(12) 52

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AD A046962

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER (14) TOP-1-1-056 ✓	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) US ARMY TEST AND EVALUATION COMMAND TEST OPERATIONS PROCEDURE (6) SOFTWARE TESTING.		5. TYPE OF REPORT & PERIOD COVERED (9) Final rept.
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s)		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS US Army Test and Evaluation Command ✓ (DRSTE-SY) Aberdeen Proving Ground, MD 21005		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS DARCOM-R 310-6
11. CONTROLLING OFFICE NAME AND ADDRESS US. Army Test and Evaluation Command (DRSTE-ME) Aberdeen Proving Ground, MD 21005		12. REPORT DATE (11) 15 Nov 77
		13. NUMBER OF PAGES 40
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) (12) 41p.		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Software Testing	Software Investigation Report (SIR)
Verification and Validation (V&V)	Hardware Monitor
Algorithm Testing	Software Monitor
Software Documentation	Timeline Analysis
Simulation Validation	Software Quality Factors

20. ABSTRACT (Continue on reverse side if necessary and identify by block number) THIS REPORT Describes 12 objectives and generalized procedures for system level testing of software in tactical-embedded-computer systems at TECOM field activities. Emphasizes the "early" areas of coordination with the developer to enable proper and complete test design, execution, and evaluation.

041 750

DDC FILE COPY

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

LB

US ARMY TEST AND EVALUATION COMMAND
TEST OPERATIONS PROCEDURE

DRSTE-RP-702-100
Test Operations Procedure 1-1-056 15 November 1977
AD No.

SOFTWARE TESTING

			Page
Paragraph	1.	Scope.	3
	2.	Objectives and Procedures.	3
	2.1	Specification Development.	3
	2.1.1	Objective.	3
	2.1.2	Procedure.	3
	2.1.2.1	Preventive Measures.	3
	2.1.2.2	Corrective Measures.	4
	2.2	Algorithm Testing.	4
	2.2.1	Objective.	4
	2.2.2	Procedure.	4
	2.2.2.1	Definition of Problem.	4
	2.2.2.2	Basic Algorithmic Adequacy	4
	2.2.2.3	Algorithm Computerization.	5
	2.2.2.4	Trade-Offs	5
	2.3	Computer Program Documentation	5
	2.3.1	Objective.	5
	2.3.2	Procedure.	5
	2.3.2.1	Data Items	5
	2.3.2.2	Review of Documents.	5
	2.4	Acquiring V&V Data	6
	2.4.1	Objective.	6
	2.4.2	Procedure.	6
	2.4.2.1	Data Collection.	6
	2.4.2.2	Analysis	6
	2.5	Simulation Validation.	6
	2.5.1	Objective.	6
	2.5.2	Procedure.	6
	2.5.2.1	Use of Simulation.	6
	2.5.2.2	Validation of Stochastic Simulations	6
	2.5.2.3	Problems in Simulation Validation for Non-Stochastic Types.	7
	2.5.2.4	How Much Testing	7
	2.5.2.5	What Tests are Necessary	7
	2.5.2.6	Measures of Validity	7
	2.5.2.7	TECOM Responsibility	7
	2.6	Software/Computer Failures	8
	2.6.1	Objectives	8
	2.6.2	Procedures	8

Page

2.6.2.1 Definition of Problem. 8
2.6.2.2 Failures During System Testing 8
2.6.2.3 Testing for Failures 9
2.6.2.3.1 Pursuing Known Problem Areas 9
2.6.2.3.2 Saturation Problems. 9
2.6.2.4 Failure Testing Resources. 9
2.6.2.4.1 Knowledge of Process 9
2.6.2.4.2 System Driving and Stimulation 9
2.6.2.4.3 Hardware Test Instrumentation. 10
2.6.2.4.4 Software Test Instrumentation. 10
2.7 Performance Bounds/Excess Capacity 11
2.7.1 Objective. 11
2.7.2 Procedure. 11
2.7.2.1 Test Resources Required. 11
2.7.2.2 Models and Simulations 12
2.7.2.3 System Driver or Stimulator. 12
2.7.2.4 Instrumentation Requirements 12
2.8 Specification Compliance 12
2.8.1 Objective. 12
2.8.2 Procedure. 12
2.8.2.1 Traceability 12
2.8.2.2 TECOM Responsibility 13
2.9 Short- and Long-Term Retesting 13
2.9.1 Objective. 13
2.9.2 Procedure. 13
2.9.2.1 Definition of Problem. 13
2.9.2.2 Short-Term Retests 13
2.9.2.3 Long-Term Retests. 14
2.10 Operating System Testing 14
2.10.1 Objective. 14
2.10.2 Procedure. 14
2.10.2.1 Operating System Overview. 14
2.10.2.2 TECOM Responsibilities 14
2.11 Timing Analysis and Testing. 15
2.11.1 Objective. 15
2.11.2 Procedure. 15
2.11.2.1 Time Line for System Functioning 15
2.11.2.2 Time Line for Software Functioning 15
2.11.2.3 TECOM Responsibility 15
2.12 Software Investigation Reports 17
2.12.1 Objective. 17
2.12.2 Procedure. 17
2.12.2.1 Purpose and TECOM Responsibilities 17
2.12.2.2 SIR Responsibilities 17
2.12.2.3 Monthly Reports. 19

		Page
2.13	Software Quality	19
2.13.1	Objective	19
2.13.2	Purpose	19
2.13.2.1	Measures of Quality	19
2.13.2.2	TECOM Responsibilities	32

1. SCOPE. This TOP describes the objectives and procedures involved in TECOM software/computer testing for tactical systems containing computers. This TOP is in consonance with AR 70-XX(Draft); Management of Computer Resources in Army Defense Systems; DARCOM Software Test and Evaluation Guidelines (Draft); and TECOM Technical Report SY-2-77, Feb 77, Testing Subcommittee Input Report to Study: Software Testing Policies and Procedures. The objectives contained in this TOP will be pursued for all tactical systems containing computers for which TECOM has test and/or evaluation responsibility. The application of this TOP will be specified in the Test Design Plan (TDP). It may be that not all of the subjects will apply to each project owing either to the nature of the system or the inability (or unwillingness) of the Project Manager to support the required analyses and/or tests. In the latter case the TECOM field test activity will prepare a risk statement to be submitted to the Project Manager through the appropriate element(s) within TECOM Headquarters. It is further not necessary that the TECOM field test activity perform all of the testing and analyses implied in the pursuit of the objectives. It is, however, required that those tests and analyses not performed by TECOM be accounted for by the field activity responsible for the system testing. The TECOM field activity will provide a member to the Computer Resource Work Group (CRWG) for the project. (Ref. AR 70-XX) If a CRWG is not formed for the project, the TIWG (Test Integration Working Group) will be used as the forum for software test matters.

2. OBJECTIVES AND PROCEDURES.

2.1 Specification Development.

2.1.1 Objective: To participate in system and software specification(s) development and planning to the end that those specifications are traceable, logical, complete, and sufficient and that they will produce testable software.

2.1.2 Procedure.

2.1.2.1 Preventive Measures: This objective can be attained through any number of ways. The preferred way is that the (MIL-STD-490) A/B5/C5 level specs be drafted using a method or "package" such as those designated on page A-4 of TECOM Tech Rpt, SY-2-77. These methods assure completeness, logic, traceability, and sufficiency of the specifications. To the degree that these methods provide a structure to support fault isolation, they aid in testability. Other facets of testability such as instrumentation, simulation, stimulation, and test case generation will be discussed in other objectives.

2.1.2.2 Corrective Measures: If the specifications exist, but were not written using one of the methods or "packages" described in TECOM TR, SY-2-77, they should be reviewed in detail, using either one of these methods or "packages" or a method similar to that provided in ARMTE Study, Test Methodology Study Final Report, dtd Aug 76. This is usually done by the Verification and Validation (V&V) contractor. If such is the case, the TECOM field activity will monitor the action. If the PM has not planned for this specification verification, the TECOM field activity should offer to provide this service using the CRWG and the TIWG as vehicles for coordination.

2.2 Algorithm Testing.

2.2.1 Objective: To acquire data in support of the evaluation of algorithm accuracy, efficiency, timing, and computer resource utilization.

2.2.2 Procedure.

2.2.2.1 Definition of the Problem: The design of embedded-computer systems, especially those that are time-critical in their functioning (real-time), involves the selection of several critical computer resources which will, together, produce the proper data transformations in a timely fashion. The transformations often include mathematical equations, sorting, statistical computation, and the like. Such transformation logic comprises the algorithm(s) set. Algorithms in digital computers can often be thought of as 2-stage entities: (1) the continuous or algebraic statement(s) and (2) the way the equation is digitally computed (numerical analysis equivalent). The designer more often than not has less choices in the first stage than in the second. The first four volumes of Donald E. Knuth's The Art of Computer Programming and Abramowitz and Segun's Handbook of Mathematical Functions illustrate this point completely. The way a data transformation is physically done in the computer is usually a trade-off among speed of computation, accuracy, and data storage and handling requirements.

2.2.2.2 Basic Algorithmic Adequacy: The CRWG representative will review and, if possible, participate in the selection of algebraic equations to be used. Here, the adequacy of the equations to satisfy user needs in terms of accuracy and completeness will be determined. A good example here is the selection of equations used to compute ballistic trajectories. The choice is usually between the three degree of freedom (modified) method and the six degree of freedom method. The six degree of freedom method provides the most accurate answers but requires the most computer resources.

2.2.2.3 Algorithm Computerization: The next consideration is the numerical analysis equivalent equations to be used. Here, the direct relation among computing speed, computer resource requirements, and accuracy become more readily apparent. This is especially true in the numerical analysis equivalent for integration or for transcendental function approximation. In these cases, accuracy obtained is often a function of the number of terms used in a series expansion or the incremental "fineness" used, which, of course, is a trade-off between accuracy and speed of computation.

2.2.2.4 Trade-offs: These trade-offs should be viewed and exercised through simulation and models. The accuracy and timing of system functions should be the subject of functional simulation models. The first stage algorithmic accuracy necessary to obtain the required system performance should be the subject of engineering models. The computerization of these equations should be the subject of a computer system simulation. The TECOM representative(s) on the CRWG should insist on this lineage of simulation/models or their documented logical equivalent being used during the entire development process and delivered with the system. These simulations/models should be retained, updated, and validated as developmental increments and parameters are established. In cases where the simulation/model parameters are not directly obtainable from contractor tests, the V&V tester should design and perform such tests. Failing this, TECOM should propose that we do such testing. Should the PM not support such testing, the TECOM field activity will prepare a statement of risk to the Project Manager through the appropriate TECOM Headquarters element.

2.3 Computer Program Documentation.

2.3.1 Objective: To determine the adequacy of the computer program documentation.

2.3.2 Procedure.

2.3.2.1 Data Items: The TECOM field activity should familiarize itself with the published Data Item Descriptions (DID's) pertaining to computer resources as found in DOD 5000.19.L, Vol II, dtd Jan 77 (w/supplement). In reviewing the Request For Proposal (RFP), special cognizance will be given to the Contract Data Requirements List (CDRL) to assure that all the documents and data that will be needed to test and maintain the software by the government are required to be delivered by the contractor prior to government testing.

2.3.2.2 Review of Documents: When delivered, the documentation will be analyzed to reveal (if any) errors, omissions, inconsistencies between documents (or between document and code), and improper configuration control identification. All problems found will be reported through the CRWG to the PM for action.

2.4 Acquiring V&V Data.

2.4.1 Objective: To acquire data obtained during independent verification and validation (V&V) actions required to accomplish TECOM software T&E tasks.

2.4.2 Procedure.

2.4.2.1 Data Collection: Much of the data to be obtained by the TECOM field activity for test design and/or evaluation purposes will come from the activities normally provided by an Independent Software Tester (IST) or Verification and Validation (V&V) Tester. This does not mean, however, that TECOM can give the V&V tester a stack of blank forms during the concept phase and have them delivered, filled in, at DT II. In order for TECOM to be responsive to software testing throughout the development process, it must closely monitor the activities of the V&V tester.

2.4.2.2 Analysis: Appropriate V&V reports will be analyzed as to their implication to TECOM test planning and activities. Special notice will be made of developmental software problem areas and areas where contractor and/or V&V testing is either inadequate or yields marginal results. Such areas will form the basis for generation of system test cases and data collection plans.

2.5 Simulation Validation.

2.5.1 Objective: To acquire data which will correlate the behavior of the system under test and its associated simulations.

2.5.2 Procedure.

2.5.2.1 Use of Simulation: Paragraph 2.2.2.4 mentions one use of simulation in computer software test and evaluation. There are others. Not the least of these is the requirement to use simulation in lieu of actual testing for extremely difficult, costly, unsafe, and/or unrepeatable test scenarios. This, however, implies that the simulation(s) must be validated to establish confidence in their use.

2.5.2.2 Validation of Stochastic Simulations: There exists a wealth of statistical theory on the comparing of data sets resulting from random processes. When simulating a statistically distributed process, the results can be compared to the results from testing that process in order to determine the degree or level of confidence that the simulated and test data sets come from the same parent: i.g., that the simulation truly represents the real thing. The methodology involved is contained in TECOM TOP 5-1-030, Simulation (Draft).

2.5.2.3 Problems in Simulation Validation for Non-Stochastic Types: Little has been written on a scientific approach to validation for non-stochastic simulations. It appears to be an area requiring a liberal application of common sense. The general idea is to generate benchmark tests for both system and simulation and make some decision based on the comparison of the results. Several questions immediately arise:

(a) How many tests should be run?

(b) How should the input loading be varied?

(c) What are the measures of validity or fidelity of the simulation?

2.5.2.4 How Much Testing: The way in which the first question is answered is limited by the criticality of the decision which must be based on the simulation and the amount of money and time available. Obviously no simulation can account for all of the exogenous variables affecting actual system performance. However, if some (say) environmental variable causes significant changes in output from standard ambient conditions, the nature and degree of these changes must be ascertained through test and the simulation adjusted accordingly. If the system displays item-to-item or day-to-day variation in output for the same input and that variation appears to be caused by a host of things not directly correlatable, the extent and nature of that variation must be ascertained by test so that Monte Carlo techniques can be applied to the simulation. The simulation then migrates from deterministic to stochastic and the methodologies of 2.5.2.2 apply. The number of tests run in any case is finally dependent upon "how good" the simulation appears to be in the eyes of the decision maker.

2.5.2.5 What Tests are Necessary: If the input-to-output transformation appears to be linear (or nearly so), generally all that is needed is "low-medium-high" input test cases to demonstrate correlation between simulation and system. However, if at some point the input displays high leverage over the output, the test cases should "cluster" around that area.

2.5.2.6 Measures of Validity: The measures of validity or fidelity usually take the form of $\pm$ % difference at various input levels. This is usually displayed graphically. For extrapolation purposes, it is important that good agreement exists between simulation and system from input midpoint (test) to input high point (test).

2.5.2.7 TECOM Responsibility: The TECOM field activity will review the plans (if any) for simulation validation in early stages of development. A system level test program designed to validate system simulations and computer resource/timing simulations will be prepared and submitted to the PM for funding and scheduling. Subsystem level tests required for simulation validation which are required but not planned will be requested by TECOM through the CRWG.

2.6 Software/Computer Failures.

2.6.1 Objective: To systematically detect and analyze software and computer related failures in system tests.

2.6.2 Procedure:

2.6.2.1 Definition of Problem: Software and the computer in which it resides are components of a larger system. In most real-time tactical systems the software receives data from input sensors (or man), transposes that data (algorithms), makes decisions based on that transformation, and directs proper and timely action to be taken. Unlike software, the computer hardware is comprised of an electronic device(s) with electromechanical peripheral devices which are all subject to physical failure... overheating, physical breakage, etc. Failures in hardware are usually easily identified because of failure signatures such as heat, discoloration, physical breakage, electrical discontinuity, etc. Software failures, on the other hand, leave no signatures. They take several forms:

(a) The software does not respond to an input.

(b) The software responds incorrectly to an input:

(1) Improperly timed response.

(2) Numerically wrong response.

(3) Response requiring more resources than are available.

2.6.2.2 Failures During System Testing: In complex software development efforts, for real-time embedded computer systems, the device on which the software resides and is tested can vary throughout the RDT&E process. First, there is the "instruction level simulator." This is usually a software program itself, operating on a large-scale general purpose "host" computer. It "acts like" the target system in that it receives simulated inputs in non-real-time, exercises the algorithms, communicates with non-real-time simulations of the peripherals, and provides outputs. The developmental software, which is written in a Higher Order Language (HOL) is compiled and assembled in accordance with the host computer's architecture, operating system and basic instruction set; i.e., to run on that computer. Its prime purpose is to see that the software statically and sequentually does what it is intended to do. However, much is often claimed for these test beds than is due them. The general purpose host computer's architecture, basic instruction set, and operating system are seldom (if ever) the same as that of the target computer. The inputs are usually assumed, as are the peripheral simulations. The next level of test bed is the "emulator." Here, the basic instruction set of the host computer is microprogrammable and the architecture is flexible so that the execution of code, including the target operating system, is procedurally the same as on the target system. However, the speed of the emulator may be different from the target computer. Here, real-time peripheral simulations or the actual peripherals themselves are included. The inputs are provided by a real-time driver (discussed later). The final level of test bed is the target system itself. It is at this level that the TECOM field activities do most of their testing. Each level calibrates or validates the previous level. So, aside from the purpose of showing specification compliance, TECOM testing must be another rung on the ladder toward a highly validated development and test process that will be purchased for the purposes of software maintenance and post deployment development and changes. Failures observed at the system level, as with failures found elsewhere in the development process, will initiate Software Investigation Reports (SIR). (See Section 2.12)

2.6.2.3 Testing for Failures.

2.6.2.3.1 Pursuing Known Problem Areas: Since the system level testing is merely another step in the RDT&E process for software, it is reasonable that emphasis be placed here on areas of the code which are marginally designed and/or tested at lower levels. Scenarios will be devised by the TECOM field activity to exercise functional areas and modules (which displayed high failure rates) through their entire design range.

2.6.2.3.2 Saturation Problems: Another aspect of development often lightly considered is the down-stream efficacy of the original computer sizing exercise. Here, the computer system simulation is used to determine scenarios which would cause saturation of various computer resources. If the scenarios are reasonably realistic, they should be used to determine the envelope of normal system functioning allowed by the computer. If these bounds can easily be reached, the software should have in its design degradation modes of operation. These would normally be involved at resource saturation. Test scenarios will be designed to both invoke and exercise the "degraded mode" software.

2.6.2.4 Failure Testing Resources.

2.6.2.4.1 Knowledge of Process: Complete understanding of the process that transpires in the computer hardware and software as well as other system components is an absolute necessity if TECOM is to be able to locate and analyze failures in embedded computer systems. Such understanding only comes through diligent monitoring of the development process, aided by an understanding and use of the various simulation and models which should be available. Complete, up-to-date documentation which reflects a strong configuration management program is also required.

2.6.2.4.2 System Driving and Stimulation: The prime reasons for utilizing a computer to control tactical systems are the computer's speed and capacity for information handling. This naturally increases the size and complexity of the scenario in which the tactical system can satisfactorily operate. The input to the computer is usually an interpretation of system experiences through its "eyes" or sensors (including man); the output of the computer is manifested through dynamic operation of weapons, display units, etc. As the complexity and size of test scenarios increase, the ability to repeatedly provide physical threat stimuli (test scenarios) and record (say) weapon dynamics (system response) becomes increasingly difficult. In many cases, such physical testing is impossible. Here, a system driver (stimulator, exerciser) is required. However, such a device will be quite sophisticated itself, often as sophisticated as the system being exercised. It therefore must be initiated for parallel development along with the target system itself. It is for this reason that the TECOM field activity must review the planned utilization of the target system very early and determine the requirements for a driver against TECOM's abilities to provide repeatable tests of those magnitudes in the field. This must be done sufficiently early in the development cycle to provide required lead time. The TECOM field activity will either actually develop the driver requirements (if any), or will actively participate with the Project Manager in their development. All other TECOM responsibilities pertaining to the driver are the same as for the target system itself.

2.6.2.4.3 Hardware Test Instrumentation: If system failures are to be analyzed as to their causes, the capability must exist for testing the hardware separate of the software. Some systems, such as TACFIRE, use a Computer Test Set (CTS) to test every logic and memory circuit in the computer electrically. The CTS is a separate piece of equipment. Another method is to use a software program which resides in the operating system and tests hardware functioning as part of the planned process or during processor "idle" time. Such a program is usually also invocable as a "utility" when the system is in an "off-line" state. The TECOM field activity should insist that such a capability be made available and deliverable with the system.

2.6.2.4.4 Software Test Instrumentation: Aside from the normal TECOM field capability and instrumentation to measure system dynamics, a capability is required to measure computer output and software activity during system testing. Computer output recording is often designed into the system driver (discussed above). This usually takes the form of a real-time recorder which records the input and output simultaneously, keyed to a common clock, in order to obtain system response timing. Within the computer at (say) the software module level, special instrumentation is required to measure such things as activity rates, timing, resources used, queues, and inter-module input/output. This can generally be done in two ways....a software monitor and/or a hardware monitor. These are described below:

(a) Hardware Monitor: A hardware monitor is an electronic device which may be wired to a computing system (including peripherals) to capture and record electrical signals created within the system during operation. It is probably the only universal piece of equipment that is available for software testing. A versatile and portable model should be available to or owned by the TECOM field activity. In general purpose computer applications, the wiring operation is done with electrically non-inductive probes which are attached to appropriate terminal points, registers, etc. However, because of ruggedization and packaging, this may not be possible in the tactical computer case. Therefore, the TECOM field activity should insist that (at least) probe points be accessible. Even better would be the creation of a test item where the probe connections were brought to say a multi-pronged plug on the system's outer surface. Since this is a design consideration, whatever influence to be brought to bear must be done in early design and specification. The measurements, which are comprised of timing and pulse counts, are recorded simultaneously from locations within the computer which would be exercised by the code in question. This means that a computer system simulation (or the like) must be used to first determine the resources and characteristics to be measured. Then, a probe point library and/or a logic wiring diagram is needed to locate the signals to be monitored. By analyzing the resultant non-volatile recordings of data, such things as average execution times, high speed buffer hits, resource utilization, and operating system efficiency can be determined.

(b) Software Monitor: The software monitor is a computer program that is incorporated into the system software which causes certain data related to the on-going processing to be recorded on disk or tape. The types of data typically recorded this way are module processing sequence, module start/stop times, executive intervention, missed time windows/deadlines, and processor idle time. Since this type of monitor is part of the development software package itself, it utilizes computer resources and time in competition with the tactical portion. It, therefore, is difficult to completely remove it when the system is fielded. This means that it should be a design consideration from the start. This, of course, implies that TECOM field activity influence must be brought to bear early in the R&D cycle.

2.7 Performance Bounds/Excess Capacity.

2.7.1 Objective: To determine the performance bounds on the system which is "allowed" by the software and to determine excess resources above and beyond the maximum utilization.

2.7.2 Procedure.

2.7.2.1 Test Resources Required: Three basic ingredients are required to determine the maximum demand placed on computer resources by the full range of tactical scenarios...test case generation, repeatable stimulation, software instrumentation. These things will be discussed in the subsequent paragraphs for system level testing.

2.7.2.2 Models and Simulations: As with analyzing failures, the most important ingredient is the total understanding of the system and its intended uses. This "understanding" will probably take the form of simulations, models, and documentation. Functional simulations of the utilization of the system will determine the range and combinations of inputs to be sensed by the system. Engineering models will determine the transformation of these sensed system inputs into a bit-stream at the pre-processor input port. The computer system simulation then predicts the dynamic utilization of computer resources during the scenario time period. Sensitivity analyses will then be performed on this sequence of simulations to define a set of system scenarios that will demand computer resources at or near saturation. Where saturation is not reached with maximum demand, the amount of resource used (or conversely, existing resource in excess of peak demand) will be determined. It will often be found that resource saturation occurs long before the scenario intensity reaches its peak. When this is the case, at a minimum, the software must not allow the system to "crash." The documentation should state required system reaction in case of resource saturation. If testing and analysis to determine this set of scenarios, this portion of input space that envelops normal system performance, is not planned by the PM, V&V tester, or contractor, the TECOM field activity will request to use the simulations for this purpose. A set of viable test cases will be determined for system level testing.

2.7.2.3 System Driver or Stimulator: At the system testing level, test input repeatability is quite difficult for large complex systems. A driver or exerciser is required, as was discussed in 2.6.2.4.2. This driver might be designed to stimulate the entire system through its sensors. However, it is usually designed to interpose at a point between the sensors and the input port to the pre-processor. It is important that the inputs generated by the driver can be easily modified or adjusted to allow for validation tolerances in the simulations.

2.7.2.4 Instrumentation Requirements: Given that the set of scenarios that demand 100% of various computer resources can be generated by a system driver, software instrumentation will be required to pinpoint the cause of any problems that may exist here. This instrumentation is described in 2.6.2.4.4.

2.8 Specification Compliance.

2.8.1 Objective: To assess the software aspects of system compliance (or lack of compliance) to A level specifications.

2.8.2 Procedure.

2.8.2.1 Traceability: One of the sub-objectives of specification development is traceability, not only within the A-B-C specification hierachy, but between the A level spec and the ROC. The techniques described in the ARMTE methodology investigation report, Validation of TECOM/ARMTE Software Methodology, Aug 76, provide a break out or mapping of B-5 software performance specs onto the A level system functions which are controlled or influenced by software. The TECOM field activity will assure that this mapping is physically done.

2.8.2.2 TECOM Responsibilities: This mapping essentially is the backbone for software test requirements at all levels. At the module, module integration, and software system levels, TECOM's role is primarily advisory. However, owing to the expense of software testing at the system level, the principles of SIDTC must be followed wherever possible. The system Detailed Test Plan will demonstrate all software requirements not sufficiently covered in lower level tests. When capability short-falls occur, the TECOM field activity will suggest tests which will help determine the cause of the difference(s) between specified and actual system performance. The TECOM field activity will suggest parameters and locations within the software (down to the module level when possible) which are likely areas for redesign or software modification considerations. The TECOM field activity will initiate a SIR in such cases and offer any and all assistance to the PM in resolving the problem within regulatory and fiscal constraints.

2.9 Short- and Long-Term Retesting.

2.9.1 Objective: To determine specific system level tests which should be rerun to demonstrate the effects of software "fixes."

2.9.2 Procedure.

2.9.2.1 Definition of Problem: Two general types of retests are considered here. First, there are retests which are short term in nature; i.e., immediately after a software "patch" is made. The second type of retest often involves long time periods and many software "patches" between original test and retest. These two types of retest are discussed in subsequent paragraphs.

2.9.2.2 Short-Term Retests: Software patches can range from single parameter changes all the way to functional area redesign. They generally involve the changing of 10 or less instructions, all within the same module. These changes usually precipitate changes in other modules working with or dependent on the original instructions changed; and so on, until no additional changes are needed. The nature and extent of the change to a module must be weighed against the way and extent that the module is used by the system to determine if any system level adverse effects have resulted.

Here, the TECOM field activity must draw heavily on its knowledge of how the system utilizes the software in performing its functions. System functions that are determined to be most affected by the changes will be proposed for immediate retest. The retesting will be proposed for immediate retest. The retesting will be designed to fully exercise the changed code.

2.9.2.3 Long-Term Retests: Long-term retests are equivalent to benchmark tests insofar as they ascertain the degree to which the system is performing its specified functions. If the software changes have been few and analysis shows that the cumulative effects are small, the retest costs may not be justified. However, if the software changes have been extensive, it is very risky to rely solely on any kind of analysis to predict system performance change, regardless of how comprehensive the configuration control is. The long-range retests should be designed to fully exercise all affected functional areas within the software. The test case generation will follow the same logic as in the resource saturation case (2.7.2.2). Complete resource utilization monitoring and tracing should be done during these tests. The results will serve as standards for assessing the impact of future software maintenance actions.

2.10 Operating System Testing.

2.10.1 Objective: To measure the performance, timing, and computer resource utilization of the Operating System (OS) functions.

2.10.2 Procedure.

2.10.2.1 Operating System Overview: The operating system is the "doer" in computer software systems. It is a software package that usually resides in core (although, segments of it may be swapped in and out of core) and essentially provides services and controls the applications programs. Among its many functions are dispatching and scheduling of tasks; allocating and freeing of memory; assigning, scheduling, and returning for service of peripheral devices; recovering from errors; process synchronization; timing of tasks, alarms, and time of day; creating, accessing, and purging of files; performance measurement; acquiring, scheduling, breaking down, and accounting for jobs; managing of shared resources.

2.10.2.2 TECOM Responsibilities: The TECOM field activity will ascertain that each function that the OS is designed to perform is tested and validated with regard to quality of performance, timing, and computer resources used in the performance of the function. This is usually done by the contractor. However, it must be done on the actual system hardware. Since such information is needed for system timing and computer system simulation, its completion will be sought as early as possible. If the contractor (or someone else) has no plans for doing this testing, the TECOM field activity will propose that it be done through the CRWG. If TECOM is to design and perform the tests, the following will be done:

(1) Devise simple scenarios that will invoke each OS function, one scenario per function.

(2) Establish driver programs to input these scenarios.

(3) Create a hardware monitoring plan for each scenario run which will dynamically record the utilization of computer resources.

(4) Perform tests and reduce monitor records.

2.11 Timing Analysis and Testing.

2.11.1 Objective: To determine the actual relationship between the computer and system time lines.

2.11.2 Procedure.

2.11.2.1 Time Line for System Functioning: At some point during B level spec writing, the developer should, especially for real-time systems, determine exactly what is to be done when, relative to normal functioning. This should result in a study or paper which defines the software problem in terms of run times, time windows, queue tolerance, etc. Figure 1 is a pictorial representation of the time line for a basic air defense (AD) mission. Notice that the labels describe what the system must do in functional terms. It essentially describes the timing relationships among the functional areas within the software. The criticality of system timing must be determined by simulation.

2.11.2.2 Time Line for Software Functioning: From the system time line analysis, the software developer designs a module level time line which essentially becomes the basis for the computer system simulation and, of course, the backbone of software design decisions. Software time marks are usually met since this is essential for the system to initially operate. However, if the computer is directing some mechanical equipment through a sequence of operations, and that mechanical equipment is subject to slow down due to wear, age, and environment, the software time line (and subsequently the software) must allow time windows of sufficient width to accommodate the largest cycle of the operational sequence. Otherwise, a failure has been "designed into" the system.

2.11.2.3 TECOM Responsibility: The TECOM field activity will assure that a complete timing analysis has been done for both system and software in real-time systems. Further, all critical time marks and windows should be validated by test. Such testing should be performed on the target system.

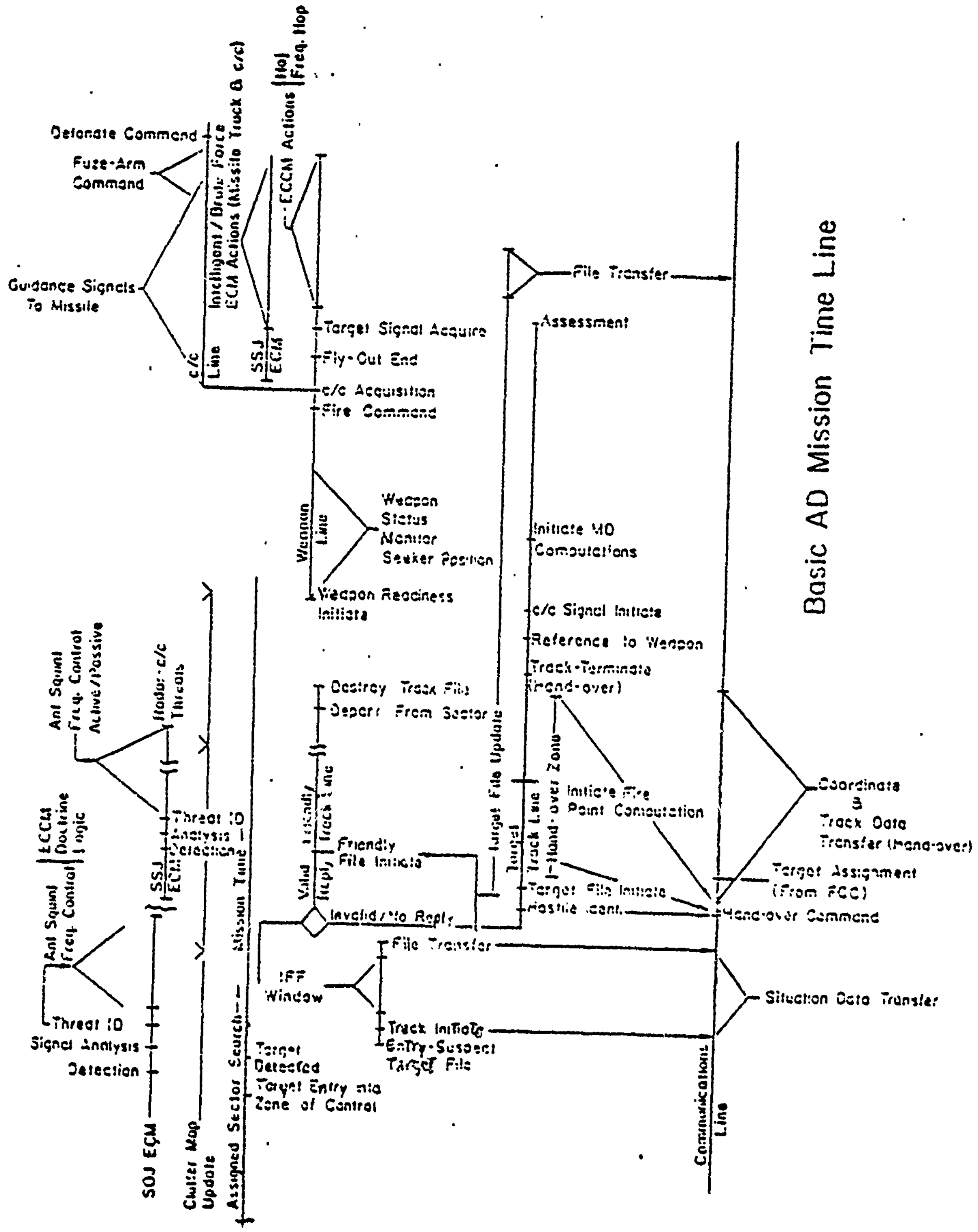
Basic AD Mission Time Line
SOJ ECM
Detection
Signal Analysis
Threat ID
Ant Squnt Freq. Control
ECCM Doctrine Logic
Ant Squnt Freq. Control Active/Passive
SSJ ECM
Radar c/c Threats
Threat ID Analysis Detection
Clutter Map Update
Assigned Sector Search
Mission Time
Target Detected
Target Entry into Zone of Control
IFF Window
File Transfer
Track Initiate Entry-Suspect Target File
Valid Friendly Reply
Friendly Track Line
Invalid/No Reply
Friendly File Initiate
Depart From Sector
Destroy Track File
Weapon Line
Weapon Readiness Initiate
Weapon Status Monitor Seeker Position
Fire Command
c/c Acquisition
Fly-Out End
Target Signal Acquire
c/c Line
SSJ ECM
Intelligent/Brute Force ECM Actions (Missile Track & c/c)
ECCM Actions
(Ho) Freq. Hop
Guidance Signals To Missile
Fuze-Arm Command
Detonate Command
Target Track Line
Target File Update
Hostile Ident.
Target File Initiate
Hand-over Zone
Initiate Fire Point Computation
Track-Terminate (Hand-over)
Reference to Weapon
c/c Signal Initiate
Initiate MO Computations
Assessment
File Transfer
Communications Line
Situation Data Transfer
Hand-over Command
Target Assignment (From FCC)
Coordinate & Track Data Transfer (Hand-over)

Fig. 1

Test cases which exercise the system through time marks and windows of interest should be used. Measurements should be made using a hardware monitor where possible. The hardware monitor is the only direct software performance device that does not utilize host computer time and resources in making the timing measurement. However, if integral software monitoring already exists in the system that will perform the timing measurements, and all time can be accounted for, including software monitor time, the use of hardware monitor may not be necessary.

2.12 Software Investigation Reports (SIR's).

2.12.1 Objective: To assure the establishment of an information reporting system on the events and eradications of software problems.

2.12.2 Procedure.

2.12.2.1 Purpose and TECOM Responsibilities: The following paragraphs describe a suggested procedure for reporting software problems and their eradication throughout the development cycle. The information contained in the SIR's, along with the monthly information and participative evaluation of development status and software quality. The TECOM test activity will assure that the SIR's and reports are provided to the evaluator in a timely manner. Although it is not necessary that the exact suggested system be implemented, the data content is essential for sound project management and evaluation.

2.12.2.2 SIR Responsibilities: The Materiel Developer will institute a Software Investigation Reporting Procedure that will collect, as a minimum, the information contained in (Fig. 2). The Materiel Developer will establish a point of contact within his organization (either the Product Assurance Office or Independent Verification and Validation Staff) that is responsible for implementing the software investigation procedures. The Materiel Developer will require that the developing contractor and all government testing agencies participate in SIR collection. SIR's will be collected during:

(1) Informal Contractor Testing: During software development the contractor will perform module integration test prior to formal Preliminary Qualification Test. SIR's will be collected by contractor testing groups who will complete the section describing the problem. The Contractor's programming section will complete the initial analysis section of the SIR and forward the SIR to the Materiel Developer point of contact. The Materiel Developer will assign a unique number to this SIR and enter the information in a SIR Data Base. The SIR will be returned to the Contractor programming section for corrective action. Upon resolution of the SIR, a completed SIR will be returned to the Materiel Developer for inclusion in the SIR Data Base.

SOFTWARE INVESTIGATION REPORT

Activity Reporting Problem: P.O.C.____________
Reporting Method: EPR#____ OD#____ Other____
Phase of Development: EDT-C____ PQT-C____ PQT-G____ ADVT-C____ ADVT-G____
SYSTEM VERSION:
Type of Error: Catastrophic____ Major____ Minor____
Suggested Improvement____
Description of Problem:

CORRECTIVE ACTION
INITIAL ANALYSIS

Report Number:____
Date Investigation Started____
Initial Analysis of Problem:
Software____ Hardware____ Operator Error____

RESOLUTION OF PROBLEM

Activity Resolving Problem: P.O.C.__________
Date:
Hardware____ Operator Error____
Software____
Program: Applications___ Executive___
Module:____
Probable Cause:
Ambiguous Requirements____
Design Error____
Coding Error____
Timing Error____
Previous Corrective Action____

Corrective Action:
Fixed____
Description of Fix____
Changed Requirement____
Deleted Requirement____

Analyst/Programmer Time in Resolving Problem____ M/Hrs
Number of other Modules that Required Changes____

IN-PLANT VERIFICATION

Date of Test:	Type of Test	System Version
____________	Static Code	TEMP____PERM____
____________	Single Thread	TEMP____ PERM____
____________ m	System	TEMP____PERM____

FIELD VERIFICATION

Date of Test:	Type of Test	System Version
____________	Single Thread	TEMP____PERM____
____________	System	TEMP____PERM____

Fig. 2

(2) Formal Contractor Testing: Preliminary Qualification Test and Formal Qualification Test procedures for handling the SIR's will be identical to that used during informal contractor testing.

(3) Government Testing: The government testing agency will complete the descriptive section of the SIR and forward SIR to Materiel Developer who will assign a unique number to this SIR and enter this in the SIR Data Base. The SIR is then forwarded to the Contractor's programming section for completion of the Initial Analysis Section and corrective action. The initial analysis information is forwarded to the Materiel Developer for inclusion in the SIR Data Base. Upon completion/correction of the SIR, the contractor's programming section will forward this information to the Materiel Developer for inclusion in the SIR Data Base. The Materiel Developer will send copies biweekly to testing agencies on those SIR's that have had some new information added or have been completed. If the test agency performs a test to verify the correction, then this information will be forwarded to Materiel Developer for inclusion in the Data Base.

2.12.2.3 Monthly Reports: The Materiel Developer will provide monthly summaries to Computer Resources Working Group (CRWG) and the Test Integration Working Group's Software Subcommittee. Figures (3-14) are samples that contain the required information to be provided. Although it is not mandatory, it is highly recommended that SIR Data Base be automated to provide flexibility in producing reports; additionally, automation will enhance the Materiel Developer's analysis and management of the software development.

2.13 Software Quality.

2.13.1 Objective: To assure that proper and sufficient data is available in a timely fashion to support the software portion of the system quality evaluation.

2.13.2 Procedure.

2.13.2.1 Measures of Quality: During the evaluation process, four measures of software quality (at a minimum) will be estimated by the evaluation. These are:

(1) Usability: The effort required to learn, operate, prepare input, and interpret output.

(2) Correctness: Extent to which a program satisfies its specifications and fulfills the user mission objectives.

SIR MONTHLY REPORT

AMBIGUOUS REQUIREMENTS

MONTH ENDING DD MMM YY

CATEGORY	EXEC PGMS	APPL PGMS	AVG. M/HRS CORR.	RESP TIME DAYS	STATUS				AVG # MODULES CHANGED	PREVIOUS SIR'S CLOSED	DELETED REQUIREMENTS	CHANGED REQUIREMENTS
					OPEN		CLOSED					
					#	%	#	%				
CATASTR.												
MAJOR												
MINOR												
SUG. IMP.												

Fig. 3

SIR MONTHLY REPORT

DESIGN ERRORS

MONTH ENDING DD MMM YY

CATEGORY	EXEC PGMS	APPL PGMS	AVG. M/HRS CORR.	RESP TIME DAYS	STATUS				PREVIOUS SIR'S CLOSED	AVG # MODULES CHANGED
					OPEN		CLOSED			
					#	%	#	%		
CATASTR										
MAJOR										
MINOR										
SUG. IMP.										

Fig. 4

SIR MONTHLY REPORT

CODING ERRORS

MONTH ENDING DD MMM YY

CATEGORY	EXEC PGMS	APPL PGMS	AVG. M/HRS CORR.	RESP TIME DAYS	STATUS OPEN #	%	CLOSED #	%	AVG # MODULES CHANGED	PREVIOUS SIR'S CLOSED
CATASTR										
MAJOR										
MINOR										
SUG. IMP.										

Fig. 5

SIR MONTHLY REPORT

TIMING ERRORS

MONTH ENDING DD MMM YY

CATEGORY	EXEC PGMS	APPL PGMS	AVG. M/HRS CORR.	RESP TIME DAYS	STATUS				AVG # MODULES CHANGED	PREVIOUS SIR'S CLOSED
					OPEN		CLOSED			
					#	%	#	%		
CATASTR										
MAJOR										
MINOR										
SUG. IMP.										

Fig. 6

SIR MONTHLY REPORT

PREVIOUS CORRECTIVE ACTIONS ERRORS

MONTH ENDING DD MMM YY

CATEGORY	EXEC PGMS	APPL PGMS	AVG. M/HRS CORR.	RESP TIME DAYS	STATUS				AVG # MODULES CHANGED	PREVIOUS SIR'S CLOSED
					OPEN		CLOSED			
					#	%	#	%		
CATASTR										
MAJOR										
MINOR										
SUG. IMP.										

Fig 7

SIR MONTHLY REPORT

BY

CATEGORY

CATEGORY	AMBIGUOUS REQUIREMENT		DESIGN ERRORS		CODING ERRORS		TIMING ERRORS		PREVIOUS CORR. ACTIONS	
	OPEN	CLSD	OPEN	CLSD	OPEN	CLSD	OPEN	CLSD	OPEN	CLSD
CATASTR										
MAJOR										
MINOR										
SUG. IMP.										

Fig. 8

SIR CUMULATIVE REPORT

AMBIGUOUS REQUIREMENT

MONTH ENDING DD MMM YY

CATEGORY	EXEC PGMS	APPL PGMS	AVG. M/HRS CORR.	RESP TIME DAYS	STATUS				AVG. # MODULES CHANGED	CHANGED REQUIREMENT	DELETED REQUIREMENT
					OPEN		CLOSED				
					#	%	#	%			
CATASTR											
MAJOR											
MINOR											
SUG. IMP.											

Fig. 9

SIR CUMULATIVE REPORT

DESIGN ERRORS

MONTH ENDING DD MMM YY

CATEGORY	EXEC PGMS	APPL PGMS	AVG. M/HRS CORR.	RESP TIME DAYS	STATUS				AVG # MODULES CHANGED
					OPEN		CLOSED		
					#	%	#	%	
CATASTR									
MAJOR									
MINOR									
SUG. IMP.									

Fig. 10

SIR CUMULATIVE REPORT

CODING ERRORS

MONTH ENDING DD MMM YY

CATEGORY	EXEC PGMS	APPL PGMS	AVG. M/HRS CORR.	RESP TIME DAYS	STATUS				AVG # MODULES CHANGED
					OPEN #	%	CLOSED #	%	
CATASTR									
MAJOR									
MINOR									
SUS. IMP.									

- Fig. 11

SIR CUMULATIVE REPORT

TIMING ERRORS

MONTH ENDING DD MMM YY

CATEGORY	EXEC PGMS	APPL PGMS	AVG. M/HRS CORR.	RESP TIME DAYS	STATUS				AVG # MODULES CHANGED
					OPEN		CLOSED		
					#	%	#	%	
CATASTR									
MAJOR									
MINOR									
SUG. IMP.									

Fig. 12

SIR CUMULATIVE REPORT

PREVIOUS CORRECTIVE ACTION ERRORS

MONTH ENDING DD MMM YY

CATEGORY	EXEC PGMS	APPL PGMS	AVG. M/HRS CORR.	RESP TIME DAYS	STATUS				AVG # MODULES CHANGED
					OPEN		CLOSED		
					#	%	#	%	
CATASTR									
MAJOR									
MINOR									
SUG. IMP.									

Fig. 13

SIR CUMULATIVE REPORT

BY

CATEGORY

CATEGORY	AMBIGUOUS REQUIREMENT		DESIGN ERRORS		CODING ERRORS		TIMING ERRORS		PREVIOUS CORR. ACTIONS	
	OPEN	CLSD	OPEN	CLSD	OPEN	CLSD	OPEN	CLSD	OPEN	CLSD
CATASTR										
MAJOR										
MINOR										
SUG. IMP.										

Fig. 14

(3) Maintainability: Effort required to locate, fix and test an error or a required change to a function.

(4) Interoperability: Efforts required to couple one system with another.

2.13.2.2 TECOM Responsibilities: The data required and the sources of the data are provided in Tables 1 thru 4. The TECOM field activity will assure that the applicable data is provided to the evaluator in a timely fashion.

TABLE 1
SOFTWARE QUALITY FACTOR CORRECTNESS

Data Required	Source
1. Cross reference of requirements to A-level specification to Design specification (B.5) to product specification (C.5).*	1. Review of documentation.
2. Does design agree with specification?	2. Review of documentation.
3. Does code agree with design?	3. Review of documentation.
4. Number of uncorrected SIR's.	4. During development SIR's will be collected.
5. Are adequacy requirements of inputs, outputs, processing, and constants defined in Design Documents?	5. Review of documentation.
6. Does code implementation achieve the accuracy requirements?	6. Review of documentation and System Field Test (PQT-C and PQT-G).
7. Percent of input variables that are checked for range tolerance.	7. PQT-C and PQT-G system test.
8. Are inputs checked for consistency with other inputs?	8. Review of documentation.
9. Are all conditions and processing defined for each decision point?	9. Review of documentation.
10. Are input errors displayed for operator action?	10. Review of documentation and System Field Test (PQT-C and PQT-G).
11. Do modules check for sufficient input data prior to start of processing?	11. Review of documentation and System Field Test (PQT-C and PQT-G).
12. Are loops and multiple transfer indices checked for range tolerance prior to execution?	12. Review of documentation.

TABLE 1 (Cont'd)
SOFTWARE QUALITY FACTOR CORRECTNESS

Data Required	Source
13. Does the design include recovery from hardware errors?	13. Review of documentation.
14. Has the hardware error recovery design been successfully implemented?	14. PQT-C and PQT-G.
15. Percent of paths that have been tested for each module.	15. PQT-C and PQT-G.
16. Percent of parameters that have been tested outside of the range of parameters tolerance.	16. PQT-C and PQT-G.
17. Percent of module interfaces that have been tested.	17. PQT-C and PQT-G.

*MIL-STD's 490 and 483 define the documentation requirements for A, B.5 and C.5 levels of specification.

TABLE 2
SOFTWARE QUALITY FACTOR USABILITY

Data Required	Source
1. Are all error conditions and response appropriately described to operator?	1. Review documentation and results of PQT-G and OT-II.
2. Are there provisions for operator to interrupt, obtain status, save, modify and continue processing?	2. PQT-C and PQT-G system test.
3. The worst case percent of time operator is busy.	3. PQT-G system test.
4. Are the operator messages consistent and responses standard?	4. Review of documentation and results of PQT-G and OT-II.
5. Number of default values defined to operator for modification.	5. Review of documentation.
6. Total number of default values in software.	6. Review of documentation.
7. Number of input formats that operator must master.	7. Review of documentation.
8. Percent of input parameter that can be operator corrected prior to execution.	8. Review of documentation and PQT-C and PQT-G system test.
9. Are there provisions for specifying inputs from other than primary sources?	9. Review of documentation.
10. Do outputs have operator selectable controls?	10. Review of documentation.
11. Do outputs have user oriented identification?	11. PQT-G and OT-II system test.
12. Number of different output formats.	12. Review of documentation.

TABLE 2 (Cont'd)
SOFTWARE QUALITY FACTOR USABILITY

Data Required	Source
13. Do error messages have special activity to alert operator?	13. Review of documentation and PQT-G and OT-II system test.
14. Will the design and implementation permit output to be selectably controlled to other than primary source?	14. Review of documentation.
15. Are realistic simulated exercises provided?	15. PQT-G and OT-II system tests.
16. Is there sufficient diagnostic information provided to operator to perform corrective action?	16. PQT-G and OT-II system tests.

TABLE 3
SOFTWARE QUALITY FACTOR MAINTAINABILITY

Data Required	Source
1. Cross reference of requirements to A-level specification to design specifications (B.5) to product specifications (C.5)*.	1. Review of documentation.
2. For each module the percent of loops that contain non-loop dependent variables.	2. Review of documentation.
3. Does the design allocate storage requirements for each module?	3. Review of documentation.
4. Does the source code have comments that reference the B.5 specifications that are being implemented?	4. Review of documentation.
5. Are all conditions and processing defined for each decision point?	5. Review of documentation.
6. Percent of modules whose design does not comply with government standards or approved contractor standards.	6. Review of documentation.
7. Percent of modules whose implementation violate the design standards for module interaction.	7. Review of documentation.
8. Percent of modules whose implementation violate the design standards for error handling conventions.	8. Review of documentation.
9. When an error is detected, is the correction procedure in the calling module?	9. Review of documentation.
10. Percent of modules that are independent of storage size, buffer size or array size.	10. Review of documentation.
11. Percent of modules that have a standard formatted prologue of comments.	11. Review of documentation.

TABLE 3 (Cont'd)
SOFTWARE QUALITY FACTOR MAINTAINABILITY

Data Required	Source
12. Percent of modules that implement a standard convention for setting off comments from source data.	12. Review of documentation.
13. Percent of modules that have comments for all transfer of controls and destinations.	13. Review of documentation.
14. Percent of modules that have comments for all machine dependent code or non-standard High Order Language Constraints.	14. Review of documentation.
15. Percent of modules that describe the attributes of all declared Variables.	15. Review of documentation.
16. Percent of modules that violate the Development Programming Standards.	16. Review of documentation.
17. Percent of program variables that utilize descriptive names.	17. Review of documentation.
18. Percent of modules that have more than one statement per line.	18. Review of documentation.
19. Percent of modules that comply with standard I/O conventions.	19. Review of documentation.
20. Percent of modules that violate data consistency, i.e., variable stores more than one data type.	20. Review of documentation.
21. Utilizing design documents construct a hierarchical chart of system flow if a chart doesn't exist.	21. Review of documentation.
22. Average man-hours expended to correct software errors per module due to: Design error, coding error, and timing error.	22. SIR summaries.

TABLE 3 (Cont'd)
SOFTWARE QUALITY FACTOR MAINTAINABILITY

Data Required	Source
23. Percent of modules that are not independent, i.e., module is dependent on the source of input.	23. Review of documentation.
24. Percent of modules where documentation (comments) describe input, output, processing and the limitation of this module.	24. Review of documentation.
25. For each module, the percent of loops that have only one entrance.	25. Review of documentation.
26. Percent of modules that use variables for only one parameter excluding variables that are used for indexing.	26. Review of documentation.
27. For each module the maximum depth of nesting.	27. Review of documentation.
28. For each module the ratio of Branch instructions to all instructions for this module.	28. Review of documentation.
29. The expected number of modules that would require modification if a change is applied to one module.	29. SIR's collected during PQT-C and PQT-G for design modification.
30. The percent of modules that perform only one function.	30. Review of documentation.
31. The percent of modules that can only return to calling modules.	31. Review of documentation.
32. Percent of memory capacity committed.	32. PQT-C.
33. Percent of CPU capacity (I/O and processer) being utilized.	33. (a) Full system load of maximum user units transmitting. (b) Medium system load with 50% of user units transmitting. (c) Load(s) predicted to stress either I/O capacity or processer capacity.

*MIL-STD's 490 and 483 define the documentation requirements for A, B.5 and C.5 level of specifications.

TABLE 4
SOFTWARE QUALITY FACTOR INTEROPERABILITY

Data Required	Source
1. Does the system design (B.5) have statement of requirements for communication with other systems?	1. Review of documentation.
2. Are the communication protocol standards defined and implemented?	2. Review of documentation and PQT-G and OT-II system test.
3. Number of modules that perform the communication interface function.	3. Review of documentation.
4. Is there a standard data representation for communication interface?	4. Review of documentation.
5. Does the design have provision for identification and password checking for communication and is this design implemented in source code?	5. Review of documentation.
6. Does the design have provisions for access control for the data base and has it been implemented in source code?	6. Review of documentation.
7. Have provisions for memory protection been included in design and has this design been implemented in code?	7. Review of documentation.
8. Does the system design include provisions for recording and reporting accesses and reporting violations of access to the system?	8. Review of documentation and PQT-G and OT-II.