Recognized as an
American National Standard (ANSI)

IEEE
Std 829-1983

IEEE Standard for Software Test Documentation

Sponsor

Software Engineering Technical Committee
of the
IEEE Computer Society

Approved December 3, 1982
Reaffirmed March 21, 1991

IEEE Standards Board

Approved August 19, 1983
Reaffirmed September 6, 1991

American National Standards Institute

The Institute of Electrical and Electronics Engineers, Inc
345 East 47th Street, New York, NY 10017, USA

IEEE Standards documents are developed within the Technical Committees of the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE which have expressed an interest in participating in the development of the standard.

Use of an IEEE Standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least once every five years for revision or reaffirmation. When a document is more than five years old, and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of all concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason IEEE and the members of its technical committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE Standards Board
345 East 47th Street
New York, NY 10017
USA

Foreword

(This Foreword is not a part of ANSI/IEEE Std 829-1983, IEEE Standard for Software Test Documentation.)

Purpose

The purpose of this standard is to describe a set of basic software test documents. A standardized test document can facilitate communication by providing a common *frame of reference* (for example, a customer and a supplier have the same definition for a test plan). The content definition of a standardized test document can serve as a completeness checklist for the associated testing process. A standardized set can also provide a baseline for the evaluation of current test documentation practices. In many organizations, the use of these documents significantly increases the manageability of testing. Increased manageability results from the greatly increased visibility of each phase of the testing process.

This standard specifies the form and content of individual test documents. It does not specify the required set of test documents. It is assumed that the required set of test documents will be specified when the standard is applied. Appendix B contains an example of such a set specification.

Overview

The documents outlined in this standard cover test planning, test specification, and test reporting.

The test plan prescribes the scope, approach, resources, and schedule of the testing activities. It identifies the items to be tested, the features to be tested, the testing tasks to be performed, the personnel responsible for each task, and the risks associated with the plan.

Test specification is covered by three document types:

(1) A test-design specification refines the test approach and identifies the features to be covered by the design and its associated tests. It also identifies the test cases and test procedures, if any, required to accomplish the testing and specifies the feature pass/fail criteria.

(2) A test-case specification documents the actual values used for input along with the anticipated outputs. A test case also identifies any constraints on the test procedures resulting from use of that specific test case. Test cases are separated from test designs to allow for use in more than one design and to allow for reuse in other situations.

(3) A test procedure specification identifies all steps required to operate the system and exercise the specified test cases in order to implement the associated test design. Test procedures are separated from test-design specifications as they are intended to be followed step by step and should not have extraneous detail.

Test reporting is covered by four document types:

(1) A test item transmittal report identifies the test items being transmitted for testing in the event that separate development and test groups are involved or in the event that a formal beginning of test execution is desired.

(2) A test log is used by the test team to record what occurred during test execution.

(3) A test incident report describes any event that occurs during the test execution which requires further investigation.

(4) A test summary report summarizes the testing activities associated with one or more test-design specifications.

Figure 1 shows the relationships of these documents to one another as they are developed and to the testing process they document.

Terminology

The words *shall*, *must*, and the imperative form identify the mandatory material within this standard. The words *should* and *may* identify optional material.

Appendixes

The examples found in Appendix A are meant to clarify the intent of the document descriptions found in the standard. Some suggestions about implementing and using the standard are in Appendix B. Appendix C contains references to related test documentation standards. Appendix D contains references to testing-related documents of general interest which are not focused on test documentation.

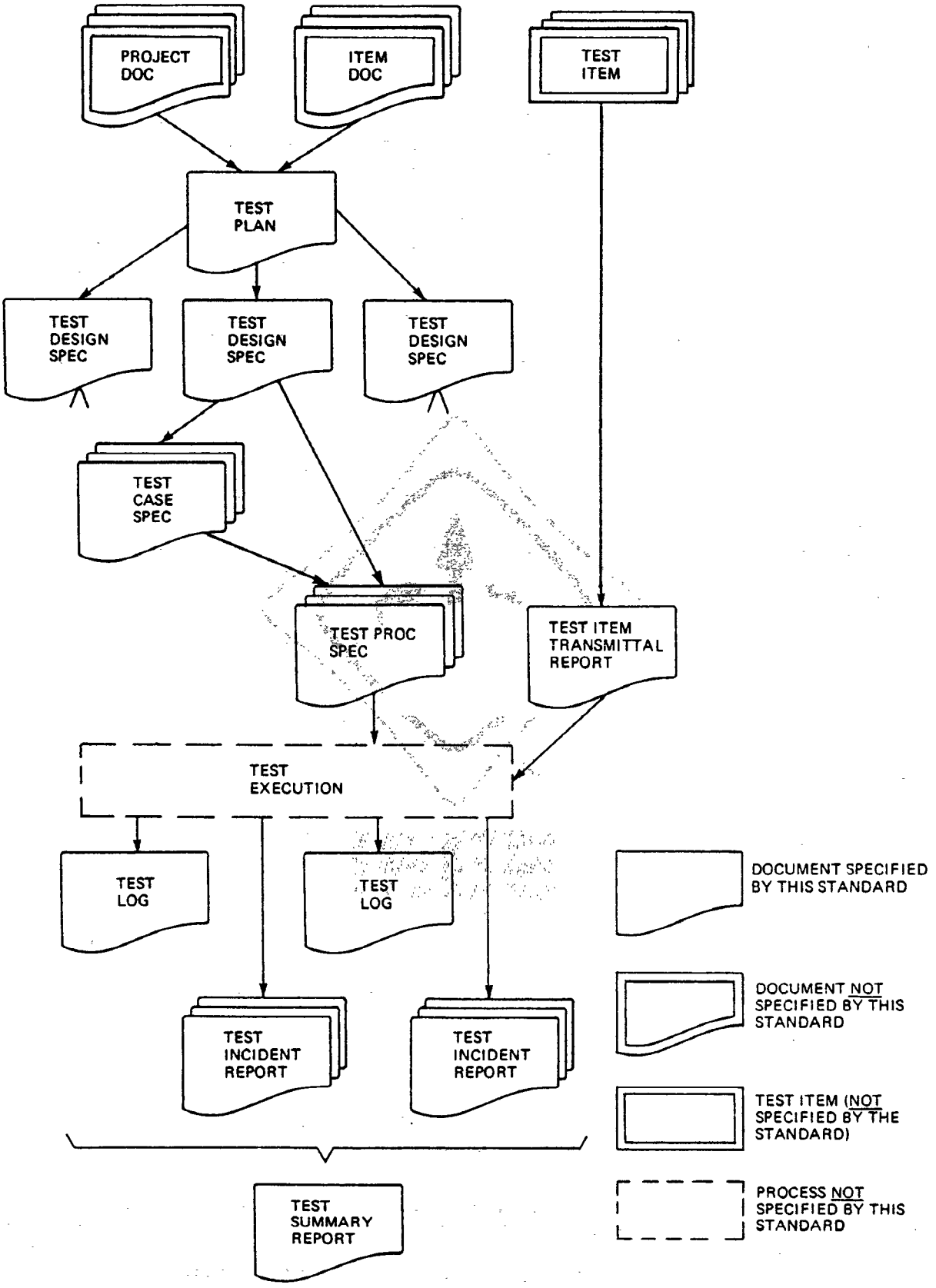


Fig 1
Relationship of Test Documents to Testing Process

Audience

The standard should be of interest to software users and software procurement personnel; to development, test, and maintenance personnel; to operations and acquisition support managers; to software quality assurance personnel and auditors; and to participants in the legal system.

History

The development of this standard began from discussions within the Software Engineering Standards Subcommittee in 1977. Some initial work was done by a group including Joan Bateman, Leonard Birns, Herb Hecht, and Bob Poston and resulted in an early draft outline. The project authorization request for this effort was approved by the IEEE Standards Board in May, 1980. Following authorization, a series of ten meetings which were held across the country from September, 1980 to May, 1982 produced the first draft submitted for balloting.

Suggestions for improvement of the standard will be welcome. They should be sent to:
Secretary
IEEE Standards Board
Institute of Electrical and Electronics Engineers
345 East 47th Street
New York, NY 10017.

At the time this standard was approved on December 3, 1982, the IEEE Standards Board had the following membership:

Irvin N. Howell, *Chairman* **Edward Chelotti,** *Vice Chairman*

Sava I. Sherr, *Secretary*

G. Y. R. Allen	D. C. Fleckenstein	A. R. Parsons
J. J. Archambault	Jay Forster	J. P. Riganati
James H. Beall	Kurt Greene	Frank L. Rose
John T. Boettger	Joseph L. Koepfinger	Robert W. Seelbach
J. V. Bonucchi	Irving Kolodny	Jay A. Stewart
Edward J. Cohen	John E. May	Clifford O. Swanson
Len S. Corey	Donald T. Michael*	Robert E. Weiler

*Member emeritus

This standard was completed by a working group with the following members:

David Gelperin, *Chairman*

Rick Adrion	Tom Gilb	Art Pollari
Jules Aronson	Loretta Guarino	Robert Poston
Joan Bateman	John Hawthorne	Sam Redwine
Charlie Bates	Herb Hecht	Charles Schult
A. Birnbaum	Mike Hennell	David Schultz
Leonard Birns	Dan Hocking	Dennis Sharp
John Bowen	Mark Holthouse	Jerry Smith
Martha Branstad	Ray Houghton	Wayne Smith
Fletcher Buckley	Paul Howley	Joan Spalding
Douglas Burt	John Kalasky	Bob Stewart
Ian Burton	Jason Kazarian	S. L. Stewart
John Cain	Ed Kit	Sarah Swales
John Center	Denise Krauss	Barbara Taute
George Chapman	Joseph Krupinski	J. R. Taylor
Santosh Chokhani	Tom Kurihara	Rudolf Van Megen
Bruce Clay	Costa J. Labovites	David Vatsaas
D. L. Cooper	Paula Levinton	John Walter
Doc Craddock	Leo Lippman	Andrew Weigel
James Cridge	Alex Long	Jonathon Wexler
Patricia Daggett	Mable Love	Paul White
Bill Dupras	Ben Manny	Bruce Wilkins
Mary Eads	Albrecht Newmann	Harry Wilkinson
Sharon Freid	Bill Newsom	F. T. Woodall
Tom Frost	C. Donald Ostler	Fred Yonda
	Varsha Pai	

The standard was approved by the Software Engineering Standards Subcommittee of the IEEE Computer Society. At the time it approved this standard, the subcommittee had the following membership:

Fletcher J. Buckley, *Chairman*

Russell J. Abbott
A. Frank Ackerman
Leo Beltracchi
Richard L. Bernstein
Nestore G. Biasi
Dennis W. Bragg
Douglas W. Burt
Homer C. Carney
John Center
Won L. Chung
Anonio M. Cicu
Lori A. Clarke
Gilmore G. Cooke
A. J. Cote, Jr
Patricia W. Daggett
George Darling
B. Dasarathy
Noah S. Davids
James A. Dobbins
Mary L. Eads
John D. Earls
Leo G. Egan, Jr
John W. Fendrich
Dennis W. Fife
Joel J. Forman
M. Galinier
Forrest Karl Gardner
David Gelperin
Gary Gladden
Shirley A. Gloss-Soler
John J.Greene
Jack W. Grigsby
Robert M. Gross
David A. Gustafson
Russell T. Gustin
Thomas L. Hannan
Herbert Hecht
Leslie R. Heselton, III
Sam Horvitz
Paul Howley
Shuenn-Chang Hwang
Jim H. Ingram
John P. Kalasky
Laurel V. Kaleda
Thomas M. Kurihara
Domenic V. La-Rosa
Robert A. C. Lane
Gregory N. Larsen
Geoffrey R. Lewis
A. Lip Lim
G. S. Lindsay
Myron Lipow
William M. Lively
Marvin Lubiofsky
Don Lundquist
Alan Cheuk-Wai Ma
Andy K. Mahindru
Philip C. Marriott
Mike McCollough
Belden Menkus
Edward F. Miller, Jr
G. Scott Morris
Gene T. Morun
Walter G. Murch
Jack Nebb
Geraldine Rajcula Neidhart
Michael A. Neighbors
Leon Osterweil
Donald J. Ostrom
William E. Perry
Donald J. Pfeiffer
Robert M. Poston
Patricia B. Powell
Farzad Raji
Jean Claude Rault
Samuel T. Redwine
T. L. Regulinski
William E. Riddle
Clarence W. Rutter, III
Norman F. Schneidewind
Antonia D. Schuman
Leonard W. Seagren
Wayne Smith
Harry M. Sneed
Lee Sprague
Edward A. Straker
Kou Chung Tai
Barbara J. Taute
George D. Tice, Jr
Terrence L. Tillmanns
William S. Turner, III
Edwin A. Ulbrich, Jr
Udo Voges
John P. Walter
Andrew H. Weigel
N. P. Wilburn
Martin Wong
David C. Wood
Ted Workman
Alfred W. Yonda
Peter F. Zoll

Special representatives to the Software Engineering Subcommittee were as follows:

William E. Perry, *DPMA*
Roy P. Pritchett, Jr, *EDP Auditors Association*
John Milandin, *ANSI Z1*

Contents

IEEE Standard for Software Test Documentation

1. Scope

This standard describes a set of basic test documents which are associated with the dynamic aspects of software testing (that is, the execution of procedures and code). The standard defines the purpose, outline, and content of each basic document. While the documents described in the standard focus on dynamic testing, several of them may be applicable to other testing activities (for example, the test plan and test incident report may be used for design and code reviews).

The standard may be applied to commercial, scientific, or military software which runs on any digitial computer. Applicability is not restricted by the size, complexity, or criticality of the software. However, the standard does *not* specify any class of software to which it must be applied. The standard addresses the documentation of both initial development testing and the testing of subsequent software releases. For a particular software release, it may be applied to all phases of testing from module testing through user acceptance. However, since all of the basic test documents may not be useful in each test phase, the particular documents to be used in a phase are *not* specified. Each organization using the standard will need to specify the classes of software to which it applies and the specific documents required for a particular test phase.

The standard does *not* call for specific testing methodologies, approaches, techniques, facilities, or tools, and does *not* specify the documentation of their use. Additional test documentation may be required (for example, code inspection checklists and reports). The standard also does *not* imply or impose specific methodologies for documentation control, configuration management, or quality assurance. Additional documentation (for example, a quality assurance plan) may be needed depending on the particular methodologies used.

Within each standard document, the content of each section (that is, the text which covers the designated topics) may be tailored to the particular application and the particular testing phase. In addition to tailoring content, additional documents may be added to the basic set, additional sections may be added to any document and additional content to any section. It may be useful to organize some of the sections into subsections. Some or all of the contents of a section may be contained in another document which is then referenced. Each organization using the standard should specify additional content requirements and conventions in order to reflect their own particular methodologies, approaches, facilities, and tools for testing, documentation control, configuration management, and quality assurance.

The standard applies to documentation on electronic media as well as paper. Paper must be used for documents requiring approval signatures, unless the electronic documentation system has a secure approval annotation mechanism and that mechanism is used.

2. Definitions

This section contains key terms as they are used in the standard.

design level. The design decomposition of the software item (for example, system, subsystem, program, or module).

pass/fail criteria. Decision rules used to determine whether a software item or a software feature passes or fails a test.

software feature. A distinguishing characteristic of a software item (for example, performance, portability, or functionality).

software item. Source code, object code, job control code, control data, or a collection of these items.

test. (1) A set of one or more test cases, or

(2) A set of one or more test procedures, or
(3) A set of one or more test cases and procedures.

test case specification. A document specifying inputs, predicted results, and a set of execution conditions for a test item.

test design specification. A document specifying the details of the test approach for a software feature or combination of software features and identifying the associated tests.

test incident report. A document reporting on any event that occurs during the testing process which requires investigation.

test item. A software item which is an object of testing.

test item transmittal report. A document identifying test items. It contains current status and location information.

test log. A chronological record of relevant details about the execution of tests.

test plan. A document describing the scope, approach, resources, and schedule of intended testing activities. It identifies test items, the features to be tested, the testing tasks, who will do each task, and any risks requiring contingency planning.

test procedure specification. A document specifying a sequence of actions for the execution of a test.

test summary report. A document summarizing testing activities and results. It also contains an evaluation of the corresponding test items.

testing. The process of analyzing a software item to detect the differences between existing and required conditions (that is, bugs) and to evaluate the features of the software item.

3. Test Plan

3.1 Purpose. To prescribe the scope, approach, resources, and schedule of the testing activities. To identify the items being tested, the features to be tested, the testing tasks to be performed, the personnel responsible for each task, and the risks associated with this plan.

3.2 Outline. A test plan shall have the following structure:

1. Test-plan identifier
2. Introduction
3. Test items
4. Features to be tested
5. Features not to be tested
6. Approach
7. Item pass/fail criteria
8. Suspension criteria and resumption requirements
9. Test deliverables
10. Testing tasks
11. Environmental needs
12. Responsibilities
13. Staffing and training needs
14. Schedule
15. Risks and contingencies
16. Approvals

The sections shall be ordered in the specified sequence. Additional sections may be included immediately prior to *Approvals*. If some or all of the content of a section is in another document, then a reference to that material may be listed in place of the corresponding content. The referenced material must be attached to the test plan or available to users of the plan.

Details on the content of each section are contained in the following sections.

3.2.1 Test-Plan Identifier. Specify the unique identifier assigned to this test plan.

3.2.2 Introduction. Summarize the software items and software features to be tested. The need for each item and its history may be included.

References to the following documents, when they exist, are required in the highest-level test plan:

Project authorization
Project plan
Quality assurance plan
Configuration management plan
Relevant policies
Relevant standards

In multilevel test plans, each lower-level plan must reference the next higher-level plan.

3.2.3 Test Items. Identify the test items including their version/revision level. Also specify characteristics of their transmittal media which impact hardware requirements or indicate the need for logical or physical transformations before testing can begin (for example, programs must be transferred from tape to disk).

Supply references to the following item documentation, if it exists:

Requirements specification
Design specification
Users guide
Operations guide
Installation guide

Reference any incident reports relating to the test items.

Items which are to be specifically excluded from testing may be identified.

3.2.4 Features to be Tested. Identify all software features and combinations of software features to be tested. Identify the test-design specification associated with each feature and each combination of features.

3.2.5 Features Not to be Tested. Identify all features and significant combinations of features which will not be tested and the reasons.

3.2.6 Approach. Describe the overall approach to testing. For each major group of features or feature combinations, specify the approach which will ensure that these feature groups are adequately tested. Specify the major activities, techniques, and tools which are used to test the designated groups of features.

The approach should be described in sufficient detail to permit identification of the major testing tasks and estimation of the time required to do each one.

Specify the minimum degree of comprehensiveness desired. Identify the techniques which will be used to judge the comprehensiveness of the testing effort (for example, determining which statements have been executed at least once). Specify any additional completion criteria (for example, error frequency). The techniques to be used to trace requirements should be specified.

Identify significant constraints on testing such as test-item availability, testing-resource availability, and deadlines.

3.2.7 Item Pass/Fail Criteria. Specify the criteria to be used to determine whether each test item has passed or failed testing.

3.2.8 Suspension Criteria and Resumption Requirements. Specify the criteria used to suspend all or a portion of the testing activity on the test items associated with this plan. Specify the testing activities which must be repeated, when testing is resumed.

3.2.9 Test Deliverables. Identify the deliverable documents. The following documents should be included:

Test plan
Test design specifications
Test case specifications
Test procedure specifications
Test item transmittal reports
Test logs
Test incident reports
Test summary reports

Test input data and test output data should be identified as deliverables.

Test tools (for example, module drivers and stubs) may also be included.

3.2.10 Testing Tasks. Identify the set of tasks necessary to prepare for and perform testing. Identify all intertask dependencies and any special skills required.

3.2.11 Environmental Needs. Specify both the necessary and desired properties of the test environment. This specification should contain: the physical characteristics of the facilities including the hardware, the communications and system software, the mode of usage (for example, stand-alone), and any other software or supplies needed to support the test. Also specify the level of security which must be provided for the test facilities, system software, and proprietary components such as software, data, and hardware.

Identify special test tools needed. Identify any other testing needs (for example, publications or office space). Identify the source for all needs which are not currently available to the test group.

3.2.12 Responsibilities. Identify the groups responsible for managing, designing, preparing, executing, witnessing, checking, and resolving. In addition, identify the groups responsible for providing the test items identified in 3.2.3 and the environmental needs identified in 3.2.11.

These groups may include the developers, testers, operations staff, user representatives, technical support staff, data administration staff, and quality support staff.

3.2.13 Staffing and Training Needs. Specify test staffing needs by skill level. Identify training options for providing necessary skills.

3.2.14 Schedule. Include test milestones identified in the Software Project Schedule as well as all item transmittal events.

Define any additional test milestones needed. Estimate the time required to do each testing task. Specify the schedule for each testing task and test milestone. For each testing resource (that is, facilities, tools, and staff), specify its periods of use.

3.2.15 Risks and Contingencies. Identify the high-risk assumptions of the test plan. Specify contingency plans for each (for example, delayed delivery of test items might require increased night shift scheduling to meet the delivery date).

3.2.16 Approvals. Specify the names and titles of all persons who must approve this plan. Provide space for the signatures and dates.

4. Test-Design Specification

4.1 Purpose. To specify refinements of the test approach and to identify the features to be tested by this design and its associated tests.

4.2 Outline. A test-design specification shall have the following structure:

(1) Test-design-specification identifier
(2) Features to be tested
(3) Approach refinements
(4) Test identification
(5) Feature pass/fail criteria

The sections shall be ordered in the specified sequence. Additional sections may be included at the end. If some or all of the content of a section is in another document, then a reference to that material may be listed in place of the corresponding content. The referenced material must be attached to the test-design specification or available to users of the design specification.

Details on the content of each section are contained in the following sections.

4.2.1 Test-Design-Specification Identifier. Specify the unique identifier assigned to this test-design specification. Supply a reference to the associated test plan, if it exists.

4.2.2 Features to be Tested. Identify the test items and describe the features and combinations of features which are the object of this design specification. Other features may be exercised, but need not be identified.

For each feature or feature combination, a reference to its associated requirements in the item requirement specification or design description should be included.

4.2.3 Approach Refinements. Specify refinements to the approach described in the test plan. Include specific test techniques to be used. The method of analyzing test results should be identified (for example, comparator programs or visual inspection).

Specify the results of any analysis which provides a rationale for test-case selection. For example, one might specify conditions which permit a determination of error tolerance (for example, those conditions which distinguish valid inputs from invalid inputs).

Summarize the common attributes of any test cases. This may include input constraints that must be true for every input in the set of associated test cases, any shared environmental needs, and any shared special procedural requirements, and any shared case dependencies.

4.2.4 Test Identification. List the identifier and a brief description of each test case associated with this design. A particular test case may be identified in more than one test design specification. List the identifier and a brief description of each procedure associated with this test-design specification.

4.2.5 Feature Pass/Fail Criteria. Specify the criteria to be used to determine whether the feature or feature combination has passed or failed.

5. Test-Case Specification

5.1 Purpose. To define a test case identified by a test-design specification.

5.2 Outline. A test-case specification shall have the following structure:

(1) Test-case-specification identifier
(2) Test items
(3) Input specifications
(4) Output specifications
(5) Environmental needs
(6) Special procedural requirements
(7) Intercase dependencies

The sections shall be ordered in the specified sequence. Additional sections may be included at the end. If some or all of the content of a section is in another document, then a reference to that material may be listed in place of the corresponding content. The referenced material must be attached to the test-case specification or available to users of the case specification.

Since a test case may be referenced by several test-design specifications used by different groups over a long time period, enough specific information must be included in the test-case specification to permit reuse.

Details on the content of each section are contained in the following sections.

5.2.1 Test-Case-Specification Identifier. Specify the unique identifier assigned to this test-case specification.

5.2.2 Test Items. Identify and briefly describe the items and features to be exercised by this test case.

For each item, consider supplying references to the following item documentation.

(1) Requirements specification
(2) Design specification
(3) Users guide
(4) Operations guide
(5) Installation guide

5.2.3 Input Specifications. Specify each input required to execute the test case. Some of the inputs will be specified by value (with tolerances where appropriate), while others, such as constant tables or transaction files, will be specified by name. Identify all appropriate data bases, files, terminal messages, memory resident areas, and values passed by the operating system.

Specify all required relationships between inputs (for example, timing).

5.2.4 Output Specifications. Specify all of the outputs and features (for example, response time) required of the test items. Provide the exact value (with tolerances where appropriate) for each required output or feature.

5.2.5 Environmental Needs.

5.2.5.1 Hardware. Specify the characteristics and configurations of the hardware required to execute this test case (for example, 132 character × 24 line CRT).

5.2.5.2 Software. Specify the system and application software required to execute this test case. This may include system software such as operating systems, compilers, simulators, and test tools. In addition, the test item may interact with application software.

5.2.5.3 Other. Specify any other requirements such as unique facility needs or specially trained personnel.

5.2.6 Special Procedural Requirements. Describe any special constraints on the test procedures which execute this test case. These constraints may involve special set up, operator intervention, output determination procedures, and special wrap up.

5.2.7 Intercase Dependencies. List the identifiers of test cases which must be executed prior to this test case. Summarize the nature of the dependencies.

6. Test-Procedure Specification

6.1 Purpose. To specify the steps for executing a set of test cases or, more generally, the steps used to analyze a software item in order to evaluate a set of features.

6.2 Outline. A test-procedure specification shall have the following structure:

(1) Test-procedure-specification identifier
(2) Purpose
(3) Special requirements
(4) Procedure steps

The sections shall be ordered in the specified sequence. Additional sections, if required, may be included at the end. If some or all of the content of a section is in another document, then a reference to that material may be listed in place of the corresponding content. The referenced material must be attached to the test-procedure specification or available to users of the procedure specification.

Details on the content of each section are contained in the following sections.

6.2.1 Test-Procedure-Specification Identifier. Specify the unique identifier assigned to this test-procedure specification. Supply a reference to the associated test-design specification.

6.2.2 Purpose. Describe the purpose of this procedure. If this procedure executes any test cases, provide a reference for each of them.

In addition, provide references to relevant sections of the test item documentation (for example, references to usage procedures).

6.2.3 Special Requirements. Identify any special requirements that are necessary for the execution of this procedure. These may include prerequisite procedures, special skills requirements, and special environmental requirements.

6.2.4 Procedure Steps. Include the following steps as applicable:

6.2.4.1 Log. Describe any special methods or formats for logging the results of test execution, the incidents observed, and any other events pertinent to the test (see Test Log, Section 8 and Test Incident Report, Section 9).

6.2.4.2 Set Up. Describe the sequence of actions necessary to prepare for execution of the procedure.

6.2.4.3 Start. Describe the actions necessary to begin execution of the procedure.

6.2.4.4 Proceed. Describe any actions necessary during execution of the procedure.

6.2.4.5 Measure. Describe how the test

measurements will be made (for example, describe how remote terminal response time is to be measured using a network simulator).

6.2.4.6 Shut Down. Describe the actions necessary to suspend testing, when unscheduled events dictate.

6.2.4.7 Restart. Identify any procedural restart points and describe the actions necessary to restart the procedure at each of these points.

6.2.4.8 Stop. Describe the actions necessary to bring execution to an orderly halt.

6.2.4.9 Wrap Up. Describe the actions necessary to restore the environment.

6.2.4.10 Contingencies. Describe the actions necessary to deal with anomolous events which may occur during execution.

7. Test-Item Transmittal Report

7.1 Purpose. To identify the test items being transmitted for testing. It includes the person responsible for each item, its physical location, and its status. Any variations from the current item requirements and designs are noted in this report.

7.2 Outline. A test-item transmittal report shall have the following structure:

(1) Transmittal-report identifier
(2) Transmitted items
(3) Location
(4) Status
(5) Approvals

The sections shall be ordered in the specified sequence. Additional sections may be included just prior to *Approvals*. If some or all of the content of a section is in another document, then a reference to that material may be listed in place of the corresponding content. The referenced material must be attached to the test-item transmittal report or available to users of the transmittal report.

Details on the content of each section are contained in the following sections.

7.2.1 Transmittal-Report Identifier. Specify the unique identifier assigned to this test-item transmittal report.

7.2.2 Transmitted Items. Identify the test items being transmitted, including their version/revision level. Supply references to the item documentation and the test plan relating to the transmitted items. Indicate the people responsible for the transmitted items.

7.2.3 Location. Identify the location of the transmitted items. Identify the media that contain the items being transmitted. When appropriate, indicate how specific media are labeled or identified.

7.2.4 Status. Describe the status of the test items being transmitted. Include deviations from the item documentation, from previous transmittals of these items, and from the test plan. List the incident reports which are expected to be resolved by the transmitted items. Indicate if there are pending modifications to item documentation which may affect the items listed in this transmittal report.

7.2.5 Approvals. Specify the names and titles of all persons who must approve this transmittal. Provide space for the signatures and dates.

8. Test Log

8.1 Purpose. To provide a chronological record of relevant details about the execution of tests.

8.2 Outline. A test log shall have the following structure:

(1) Test log identifier
(2) Description
(3) Activity and event entries

The sections shall be ordered in the specified sequence. Additional sections may be included at the end. If some or all of the content of a section is in another document, then a reference to that material may be listed in place of the corresponding content. The referenced material must be attached to the test log or available to users of the log.

Details on the content of each section are contained in the following sections.

8.2.1 Test-Log Identifier. Specify the unique identifier assigned to this test log.

8.2.2 Description. Information which applies to all entries in the log except as specifically noted in a log entry should be included here. The following information should be considered.

(1) Identify the items being tested including their version/revision levels. For each of these items, supply a reference to its transmittal report, if it exists.

(2) Identify the attributes of the environments in which the testing is conducted. Include facility identification, hardware being used (for example, amount of memory being

used, CPU model number, and number and model of tape drives, and/or mass storage devices), system software used, and resources available such as the amount of memory available.

8.2.3 Activity and Event Entries. For each event, including the beginning and end of activities, record the occurrence date and time along with the identity of the author.

The following information should be considered:

8.2.3.1 Execution Description. Record the identifier of the test procedure being executed and supply a reference to its specification. Record all personnel present during the execution including testers, operators, and observers. Also indicate the function of each individual.

8.2.3.2 Procedure Results. For each execution, record the visually observable results (for example, error messages generated, aborts, and requests for operator action). Also record the location of any output (for example, reel number). Record the successful or unsuccessful execution of the test.

8.2.3.3 Environmental Information. Record any environmental conditions specific to this entry (for example, hardware substitutions).

8.2.3.4 Anomalous Events. Record what happened before and after an unexpected event occurred (for example, *A summary display was requested and the correct screen displayed, but response seemed unusually long. A repetition produced the same prolonged response*). Record circumstances surrounding the inability to begin execution of a test procedure or failure to complete a test procedure (for example, a power failure or system software problem).

8.2.3.5 Incident-Report Identifiers. Record the identifier of each test-incident report, whenever one is generated.

9. Test-Incident Report

9.1 Purpose. To document any event that occurs during the testing process which requires investigation.

9.2 Outline. A test-incident report shall have the following structure:

(1) Test-incident-report identifier
(2) Summary
(3) Incident description
(4) Impact

The sections shall be ordered in the specified sequence. Additional sections may be included at the end. If some or all of the content of a section is in another document, then a reference to that material may be listed in place of the corresponding content. The referenced material must be attached to the test-incident report or available to users of the incident report.

Details on the content of each section are contained in the following sections.

9.2.1 Test-Incident-Report Identifier. Specify the unique identifier assigned to this test incident report.

9.2.2 Summary. Summarize the incident. Identify the test items involved indicating their version/revision level. References to the appropriate test-procedure specification, test-case specification, and test log should be supplied.

9.2.3 Incident Description. Provide a description of the incident. This description should include the following items:

Inputs
Expected results
Actual results
Anomalies
Date and time
Procedure step
Environment
Attempts to repeat
Testers
Observers

Related activities and observations that may help to isolate and correct the cause of the incident should be included. For example, describe any test-case executions that might have a bearing on this particular incident and any variations from the published test procedure.

9.2.4 Impact. If known, indicate what impact this incident will have on test plans, test-design specifications, test-procedure specifications, or test-case specifications.

10. Test-Summary Report

10.1 Purpose. To summarize the results of the designated testing activities and to provide evaluations based on these results.

10.2 Outline. A test-summary report shall have the following structure:

(1) Test-summary-report identifier
(2) Summary
(3) Variances
(4) Comprehensive assessment
(5) Summary of results
(6) Evaluation
(7) Summary of activities
(8) Approvals

The sections shall be ordered in the specified sequence. Additional sections may be included just prior to *Approvals*. If some or all of the content of a section is in another document, then a reference to that material may be listed in place of the corresponding content. The referenced material must be attached to the test-summary report or available to users of the summary report.

Details on the content of each section are contained in the following sections.

10.2.1 Test-Summary-Report Identifier. Specify the unique identifier assigned to this test-summary report.

10.2.2 Summary. Summarize the evaluation of the test items. Identify the items tested, indicating their version/revision level. Indicate the environment in which the testing activities took place.

For each test item, supply references to the following documents if they exist: test plan, test-design specifications, test-procedure specifications, test-item transmittal reports, test logs, and test-incident reports.

10.2.3 Variances. Report any variances of the test items from their design specifications. Indicate any variances from the test plan, test designs, or test procedures. Specify the reason for each variance.

10.2.4 Comprehensiveness Assessment. Evaluate the comprehensiveness of the testing process against the comprehensiveness criteria specified in the test plan (3.2.6) if the plan exists. Identify features or feature combinations which were not sufficiently tested and explain the reasons.

10.2.5 Summary of Results. Summarize the results of testing. Identify all resolved incidents and summarize their resolutions. Identify all unresolved incidents.

10.2.6 Evaluation. Provide an overall evaluation of each test item including its limitations. This evaluation must be based upon the test results and the item level pass/fail criteria. An estimate of failure risk may be included.

10.2.7 Summary of Activities. Summarize the major testing activities and events. Summarize resource consumption data, for example, total staffing level, total machine time, and total elapsed time used for each of the major testing activities.

10.2.8 Approvals. Specify the names and titles of all persons who must approve this report. Provide space for the signatures and dates.

Appendixes

(The following Appendixes are not a part of IEEE Std 829-1983, IEEE Standard for Software Test Documentation.)

A. Examples

The following examples are taken from commercial data processing. This should not imply any limitations on the applicability of the standard to other classes of software.

Contents

A1. Corporate Payroll-System Test Documentation

A1.1. Introduction

A1.1.1. Scope. The system test documentation example presented here is done in accordance with the IEEE Standard for Software Test Documentation. Each document is represented as it might be used for the system test of a payroll system.

The payroll system used in this example contains the following major functions:

(1) Maintain employee information
(2) Maintain payroll history information
(3) Prepare payroll checks
(4) Prepare payroll tax reports
(5) Prepare payroll history reports

A Phase 2.0 development plan exists for the payroll system which will be started at some future time. This phase covers, primarily, a personnel reporting system.

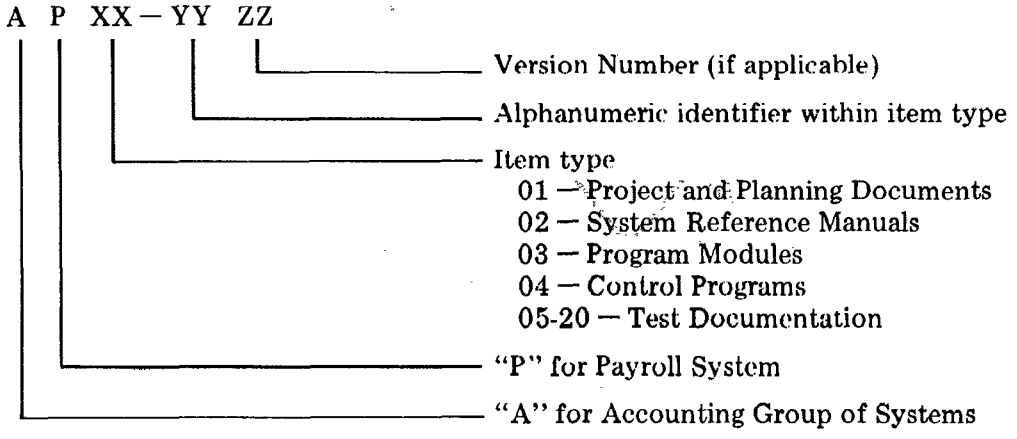
A1.1.2. Assumptions. The following assumptions were made when preparing this example:

(1) System testing activities assume that *module* and *integration* testing have been done. This implies that single program functionality has been comprehensively tested. System level testing, therefore, focuses on the testing of multiprogram functionality (for example, year-end processing) as well as external interfaces, security, recovery, and performance. In addition, operator and user procedures are tested.

(2) The payroll system will be system tested at only one site.

A1.1.3. Naming Conventions. The naming conventions which follow are used throughout the payroll-system example.

Corporate Payroll System



Project Planning Documents

AP01-01	Statement of Requirements
AP01-02	Preliminary Development Plan
AP01-03	Project Authorization
AP01-04	System Design Description
AP01-05	Business Plan
AP01-06	Final Development Plan
AP01-08	Quality Assurance Plan
AP01-09	Configuration Management Plan
AP01-12	Statement of Completion

System Reference Manuals

AP02-01	System Reference Manual
AP02-02	Operation Reference Manual

AP02-03	Module Reference Manual
AP02-04	User Transaction Reference Manual
Program Modules	
AP03-	Program Modules
Control Programs	
AP04-	Control Programs, Utilities, Sorts
Test Documentation	
AP05-YYZZ	Test Plan
AP06-YYZZ	Test Design Specification
AP07-YYZZ	Test Case Specification
AP08-YYZZ	Test Procedure Specification
AP09-YY	Test Log
AP10-00	Test Incident Report Log*
AP11-YY	Test Incident Report
AP12-YY	Test Summary Report
AP13-YY	Test Item Transmittal Report

*Note: This *test* document is not specified by this standard.

A.1.2

System Test Plan
for the
Corporate Payroll System

XYZ Corporation

AP05-0101

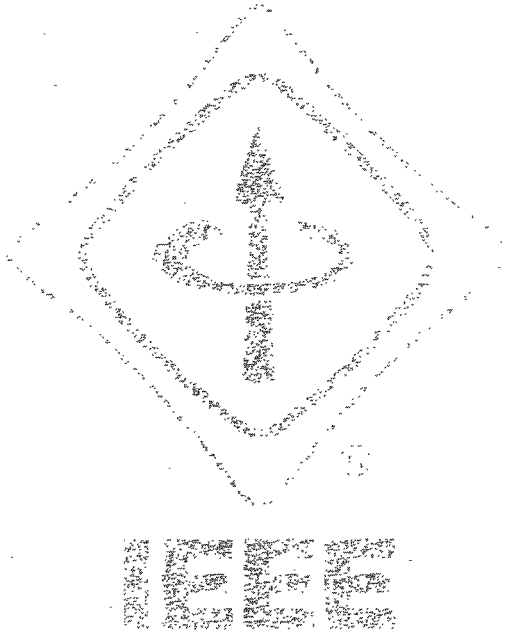
Prepared by
Manager, System Test Group
Manager, Corporate Payroll Department

January 21, 1982

System Test Plan
Corporate Payroll System

Contents

SECTION PAGE



1. Test Plan Identifier

AP05-0103

2. Introduction

2.1 Objectives. A system test plan for the corporate payroll system should support the following objectives.

(1) To detail the activities required to prepare for and conduct the system test.

(2) To communicate to all responsible parties the tasks which they are to perform and the schedule to be followed in performing the tasks.

(3) To define the sources of the information used to prepare the plan.

(4) To define the test tools and environment needed to conduct the system test.

2.2 Background. Last year the XYZ Corporate Systems and Programming Department developed a new General Ledger System at the request of the Corporate Accounting Department. A request was made at the same time for a new corporate payroll system to be developed which would interface with the general ledger system.

The Management Systems Review Committee approved the request for the payroll system in September of 1981 and named a corporate payroll system advisory group to decide on the system requirements. The group finished a Statement of Requirements (AP01-01) and a Preliminary Development Plan (AP01-02) in December, 1981.

2.3 Scope. This test plan covers a full systems test of the corporate payroll system. This includes operator and user procedures, as well as programs and job control. In addition to comprehensively testing multiprogram functionality, external interfaces, security, recovery, and performance will also be evaluated.

2.4 References. The following documents were used as sources of information for the test plan.

Corporate Payroll System Preliminary Development Plan (AP01-02)
Corporate Payroll System Authorization (AP01-03)
Corporate Payroll System Final Development Plan (AP01-06)
Corporate Payroll System Quality Assurance Plan (AP01-08)
Corporate Payroll System Configuration Management Plan (AP01-09)
XYZ Corporate Systems Development Standards and Procedures (XYZ01-0100)
Corporate General Ledger System Design Description (AG01-04)
Corporate General Ledger System Test Plan (AG05-01)

3. Test Items

All items which make up the corporate payroll system will be tested during the system test. The versions to be tested will be placed in the appropriate libraries by the configuration administrator. The administrator will also control changes to the versions under test and notify the test group when new versions are available.

The following documents will provide the basis for defining correct operation.

Corporate Payroll System Statement of Requirements (AP01-01)
Corporate Payroll System Design Description (AP01-04)
Corporate Payroll System Reference Manual (AP02-01)
Corporate Payroll System Module Reference Manual (AP02-03)

The items to be tested are:

3.1 Program Modules. The program modules to be tested will be identified as follows:

Type	Library	Member Name
Source Code	SOURLIB1	AP0302 AP0305

Executable Code	MACLIB1	AP0301
	SYSLIB1	AP0302
		AP0305

3.2 Job-Control Procedures. The control procedures for application programs, sorts, and utility programs will be identified as follows:

Type	Library	Member Name
Application Programs	PROCLIB1	AP0401
Sorts	PROCLIB1	AP0402
Utility Programs	PROCLIB1	AP0403

3.3 User Procedures. The online procedures specified in the Corporate Payroll Sytem User Transaction Reference Manual (AP02-04) will be tested.

3.4 Operator Procedures. The system test includes the procedures specified in the Corporate Payroll System Operation Reference Manual (AP02-02).

4. Features to be Tested

The following list describes the features that will be tested.

Test Design Specification Number	Description
AP06-01	Data base conversion
AP06-02	Complete payroll processing for salaried employees only
AP06-03	Complete payroll processing for hourly employees only
AP06-04	Complete payroll processing for all employees
AP06-05	Periodic reporting
AP06-06	General Ledger transaction building
AP06-07	Security
AP06-08	Recovery
AP06-09	Performance

5. Features not to be Tested

The following features will not be included in the system tests because they are not to be used when the system is initially installed.

Equal Employment Opportunity Commission Compliance Reports
Internal Training Schedule Reports
Salary/Performance Review Reports

The development Phase 2.0 documentation will contain a test plan for these features.

The test cases will not cover all possible combinations of options within the transaction or report being tested. Only combinations that are known to be required for current XYZ Corporate Payroll processing will be tested.

6. Approach

The test personnel will use the system documentation to prepare all test design, case, and proce-

dure specifications. This approach will verify the accuracy and comprehensiveness of the information in the documentation in those areas covered by the tests.

Personnel from the Payroll and Corporate Accounting Departments will assist in developing the test designs and test cases. This will help ensure that the tests represent the production use of the system.

In order to ensure privacy, all test data extracted from production files will have privacy sensitive fields changed.

6.1 Conversion Testing. In addition to counting the input and output records, the validity of the converted data base will be verified in two ways. The first verification method involves the use of a *data base auditor* which must be built by the development group. When run against the converted data base, the data base auditor will check value ranges within a record and the required relationships between records.

The second verification method involves the random selection of a small subset of old records and than a direct comparison against a corresponding subset of the new records. The number of direct comparisons, c, and the number of old records, r, must be specified. A set of c random numbers will be generated from the range 1 to r. This set will be sorted and used during the conversion process to drive the selection of records for direct comparison.

NOTE: This same two-pronged verification approach should be used during the actual conversion.

6.2 Job Stream Testing. A comprehensive set of records of salaried employees, hourly employees, and a merged set of these two should be used to test payroll processing. The standard job stream testing approach should be used.

Run each of the periodic reporting job streams at least once.

6.3 Interface Testing. In order to test the interface between the payroll and general-ledger systems, the payroll system will build a comprehensive set of general-ledger transactions. These transactions will then be input to the general-ledger test system. The resulting general-ledger entries must be extracted, printed, and compared with a printout of the general-ledger transactions prepared by the payroll system.

6.4 Security Testing. Attempted access without a proper password to the online data entry and display transactions will be tested.

6.5 Recovery Testing. Recovery will be tested by halting the machine during stand alone time and then following the recovery procedures.

6.6 Performance Testing. Performance will be evaluated against the performance requirements (AP01-01) by measuring the run times of several jobs using production data volumes.

6.7 Regression. It is assumed that several iterations of the system test will be done in order to test program modifications made during the system test period. A regression test will be performed for each new version of the system to detect unexpected impact resulting from program modifications.

The regression test will be done by running all of the tests on a new version that were run on the previous version and then comparing the resulting files. The standard comparator program, UT08-0100, will be used to compare all system outputs.

6.8 Comprehensiveness. Each of the system features described in the Corporate Payroll System Reference Manual (AP02-01) will have at least one associated test-design specification. Each of the user procedures specified in the Corporate Payroll-System User Transaction Reference Manual (AP02-04) will be tested at least once. Each of the operating procedures specified in the Corporate Payroll-System Operation Reference Manual (AP02-02) also will be tested at least once. In addition, each job control procedure will be executed at least once.

A coverage matrix will be used to related test-design specifications to each of the areas described above.

6.9 Constraints. A final implementation date of August 31, 1982 has been planned for the Corpo-

rate Payroll System. It will be necessary to meet this date because the new ABC Division begins full operation on September 1, and they must have this payroll system to pay their employees.

7. Item Pass/Fail Criteria

The system must satisfy the standard requirements for system pass/fail stated in the XYZ Corporate Systems Development Standards and Procedures (XYZ01-0100).

The system must also satisfy the following requirements:

Memory requirements must not be greater than 64K of real storage

Consistency of user procedures with other accounting systems must satisfy the Payroll Supervisor

8. Suspension Criteria and Resumption Requirements

8.1 Suspension Criteria. Inability to convert the Employee Information Data Base will cause suspension of all testing activities.

8.2 Resumption Requirements. When a new version of the system is transmitted to the test group after a suspension of testing has occurred, a regression test as described in 6.7 will be run.

9. Test Deliverables

The following documents will be generated by the system test group and will be delivered to the configuration management group after test completion.

Test Documentation:

System Test Plan
System Test Design Specifications
System Test Case Specifications
System Test Procedure Specifications
System Test Logs
System Test Incident Report Log
System Test Incident Reports
System Test Summary Report

Test Data:

(1) Copies of all data entry and inquiry screens and the reply screens are to be attached to the related test case document.

(2) Copies of the input and output test files should be delivered to the configuration management group.

(3) Microfiche copies of the printed output from the final execution of each test procedure are to be delivered to the configuration management group along with the test documentation.

10. Testing Tasks

See Task List, Attachment A, page 28.

11. Environmental Needs

11.1 Hardware. The testing will be done on the XYZ hardware configuration.

Since most testing must be done during prime operating hours, 3 on-line terminals must be available to the test group during this period.

11.2 Software

11.2.1 Operating System. The production operating system will be used to execute these tests.

11.2.2 Communications Software. All on-line programs will be tested under the control of the test communication software.

11.3 Security. Security will be limited to existing controls.

11.4 Tools. The following test tools are required to develop and evaluate the system tests.

(1) Test Data Generator (UT09-0200). This program will be used to generate the majority of the test data. It is located in the standard system library, SYSLIBA.

(2) Comparator Program (UT08-0100). This program will be used to compare system results during the regression tests. It is located in the standard system library, SYSLIBA.

(3) Data Base Auditor. This program audits value ranges and interrecord relationships in the data base. It must be supplied by the development group.

11.5 Publications. The following documents are required to support systems testing.

Corporate Payroll System Statement of Requirements (AP01-01)
Corporate Payroll System Design Description (AP01-04)
Corporate Payroll System Reference Manual (AP02-01)
Corporate Payroll Operation Reference Manual (AP02-02)
Corporate Payroll System Module Reference Manual (AP02-03)
Corporate Payroll System User Transaction Reference Manual (AP02-04)

12. Responsibilities

The following groups have responsibility for segments of the testing.

12.1 System Test Group. This group provides the overall management of the testing and the technical testing expertise.

12.2 Corporate Payroll Department. This group is the end user of the Corporate Payroll System and will provide assistance to the test group in the following activities:

Reviewing the test-design specifications.
Executing the on-line tests.
Checking output screens and reports.

12.3 Development Project Group. This group transmits the system to be tested and responds to the System Test Incident Reports. This group does any program debugging that is required. It also supplies the data-base auditor.

13. Staffing and Training Needs

13.1 Staffing. The following staff is needed to carry out this testing project.

13.1.1 Test Group.

Test Manager	1
Senior Test Analyst	1
Test Analysts	2
Test Technician	1

13.1.2 Payroll Department.

Payroll Supervisor	1

13.2 Training. The Corporate Payroll Department personnel must be trained to do the data entry transactions. The User Transaction Reference Manual (AP02-04) will be the basis of this training.

14. Schedule

See attached Task List (Attachment A).

Hardware, software, and test tools will be used for testing during the period from June 1, 1982 through August 1, 1982.

15. Risks and Contingencies

If the testing schedule is significantly impacted by system failure, the development manager has agreed to assign a full-time person to the test group to do debugging.

If one payroll supervisor is not sufficiently available for testing, then the payroll manager has agreed to identify a second supervisor.

If hardware problems impact system availability during the day, then the test group will schedule their activities during the evening.

The first production runs of the Corporate Payroll System must be checked out in detail before the payroll checks are distributed, and any checks in error must be corrected manually.

16. Approvals

Test Manager	Date
Development Project Manager	Date
Quality Assurance Manager	Date

Attachment —A. Task List

	Task	Predecessor Tasks	Special Skills	Responsibility	Effort	Finish Date
(1)	Prepare test plan.	Complete payroll system design description (AP01-04) and preliminary development plan (AP01-02)		Test manager Senior test analyst	4	01-21-82
(2)	Prepare test-design specifications.	Task 1	Knowledge of corporate payroll procedures	Senior test analyst	9	04-01-82
(3)	Prepare test-case specifcations.	Complete corresponding test designs (Task 2)		Test analyst	4	04-15-82
(4)	Prepare test-procedure specifications.	Complete corresponding test cases (Task 3)		Test analyst	6	05-15-82
(5)	Build the initial employee-information data base.	Task 4		Test analyst	6	06-01-82
(6)	Complete test-item transmittal and transmit the corporate payroll system to the test group.	Complete integration testing		Development project manager		06-01-82
(7)	Checkout all job-control procedures required to execute the system.	Task 6	Job control experience	Test technician	1	06-08-82
(8)	Assemble and link the corporate payroll system.	Task 6		Test technician	1	06-08-82
(9)	Execute data-entry test procedures.	Task 5 Task 8		Test analyst	1	06-22-82
(10)	Execute batch test procedures.	Task 5 Task 8		Test technician	3	06-30-82
(11)	Check out batch test results.	Task 10	Knowledge of payroll-report requirements	Test analyst	1	07-02-82
(12)	Resolve test-incident reports.	Task 9 Task 11		—Development group manager —System test-group manager —Corporate payroll department manager	2	07-16-82
(13)	Repeat tasks (6)—(12) until all test procedures have succeeded.	Task 12			2	07-30-82
(14)	Write the system test summary report.	Task 13		—System test-group manager —Corporate payroll-department manager	1	08-06-82
(15)	Transmit all test documentation and test data to the configuration management group.	Task 14		System test group	1	08-06-82

A1.3. Corporate Payroll System Test-Procedure Specification

1. Test-Procedure Specification Identifier

AP08-0101 March 5, 1982

2. Purpose

This procedure describes the steps necessary to perform the test specified in the test-design specification for data-base conversion (AP06-0101). The procedure describes the execution of the test case described in System Test-Case Specification AP07-0101. (NOTE: Neither the test-design specification nor test-case specification are included in this set of system test examples). This test will exercise the Employee Information Data Base Conversion Procedures specified in the Corporate Payroll System Reference Manual (AP02-01) and the conversion program (AP03-07) described in the Corporate Payroll System Module Reference Manual (AP02-03).

3. Special Requirements

In order to execute this procedure, the "random subset" program, the old data extract program, the new data extract program, and the data base auditor specified in AP06-0101 must be available.

4. Procedure Steps

4.1 Log. Record the execution of this procedure on a standard test log (AP09-YY).

4.2 Set Up

(1) Generate a test version of the old employee data base according to the test-case specification in AP07-0101 using the test data generator (UT09-0200).

(2) Execute the random subset program requesting 50 random numbers in the range 1 to 500.

(3) Sort the random number file into an increasing sequence.

(4) Execute the old data extract program with the test version of the old employee-information data base using the sorted random number file.

(5) Print the extracted records.

4.3 Proceed. Execute the conversion program with the test version of the old data base generating the new employee information data base.

4.4 Measure

(1) Execute the data-base auditor with the new employee information data base. Report violations in test-incident reports.

(2) Execute the new data extract program with the new data base using the sorted random-number file.

(3) Print the extracted records.

(4) Compare the extracted old records with the extracted new records. Report differences in test-incident reports.

4.5 Wrap Up. Delete both extracted files and the random number file.

A1.4. Corporate Payroll System Transmittal Report

1. Transmittal Report Identifier

AP13-03 June 24, 1982

2. Transmitted Items

A new version of the data conversion program (AP03-0702) is being transmitted.

The program is described in the Module Reference Manual (AP02-0305). The associated conversion procedures are specified in the System Reference Manual (AP02-0109). The transmitted program is associated with system test plan AP05-0103.

Communication about this program should be directed to the manager of the payroll system development project.

3. Location

The transmitted code is located as follows:

Source Code	SOURLIB1 (AP0307)
Object Code	SYSLIB1 (AP0307)

The system documentation and test plans are available in the documentation library.

4. Status

The conversion program has been fully retested at the unit and integration levels. The three incident reports (AP11-15, 16, and 17) generated by the June 10th execution of AP08-0101 are resolved by this new version.

The *invalid department code* messages (AP11-15) and the *blank home addresses* (AP11-16) resulted from insufficient logic in the conversion program. Additional logic was added. The *number of dependents* field processing problem (AP11-17) resulted from an imprecise program specification. The logic has been changed and comments have been added for clarity.

5. Approvals

Development Manager	Date
Test Manager	Date

A1.5 Corporate Payroll-System Test Log

1. Test Log Identifier

AP09-04 June 10, 1982

2. Description

The first version of the data conversion program (AP03-0701) is being tested. The program was transmitted (AP13-01) to the test group along with the entire payroll system.

This batch testing is being conducted using the standard corporate data-center facilities.

This log records the execution of the data conversion test procedure (AP08-0101). The tests are being submitted to background processing through a CRT by a senior test analyst.

3. Activities and Event Entries

June 10, 1982		Incidents
2:00 PM	— Dick J. started testing.	
2:15 PM	— Began to generate the old test data base.	
3:30 PM	— Discovered a possible bug in the test data generator. Filled out an incident report and worked around the problem.	AP11-14
6:00 PM	— Completed the old test data base generation. It is located on TEST1.	
6:15 PM	— Dick J. stopped testing.	

June 11, 1982		Incidents
9:45 AM	— Dick J. started testing.	
10:00 AM	— Began to create the random number file.	
10:45 AM	— Generated a sorted random number file.	
11:30 AM	— Selected and printed a random subset of records from the old test data base.	
12:30 PM	— Dick J. stopped testing.	
12:45 PM	— Jane K. started testing.	
1:00 PM	— Ran the conversion program against the old test data base. The new data base is on TEST2. The status report from the run contained 3 messages warning of invalid data in the department code field. The three records were checked and the values appeared valid. An incident report was generated.	AP11-15
3:30 PM	— Ran the data-base auditor against the new data base. The auditor reported multiple instances of blank home addresses. A check found these addresses nonblank in the old data base. The incident was reported.	AP11-16
4:00 PM	— Jane K. stopped testing.	

June 12, 1982

Incidents

8:15 AM — Jane K. started testing.

8:30 AM — Selected and printed the random subset of records from the new data base. In one case, the *number of dependents* field was changed from three to zero (possibly because no names were present). The incident was reported. AP11-17

11:30 AM — The extract and random number files were deleted.

11:45 AM — Jane K. stopped testing.

A1.6. Corporate Payroll System Test Incident Report

1. Report Identifier

AP11-17 June 12, 1982

2. Summary

Changes in the *number of dependents* field were found by comparing records from the new employee data base created by the conversion program (AP03-0701) with those from the old data base. Test log AP09-04 records this incident. The incident occurred during execution of test procedure AP08-0101.

3. Incident Description

June 12, 1982 8:30 AM Jane K.

A test version of the old employee data base was converted to its new format. The value in the *number of dependents* field was not expected to change during this process. This field value changed in the record indicated on the attached printouts.

Note that although the dependent count is three in the original record, none of the names appear. The number of names matches the count in all of the other records.

Perhaps the program is counting the names and forcing consistency.

4. Impact

Testing activity is suspended until this incident is resolved.

A2. Normalize Numeric Expression
Module-Test Documentation

The following example describes the testing of a module which reformats a numeric expression entered on a CRT. The module removes all commas, the sign, and the decimal point. It also checks the validity of the input expression.

A2.1 Introduction

General Requirements. To provide user-friendly entry of numeric data on a CRT, a system permits the keying of numeric expressions containing optional non-numeric symbols such as commas, a decimal point, and a leading sign. Any of the following examples would be valid entries:

+0
1234.
-.012
12,345.6

To facilitate editing of such input, a routine is required to normalize the numeric expression to a decimal point aligned value and to describe it. An expression is described by various characteristics such as:

Includes sign
Includes commas
Includes decimal point
Number of fractional digits and
Number of integer digits

A return code should identify the specific nature of any edit error.
The routine will be accessed by COBOL programs.

Functional Design.

Input: A character string of length 25 called NUMERIC-EXPRESSION contains a numeric expression. The expression must contain at least 1 digit. It may contain no more than 14 integer digits and no more than 4 fractional digits. It may contain valid combinations of

Leading sign
Decimal point and
Grouping commas.

A valid entry field may have spaces on the left, the right, or both. Interior spaces are invalid.

Process: The input expression is edited and if invalid an error condition is recorded in the return code. If valid, any signs, decimal points, and commas are removed and the resulting numeric value is decimal-point aligned in a signed field. In addition, a set of input descriptors is calculated.

Output: A decimal-point aligned, signed numeric value in a PIC S9(14)V9(4) field called ALIGNED-NUMERIC-VALUE

A set of input descriptors

INTEGER-DIGIT-COUNT	(0 - 14)
FRACTIONAL-DIGIT-COUNT	(0 - 4)
WAS-SIGN-FOUND	(N-0, YES)
WERE-COMMAS-FOUND	(N-0, YES)
WAS-DECIMAL-POINT-FOUND	(N-0, YES)

A RETURN-CODE with the following values

- NORMALIZATION-OK
- INVALID-FIRST-CHAR

First character is other than a digit, period, or sign

- INVALID-NONFIRST-CHAR

Nonfirst character is other than a digit, period, or comma

· NO-DIGIT-FOUND
No numeric character was entered
· TOO-MANY-INTEGER-DIGITS
More than 14 consecutive digits without a decimal point
· TOO-MANY-FRACTIONAL-DIGITS
More than 4 digits to the right of a decimal point
· TOO-MANY-DECIMAL-POINTS
More than 1 decimal point
· COMMA-RIGHT-AFTER-SIGN
Comma immediately follows a sign
· INVALID-COMMA-INTERVAL
Less than 3 consecutive-digits following a comma
More than 3 consecutive digits preceding or following a comma
· COMMA-AFTER-POINT
Comma appears to the right of a decimal point

If the value of RETURN-CODE is *not* NORMALIZATION-OK, then the values of the other output fields are *undefined.*

TECHNICAL DESIGN.

LANGUAGE: COBOL

ACCESS: PERFORM of included sub-routine

HIERARCHY: Normalize-Numeric-Exp
CHART
- Left-justify Expression
- Find Right-most Non-space
- Validate Expression
  - Initialize Descriptor Fields
  - Set Return OK
  - Do Validation Scan
  - Wrap Up Validation Scan
- Normalize Valid Expression
  - Save Digit
  - Delete Specials
  - Align Output Value
  - Establish Sign

NOTES:

Output Fields	Setting Procedures
Return Code (Error)	Do Validation Scan Wrap Up Validation Scan
Return Code (OK)	Set Return OK
Input Descriptors	Initialize Description Fields Do Validation Scan Wrap Up Validation Scan
ALIGNED-NUMERIC-VALUE	Align Output Value Establish Sign

Module Test Documentation for Normalize Numeric Expression

- Test Design Specification
- Test Case Specification
- Test Summary Report

Prepared by Module Developer
March 23, 1982

A2.2. Normalize Numeric Expression Module Test-Design Specification

1. Test-Design Specification Identifier

NNE.TD.01.05 15 March 1981

NOTE: No test plan is associated with this module, because its development was not associated with any particular application project (so there is no project level test plan) and because the special projects manager decided that a specific module test plan was unnecessary. The quality support manager concurred.

2. Features to be Tested

Individual Features
- 2.1 Digits Only Processing
- 2.2 Sign Processing
- 2.3 Decimal Point Processing
- 2.4 Commas Processing

Combinations
- 2.5 Sign and Decimal Point
- 2.6 Sign and Commas
- 2.7 Decimal Point and Commas
- 2.8 Sign, Decimal Point and Commas

All of these features are specified in the functional design description contained in the *common routines* section of the programmer's guide.

3. Approach Refinements

The individual processing features of the module will be tested first with valid and invalid input. All of the combinations will then be used.

A program will be written to drive the module. A file will be created with each record containing a single input value and fields to store the resulting values. The driver program will read a record, pass the corresponding input value to the module, store the resulting values in the record and rewrite it. The current version id of the module should be stored in each rewritten record.

Before testing begins, a test-case file will be generated in the same format as the driver file. The records will contain the input values along with the *predicted* resulting values. Following a test run, the driver file will be compared with the case file. The file comparison utility program will report any differences.

Since generation of all possible input values is impractical, test-set comprehensiveness will be evaluated based upon the following criteria:

(1) Requirements coverage — has each of the requirements been satisfied?
(2) Design coverage — has each of the functional design specifications been satisfied?
(3) Domain coverage — has each of the input constraints (for example, maximum of one decimal point) been tested? Have representative values been included? Have all error messages been generated?
(4) Branch coverage — has every branch been taken at least once?
(5) Statement coverage — has every statement been executed at least once?

Appropriate checklists will be generated to evaluate criteria 1-3. Existing code instrumentation tools will be used to evaluate 4 and 5.

The test set must satisfy each component of the five criteria specified above at least once.

Test Case Selection Rationale

Input constraints

(1) No more than 14 integer digits
(2) No more than 4 fractional digits
(3) No more than one decimal point
(4) Between 1 and 3 contiguous digits to the left of each comma
(5) Exactly 3 contiguous digits to the right of each comma
(6) No commas after the decimal point

There are no relevant internal or output constraints.

Common Test-Case Characteristics

All test cases require a module driver.

4. Test Identification

Cases

Digits Only	
Valid	
14 integer digits	NNE.TC.001
centered 6 integer digits	NNE.TC.002
left justified 1 integer digit	NNE.TC.003
Invalid	
15 integer digits	NNE.TC.010
digit string with imbedded space	NNE.TC.011
digit string with leading invalid character	NNE.TC.012
digit string with imbedded invalid character	NNE.TC.013
digit string with trailing invalid character	NNE.TC.014
Sign	
Valid	
right justified + signed 14 integers	NNE.TC.020
— signed integers	NNE.TC.021
Invalid	
imbedded sign	NNE.TC.030
trailing sign	NNE.TC.031
sign alone without digits	NNE.TC.032
2 leading signs	NNE.TC.033
2 separated signs	NNE.TC.034
Decimal Point	
Valid	
leading point with 4 fractional digits	NNE.TC.040
embedded point with 1 fractional digit	NNE.TC.041
trailing point with 14 integers	NNE.TC.042

Cases	
Invalid	
5 fractional digits	NNE.TC.050
2 points	NNE.TC.051
point without digits	NNE.TC.052
Commas	
Valid	
1 comma	NNE.TC.060
4 commas with 14 integer digits	NNE.TC.061
Invalid	
leading comma	NNE.TC.070
4 digits to left of a comma	NNE.TC.071
2 digits to right of a comma	NNE.TC.072
4 digits to right of a comma	NNE.TC.073
trailing comma	NNE.TC.074
comma without digits	NNE.TC.075
15 integer digits	NNE.TC.076
Sign and Decimal Point	
Valid	
sign and trailing point with 1 digit	NNE.TC.080
sign adjacent to point with 1 digit	NNE.TC.081
sign and point with 14 digits	NNE.TC.082
Invalid	
sign and point without digits	NNE.TC.090
Sign and Commas	
Valid	
sign and comma with 14 digits	NNE.TC.100
sign and comma with 4 digits	NNE.TC.101
Invalid	
sign adjacent to comma	NNE.TC.110
Decimal Point and Commas	
Valid	
comma with 14 integer digits and 4 fractional digits	NNE.TC.120
one comma with 4 digits and trailing point	NNE.TC.121
Invalid	
no digits between comma and point	NNE.TC.130
4 digits between comma and point	NNE.TC.131
comma following point	NNE.TC.132
Sign, Decimal Point and Commas	
Valid	
longest valid expression	NNE.TC.140
shortest valid expression	NNE.TC.141
representative valid expression	NNE.TC.142
Invalid	
15 integer and 4 fractional digits	NNE.TC.150
14 integer and 5 fractional digits	NNE.TC.151

Procedures. There are no *formal* test procedures associated with this design.
The procedure for using the module driver is in the *test tools* section of the programmer's guide.

5. Feature Pass/Fail Criteria

Each feature must pass all of its test cases in order to pass this test.

A2.3. Normalize Numeric Expression Module Test-Case Specification

1. Test Case Specification Identifier

NNE.TC.121.01 17 March 1981
One comma with 4 digits and trailing point.

2. Test Items

Normalized Numeric Expression Subroutine — This routine strips signs, commas, and decimal points from numeric expressions.

The requirements, functional design, and technical design specifications are contained in the *common routines* section of the programmer's guide.

3. Input Specifications

1,234. in NUMERIC-EXPRESSION

4. Output Specifications

+12340000 in ALIGNED-NUMERIC-VALUE
NORMALIZATION-OK in RETURN-CODE
4 in INTEGER-DIGIT-COUNT
0 in FRACTIONAL-DIGIT-COUNT
N-0 in WAS-SIGN-FOUND
YES in WERE-COMMAS-FOUND
YES in WAS-DECIMAL-POINT-FOUND

5. Environmental Needs

A module driver is required to execute this case.

6. Special Procedural Requirements

The procedure for using the module driver is in the *test tools* section of the programmer's guide.

7. Intercase Dependencies

None.

A2.4. Normalize Numeric Expression Module Test Summary Report

1. Test Summary Report Identifier

NNE.TS.01 23 March 1981

2. Summary

After correcting three faults, the Normalize Numeric Expression Module (Revision 5) passed all tests. The routine was tested using a module driver.

The following test documents are associated with this module.

(1) Module Test Design Specification
NNE.TD.01.05
(2) Module Test Case Specifications
NNE.TC.001 — .151

3. Variances

Conditions identified during testing resulted in enhancements to the set of invalid conditions described in the original functional design. This in turn resulted in the specification of eleven additional test cases. All of these changes are included in the current documentation.

4. Comprehensiveness Assessment

The attached (but not included with example) checklists and execution trace reports demonstrate that the minimum comprehensiveness requirements specified in the test design specification have been satisfied.

5. Summary of Results

Three of the test cases (071, 073, and 131) exposed faults involving insufficient logic. Additional logic was added, some new test cases were defined and the test set was rerun. All features passed their tests.

6. Evaluation

The module passed comprehensive testing with only three faults being detected. No more than one additional fault in the first six months of use is specified.

7. Summary of Activities

	Est	Actual
Begin Testing 03/12/82		
Test Design (including cases)	2.0 days	3.0 days
Module Driver Development	1.0 days	1.5 days
Test Execution	2.0 days	2.0 days
Module Revision	2.0 days	1.5 days
Test Reporting	0.5 days	0.5 days
End Testing 03/23/82	7.5 days	8.5 days

8. Approvals

_____ _____

Development Project Manager Date

Appendix B
Implementation and Usage Guidelines

B1. Implementation Guidelines

When the standard is adopted by an organization, it is recommended that it be implemented in phases.

(1) Initial Phase. Begin by introducing the planning and reporting documents. The test plan will provide a foundation for the whole testing process. The reporting documents will encourage the testing organization to record the appropriate data in an organized manner.

Begin by implementing test documentation at the system level. The need for rigor and control during system testing is critical. System test documentation is a key element in meeting this need.

(2) Subsequent Phases. Introduce the balance of the documents in subsequent phases. Their sequence of introduction will depend upon the results of prior phases.

The test documentation eventually will form a document hierarchy corresponding to the design hierarchy, that is, system test documentation, subsystem test documentation, and module test documentation.

B2. Additional Test-Documentation Guidelines

Develop guidelines for the documentation of the specific testing techniques used in your organization (for example, code inspections or simulation). This documentation will supplement the basic documents of the standard.

B3. Usage Guidelines

(1) In the project plan or the organization's standards, identify which test documents are required during which testing activities. Provide guidelines for using these documents in your organization.

Figure B1 (below) is an example of a specification for the test documents required for various testing activities. The amount of documentation required will vary from organization to organization.

(2) Add sections and material within sections in order to tailor each document to a particular test item and a particular test environment.

(3) Consider documenting sets of modules at the module test level. For example, if might be useful to develop a module test design specification for a set of modules which generate reports. While different test cases would be required, a common test procedure specification might be appropriate.

Documents / Activities	Test Plan	Test Design Spec	Test Case Spec	Test Proc Spec	Test Item Trans Report	Test Log	Test Incident Report	Test Summary Report
Acceptance	X	X	X	X	X		X	X
Field	X	X			X		X	X
Installation	X	X	X	X	X		X	X
System	X	X	X	X	X	X	X	X
Subsystem		X	X	X	X	X	X	X
Program		X	X					X
Module		X	X					X

Fig B1
Example of a Required Test Documentation Specification

Appendix C
Related Documentation Standards

Several other standards which relate to test documentation are described below. Related testing standards can be found in Appendix D.

C1. National Standards

These references establish nationally recognized requirements for software documentation.

ANSI N413-1974, American National Standard Guidelines for the Documentation of Digital Computer Programs.[1]

ANSI/IEEE Std 100-1977, IEEE Standard Dictionary of Electrical and Electronics Terms.[2]

ANSI/IEEE Std 730-1981, Standard for Software Quality Assurance Plans.

IEEE Std 729-1983, IEEE Standard Glossary of Software Engineering Terminology.

Guide for Technical Documentation of Computer Projects, American National Standards Institute Technical Committee X3. Technical Report no 6, New York: June 1982.[3]

C2. Federal Standards

These references include standards and guidelines adopted for use by Federal agencies.

Guidelines for Documentation of Computer Programs and Automated Data Systems. Federal Information Processing Standards Pub 38. National Bureau of Standards, (FIPS PUB 38), February 1976.

C3. Military Standards

These references include standards and guidelines which may be invoked in Department of Defense contracts.

Automated Data Systems Documentation Standards, Standard 7935.1-S, Department of Defense, Sept 1977.

Tactical Digital Systems Documentation Standards SECNAVINST 3560.1, Department of the Navy, 1974.

WWMCCS CCTC Test Package Development Guidelines, Technical Memorandum TM 241-80, Defense Communications Agency, Nov 1980.

[1] ANSI standards are available from the Sales Department of American National Standards Institute, 1430 Broadway, New York, NY 10018.

[2] IEEE standards are available from the Institute of Electrical and Electronics Engineers, IEEE Service Center, 445 Hoes Lane, Piscataway, NJ 08854.

[3] This document is available from ANSI X3 Secretary, CBEMA, 311 1st Street NW, Suite 500, Washinton, DC 20001.

Appendix D
Annotated Bibliography

This bibliography contains two kinds of references. One set of references identifies a set of basic sources on software verification and testing. The objective is to identify a few basic references on testing for those who wish information on techniques and tools. The second set of references identifies some military testing standards which might serve as source material for those developing testing standards within their organization. These standards focus on testing rather than documentation.

D1 Basic Sources

ADRION, W.R., BRANSTAD, M.A. and CHERNIAVISKY, J.C. *Validation, Verification and Testing of Computer Software.* National Bureau of Standards: NBS Special Pub 500-75 1980.

This survey discusses testing and analysis techniques that can be used to validate software. Verification throughout the development process is stressed. Specific tools and techniques are described.

BARRY, M. *Airborne Systems Software Acquisition Engineering Guidebook for Software Testing and Evaluation.* TRW: 30323-6011-TU-00 1980.

This guidebook describes the planning and testing activities necessary to a successful software testing effort. It presents checklists and references to supplemental information in government documents and to summarized data from professional journals and books.

DEUTSCH, M.S. *Software Verification and Validation.* Prentice-Hall: 1982.

This book describes a testing methodology based upon techniques used at the Hughes Aircraft Company. It describes the application of automated verification systems and summarizes the verification activities used throughout the software life cycle. It provides insight into one effective approach to dealing with the verification of large software systems.

GLASS, R. *Software Reliability Guidebook.* Prentice-Hall: 1979.

This book defines reliability as the degree to which a software system both satisfies its requirements and delivers usable services. The book surveys many methodologies that are supposed to increase software reliability. These methodologies span the software life cycle requirements, design, implementation, checkout, and maintenance. The handbook rates each methodology as to its value, as a function of the degree of reliability required for the software program. The book is written for the practitioner but includes an annotated bibliography at the end of each chapter for readers desiring a more academic discussion of the topics. The section devoted to the checkout phase (testing) contains almost as much material as all the other life cycle sections combined. Test plans, procedures and reports are briefly described.

GUNTHER, R. *Management Methodology for Software Product Engineering.* John Wiley and Sons: Wiley-Interscience, 1978.

This book was selected for inclusion from the many management oriented books on software development because of its chapter devoted to managing the software product test. This chapter maps the product test groups activities into the traditional software life cycle phases. The chapter discusses software test documentation and even presents a table of contents for test plans and test logs. The author writes from his experience as a product planning manager for Amdahl Corporation.

HETZEL, W.C. Ed. *Program Test Methods.* Prentice-Hall: 1973.

This book is a collection of papers from a testing symposium held in 1972. It covers testing concepts, design of programs to facilitate testing, design of languages to facilitate testing, testing mathematical software, and testing large software systems. The book contains a comprehensive bibliography.

MILLER, E. and HOWDEN, W.E. *Tutorial: Software Testing and Validation Techniques (2nd Ed).* IEEE Computer Society Press: Catalog no EHO 180-0, 1981.

This IEEE Tutorial is a collection of papers that represent new developments in program structure analysis, test coverage, test results analysis, test project management techniques, and test tools. The publication also includes an extensive bibliography on testing.

MYERS, G. *The Art of Software Testing.* Wiley — Interscience, John Wiley and Sons: 1979.

This book is based upon material from Software Reliability: Principles and Practices by the same author. The book emphasizes testing as an activity which tries to find errors in a program, not an activity that attempts to show that a program works. The book expands this thesis into a set of testing principles. Testing techniques and methodology are covered along with a survey of existing tools. Test documentation is not addressed. The book does a good job of defining testing terms and classifying testing techniques.

PERRY, W. *Effective Methods of EDP Quality Assurance.* Wellesly Massachusetts: QED Systems, 1977.

This handbook contains the following four sections: The Quality Assurance Function, Planning The Quality Assurance Function, Quality Assurance Reviews, and Relationships and Other QA Tests. The appendix presents a sample software QA manual. Software testing is presented as one activity requiring a QA review. The handbook presents what points should be addressed by a detailed test plan as well as the test result report. The handbook's strength is its broad coverage of all the aspects of a well organized QA group. The book addresses software testing as an important activity in software development that should be reviewed by a QA function.

POWELL, P.B. Ed. *Planning for Software Validation, Verification and Testing.* National Bureau of Standards: NBS Special Pub 1982.

The document is for those who direct and those who implement computer projects. It explains the selection and use of validation, verification, and testing (VV and T) tools and techniques. It explains how to develop a plan to meet specific software VV and T goals.

POWELL, P.B. Ed. *Software Validation, Verification, and Testing Technique and Tool Reference Guide.* National Bureau of Standards: NBS Special Pub 1982.

Thirty techniques and tools for validation, verification, and testing (VV and T) are described. Each description includes the basic features of the technique or tool, the input, the output, an example, an assessment of the effectiveness and of the learning time and training, an estimate of the needed resources, and references.

WOOLRIDGE, S. *Systems and Programming Standards.* New York: Petrocelli/Charter, 1977.

The author's major purpose is to outline the contents of a programming standards manual. Chapter 8 addresses system testing and presents the table of contents for a system test plan as well as a checklist for system testing. The presentation differs from the other references because of the book's focus on standards and procedures as contrasted to testing philosophies and software QA techniques. The book shows that testing procedures can be integrated into general software development procedures.

YEH, R. *Current Trends in Programming Methodology.* vol II Program Validation. Prentice-Hall, 1977.

This book is a selection of papers published during the first half of the seventies. The collection of papers is indexed and the volume also includes an extensive annotated bibliography. Test documentation is not covered but the book provides a reference point to theoretical treatments of software testing.

D2. Military Testing Standards

Acquisition and Support Procedures for Computer Resources in Systems. AF Regulation 800-14, vol 2, Department of the Air Force, 1975.

The volume defines the Air Force procedures for the acquisition of computer programs used in military systems. It defines a software acquisition life cycle. In chapter four, it defines the required types, levels and phases of software testing and their relationship to this life cycle. It specifies policy for software testing at the following levels: informal testing, preliminary qualification testing, formal qualification testing, and system level testing. This volume identifies the required contents for test plans and procedures and it also discusses general methods for computer program verification and validation.

Testing of Computer Software Systems. Technical Bulletin, TB 18-104, Department of the Army, September 1981.

This technical bulletin defines the Army's computer software testing methodology. It

identifies required tests and test documentation. It also identifies a variety of participants in the testing process and their responsibilities.

Weapon System Software Development. MIL-STD-1679. Department of the Navy, 1978.

This specification requires quality assurance procedures at each stage of development to validate accuracy, correctness, and performance of the product programs and to verify the accuracy and conformance of program documentation. In the area of software trouble reporting, the specification requires the development and implementation of procedures for handling and reporting software problems. Section 5.8 describes various testing requirements.