Replacement Program V&V Subprogram. Subsequent deliverables under this task deal with reliability factors, and with design techniques and development techniques that enhance reliability and maintainability.

The following paragraphs of this introduction describe the scope of the maintainability factors study and discuss definitions of maintainability applicable to the ATC Computer Replacement Program.

1.1 SCOPE OF THE MAINTAINABILITY FACTORS STUDY

The purpose of the Maintainability Factors Study is to identify detail elements of software design and development that enhance the maintainability of the resulting software product. Requirements for the design structures and development techniques that will provide a highly maintainable software base for the ATC Computer Replacement Program will be generated in subsequent effort under Task 10.

As part of the Maintainability Factors Study the extensive literature on software maintainability has been surveyed, and key concepts that deal with maintainability factors have been synopsized in section 2 of this report. These approaches are then analyzed in the light of specific requirements of the ATC environment in section 3, and a set of factors that is well suited for that environment is identified in section 4.

1.2 DEFINITIONS OF MAINTAINABILITY

The definitions of maintainability offered in the existing

software literature fall into three broad categories: those based on computer systems concepts (applicable to hardware and software), those viewing maintainability as a primary measurable quantity, and those viewing maintainability as composed of a number of more elementary factors (composite definitions).

1.2.1 Computer System Based Definition

The general definition of maintainability applicable to defense systems is the prototype of this category:

> A characteristic of design and installation which is expressed as the probabilty that an item will be retained in, or restored to, a specified condition within a given period of time, when the maintenance is performed in accordance with prescribed procedures and resources [1].

The draft of a recent computer dictionary offers the following related formulation:

> The ease with which maintenance of a functional unit can be performed in accordance with prescribed requirements [2].

In the same dictionary, maintenance is defined as "Any activitity intended to keep equipment or programs in satisfactory working condition".

1.2.2 Direct Measurement Definitions

A typical definition of this type is:

> Maintainability is the probability that, when maintenance action is initiated under stated conditions, a failed system will be restored to operable condition within a specified time. [3]

Although no specific software terms are mentioned in the definition, it appears under a heading of "Software

Maintainability Measurement" and it is definitely aimed at computer programs. This defintion has also been adopted as an alternative in a recent software glossary [4].

1.2.3 Composite Definitions

A very concise definition in this category is:

> Code possesses the characteristic maintainability to the extent that it facilitates updating to satisfy new requirements or to correct deficiencies. This implies that the code is understandable, testable, and modifiable [5].

This definition has been adopted as the primary one in the above mentioned software glossary [4]. Related definitions are presented in several sources under the entry "Maintainable". Thus,

> A maintainable software product is one which is understandable, testable, and easy to modify [5].

In other formulations, the identification of elements has been expanded [3,4], e. g., modifiability is described as applicable to both the program and the documentation. This expansion does not represent a benefit for the present effort because it is only an intermediate step toward an analysis of the factors of maintainability which is the subject of the following section.

1.2.4 Definition Adopted Here.

Although the ATC environment places a high premium on systems oriented measures of reliability and maintainability (i. e., those that are consistent for hardware and software), it is not believed that the definitions presented in 1.2.1 are suitable. The primary difficulties are (a) that software maintenance is frequently undertaken while the system is

operable, and that therefore the time to restore service is not an appropriate measure, and (b) that restoration of service can frequently be accomplished by temporary measures that do not constitute software maintenance (e. g., program restart). Essentially the same objections also apply to the definitions discussed under 1.2.2.

The composite definitions are found to be more suitable, and the first one discussed under that heading is the most general one. Because it is the purpose of this volume to identify factors that contribute to maintainability we do not at this time wish to prejudice this effort by including the implicit factors identification in the second sentence of that definition. For this reason it will be modified to the form:

> Maintainability is a characteristic of software that permits it to be easily updated to satisfy new requirements or to correct deficiencies.

SECTION 2
FACTORS IDENTIFIED IN PRIOR LITERATURE

There is a fairly extensive literature on factors that affect the maintainability of software. The significant contributions to the field can be broken down into three distinct categories:

Quality factors
Reason for change factors
Activity factors

These are discussed in this sequence in the body of this section.

2.1 QUALITY FACTORS

The pioneer work in this category was published as a TRW report in 1973 [5], and excerpts or updates of this can be found in the professional as well as in the trade literature, e. g., [6, 7]. As part of an overall hierarchical structure of software quality factors, the primary ones affecting maintainability were identified as testablity, understandability, and modifiabilty. Note that this structuring has now been included in some of the definitions of maintainability (see 1.2.3 of this volume). On detailed examination, these factors were seen to be decomposable into primitives of consistency, accessibility, communicativeness, structuredness, self-descriptiveness, conciseness, legibility, and augmentability.

A further significant work in this category was published as an RADC report by a group of researchers at General Electric, Sunnyvale [8]. Maintainability is here broken down directly into five primitives: consistency, simplicity, conciseness, modularity, and self-descriptiveness. The reference equates modularity with structuredness, and it is thus seen that four out of these five primitives are identical with those identified in

the TRW study. The one least matchable is simplicity, which partly overlaps the legibility factor among the TRW primitives. Definitions of the factors are provided in the Appendix.

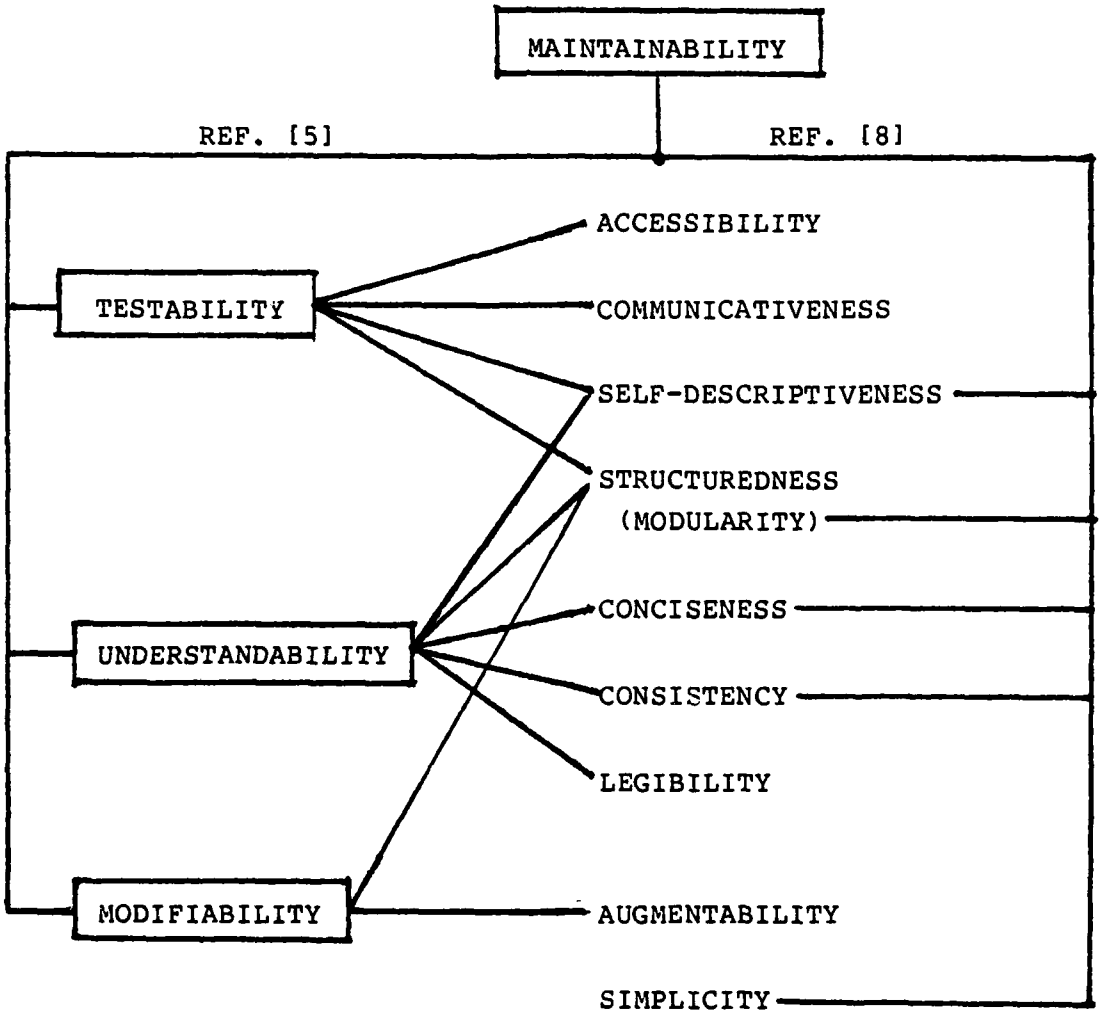


FIGURE 2 - 1
MAINTAINABILITY FACTORS FROM REFS. [5] AND [8]

A comparison of the quality factors for reliability as defined by TRW and GE is shown in figure 2-1. It is seen that the GE study does not consider communicativeness, accessibility, and augmentability. These omissions are due at least partly to a very narrow definition of maintainability as exclusively concerned with error correction: "Effort required to locate and fix an error in an operational program". As will be discussed below, this does not appear to be an adequate scoping of software maintenance for the ATC environment.

2.2 REASON FOR CHANGE FACTORS

In a 1976 paper [9], E. B. Swanson called attention to the variety of reasons for performing software maintenance, and to the different capabilities and activities that are involved in these. He classified the reasons for change into major categories, termed bases for maintenance. A summary of his classification scheme is presented in table 2 - 1.

TABLE 2 - 1
REASON FOR CHANGE FACTORS

A. Corrective Maintenance
    1. Processing Failure
    2. Performance Failure
    3. Implementation Failure

B. Adaptive Maintenance
    1. Change in Data Environment
    2. Change in Processing Environment

C. Perfective Maintenance
    1. Processing Inefficiency
    2. Performance Enhancement
    3. Maintainability

Processing failures are those which result in abnormal output; performance failures are those in which normal output is furnished but system requirements are not met. All other corrective maintenance actions are termed implementation failures, and these include specifically changes to comply with design or coding standards. Although specific data are not provided in the paper, it can be assumed that performance failures will usually require a large maintenance effort, processing failures a smaller one, and implementation failures the least. The structure within the Corrective Maintenance category is therefore meaningful in scoping both the magnitude of the maintenance effort and the techniques that can be brought to bear on it. Note also that only types A1 and A2 are covered by the definition of maintainability adopted in [8], and that a strict interpretation of the definition may even exclude A2.

The subclassifications under Adaptive Maintenance are self-explanatory. Here it is not possible to associate specific levels of maintenance effort with each entry (e. g., change in data environment may involve a single data item or an entire restructuring of the data base). However, the classes are significant in term of the applicable maintenance techniques.

Perfective maintenance in substituting a more efficient algorithm or in use of a more appropriate language construct falls into the classification of "Processing Inefficiency". The "Performance Enhancement" class includes changes in report formats (e. g. for better readability) or in providing additional output data. The last entry in the Perfective Maintenance base are changes that are made to improve the maintainability of the code and which presumably do not affect either the output or the processing proper. The classification here is particularly significant in identifying different requester groups: performance enhancements are typically requested by the user, changes in processing to remove inefficiencies may be requested

by the computer operations group, and maintainability changes by the software maintenance personnel. Again, attention to these differences will be helpful in pointing to appropriate maintainability techniques.

Swanson proposed this classification in order to analyze maintenance activities and to generate measures of maintenance performance. The classification is of interest also for assignment of proper tools and techniques and should therefore be regarded as a contributor to the maintainability factors that are being evaluated here. A vague attempt to incorporate the bases in a classification of maintenance activities has already been reported [10].

2.3 ACTIVITIES FACTORS.

In a recent paper, T. Gilb has described the activities required for software maintenance in considerable detail [11]. The resulting activities categories may be of value in analyzing the maintenance effort and are therefore evaluated here. The specific activities described in the reference are:

TABLE 2 - 2
ACTIVITIES FACTORS FROM [11]

1. Problem Occurrence
2. Problem Recognition
3. Administrative Delay
4. Maintenance Personnel Assigned
5. Collection of Maintenance Tools and Documentation
6. Analysis of Need
7. Evaluation of Alternative Corrections
8. Active Correction
9. Test
10. Side-effect Test
11. Independent Quality Inspection

To this list should be added documentation of the maintenance action and all steps necessary to introduce the changed program to the user (in the ATC environment this will involve administrative as well as technical activities).

Another list of activities is presented in [9], apparently at least in part derived from the classification discussed above. These activities have been related to three major functions as shown in table 2 - 3.

TABLE 2 - 3
FUNCTIONS/ACTIVITIES MATRIX FROM [10].

Activities	Affected Functions		
	Understanding	Modifying	Revalidating
Error Identification	x	x	
Requirements Analysis		x	
Redesign	x	x	x
Code Production	x	x	
Test & Integration			x
Documentation	x		x
Quality Assurance		x	x
Configuration Mangmt.	x	x	x

It is interesting that the functions identified here are closely related to the major quality factors (testability, understandability and modifiability) used in [5]. This matrix therefore represents a tentative step in associating activities with quality factors.

SECTION 3
EVALUATION

In this section the maintainability factors described in the preceding section are evaluated in the light of requirements for an advanced air traffic control system. The structure of this section follows that of the preceding one in discussing quality, reason for change, and activities factors in that order.

3.1 QUALITY FACTORS

All of the factors discussed in the preceding section are useful in analyzing the software maintenance process. Quality factors have the additional merit that they offer an explicit means of affecting the process in a beneficial manner. It is like not only talking about the weather but also doing something about it. They are therefore a prime candidate for inclusion in a Reliability/Maintainability Program for the ATC Computer Replacement Program.

The quality factors proposed in the TRW report [5] have been scrutinized both inside the company and by the software community at large, and in general they have been found valid. The major factors (testability, understandability, and modifiability) have found their way into widely accepted definitions, and as shown in the preceding section, have been used as a cross-reference in other factor studies. The factors identified in [5] cover maintenance initiated because of errors in the existing software as well as maintenance requested for other reasons. A weakness of the approach is that the metrics for the primitives are not uniformly precise, and that evaluation of programs against these factors is therefore necessarily subjective.

The quality factors used in the GE study [8] are all expressed as precise metrics, and, due to this, quantitative effects on maintenance effort can be more readily demonstrated.

This represents a desirable feature, but it also involves some limitations: because of the exclusively metric approach, some factors that are not easily quantified had to be omitted; and the metrics are so specific that they may become obsolete as new programming practices or computing equipment come into use. The quality factors used by GE are targeted at corrective maintenance. Factors that are significant primarily for adaptive or perfective maintenance are not covered. In this respect these metrics do not meet the requirements of the ATC Computer Replacement Program.

3.2 REASON FOR CHANGE FACTORS

The reason for change factors are by themselves primarily an analytic tool. Classification along these lines is obviously desirable to control the workload in a maintenance organization or when major procedural changes are being considered. Such decisions are not within the present scope of the ATC Computer Replacement Program. Nevertheless, the reason for change has to be considered in evaluating the importance of quality factors. Thus, qualities contributing to testablity will be particularly important when corrective maintenance is being performed while modifiability will be more important when adaptive or perfective maintenance is undertaken.

It is expected that the reasons for change will vary with the maturity of the principal software run on the ATC system. When new software is introduced, corrective maintenance will predominate, followed by a phase during which considerable perfective maintenance will be accomplished, and finally a phase during which the emphasis will shift to adaptive maintenance. For this reason, the weighting of quality factors may also change with the software life cycle. Reason for change factors are considered in this context in the recommendations discussed in the next section.

3.3 ACTIVITIES FACTORS

Activities classifications are also primarily of interest in the analysis of workloads and in organizational decision making. It is difficult to infer from either of the two activities classifications presented in 2.3 specific software qualities that will reduce maintenance requirements. Table 2 - 3 implies that a high rating with regard to the major quality factors will have a beneficial effect on the specific maintenance activities listed there. It is therefore not considered necessary to include an activities classification with the primary maintainability criteria that will be developed under this task.

Some insights into features that benefit maintainability can nevertheless be gained by considering the detail maintenance activities. E. g., that seven steps are necessary according to [11] before the active correction phase is entered suggests that aids to problem location and isolation will have a high payoff. For this reason, a modified form of the activities list presented in [11] is included in the recommendations as an aid to further work in the maintainability area. The activities factors in [11] were selected as a starting point because they are more detailed than those identified in [10].

SECTION 4
RECOMMENDATIONS

As indicated in the preceding section, the quality factors provide the most direct approach for assuring that software generated for the ATC Computer Replacement Program will be easy to maintain, and it is therefore recommended that these factors be used in the formulation of software requirements. Among the two formulations of quality factors, the one proposed by TRW [5] appears more seasoned and comprehensive. Deficiencies in quantitative metrics can be corrected by either adapting suitable metrics from [8] (the agreement at the primitive factors level permits this in many cases), and by generating new metrics. It is also noted that metrics are less significant for software procurement than for the evaluation of existing software. E. g., adherence to a hierarchical structure can be made a requirement, and it is then not necessary to have a metric for deviations from hierarchical structure.

Because there can be conflicts among requirements imposed by various quality factors, and because imposition of any requirement usually involves a cost, it is recommended to identify priorities among the quality factors. At present a binary classification (high/low priority) appears adequate. The priority rating is to some extent affected by the type of maintenance to be undertaken, and that characteristic can be described in terms of reason for change factors. Table 4 - 1 shows the priority assigned to the maintainability primitives from [5] under the major reason for change classifications developed in [9].

TABLE 4 - 1
MAINTAINABILITY FACTORS AND PRIORITY ASSIGNMENTS

Factor	Reason for Change		
	Correct.	Adaptive	Perfective
Consistency	x	x	
Accessibility		x	
Communicativeness		x	x
Structuredness	x	x	x
Self-descriptiveness	x		
Conciseness		x	
Legibility	x		
Augmentability		x	x

x = high priority, all others low priority

The factors identified here will be used to guide the effort for software maintenance in the continuation of task 10. In addition, it is recommended that this effort consider the specific activities listed in table 4 - 2 in the establishment of maintainability criteria for the ATC Computer Replacement Program software.

TABLE 4 - 2
ACTIVITIES FOR SOFTWARE MAINTENANCE

1. Problem Occurrence
2. Problem Recognition
3. Administrative Delay
4. Maintenance Personnel Assigned
5. Collection of Maintenance Tools and Documentation
6. Analysis of Need
7. Evaluation of Alternative Corrections
8. Active Correction
9. Test
10. Side-effect Test
11. Independent Quality Inspection
12. Documentation
13. Trial Use
14. Configuration Management

REFERENCES

[1] MIL-STD-721B "Military Standard, Definitions of Effectiveness Terms for Reliability, Maintainability, Human Factors and Safety", NAVAIR for Dept. of Defense, 25 Aug 1966 (w/Notice 1, 10 Mar 1970)

[2] Computer Dictionary (Draft Standard), IEEE Computer Society, 1979

[3] T. Gilb, Software Metrics,Winthrop Publishers, Cambridge MA, 1977

[4] S. A. Gloss-Soler, The DACS Glossary, A Bibliography of Software Engineering Terms, Data and Analysis Center for Software, Griffiss AFB NY, October 1979

[5] B. W. Boehm, J. R. Brown, H. Kaspar, M. Lipow, G. J. McLeod, and M. J. Merritt, Characteristics of Software Quality, TRW-SS-73-09, TRW Systems, Systems Engineering and Integration Division, Redondo Beach CA, December 1973

[6] B. W. Boehm, J. R. Brown, and M. Lipow, "Quantitative Evaluation of Software Quality", Proceedings of the Second International Conference on Software Engineering, San Francisco CA, Oct 1976, IEEE Cat No 76 CH1125-4C, pp. 592 - 605

[7] B. W. Boehm, "Software and its Impact: A Quantitative Assessment", Datamation, Vol 19 No 5, May 1973, pp.48 - 59

[8] J. A. McCall, P. K. Richards, and G. F. Walters, Factors in Software Quality, RADC-TR-77-369 (3 Vols.), November 1977

[9] E. B. Swanson, "The Dimensions of Maintenance", Proceedings of the Second International Conference on Software Engineering, San Francisco CA, Oct 1976, IEEE Cat No 76 CH1125-4C, pp. 492 - 497

[10] J. D. Donahoo and D. Swearinger, A Review of Software Maintenance Technology, RADC-TR-80-13, February 1980

[11] T. Gilb, "Controlling Maintainability: A Quantitative Approach to Software". Available from Data and Analysis Center for Software, Griffiss AFB NY 13441.

APPENDIX
DEFINITIONS OF MAINTAINABILITY FACTORS

The following are definitions of detailed maintainability factors, termed 'primitives' in [5] and 'criteria' in [8], as given in these respective references. The listing is alphabetical. Where definitions from both references are available, that considered more applicable to the ATC Computer Replacement Program is preceded by ***.

ACCESSIBILITY

A Software product possesses accessibility to the extent that it facilitates the selective use of its components [5].

AUGMENTABILITY

*** A software product possesses augmentability to the extent that it easily accomodates expansions in data storage requirements or component computational functions [5].

(EXPANDABILITY - not a criterion for maintainability)
Those attributes of software that provide for expansion of data storage requirements or computational functions [8].

COMMUNICATIVENESS

*** A software product possesses communicativeness to the extent that it facilitates the specification of inputs and outputs whose form and content are easy to assimilate and useful [5].

(not a criterion for maintainability) Those attributes of software that provide useful inputs and outputs that can be assimilated [8].

CONCISENESS

*** A software product possesses conciseness to the extent that no excessive information is present [5].

Those attributes of software that provide for implementation of a function with a minimum amount of code [8].

CONSISTENCY

A Software product possesses internal consistency to the extent that it contains uniform notation, terminology and symbology within itself. The product possesses external consistency to the extent that the content is traceable to the requirements [5].

*** The attributes of software that provide uniform design and implementation techniques and notation. [8].

LEGIBILITY

A software product possesses legibility to the extent that its function and those of its component statements are easily discerned by reading the code [5].

SIMPLICITY

Those attributes of software that provide implementation of functions in the most understandable manner. (Usually the avoidance of practices that increase complexity) [8].

SELF-DESCRIPTIVENESS

*** A software product possesses self-descriptiveness to the extent that it contains ehough information for a reader to determine its objectives, assumptions, constraints, inputs,

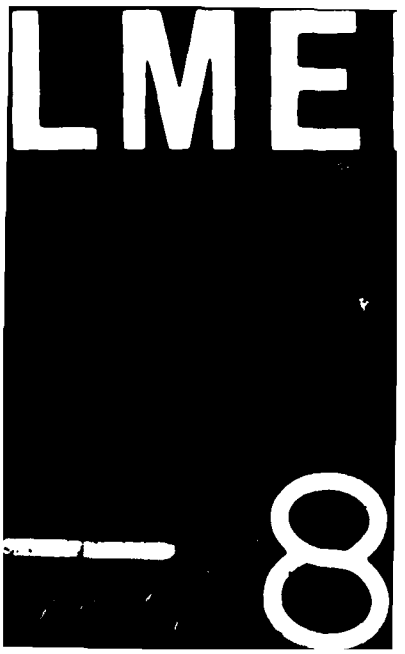
outputs, components, and status [5].

Those attributes of software that provide explanation of the implementation of a function [8]

STRUCTUREDNESS

*** A software product possesses structuredness to the extent that it possesses a definite pattern of organization of its interdependent parts [5].

(MODULARITY) Those attributes of software that provide a structure of highly independent modules [8].


LME
8