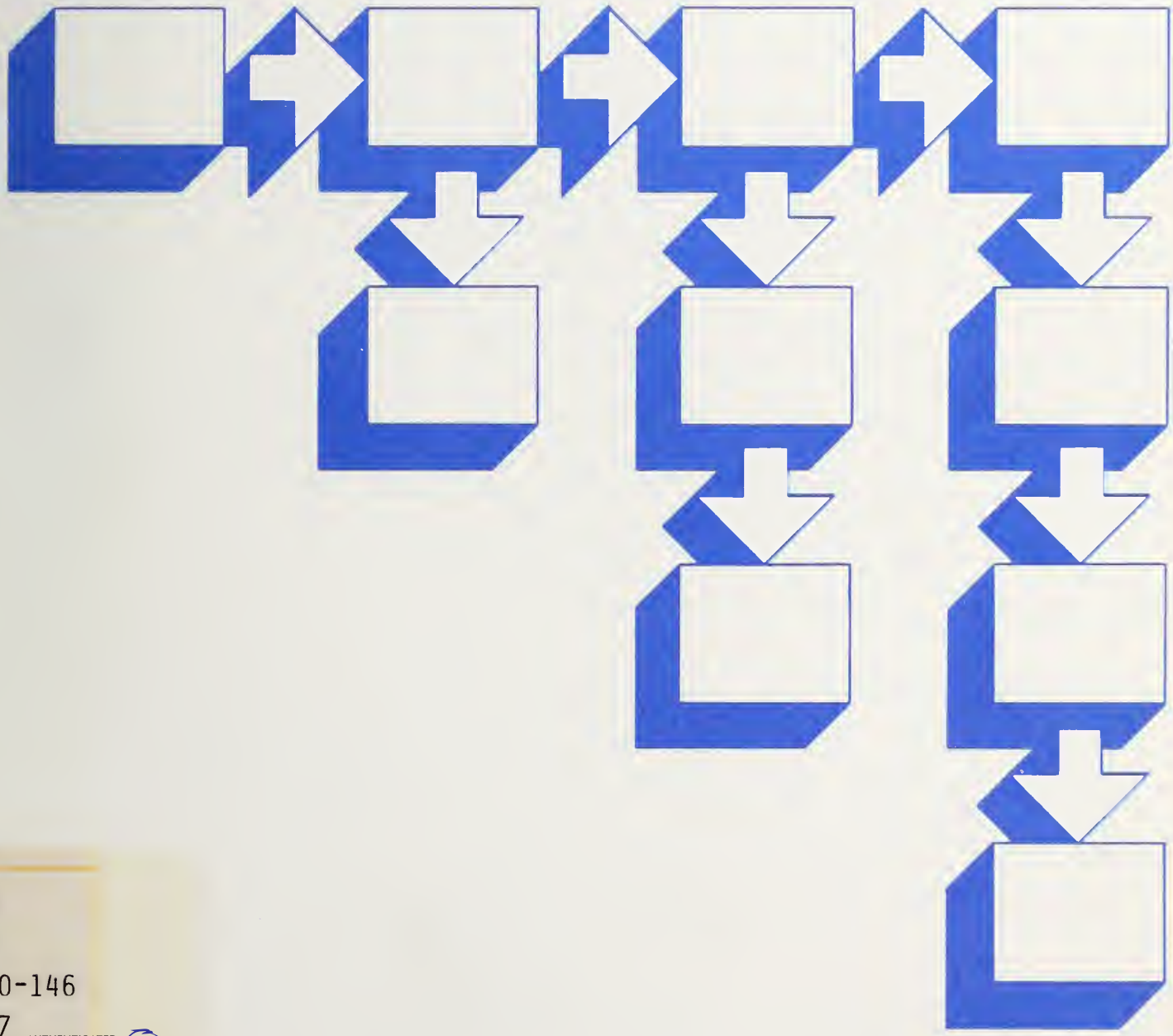
Computer Science and Technology

NBS Special Publication 500-146

Report on the NBS Software Acceptance Test Workshop April 1-2, 1986

Dolores R. Wallace and John C. Cherniavsky



The National Bureau of Standards[1] was established by an act of Congress on March 3, 1901. The Bureau's overall goal is to strengthen and advance the nation's science and technology and facilitate their effective application for public benefit. To this end, the Bureau conducts research and provides: (1) a basis for the nation's physical measurement system, (2) scientific and technological services for industry and government, (3) a technical basis for equity in trade, and (4) technical services to promote public safety. The Bureau's technical work is performed by the National Measurement Laboratory, the National Engineering Laboratory, the Institute for Computer Sciences and Technology, and the Institute for Materials Science and Engineering.

The National Measurement Laboratory

Provides the national system of physical and chemical measurement; coordinates the system with measurement systems of other nations and furnishes essential services leading to accurate and uniform physical and chemical measurement throughout the Nation's scientific community, industry, and commerce; provides advisory and research services to other Government agencies; conducts physical and chemical research; develops, produces, and distributes Standard Reference Materials; and provides calibration services. The Laboratory consists of the following centers:

- Basic Standards[2]
- Radiation Research
- Chemical Physics
- Analytical Chemistry

The National Engineering Laboratory

Provides technology and technical services to the public and private sectors to address national needs and to solve national problems; conducts research in engineering and applied science in support of these efforts; builds and maintains competence in the necessary disciplines required to carry out this research and technical service; develops engineering data and measurement capabilities; provides engineering measurement traceability services; develops test methods and proposes engineering standards and code changes; develops and proposes new engineering practices; and develops and improves mechanisms to transfer results of its research to the ultimate user. The Laboratory consists of the following centers:

- Applied Mathematics
- Electronics and Electrical Engineering[2]
- Manufacturing Engineering
- Building Technology
- Fire Research
- Chemical Engineering[2]

The Institute for Computer Sciences and Technology

Conducts research and provides scientific and technical services to aid Federal agencies in the selection, acquisition, application, and use of computer technology to improve effectiveness and economy in Government operations in accordance with Public Law 89-306 (40 U.S.C. 759), relevant Executive Orders, and other directives; carries out this mission by managing the Federal Information Processing Standards Program, developing Federal ADP standards guidelines, and managing Federal participation in ADP voluntary standardization activities; provides scientific and technological advisory services and assistance to Federal agencies; and provides the technical foundation for computer-related policies of the Federal Government. The Institute consists of the following centers:

- Programming Science and Technology
- Computer Systems Engineering

The Institute for Materials Science and Engineering

Conducts research and provides measurements, data, standards, reference materials, quantitative understanding and other technical information fundamental to the processing, structure, properties and performance of materials; addresses the scientific basis for new advanced materials technologies; plans research around cross-country scientific themes such as nondestructive evaluation and phase diagram development; oversees Bureau-wide technical programs in nuclear reactor radiation research and nondestructive evaluation; and broadly disseminates generic technical information resulting from its programs. The Institute consists of the following Divisions:

- Ceramics
- Fracture and Deformation[3]
- Polymers
- Metallurgy
- Reactor Radiation

[1] Headquarters and Laboratories at Gaithersburg, MD, unless otherwise noted; mailing address Gaithersburg, MD 20899.
[2] Some divisions within the center are located at Boulder, CO 80303.
[3] Located at Boulder, CO, with some elements at Gaithersburg, MD.

Computer Science and Technology

NBS Special Publication 500-146

Report on the NBS Software Acceptance Test Workshop April 1-2, 1986

Dolores R. Wallace

John C. Cherniavsky[1]

Center for Programming Science and Technology
Institute for Computer Sciences and Technology
National Bureau of Standards
Gaithersburg, Maryland 20899

[1] Dr. Cherniavsky is also Chair, Department of Computer Science, Georgetown University, Washington, DC 20057

March 1987

U.S. DEPARTMENT OF COMMERCE
Malcolm Baldrige, Secretary

National Bureau of Standards
Ernest Ambler, Director

Reports on Computer Science and Technology

The National Bureau of Standards has a special responsibility within the Federal Government for computer science and technology activities. The programs of the NBS Institute for Computer Sciences and Technology are designed to provide ADP standards, guidelines, and technical advisory services to improve the effectiveness of computer utilization in the Federal sector, and to perform appropriate research and development efforts as foundation for such activities and programs. This publication series will report these NBS efforts to the Federal computer community as well as to interested specialists in the academic and private sectors. Those wishing to receive notices of publications in this series should complete and return the form at the end of this publication.

National Bureau of Standards Special Publication 500-146
Natl. Bur. Stand. (U.S.), Spec. Publ. 500-146, 50 pages (Mar. 1987)
CODEN: XNBSAV

Library of Congress Catalog Card Number: 87-619806

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1987

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington DC 20402

ABSTRACT

This document is a report on the Software Acceptance Test Workshop held at the National Bureau of Standards, April 1-2, 1986. The workshop consisted of eight sessions divided over two days. The topics of the first day's sessions were acceptance testing of off-the-shelf software, test case selection techniques, automated support for software acceptance testing, and software acceptance criteria. The topics of the second day's sessions were the management of software acceptance testing, standardization issues in software acceptance testing, research areas for software acceptance testing, and the state of practice in software acceptance testing. This report describes the charges given to all of the sessions, highlights of discussions from each of the sessions, and the conclusions of the workshop. This report is intended for those who purchase, market, develop or maintain software and for those who are responsible for software acceptance testing.

KEYWORDS

automated tools; custom software; management; off-shelf software; research; software acceptance criteria; software acceptance testing; standardization; test case selection; test planning; test practices.

TABLE OF CONTENTS

LIST OF FIGURES

1.0 INTRODUCTION

This is a report on the Software Acceptance Testing Workshop held at the National Bureau of Standards, April 1-2, 1986. The workshop was held to provide a forum for discussion by researchers and practitioners in Government, industry and academia on topics related to software acceptance testing. The attendees were asked to assess four general areas and four specific topics regarding software acceptance testing. The general areas were current state of practice, research, standardization, and management. The specific areas were test case selection techniques, software acceptance criteria, automation of software acceptance test, and software acceptance testing of off-the-shelf software (hereafter referred to as off-shelf software) vs custom software.

On the first day of the workshop, groups addressed specific issues related to software test case selection, criteria for judging acceptance of the software, tools for aiding in the acceptance of software, and special problems related to the acceptance of off-shelf software. The leader of each group reported on the conclusions of the group at the end of the day.

On the second day of the workshop, groups again formed to address more general themes of standards, research, state-of-practice, and management as they relate to software acceptance testing. Participants were from each of the four groups that had met on the previous day. At the end of the workshop, the leaders of the second day's groups presented their conclusions.

The workshop coordinators intentionally charged the participants with more material than could reasonably be dealt with in two days. They hoped to stimulate selections of topics that were of most interest to the individual groups. Before the workshop, participants received a NBS publication on software acceptance testing [NBS136] and other materials describing the topics for the workshop. These documents, though not binding, provided the participants with a common starting point for discussion. Central to discussions within individual sessions was definition of software testing terms. Most sessions defined software acceptance testing, specific tools, procedures, and other terms according to each group's experience.

This workshop report contains background information concerning the workshop, highlights of the workshop group discussions and their conclusions, and a summary of the workshop accomplishments. This report was written by the editors from the materials provided by the moderators and recorders of the eight groups. Opinions, definitions, and conclusions are those of the participants and do not necessarily represent the opinions, definitions, and conclusions of the National Bureau of Standards or the Federal Government.

ACKNOWLEDGMENT

The Software Acceptance Test Workshop served as a forum for the discussion of software acceptance testing. All attendees were contributors to the outcome of their sessions. The efforts of this group over two days resulted in the approaches to software acceptance testing described in this report. The moderators successfully combined their leadership abilities and their knowledge of software engineering to unite a diverse group from universities, government, and industry. Several attendees presented results of their experiences in software acceptance testing. Without the session recorders, we might not have been able to compile this report. We are grateful to the all participants; their names are provided in the appendix at the end of this report. The names of the session moderators, recorders and presenters are presented below.

WELCOME

o James Burrows, Director, Institute for
Computer Sciences and Technology

KEYNOTE ADDRESS

o W. Richards Adrion, Deputy Director of Computer Research,
National Science Foundation[1]

MODERATORS

o Jerry Raveling, Sperry Corporation
o Elaine Weyuker, Courant Institute of Mathematical Sciences, NYU
o Ronnie J. Martin, Georgia Institute of Technology
o Samuel T. Redwine, Jr., Institute for Defense Analyses
o W. Richards Adrion, National Science Foundation
o Keiji Tasaki, NASA Goddard Space Flight Center
o Lori Clarke, University of Massachusetts
o William H. Farr, Naval Surface Weapons Laboratory

RECORDERS

o Philip Marriott, Computer Technology Associates, Inc.
o Sheila Frankel, National Bureau of Standards
o D. Richard Kuhn, National Bureau of Standards
o Mike Slingluff, Federal Home Loan Mortgage Corporation
o Charles Eater, Veterans Administration
o Millie Ingels, Department of Commerce
o Mark Palmer, National Bureau of Standards

PRESENTERS

o Peggy Brouse, Mitre Corporation
o Nander Brown, Nander Brown & Company
o Vincent Dell'Orto, Internal Revenue Service
o Bill Dupras, NESTAR Systems
o Richard Fath, Federal Communication Commission
o Jean Philippe Favreau, National Bureau of Standards
o Dave Gelperin, Software Quality Engineering
o Phil Marriott, Computer Technology Associates
o Dan Schneider, Department of Justice
o Dave Siefert, NCR
o Mike Slingluff, Federal Home Loan Mortgage Co.
o Lou Smith, Bell Communications Research

[1] Dr. Adrion is currently Department Chair, Computer and Information Science Department, University of Massachusetts.

2.0 BACKGROUND

The Software Acceptance Test Workshop, April 1-2, 1986, was sponsored by the Institute for Computer Sciences and Technology of the National Bureau of Standards. ICST has responsibility under Public Law 89-306, also known as the "Brooks Act", to develop standards and guidelines for effective management and use of automatic data processing (ADP) by the Federal Government.

One particularly critical area in ICST's program is software engineering, that is, the engineering of software development and maintenance. The primary focus of the software engineering activities at NBS has been to develop and identify methods, techniques, tools and systems that can lead to the production and procurement of quality software and systems by the Federal Government. To assist agencies in this area, ICST has sponsored workshops and conferences and has issued Federal Information Processing Standard (FIPS) publications, reports, and guidelines on software engineering topics, including validation, verification and testing (VV&T), documentation, and software tools and environments.

Currently, the software engineering program has projects on software management, software requirements and design specifications, software acquisition with emphasis on reuse, software engineering environments, software maintenance, and VV&T. One area of VV&T is software acceptance testing. The Federal Government spends about $15 billion per year on computers and related equipment and services; this sum includes the acquisition and continued operation of many computer software applications. How well these computer software applications will perform is of significant concern to the Federal Government.

Software acceptance testing is usually the final opportunity to locate problems before the software system is accepted by the customer. After this point, depending on contractual arrangements, the vendor may have no further responsibility for the software. When acceptance testing has not been performed, or when it has failed to uncover a major problem or omission among the deliverables, an organization may suffer financial losses and operational losses from which it may be difficult to recover.

Software acceptance testing helps assure software customers that:

- the software system will perform as expected, and

- the collection of software deliverables will provide the information necessary
  to operate the software, to maintain the software, and
  to reuse the software in other systems.

These requisite assurances of software acceptance testing reflect changing approaches to software acquisition. In the earliest days of computers, almost all software acquisition involved a purchase of a custom designed software system for a single customer. Code, with perhaps some documentation, was thought of as the total product. Over time, customers have recognized that to use software productively in their environment more than code must pass acceptance. At a minimum, training and user documentation are part of the software deliverables for acceptance testing. Customers also have recognized that the long operative life of most software requires continuous software maintenance. And, finally, customers, especially in large organizations, are seeking ways to reuse part, or all, of a software system in other systems in their organizations. For these purposes, many other items are included in the software deliverables for

acceptance testing. In recent years, customer purchases of off-shelf packages, or packages modified to their specifications, may require somewhat different approaches to software acceptance testing.

Traditional custom acquisition and its accompanying software acceptance testing are fairly well understood. Yet, better methods and supporting tools are needed for complete, successful acceptance testing. Changing approaches to software acquisition may also require changing approaches to software acceptance testing. Even within the custom purchase, the criteria for acceptance go beyond functional specifications and include such criteria as quality and performance features. For these, additional techniques and tools are necessary. As the software market continues to change, approaches to software acceptance testing may have to change to fit the needs of the software customer.

In summary, software acceptance testing is the final opportunity for software customers to examine their software products before they accept them from the vendor and before they expect their organizations to use them. For some software (e.g., off-shelf purchases) acceptance testing may provide the only opportunity to verify that the software will satisfy customer requirements. Software acceptance testing is a complex process that often is not allowed enough time and resources. Management and technical concerns of software acceptance testing need further definition so that the assurances of software acceptance testing may be a reality for software customers.

NBS has held workshops, such as this one on software acceptance testing, to define specific software engineering topics in terms of:

- o the areas where research is needed;
- o the current state of practice and problems within that practice;
- o the problems with transferring concepts and methodologies to wide practice;
- o the applicability of existing standards;
- o the methodologies requiring further development before standardization;
- o the features ready for inclusion in guidelines or standards;
- o the approaches for transferring technical recommendations into practice.

ICST is especially interested in helping Federal agencies to identify their needs for software acceptance testing and to find ways to achieve them. The purpose of this workshop was to provide a forum for discussion of management and technical concerns to assist ICST in gaining a better understanding of the software acceptance test process and to explore approaches for improvement. The goal is twofold:

- o to provide information that is immediately useful to the participants; and
- o to support NBS's research program in software engineering.

The workshop was structured to lead to a synergy of views that will aid in defining principles underlying both problems and successes in software acceptance testing. The participants consisted of both researchers and practitioners in software acceptance testing from different federal agencies, industries, and academia, with a great deal experience and diverse viewpoints. The results of this workshop's findings will be used in developing material that will contribute to a FIPS guideline on software acceptance testing.

3.0 SUMMARIES OF WORKING GROUP SESSIONS

3.1 Off-Shelf Software Acceptance Testing

3.1.1 Charge

The working group for the acceptance testing of off-shelf software was charged with defining differences and similarities between acceptance testing of custom software and off-shelf software. Software acceptance testing practices for custom software may have features that are transferable to off-shelf software testing. The group was asked to identify these practices and the critical issues to be resolved before a standard acceptance methodology for off-shelf software can be defined. Finally, they were asked to recommend, if feasible, a consensus approach to testing off-shelf software.

3.1.2 Rationale

One approach to software acceptance testing involves a complete lifecycle, in which the customer is involved from the beginning of the software project. The customer has defined the requirements of the software system; the customer may have "accepted" intermediate products; the customer next accepts the final product. Finally, after acceptance, the customer may require enhancements. In this approach the requirements are customized to suit the customer's needs. Superficially at least, defining acceptance tests for custom software should be easy.

For off-shelf software acquisition, the vendor determines the requirements. A customer accepts the software based on those requirements. The customer is directly involved only from acceptance of the completed software product. The off-shelf software customer has limited or no access to development materials, including developer's requirements specifications and test cases. The requirements and constraints on off-shelf software may be sufficiently different from custom software to warrant completely different approaches to software acceptance testing.

3.1.3 Discussion

The off-shelf software acceptance test group first agreed upon a working description of off-shelf software. Instead of a formal definition, off-shelf software was defined by the generic descriptors shown in Figure 1. All discussions of the working group were based on software with these characteristics.

Many parameters affect how off-shelf software acceptance testing will be performed. The effects of some parameters, shown in Figure 2, on the approach to acceptance testing need to be better understood. The general discussions of the group focused on how to use these parameters to define software acceptance testing for a specific purchase. Once the process for a few specific examples is understood, more general guidelines may be defined.

OFF-SHELF SOFTWARE

software which is in common use
software which is readily available
software which has broad application with little modification

Figure 1. Generic description of off-shelf software

Users who are purchasing off-shelf software need to recognize that the flexibility of off-shelf software may be lower than that of custom software. This restriction is not stated explicitly in the generic characteristics but follows logically from them. By carefully examining their own needs relative to the types of characteristics shown in Figure 3, purchasers/users can establish a precise definition of the off-shelf software they want. They must define the essential features but must also define what functions are not required. Then they can go on to develop measurable acceptance criteria. Other types of criteria apply to an off-shelf software purchase. One might be a response to a question like "would you pay for this product out of your own pocket?"[2] .

users' understanding of their needs
precise definition of the off-shelf software
acceptance criteria for the off-shelf software
acceptance test procedures
constraints
standards

Figure 2. Parameters affecting off-shelf software acceptance test

As in custom software, acceptance criteria need to be developed for functional and performance measures. Other criteria can be built around the level of effort it might take for the product to be usable (e.g., change in business practices or substantial changes to the software or the software documentation). The purchaser must also consider to what extent the package can directly use inputs from other software or have its outputs used directly by other software. Can the software be immediately installed, or must changes be made for its integration into the total system? The purchaser needs to look at the total lifecycle costs of the purchase (e.g., vendor support, user training, hardware costs.)

The definition of specific acceptance test procedures may depend on factors unique to off-shelf software. These factors take into account that several off-shelf products might be considered before one is purchased. Other tasks must be performed to determine how much testing can be done:

- o establish clear selection criteria up front
- o reconcile market forces to requirements
- o determine needed level of testing vs reality (available resources).

[2]Ed. note: This question was asked relative to low cost off-shelf software packages that might be purchased in large quantity.

Usually, the acquisition of off-shelf software has some constraints and differences from the acquisition of custom design software. These differences will affect the performance of software acceptance testing. The marketplace, not individual requirements, drives what items are built for off-shelf purchase. The user may never see the requirements specification for a product, and, more importantly, has limited control of the requirements and the design. What the user sees in a user manual is likely to be the capability of the product and that capability might not be easily enhanced or modified. The user likely does not have any knowledge of the vendor's quality procedures or the extent of the vendor's qualification tests, i.e., those tests the vendor executes to demonstrate that the product meets vendor requirements. The potential purchaser may consider the vendor's reputation for building and supporting quality products. For a custom software product, users are likely to test for inclusion of all functions, which have been specificed by them. For an off-shelf software acceptance test, users may need to develop an application for the off-shelf software to determine if desired functions are included.

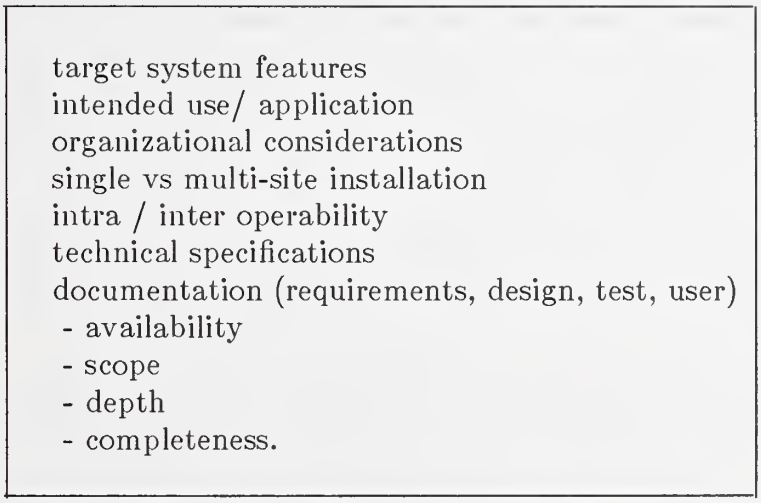


Figure 3. Some characteristics of users' needs

Economic considerations may drive the acceptance testing. Sometimes it doesn't make sense to spend large sums of money testing a package that may be used by only one person for a non-critical application. Vendors may be reluctant to permit access to some materials. If the same package is to be purchased in large quantity, and/or its application may be critical to the financial or operational capability of the purchasing organization, economic common sense may warrant the extra expense of complete testing. For a large purchase a vendor may allow access to development materials and quality assurance evidence. The characteristics of the acquisition and the user environment should determine how acceptance testing is done.

Some aspects of the software acceptance testing can be standardized. These consist of established test approaches, test tools, test cases, and diagnostics to be used during testing. Instead of developing standards for off-shelf software acceptance testing, perhaps the emphasis should be on evolving *de facto* standards for off-shelf software. If this were the case, then acceptance testing would consist of standard tests to show conformance to these *de facto* standards. A constraint to this approach is that new technology could get locked out through exclusive use of standardized tests on products. In selected areas some *de facto* standards could be achieved. Further, the use of off-shelf

software would promote working faster and smarter. Examples of possible *de facto* standards include data exchange, operating systems, documentation, and higher order languages standards.

A final discussion item involved the management of testing of *de facto* standards. Discussion ranged from the possibility of a "validation facility" to public newsletters. In the former situation, purchasers would send acceptance test suites to a common facility. New purchasers could match their needs against an already developed test suite. A newsletter could carry information of what off-shelf software has undergone extensive acceptance testing. For further information, prospective purchasers could consult those who have executed the test suite.

3.2 TEST CASE SELECTION

3.2.1 Charge

The test case selection group was asked to study test case selection techniques appropriate for software acceptance testing. Suggested topics included separating generic test case selection techniques from those more appropriate to a specific software product or to a specific means of software acquisition; determining criteria for the appropriateness of different types of selection techniques; defining the general effectiveness of various test case selection techniques including the maturity of the technique and the software tools necessary to exploit the technique. Examples of some specific topics include: path selection criteria, functional test case selection, test coverage metrics, maintainability, reliability, and other quality concerns.

3.2.2 Rationale

Various test case selection techniques have been extensively studied, but criteria for their use in acceptance testing have not been fully developed. In some acceptance test situations (e.g., testing off-shelf software) information normally available for testing is absent. This may include the code, the system design, and the component interface information. In the absence of this information what techniques should be used to test the software product adequately?

Terms such as robustness, maintainability, and reliability are often used to describe desired software features. Quantitative criteria are yet to be defined for many of these. Once they are, test case selection techniques should be used to test for these quality criteria. Other parameters such as the application area, performance requirements, and criticality of the software also affect the test selection techniques. There may be general techniques to guide the choice of test case selection techniques. Software tools may be required for effective use of the chosen test case selection techniques.

3.2.3 Discussion

The working group first established a definition of software acceptance test. As they developed their theories on test case selection, the scope of the initial definition was expanded. Time ran out before they could revise their initial definition to reflect their conclusions. The following definition for software acceptance test served as their basis:

"A software acceptance test is a system test used to measure the conformance of the delivered software system to current contractual requirements."

For the technical selection of test cases appropriate to a specific software acceptance test project, information is needed from several management-oriented tasks:

- o determine the contractual requirements,
- o establish the appropriate background and,
- o select a test strategy based on that background.

Contractual Requirements

Contractual requirements are significant and affect all the parameters of software acceptance testing. Acceptance testing, whether or not it includes some of the development information, is for the user's benefit. Development testing, whether or not used by the user, is primarily for the developer's benefit. An important factor in effective acceptance testing is the drafting of the contractual requirements that specify, as precisely as possible, the conditions that the software must satisfy before it is accepted. Many items need to be considered in establishing the contractual requirements (Figure 4).

Software is considered critical if its failure could be responsible for loss of life or large financial losses [IE729, IE730]. At the Software Acceptance Test Workshop, some asserted that the criticality level of the software increases as the value of the software to the purchasing organization increases. As the criticality increases, so should the scope of the acceptance testing.

If acceptance testing is to determine satisfaction of the contractual requirements, then the contract has to state its requirements in terms that can be transformed into test conditions. The functionality of the software, that is, the tasks the software must perform, should be clearly stated in the contract to ensure their inclusion in the software product. The contract should specify other information about individual functions and the system as a whole, e.g., expectations on function performance, required security levels, and the most important software qualities. One difficulty lies in determining criteria and test cases to test for compliance with a quality. Many quality levels are application dependent. For example, an unacceptable software safety level may mean 1 failure/10**9 years for automated aircraft landing systems, while 1 failure/year may be acceptable for an operating system (depending upon the type of failure). Other

---

1. The criticality of the software.
2. The functionality of the software.
3. Security of the software.
4. Performance considerations.
5. Metrics for quality measurements.
6. Documentation requirements.
7. The integration of the software with existing hardware and software.
8. Certification methods for the software.
9. Problems of changing requirements

Figure 4. Contractual considerations

qualities, such as maintainability, traditionally have appeared to have no testable meaning aside from the imposition of a development methodology (e.g., structured programming, system development using a 4th generation language). As software engineering practices and maintenance needs [FIPS106] become better understood, other contractual requirements can be used to support a maintainability requirement (e.g., use of standards, inclusion of development test materials as deliverables). In summary, metrics against which any function and any quality can be tested need to be quantified as much as possible in the contractual requirements. At the very least, guidelines for metrics must be included.

Frequently software is purchased as an enhancement to existing software or for use with existing hardware and software. Examples range from word processors to space station software. Contractual requirements include specifications for the interfaces between the existing software and the product; these interfaces undergo acceptance testing. For the "new" software, the purchaser may request evidence that the product has met specified quality criteria during development. For off-shelf software that will be used in the "new" product, earlier certification evidence might not be available.

The last topic in contractual considerations concerns the dynamic nature of the requirements themselves. During a project, some requirements usually change, perhaps even during a final evaluation period. The original contract needs to allow for changes in the acceptance test plan to accommodate changes to the contractual requirements.

Background

When the contractual requirements are being developed, the purchasers should be concerned with acceptance testing. At this time, they should establish the background parameters on which they will base their acceptance test strategy (Figure 5).

The scope of software acceptance testing (e.g,. the activities, the testers, requirements for a good test) is affected by differences between custom and off-shelf software as well as by contractual and business requirements. Compliance to both contractual and business requirements of the software system need to be tested.

---

o ESTABLISH BACKGROUND
- Define acceptance test scope
- Define acceptors
- Define entry conditions
- Perform risk assessment
- Perform resource assessment

o SELECT STRATEGY
- Match resources to risk
- Determine techniques

Figure 5. Preliminary tasks for test case selection

One example where software type affects the scope is a contractual requirement for either delivery of evidence of testing or purchaser participation in testing throughout the lifecycle. Due to the proprietary nature of off-shelf software, the customer does not always have access to source code and development information and participation in life-cycle testing during development is generally not possible. Instead, the software must be tested in its operational environment with data intended to find errors and particularly to test for compatibility with existing software and functionality requirements. The functionality requirements include the ability of the software to perform the desired function and also ergonomic considerations such as ease of use, ease of training, and satisfactory performance. For most off-shelf software, these requirements could be written into the requests for proposals (RFPs) and contracts.

For custom software, acceptance tests should be constructed for all stages of the lifecycle. Requirements, specifications, design documents, and code should all be used in developing an acceptance test suite. This testing could be performed by the developer and certified by the user or performed by an independent party. This process does not necessarily occur at the time of the delivery of the software, but may occur during software development at contractually agreed upon review points.

Two problems with acceptance testing involve the user of the software product. First, the end user should be involved with defining the contractual requirements and the early planning for acceptance. Second, identification of the user is a problem, for there are several users of the end product, (e.g., the operations staff, the maintainers, the application users, and auditors). Each group may have different acceptance requirements, and different acceptance procedures. Each group should play a role in the acceptance of software. Other problems occur when the requirements for beginning the execution of acceptance test are not clear. It is important to define what documentation is to be delivered, what software is in place, and what training is needed by the test staff.

Several kinds of risks also need to be identified. The management risks associated with accepting a software product concern the assessment of the consequences of failure of the product (e.g., premature product announcement or loss of life due to failure). The results of this assessment may be used to support the decision of the scope of the acceptance testing. The assessment of the technical risks that the product might fail is supported by the results of the acceptance testing. Another type of risk occurs when software acceptance testing cannot be performed in the complete operational environment. In this case, acceptance testing should use reliability tools for risk assessment.

The final object of establishing the background information for software acceptance testing concerns the resources available. Software acceptance management should examine the background information to establish the software acceptance test needs and then use the information for leverage to ensure the appropriate allocation of resources.

Strategy

Resources must be matched to the risks associated with the software product. Then priorities can be established. Different acceptance techniques can be applied according to project needs. The strategy of selecting test cases must be based on these management assessments. Effective test case selection techniques have been the subject of many research papers. Some require access to source code while others require sophisticated tools. The resources available to the software acceptance testing effort will affect the selection of test cases. Some of these techniques may be applied to the final software acceptance test case selection.

3.3 AUTOMATION OF SOFTWARE ACCEPTANCE TESTING

3.3.1 Charge

The participants in the software tools group were asked to make recommendations concerning tools for software acceptance testing. They were also asked to recommend a minimum set of tool features that should be used in software acceptance testing and to relate tool characteristics to development testing and off-shelf software testing.

3.3.2 Rationale

Acceptance testing is different from other software testing because the customer or end user is more likely to perform the testing than the developer. When not involved in the development effort, how can the end user ensure that the product is of high quality? Approaches vary from a contractual requirement permitting end user involvement in the development effort to contractual clauses that impose liability for the failure to meet criteria on the developer (in which case no testing may be required). For the one-of-a-kind software project the former is frequently desirable, whereas for an off-shelf product the latter may be feasible.

A particular concern is the type of tools that should be used to ease the acceptance process. Concerns include how these tools may differ from development tools, how they can be used to support acceptance (as opposed to development), their availability, and their usability. Among the tools that may be considered are the following: tools for test case generation; tools for measurement of compliance with acceptance criteria; tools for the development of benchmarks; software configuration management tools; test reporting and analysis tools; traceability tools; and tools for documentation examination.

3.3.3 Discussion

Presentation and discussion of two automated test activities provided insights into the appropriateness of some tools for software acceptance testing. The first was the Mitre Automated Test Environment (MATE). MATE, developed for testing a computerized employment compensation system, was used on a microcomputer to develop, track, and schedule acceptance testing for the system. The second automated test activity was an NBS effort to develop tests for communication protocols. Analytic techniques were used to determine inputs that would trace execution paths in a finite automaton.

To determine the functionality requirements for a minimal tool set for software acceptance testing, several factors relative to automation (Figure 6) need clarification. There will be exceptions to any set of reasons for selecting, or not selecting, a tool for software acceptance testing.

Definitions of acceptance testing in current guidance documents were viewed as inadequate. A definition of software acceptance test should be rigorous enough to ensure that a broad scope of acceptance activities would be considered. Acceptance is a continuing activity throughout the software lifecycle of a custom software project. This is a major difference between acceptance for custom and off-shelf software. For the latter, some business or user requirements might not be included within the written specifications but should be tested. Under the current definitions, no provision is made for such testing. Various interpretations of the current definitions could permit omissions of other acceptance test tasks, e.g, examination of documentation.

o definition of software acceptance testing,
o differences between development and acceptance testing,
o responsibilities of developer and/or customer management to prepare product for acceptance,
o procedures for software acceptance testing,
o circumstances when automation is needed,
o risks associated with tools, and
o purposes of automation.

Figure 6. Factors affecting automation of software acceptance testing

The group chose to define software tool needs based on their experience instead of a software acceptance test definition. They felt that acceptance test planners should determine their tool needs relative to a broad software acceptance test definition tailored to their project needs.

The group discussed differences between development and acceptance testing that might affect the types of software tools useful to acceptance testing. Generally, user acceptance testing is based on the functional specifications whereas early development testing requires design specifications and source code. Development testing is usually performed by the developer on the developer's site Acceptance testing is usually performed with customer involvement in an operational environment. Those executing acceptance tests may consist entirely of users who have not seen the requirements specifications document. Their assessment is based on how well the system provides the functionality they have been told to expect via a more general user document or directive. They would not compare their user experience one-to-one with the functional specifications, but rather with how well the system enables them to fulfill their job responsibilities.

What should the developer do to prepare the product for acceptance testing? Are tools and documents from the developer's environment essential to the customer? These may be contractual responsibilities of the developer management to ensure that all items for adequate acceptance testing are available to the customer.

Acceptance testing includes executing the system to demonstrate how the customers will use it. Preparing the demonstration includes planning, designing and developing test cases, executing the tests, collecting and analyzing the test results, and preparing the test reports on which the acceptance/rejection decision is based. Acceptance activities may also include evaluation of documents, installation procedures, maintainability needs, and other customer requests. Automation can be used for many of these tasks.

Automation is desirable, but when the cost of test tools exceeds the cost of the product being accepted, additional factors (e.g., reuse of the tool, criticality of the software) must be considered. Cost includes time as well as the actual dollar cost of a tool. If a product is simple enough to be tested adequately in two days, then five spent learning to use a test tool may not be cost effective. If the same tool is likely to be reused for other software acceptance testing, then the training time may be cost-effective. Of course, if the product is vital to a company's business operations, the dollar cost of the

tool or training costs become less than the *value* of the software product. If the product will itself be used in a critical application, added expense and time due to learning a tool may be worth it. For complex products, tools may provide the only means of executing complete test cases. Those selecting the automation level for software acceptance testing should take all these factors into consideration.

There are risks associated with tool use. The tool may not exist and will need to be built or purchased. Whether the test group builds or purchases the new tool, there is always risk that it will not be completed in time for the acceptance test project. The new tool's reliability, usability, and general value may not have been demonstrated before its use.

The primary purpose of test tools should be to organize and control necessary test data (e.g., test cases, functional specifications, test results, and reports). The tools should provide support in the preparation, execution, and analysis of test cases throughout both development and maintenance of the software. Candidate tools should be examined in the context of at least the five characteristics, defined below, which the group used to build a suggested tool set (Figure 7).

Generic -
Can the tool be used in many applications?

Development & Maintenance -
Can the tools be used by both the development and the maintenance phases of the lifecycle?

Availability -
Is the tool commercially available? Does the tool even exist?

Off - Shelf -
Can the tool be applied to a packaged product (e.g., off-shelf software)?

Minimal -
Is the software part of the minimal package a facility should have to perform automated testing? "On-line Systems Only" refers to the minimal required software capability to use the tool.

Several automated support tools may be used for software acceptance testing (Figure 7). No one in the group has used a document evaluator. Otherwise, each type of tool has been used for software acceptance testing in at least a primitive form by one or more members of the working group. No one claimed to have used an integrated set of the majority of the tools. Some of the group's assessments, particularly as to availability, have not been substantiated. As some of these tools become more widely-used, their effectiveness will be better known and can be added to the characteristics. The tools were discussed in the context of their definitions given in this section of the report. Some constraints may apply to these tools. For example, a white box test data evaluation tool, for use in software acceptance testing, might be applied to critical pieces of the software; in most cases the source code would not be available.

Automated Support for Software Acceptance Testing					
Automated Capability	Generic	Development Maintenance	Off Shelf	Minimal	Available
Test Data Generator Black Box	X	X	X		X
Test Data Generator White Box	X	X			X
Test Data Generator Random	X	X	X		X
Test Data Evaluation Black Box	X	X	X		
Test Data Evaluation White Box	X	X			X
Comparators	X	X	X	X	X
Documentation Verifiers-Evaluators			X		
Performance Evaluators	X	X	X	X	X
Test Driver	X	X	X	X	X
Script Recorders	X	X	X	X	x
Test Documentation And Configuration Controllers - Object Manager	X	X	X	X	X
Emulators	X	X	X		X
Filters	X	X	X		X
Standards Enforcers Evaluators	X	X		X	X

Figure 7. Suggested tool set

The working group defined the tools shown in Figure 7 according to their common level of experience. These definitions are provided below.

Test Data Generator (Black Box) -

A tool which generates test data and test output results. Information regarding data relationships is not used.

Test Data Generator (White Box) -
A tool which generates test data and results in which information regarding data relationships (e.g., path coverage) is known. The tool's application for acceptance testing may be somewhat limited because of its use on software development products.

Test Data Generator (Random) - A tool that generates random test data.

Test Data Evaluation (Black Box) -
A tool which examines the adequacy of the test data(e.g., completeness of data, correct boundaries) for determining whether the relationships between the test data and the results are correct. The user is unable to view the relationships.

Test Data Evaluation (White Box) -
A tool which examines the adequacy of the test data for use in determining whether the relationships between the test data and the results are correct. This is normally a tool used during program development rather than acceptance testing, but may be used for critical modules during acceptance testing for internal structural testing.

Comparators -
A tool that compares and cites differences between two files.

Document Verifiers/Evaluator -
A tool which verifies that the documentation matches the code. The evaluator is a tool that compares the documentation to the specifications and evaluates it by such criteria as clarity, ease of use, readability, user friendliness, etc.

Performance Evaluator -
A tool that evaluates the performance of a system. The items measured may include response time, the amount of machine resources used, the interface with other software, and performance under full load (stress testing).

Test Driver -
A tool that automates the execution of tests.

Script Recorders -
An on-line tool that records test scenarios. The tool may be used for code checking, for regression testing, and for simultaneous use by more than one tester.

Configuration Manager, Controller -
A tool that manages and controls the code, specifications, requirements, documentation, test data, etc. generated during the development of a system.

Emulator - A tool that exactly simulates another system.

Filter - A tool that manipulates input data from one file to produce output data for another file using simple text based transformations. The term arose from a typical use of *pipes* in UNIX[3] in which input text files are manipulated by programs into output text files where extraneous information has been removed or "filtered".

Standards Enforcer/Evaluators -
A tool that ensures that development standards are followed. An evaluator checks that standards are followed; an enforcer requires that standards be followed.

3.4 SOFTWARE ACCEPTANCE CRITERIA

3.4.1 Charge

The participants in the software acceptance criteria group were asked to define a comprehensive list of acceptance criteria qualities. Suggested topics included a taxonomy of acceptance criteria, metrics for acceptance criteria, tradeoffs between criteria, management issues in defining and implementing acceptance criteria, the relationship of actual requirements to acceptance criteria, and the definition of existing terminology and its relationship to existing standards.

3.4.2 Rationale

The simplest criteria for software acceptance is that the software meets the customer's requirements. Frequently, however, these requirements are stated in terms with ambiguous definitions. An example is this requirement, "The software must be maintainable, robust, fault tolerant, and secure." While definitions for some acceptance criteria exist [DOD2167], one of the tasks of the acceptance criteria group is to define those criteria used by customers in specifying desired qualities of software.

A second task is to determine how these criteria should be used in the acceptance testing process. That is, how do the criteria map to the software deliverables? A related task is the development of tests to ensure that the quality criteria are met. Some quality criteria, (e.g., specific security requirements), are testable while others, (e.g., maintainability), appear to be more difficult to test.

Finally, the question of automated tools is important. Is software acceptance testing sufficiently understood so that software tools can be used to ascertain whether or not a software product meets a quality criterion?

3.4.3 Discussion

The working group established the following goals to enable them to accomplish the tasks of their charge:

- o understanding the role(s) of acceptance testing
- o defining a comprehensive list of criteria
- o developing a taxonomy.

[3]UNIX is a registered trademark of AT&T.

Understanding acceptance

The working group addressed software acceptance testing relative both to custom software developed under contract and in-house and to compromises in the final software acceptance. Their discussions and conclusions were built around their interpretation of the keywords in the statement that software acceptance testing is the demonstration that the software meets its requirements:

- o demonstration: evidence
- o requirements: contractual description on which the criteria are defined
- o software: the deliverables
- o testing: actions that produce the evidence.

There are two views to acceptance: as a contractual event, and as a milestone event. As a contractual or legal event, the software contractually meets specifications or fails to meet those specifications. Simply put, when accepted, the payment for the software must be made. As a milestone event (e.g., during an in-house development), acceptance may be considered as a green light to proceed to the next stage. The next stage can mean several things. In development, it may mean the step from design into code activities. It may also mean a conditional acceptance, until other components are integrated and the system further accepted. It may mean acceptance so a company may release a product. In some organizations, the term "qualification" is used instead of acceptance to indicate that some pieces are ready and further development dependent on those pieces may proceed. It could also entail promises to management, production commitments, commitments of money and manpower to another stage of software development, etc. Acceptance may range from a minimal meeting of specifications to an exhaustive assessment of all parts. Organizations are driven by project parameters (e.g.,target dates, legislative mandates, risk assessments). How do these affect acceptance of milestone events?

The scope of acceptance testing is greater than functional testing of the software. With respect to the executable software itself, there are questions concerning its reliability and its maintainability. The *software* consists of many items, all of which should undergo some acceptance.

The types of items included in software acceptance are the meeting of the software's specifications, delivery of acceptable documentation, and assessments of risks, adherence to standards, ease of use and maintenance, meeting of performance requirements, and the meeting of functional requirements.

There is also a relationship between quality assurance and acceptance testing: acceptance testing is a quality assurance tool to support quality assurance activities.

Criteria

The software acceptance criteria for a software product can include only the criteria that have appeared in the specifications for that product. The workshop group prepared a working checklist of criteria categories on which they built a taxonomy to aid in measuring how well a product satisfies its criteria. The criteria checklist could be used as an aid in defining software requirements, but once the requirements have been defined, the checklist has to be tailored to a specific acquisition. The approach taken by the group was to develop a list of criteria, based on a set of quality characteristics [RADC], which would apply to different types of deliverables. Different kinds of evidence may be used for different criteria.

Criteria are applied to the baselines against which the software requirements will be tested. Baselines change as the software development or maintenance activities progress. The emphasis will be on different criteria during different aspects of software acceptance test. The development of acceptance criteria necessarily involves the development of metrics to measure compliance with the criteria. These criteria, however, are not a panacea. There should be some threshold of error so that if the software exceeds that threshold, then acceptance must not occur. There also should be a measure of severity. A severe error should cause the software to be rejected, while a number of non-severe errors may still allow acceptance (subject, of course, to later correction).

In addition to functionality, quality requirements (e.g., usability, maintainability, safety) should be built into requirements and specifications before the development process begins.

Taxonomy

The taxonomy developed by the working group involves a three-dimensional approach to solving the problem of quantifying acceptance criteria. The three dimensions may be applied somewhat differently, depending on the software acquisition, but need to include the following:

- o the products (e.g., design documentation, user manual)
- o the categories of criteria
- o the types of evidence or metrics to measure compliance.

Dynamic methods

DT	Dynamic tests

Static methods

RW	Reviews/ Walkthroughs
AA	Automated Analysis
MS	Measurement

Experience

EI	Internal
EX	External
WR	Warranties

Other Support

HL	Hot line
TR	Training
CN	Consulting
EN	Enhancements

Figure 8. Some suggested evidence types

All three dimensions are used to provide an overview of the project's acceptance needs. Different forms of evidence may be used to demonstrate how specific software products meet each category of applicable criteria. Some forms of evidence, indicated in Figure 8, have been selected for the example of Figure 9. In this example, acceptance testers would first identify the products, then the criteria, and finally, the types of evidence. Evidence may be demonstrated at different time points of the acquisition. The product may be examined as a milestone event, readiness to go on to next step, or as a contractual event, as a deliverable. Not all evidence may be requested, and only some possible types are indicated in Figure 8.

The software consists of more than the code which is executed by a computer. It consists of a collection of products that make up the complete system:

- o the specifications for requirements and designs
- o prototypes
- o the VV&T documentation
- o user manuals
- o operator manuals
- o maintenance information
- o configuration information
- o training
- o code
- o any other information required in the deliverables.

	ACCEPTANCE CRITERIA			
	Usability	Maintainability	Installability	Traceability
Management Plans				
Requirements Specs	RW	RW, AA	RW	RW
Prototypes				RW
Designs				RW, AA
Code, Docs				RW, AA
Tests				RW, AA
Test Results				
User Aids				
Installation Aids			DT, EX	
Maintenance Aids				
Operator Aids				
Configuration Aids				
Marketing Aids				

Figure 9. Example of mapping evidence to product criteria

The acceptance criteria form the basis for acceptability of these products. The criteria are derived from the specifications of these products. It is important to ensure that these specifications are measurable. Some examples of qualities are the following:

Functionality - Output
Performance- Resources, Time, Capacity
Usability
Maintainability/ Enhancements

Installability
Standards Adherence
Reliability and Availability
Security
Regulatory
Traceability
Risk Assessment
Safety
Learning.

Some of the products may have associated quality criteria, which are more difficult to demonstrate. Some approaches to measuring quality are to demonstrate the lack of errors, degree of functionality, and satisfaction of performance measures and the use of static techniques.

The types of evidence that can be used vary widely. Static evidence is obtainable from code and design reviews/walkthroughs, automated analysis tools, and performance measurements. Other evidence of compliance could be a warranty, previous project experiences, sufficient vendor support. The evidence for the satisfaction of a criterion could be based solely on trust (whether contractually enforced or not) or some objective information could be used. The satisfaction of some criteria, such as maintainability, might be possible through legal means such as maintenance contracts with the vendor.

The acceptance process then consists of determining the criteria during the development of the requirements/specifications, certifying that each product is adequately tested (this should also be in the specification), and ensuring that the entire project meets the final acceptance tests.

3.5 MANAGEMENT OF SOFTWARE ACCEPTANCE TESTING

3.5.1 Charge

The participants in the management group were asked to address issues concerning the management of software acceptance testing. They were asked to separate management from technical concerns and to define effective management strategies for solving software acceptance test problems. This working group had members from vendor organizations, testing organizations, in-house developers, and purchasers of contracted software. Their unique perspectives are represented in the session discussion.

3.5.2 Rationale

Project management is responsible for the definition of contractual requirements for acceptance of software and the definition of acceptance testing in the context of particular software products. Management is also responsible for the ultimate acceptance of the software. Since management defines software acceptance testing and is ultimately responsible for the acceptance of the software, management is critical to the software acceptance testing process. The management of software acceptance testing differs little from traditional management (e.g., planning, controlling, providing support, and performing cost-benefit and risk analyses) except that it is done in the context of software acceptance.

Management has the responsibility of planning to ensure that contractual requirements meet user needs. This may require planning for validation, verification, and

testing activities throughout the software development lifecycle, or for testing to ensure that off-shelf software meets the organization's needs. In planning the testing required for acceptance, management has responsibility for designating the persons who ensure that tests are performed.

Management is responsible for defining and implementing controls over the vendor, the vendor testers, and within the end user organization. Controls may include reporting and change procedures throughout the acceptance test period.

Management is responsible for choosing the test methodology. Management is also responsible for ensuring that ergonomic issues are settled, for putting in place required training, and for making available the required documentation to effectively use the delivered software product. Management ensures that schedules are appropriate for the tasks involved.

Finally management is responsible for the risk and cost-benefit analyses that estimate how much testing is required before acceptance of the software product. Typically these estimates will be most affected by the criticality of the system but other factors (e.g., size, expected lifetime, complexity) are important.

3.5.3 Discussion

The group established the primary goal of a well-managed acceptance test effort to be user satisfaction in the accepted product. The tasks for the session were to define management's role in achieving this goal and to determine how management could solve some of the technical problems of acceptance testing. The group discussed specific issues management should address, possible approaches to each of these issues, and constraints on applying these approaches (Figure 10). The group established some differences between development testing and acceptance testing of software, and between custom software and off-shelf software acquisition.

In software development testing the primary goal is to locate errors and discover omissions. In software acceptance testing, the goal is to determine if the product can be satisfactorily used by the customer, even if some errors go uncorrected until the next version is released. In acceptance testing, an additional goal is the assessment of maintainability of the product. Another major difference between development testing and acceptance testing involves perspective: testing against evolutionary development materials vs testing against contract specifications. The administration of the testing has similarities (e.g., appropriate documentation for planning, test materials and reporting) and differences (e.g., type of information included in the documentation, change procedures caused by rejection). Some types of automated support for testing can be used for general testing (e.g., test driver), but others may serve best in only one type of testing (e.g., source instrumenter).

Management responsibilities differ for custom and off-shelf software. For custom software, the customer establishes contractual requirements placed on the development process. These may include the development methodology, documentation standards, quality assurance standards, the specific deliverables, and the criteria for their acceptance. The contractual requirements may include delivery of test data suites and data related to the number, kind, and severity of discovered errors. The developer has responsibility for meeting scheduled milestones and for providing information from which certification of both modules and the integration of modules can be deduced. An effective configuration control system would be required to provide useful information.

Issue	Possible Approach	Constraints
Management		
Custom	User involvement through lifecycle Accurate, timely information provided to the user	Integrity of vendor Accuracy and completeness of data
Off-Shelf	Demonstration Functional testing Planning for integration with existing software	Integrity of vendor
Contractual		
Initial acceptance	Development of stable & quantifiable acceptance criteria for contract	Existence of contract
Maintenance	Warranty Support agreements	Enforceability
Risk		
Consequences of failure	Risk analysis consistent with cost+benefit analysis	Size of application Criticality
Modes of failure	Effective identification of risks Development of quantifiable tests for risk	
Organization		
Structure + Commitment	Define authority & responsibility Develop organizational standards Develop effective reward systems	Existing organizational structure Conflicting objectives
Planning	Develop test strategy Integrate all other management issues.	Time, money Quality of personnel Quality of external organization Availability of tools

Figure 10. Management issues of software acceptance testing

When information from the development process cannot be contractually required (e.g., with off-shelf software), management is forced to rely on other mechanisms. These could include previous experiences with the vendor, demonstrations of the software, and trial versions of the software. Management would still be required to specify criteria for the acceptability of such software. These criteria include functionality of the software and whether the software works well with existing software.

The constraints on the above management approaches include the availability of the information, its accuracy, and the integrity of both the information and the developer.

Contractual issues (e.g., items for inclusion in contracts, enforcement) are another management responsibility. Two principal types of contractual items involve different times within the life of the product:

- o the development and initial acceptance of the software, and
- o problems that arise with the software after initial acceptance.

For the former, contractual documentation identifies deliverables and assigns responsibilities to the developer and the user. For the latter, it is very important to establish criteria that can be measured. For example, a contract requiring only that the product be maintainable does not provide measurable criteria. A contract requiring material with specific content and format (e.g., design documentation, development test data) to support maintainability is better. Management, through its planning responsibilities, can help to solve the technical software acceptance test problem of testing for maintainability. A contract should specify materials needed by the customer for support of a feature or quality characteristic, and the acceptance test plan will ensure examination of these materials.

For working with later problems, the contract should have a warranty. Support, maintenance, and training agreements should be specified. The constraints on contractual items are the existence of a real contract and the enforceability of contractual clauses.

The theme of risk appeared in almost every discussion. The risks are of two kinds: managerial and technical. Management is concerned with the consequences of failure of the accepted product, or, in the case of a vendor releasing a product, the marketed product. Liability, marketing capability, and financial matters (e.g., the cost of maintenance due to acceptance of an untested, poor product) are managerial concerns while reliability, performance, and maintainability are examples of technical risks. Acceptance testing and the analysis of the results provide measures for the assessment of potential failure and of the amount of maintenance a product might require. The management strategy includes risk analysis that

- o identifies risks,
- o establishes their priority, and
- o quantifies the risks.

One outcome of a risk analysis is the identification of acceptable and unacceptable risks in terms of costs to the organization. A risk analysis might affect vendors' decisions on the level of acceptance testing and on the release of a product. For example, should the company market a product with known errors because entry into the marketplace is critical at this time or would damage to its reputation be unrecoverable with this entry?

Organizational structure and commitment issues refer to the responsibilities of individuals within the organization and the organization's commitment to effective acceptance testing. The management strategy is to develop plans that define both responsibilities and the authority to manage aspects of acceptance testing; to develop a uniform acquisition procedure; to be involved in acceptance testing from the beginning and throughout the project; to develop organizational quality standards that acquired software must meet; and to have an organizational structure rewarding individuals who perform effectively in the software acceptance process. The management constraints

consist of existing organizational structures; conflicting organizational objectives; and the time and money allocated to acceptance testing.

A principal management responsibility is planning for ensuring maintainability and continued usability of the software. Software acceptance testing needs to include examination of the product for these qualities. A maintained product should undergo acceptance testing and repeatability of previous acceptance tests is important. It is a management responsibility to insist that acceptance tests are repeatable. Management support in planning and directing software acceptance testing can help to solve some technical problems.

Acceptance testing should include the kinds of users who may be involved with a software product:

- o those who use the end product,
- o those who operate the end product, and
- o those who maintain the end product.

The group suggested that all involved parties should be required to sign for acceptance or rejection as a result of the acceptance tests.

Planning is a management issue that overlays all the rest and includes the development of a software acceptance test strategy flexible enough to evolve over time. This strategy should include scheduling, methodology, required standards, and early identification of alternatives if problems arise in the software development or the acquisition of off-shelf software. Management must also identify and quantify quality criteria, perform risk analysis, develop evaluation procedures (with feedback), and coordinate the external development with internal acceptance testing. The constraints in this planning include the quality and number of people, the amount of money and time available for the acceptance testing effort, the expertise of the personnel, the accountability of individuals within the organization, the quality of the external organization, and the availability of tools to aid the acceptance process.

Planning was determined to be the key task of the management of software acceptance testing. Perhaps the most important factor in planning for acceptance testing is the risk analysis that provides management with the ability to determine how much acceptance testing is needed. The planners may develop requirements and constraints based on these risks. The plan may provide direction for some technical problems of testing. Once the plan is written, management must ensure its complete implementation.

3.6 STANDARDIZING SOFTWARE ACCEPTANCE TESTING

3.6.1 Charge

The participants in the Standardization group were asked to examine standardization as applied to software acceptance testing. Issues include aspects of software acceptance that could be standardized; aspects that should be standardized; aspects that should not be standardized; and aspects that currently cannot be standardized. Existing standards should also be addressed.

3.6.2 Rationale

Acceptance testing for products in domains other than software has led to standard that ensure the quality of those products. These range from physical standards such as strength of materials and tolerances of parts to engineering standards for the design of structures such as bridges and buildings. Software is unusual because few standards exist to measure either the quality of the delivered product or the quality of the engineering process that went into the design of the product.

Successful methodologies and engineering practices for the development of software have matured; the issue of software quality standards can be addressed. Categories of standards could include standards for acceptance of a delivered product, standards for the development of software, standards for the products to be delivered with the software (e.g., documentation), and standards for tools to measure compliance with software acceptance test standards.

The main benefit of a standard is obvious - the assurance that a product meets specified standards for its development practices and for its product specifications. There are dangers, however, to standards. Standards can stifle innovation for there may be little incentive for producing a product that "exceeds" a standard. Standards may prevent competition because new methodologies may be precluded from use. Standards may be imposed prematurely. To what extent is the area of software acceptance testing mature enough to benefit from standardization?

The standards group was asked to investigate the suitability of various areas of software acceptance testing for standardization. They were to investigate areas where existing standards could be adapted to acceptance testing (e.g., test planning) and areas where standards are needed or could be developed. These questions were to be addressed specifically and in the general context of the need and desirability for standards at this time.

3.6.3 Discussion

The group considered the benefits and limitations, usage guidelines, and specific areas for standardization.

Benefits and Limitations

A summary of benefits and limitations of standards in software acceptance testing appears in Figure 11. A software acceptance test standard would be used by vendors, testers, quality assurance personnel, legal staff, auditors, and customers, including both the contracting office and the end users. A common set of definitions may help to clarify legal misunderstandings. For example, the users of the standard could understand how deliverables in a contract are to be measured, based on that standard's common set of definitions. Managers may use a standard as a vehicle for planning. Personnel new to software acceptance testing may have a minimum basis from which to work. In general, a standard provides a framework from which a particular task may be tailored to project needs. Different audiences derive different benefits from using standards.

BENEFITS	LIMITATIONS
Common definition	Too weak
Minimum set	Technology hindrance
Planning aid	False security
Large audience	Art; not practice

Figure 11. Standardization: benefits and limitations

One of the major problems associated with technical standards is that new technologies cannot always be adopted when a standard is in place. Others complain that because standards are written in the broadest manner possible, they may become so weak that no benefit is derived from using them. Some standards may contain examples or other material that users apply directly to their projects without incorporating any additional material into their application. This approach may provide a "false sense of security" when a standard is intended to provide only a minimum set of standardized activities.

An inherent danger of attempting to standardize a technology lies in not separating the state of art from the state of practice. A technology that is not yet mature (e.g., wide-spread disagreement on how to perform a process) is not ready for standardization. In some cases, it might be better to develop a recommended practice rather than a standard. This would be especially helpful where there may be political difficulties in putting a standard into place. Finally, another limitation of standards concerns the content. Should the content define a minimum set of tasks and directives that all projects must perform or use, or should a standard set a goal that all projects should strive to achieve?

Usage Guidelines

The working group discussed the implementation of a standard. A standard should include an appendix with guidelines on usage, with suggestions for tailoring to specific types of projects and with examples. A standard in software acceptance testing will have several types of users, ranging from vendors' internal organizations to those of the customers. The vendor, for example, may perform a set of qualification tests first; the actual dynamic acceptance tests may be a subset of the tests the customer will perform. The audience determines how the standard will be implemented.

Standardization Topics

For software acceptance testing, the topics indicated in Figure 12 need to be clearly understood before a standard can be developed.

Users of a standard for software acceptance testing may include developers, quality assurance personnel, test personnel, and customers, including both end users and the acquisition agency. Project managers may use such a standard to help with their resource and schedule planning for the software acceptance testing. On small projects some of these groups can overlap; on very large or critical projects, the project organization can be large with little overlap among the groups. In the former case, a quality assurance organization might be tasked with software acceptance testing but in the latter situation might only approve the use of, and ensure compliance with, a software acceptance test standard. It may be possible to develop a versatile standard useful to any of the audiences, in any organizational arrangement, but usage guidelines should be included.

o users of the standard
o requirements and testing relationship
o the type of software system: off-shelf vs custom
o tool for implementation of the standard
o specific items for standardization
  - test methodology
  - test plan
  - test results report
  - test evidence
  - test evaluation and recommendations.

Figure 12. Topics to understand before standardization

Requirements for the product(s) and acceptance testing go together. For acceptance testing, the basis for the testing must be identified, (e.g., requirements documentation, contractual agreement). Often an assumption is made that the requirements fully represent the user needs; unfortunately this is not always true. How are user needs represented in acceptance tests? The group emphasized that "contractual documents" for custom software should include user needs.

Acceptance test for off-shelf software is a very different process from acceptance test for custom design software. For off-shelf software, a comparison is usually made among several products before one is selected. The user documentation and user requirements provide materials for the acceptance test. With custom software, the functional specifications may evolve during development, and some parts of the acceptance process may occur during development.

One member of the group described a very limited set of acceptance test tools applied on a project. On this project, tools were developed to provide total data capture during testing. Testing was easier to track and more consistently performed with standard formats for test scripts, test data, and documentation. The testing itself was automated. The experience from this set of tools indicates that generic types of tools for software acceptance testing could be specified in a standard.

The test activities may be performed in a standardized sequence. One standard [IE829] defines a generic test methodology that orders a given sequence of test documentation (e.g., test plan, design, cases, procedures). Another standard [IE1012] defines test types according to their sequence of execution and requires a requirements trace, i.e., a matrix of requirements vs test documentation. A software acceptance test standard can provide a standard format for the following items:

o scope of objectives for the acceptance test
o sequence of test activities
o test plan
o control procedures, (stop & restart)
o form for pre-determined cases (form for test scripts)
o type of evidence
o tool types

o test results report (analysis, evidence)
o test summary and recommendations.

The initial software acceptance test standard should specify the type of information to be addressed in the test plan, the test evaluation and analysis report, and the test summary and recommendation report. In Figure 13, the types of information vary according to who will use the test plan: the vendor or purchaser of off-shelf software or those involved with custom software. The amount or type of information for a specific information product may also vary according to the perspective of the user of the plan. The timing sequence of development of test items is specified. The traceability matrix is required for all software. Information on constraints should be kept separate from the actual plan but is vital in establishing the acceptance test plan.

Time ran out before the group could take what they considered the next logical step: to expand on each of the information products in Figure 13 from the vendor's view and the user's view for off-shelf software, and the customer's view for custom-built software.

Types of Software			
Information products	Off-Shelf		Custom
	Vendor's View	User's View	
Test Plan			
1. Criteria Objectives	Marketing user documentation	User Documentation and Needs	Contractual
2. Methodology	Needs to be identified	Needs to address "qualification" tests	Based on Criteria
3. Procedure	Needs to be identified	Based on Methodology	Based on Methodology
4. Cases	Subset of qualifications	Built by user	Based on Procedure
5. Data	Subset of qualifications	Range from generic to specific	Based on test data
6. Traceability Matrix	Applicable	Applicable	Applicable
7. Controls for stops & restart	Needs to be identified	Not Applicable	Needs to be identified
8. Requirements	Needs to be identified	User's needs	Needs to be identified
Constraints			
1. Resources	Simulation of operating environment	Training	Simulation of operating environment People; Hardware; Tools
2. Schedule	Release driven	Need driven	Contract driven

Figure 13. Information described in Software Acceptance Test Plan

The expansion would have given more details in these areas without establishing any standards, for it is too early to standardize those details.

In closing, the group made the suggestion that a standards developing organization might form a committee to consider a standard for software acceptance testing.

3.7 RESEARCH FOR SOFTWARE ACCEPTANCE TESTING

3.7.1 Charge

The participants in the research session were asked to define areas where research in software testing could lead to techniques permitting more effective software acceptance testing. The topics to be addressed included research in automated support of acceptance testing; research results that are ready for transfer into practice; topics in acceptance testing that might be amenable to precise definition (e.g., robustness, maintainability); and programming language research that could lead to the development of good specification and requirements languages.

3.7.2 Rationale

Although some areas of software testing (e.g., test case selection) have received attention from the research community, they lack direct connections to the problems of acceptance testing. The identification of those areas is important for facilitating the transfer of technology from the research community to the practice of software acceptance testing. The research areas session was asked to identify those areas which show the most promise for technology transfer.

Terms describing quality (e.g., robustness, functionality, maintainability) are generally understood by the testing community but lack a precise definition. This leads to great difficulties in defining test data to test for these qualities and to difficulties in measuring when a sufficiently high level of each quality is achieved. One solution might be to define models of software development in which such terms can be given precise meanings. Do existing models of software development permit precise definition of such terms and if so can the definitions be extended to more general settings? Part of the development model includes representation of requirements and specifications. The research areas group was asked to discuss the development of requirements and specification languages and to discuss the effectiveness of any known to be in use.

Generic areas of research (e.g., artificial intelligence) might lead to more effective techniques for software acceptance testing. For example, could a system understanding natural language aid in solving the problem of requirements validation? The participants were asked to investigate which generic research areas might lead to the solution of outstanding problems in software acceptance testing.

3.7.3 Discussion

The research areas group concentrated on acceptance testing for one of a kind software projects in which access to software development data is available. The acceptance test is then viewed as a two part process. The first is certification that the software, as developed, is thoroughly tested (development test). The second is that the software works correctly (operational test) in its intended environment. The group considered the software acceptance test concerns relative to the initial delivered version of a product. If the software is certified properly during the development stages, and if the

information is kept for use during maintenance, then the same techniques should be applicable during maintenance.

The certification of the development test process can then be viewed as satisfying testing criteria at various points in a general software lifecycle. Many products must be developed during various stages of the software development process, beginning with definition of the user requirements and ending with the final product.

The definition of software acceptance testing is the "formal testing conducted to determine whether a software system satisfies its acceptance criteria and to enable the customer to determine whether to accept the system" [FIPS101]. The research group felt that there are two common scenarios. One occurs when off-shelf software is purchased. Then the customer has little influence on the products that are provided (e.g., design documentation, test cases), but the customer can do comparison shopping (e.g., benchmarking of similar products). The second scenario occurs when software is specified, for one of a kind systems. Under this scenario, the group strongly felt that doing acceptance testing only after the system is complete is a big mistake: that development testing must be considered as part of acceptance testing. Points in the lifecycle earlier than the acceptance point were to be considered as possible research areas for acceptance testing.

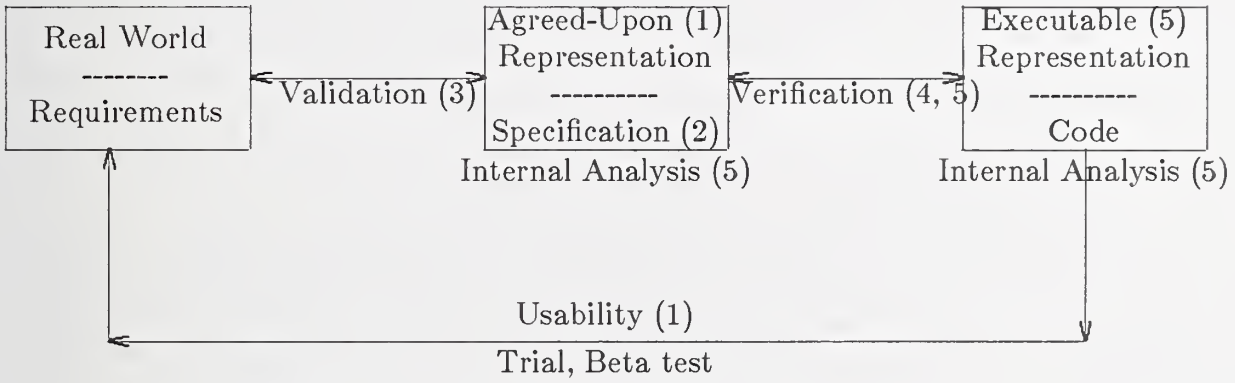


Figure 14. Simplified software development model

To enable a customer to determine whether or not to accept a system implies building customer confidence in a system. One question concerns how to build that confidence when there are many different lifecycles, applications, and development paradigms. The research areas group developed a simplified lifecycle model, shown in Figure 14, that also indicates the iterative nature of software development. The model would be applied to software maintenance also. Possible research areas from Figure 15 were mapped to this simplified model.

Some of the research areas include development testing. The group felt strongly that customers must have confidence in the development testing and that research is needed to learn how to take advantage of white box certification during acceptance testing. White box coverage techniques enable the acceptance tester (and customers) to know what percentage of modules (or source lines, or branches) have been tested. This knowledge, combined with other types of evidence during acceptance testing, enables the customers to determine if the system does what they want; the combination may help to build customer confidence. The acceptance testing may perhaps be focused on usability at the operational level. For usability, research is needed to quantify human factors. This research area appeared frequently in group discussions. Human factors

would be taken into account to represent a system to the customers as part of the process in enabling them to make an acceptance decision. An example is that user functions need to be analyzed before accepting a very high level language as usable.

At the beginning of the lifecycle, product development begins with the statement of user requirements. Several issues arose concerning this topic. There was clear agreement that prototyping is an effective way to obtain user requirements in many situations. A prototyping tool would then act as a requirements validation mechanism. It was also generally agreed that the problem of notation for requirements has not been solved. Suggestions for such notation ranged from subsets of English, to graphics, to other unspecified media. It was agreed that the notation would depend very much on the specific product being developed.

Further into the lifecycle it becomes necessary to convert the user requirements into some form of specification (e.g., design). These specifications must then be shown to be consistent internally and with the requirements. At a minimum, the form for specifications should permit effective consistency checking. Ideally there would be an automatic conversion from requirements to specifications; this was viewed as a very long term research goal.

Still further into the lifecycle it becomes necessary to convert the specifications to code. At this point unit testing becomes important. Advances in unit test have been made, but they are costly to implement. One research issue is the automation of the unit test process to reduce cost. If modules can be given certificates of correctness, then what does this imply about the correctness of the software system as a whole? It was agreed that such certificates were necessary for acceptance of the software, but not sufficient. Thus the integration of the modules must be tested.

The major concern regarding integration, and for that matter the entire software development effort, was the tremendous amount of data that accumulates over the life of a major project. Research should be directed towards the development of a sophisticated configuration manager. The configuration manager tool would track changes in code through both requirements and specifications. It would track those modules affected by changes in requirements or specifications. Advances in several areas (e.g., database theory, incremental data flow, artificial intelligence) might contribute to the development of such a configuration manager.

---

1. Quantification of human factors
2. Notation for requirements specifications
3. Methods to validate requirements specifications
4. Black box testing techniques
5. White box testing techniques
6. Test adequacy measures throughout the lifecycle
7. Measures of the effectiveness of testing and adequate data collection
8. Control of test documentation and configuration control
9. Automation of testing throughout the lifecycle

Figure 15. Suggested research areas for software acceptance testing

Problems associated with the the final testing of the system (operational testing) were then brought up. The first was the possibility that the system could not be operationally tested. This type of problem exists with military systems (e.g., SDI software), as well as in civilian systems (e.g., nuclear reactor control systems). Simulation helps in such situations, but is not adequate in many cases. Another problem is in the testing of expert systems. A solution to this problem is reliance on some sort of Turing test (e.g., testing the system's recommendations against those of three experts). The system is accepted if it agrees with a majority of the experts sufficiently often or sufficiently closely.

Several types of tools might be useful for software acceptance testing. Among these are data collection tools. Collected data on project history could be used by the forecasting tools to predict time and effort needed for retesting. Or, data may be used by evaluation tools to indicate project testpoints. Comparison tools for use at different points in the lifecycle might also be helpful. Other tool areas where research efforts might contribute to advances are test support tools and version control tools.

While some tools do exist, research is needed to improve their utility, especially in tool integration. The group proposed an ambitious research project: development of an integrated data base containing software development information (e.g., requirements, specifications, modules, unit tests). Tools would operate against the data base for document generation, tracking, forecasting, and other tasks. With such a system, those performing the final acceptance tests would require delivery of the data base, the tools to access it, and the operational tests; then they might be able to perform complete acceptance tests.

The group addressed two related issues: How does one make an acceptance decision? And, how does one learn to make an acceptance decision? Research in the areas of black box testing (4), test adequacy measures (6), white box techniques (5), and measures of the effectiveness of testing and data collection (7) as indicated in Figure 14 might help to answer the first question. Research on quantifying human factors (1) and again effectiveness measures (7) might provide advances in answering the second question. And finally, research on controls (8) and automation (9) might be used to support advances in management, configuration management, automation, and data collection problems.

3.8 PRACTICES OF SOFTWARE ACCEPTANCE TESTING

3.8.1 Charge

Through the study of examples of current practices in government and industry, the participants in the state of practice session were asked to determine the current state of practice and to recommend procedures to improve the state of practice in software acceptance testing.

3.8.2 Rationale

The state of practice in software acceptance testing varies widely from organization to organization. From examination of the practices in a number of organizations and analyses of their successes and failures, optimal methods for acceptance testing might be uncovered.

First, it is necessary to identify current practices, to separate the successes from the failures, and to identify common features among the successes. Other useful information concerning current practices includes the extent to which automation is utilized, the levels of required documentation, the commonality of acceptance test deliverables, and the kinds of standards, both developmental and acceptance test, that are used. It will also be necessary to determine how the acceptance test practices differ with the type of software undergoing test.

The results of the information gathering can lead to the recommendation of practices ready for standardization, the identification of problem areas that require improvement, and the recommendation of a software acceptance test methodology including automated support.

3.8.3 Discussion

The participants' discussions on the state of practice of software acceptance testing primarily concerned custom software. The software may be custom built for a specific customer, or developed by a vendor to become an off-shelf product. In the latter situation, the vendor becomes the acceptance tester of the software for acceptance as a product to be marketed.

The participants discussed their practices, citing successes and problems, in a lifecycle development framework. The firmest recommendation was for a strong acceptance test plan, supported with other management and quality practices. Practices, described according to successes and problems (Figure 16), involve different parties:

- o the users (acceptance testers)
- o the acceptance test management
- o the hardware/ software system.

Acceptance testers, who represent users of the software, must be involved throughout the software lifecycle. Communication between developers and users of the product is essential. An experienced developer may provide help to purchasers who, for a variety of reasons, may not be totally familiar with their own business practices or who may not have used computer systems previously. When the users of the intended product are not the contracting officers, communications are vital during requirements specifications to ensure that all the functions for the product are included, and are clearly stated. The users who are either represented by, or who are, the acceptance testers, need to work closely with the developers to ensure testability of any requirement.

Acceptance testers, especially those who are the intended end users, may be unfamiliar with computers. Often an application is part of a much larger system, including the hardware to be maintained by the purchasing organization. Steps must be taken to ensure that the acceptance testers do not become frustrated by problems unrelated to the software application. Operations and systems personnel need training in advance of acceptance testing to prevent test disruption due to non-application problems (e.g., communication protocols incorrectly initiated). The acceptance testers themselves need to be trained in the application. They should have access to all the documentation, and should know how each of their tests relates to contractual requirements.

	SUCCESSES	PROBLEMS
USERS	Standardized internal procedures	Test staff turnover
	Quality assurance	Staff working in shifts
	Configuration management	
	System training	Inexperienced operations staff
	Trial use before testing	
	User business training	Inexperience with the business
	Computer and application training	Inexperience with computers
MANAGERS	Strong acceptance test plan	Lack of quality assurance
		Haphazard practices
	Clear requirements	
	Testable criteria	Lack of testable requirements
	Designation of responsibility	Uncorrected errors
	Risk assessment	Schedule
	Rewards, reasonable schedules	Frustrated testers
SYSTEM	Configuration management tools	Changing versions
		Changing environment (developer to operations) (operation changes)
	Test-aid tools	
	Trace tools	System interfaces
	Repeatable, maintained test sets	Regression testing

Figure 16. Practices of software acceptance testing

One organization solved the problem of severe delays caused by staff turnover by developing a standardized internal system for building and maintaining test cases. The system included the rules for test case development, a test case matrix for tracing the test materials to requirements, and predetermined test results. Quality assurance and configuration management on acceptance testing also made regression testing easier to perform correctly. Tools were cited as helpful for all test-related activities; those cited as most useful are for test coverage checking, test reporting, and configuration management.

Strong management can solve many of the technical problems by insisting on communications with the developers, training for all parties involved in the acceptance testing, appropriate tools, and reasonable schedules. The management has responsibility for ensuring that the requirements to be tested are testable, and the acceptance criteria measurable. The managers need to plan how the testers are to report problems, what conditions warrant stopping of tests, and how much regression testing on changes is to be done. In some organizations, testers are detailed from their daily jobs as needed for the test execution. Sometimes when acceptance tests last over several months, the testers will sign off simply to be done with the task. Management needs to have clear procedures and designated responsibility for signoff. Finally, the testers are often frustrated because they have been removed from their normal duties and career paths. Managers need to consider incentives for testers, (e.g., reward system).

The product itself is the cause of some acceptance test problems. Differences in the product may occur in the transition versions from the developer to the operational environment ; these differences can be detected by using configuration management tools to ensure all the appropriate pieces are in place. Comparators used on code compiled in different environments (e.g., developer-user; two or more user installations) can check for differences in the object code. When the user environment itself changes, acceptance testers need to examine the requirements to understand how the acceptance tests should change. Several organizations represented at the workshop cited use of test sets

MANAGEMENT PRACTICES

- o Take a lifecycle approach to acceptance testing
- o Be involved early to establish acceptance criteria and test requirements
- o Insist on documentation well in advance of various development progress
- o Develop QA, CM plans
- o Perform, follow through on, risk assessment
- o Keep all parties informed
- o Allow sufficient time for auxiliary tasks
  - documentation review
  - training of different types of personnel
- o Use results of verification and validation tasks
- o Specify test reporting, stopping policies.

TEST PLAN TOPICS

- o Automated Tools
- o Testable Criteria
- o Process for Trouble Reporting & Monitoring
- o Traceability of Test Requirements
- o Adequate Test Case Coverage
- o Test Case Matrix
- o Predetermined Test Results

Figure 17. Recommended practices for software acceptance testing

maintained on tape; these are executed after any environmental change and checked by means of comparators with previous tests.

The working group prepared a summary of items that acceptance test managers must address in software acceptance test plans (Figure 17). The complement to items appearing in a plan are the policies that software acceptance test management must develop and implement, and ensure that the staff also implement them. Planning ahead for software acceptance testing, management implementation of policies, and support for the acceptance testers are important steps for ensuring good practice in software acceptance testing.

4.0 SUMMARY

The Software Acceptance Test Workshop held at the National Bureau of Standards on April 1-2, 1986, has resulted in the expansion of the software acceptance test project into the software acceptance project. Consensus of the various sessions is that software acceptance, whether for custom or off-shelf software, is a life-cycle activity. Initial planning for software acceptance occurs in conjunction with project planning, and continued management of the software acceptance must occur throughout the lifecycle. Execution of software acceptance tests at the delivery of the software is a subset of the processes

of software acceptance. Software acceptance incorporates many evaluation methods and procedures from software development. These methods and procedures are used to determine whether quantifiable acceptance criteria have been satisfied.

---

o software acceptance

o maintainability

o automation
  - recommended types
  - emphasis on research

o initial and continued management

o learning from successes

o acceptance criteria and the end user
  - during requirements (deliverables, evidence)
  - throughout development
  - human factors research

o role of development testing
  - vendors of custom and off-shelf
  - purchaser

o standards
  - *de facto*
  - conformance testing

o guide, rather than standard

Figure 18. Key discussion topics

Some key points of the workshop are presented in Figure 18. In some cases, discussion of these topics led to the software acceptance approach. In other cases, consensus on the approach led to the discussion of these topics as key to developing successful acceptance procedures. The life-cycle approach to software acceptance was taken by all the groups, except the off-shelf group; their view was that the lifecycle affects the vendors but purchasers have a different perspective. A more general disagreement occurred when several groups defined either software acceptance or acceptance test as a starting point for their discussions. These definitions differed in detail from one another and also from published definitions of the terms. Further work is needed to form a consensus definition. Some recommendations within groups were based on their perceptions of how development testing and acceptance testing differ. At least three groups had different opinions (test case selection; automation; management).

Maintainability was the software quality referred to most often during the workshop. The participants felt that end users needed to be involved early to ensure that maintainability was established as a contractual requirement. The suggested contractual requirements ranged from the inclusion of intermediate development products as deliverables (design documentation, test materials, tools) to required development practices (e.g., standards for design documentation, test materials, tools). Evaluation for maintainability could include reviews, walkthroughs, or automated analyses of these materials. Although the human factors aspect of maintainability was felt to be important, the groups concluded that further research would be needed to determine how to measure this aspect of maintainability.

Automation for software acceptance testing was addressed by almost every group. The automation group suggested tool types to aid in test tasks. The research group recommended test support tools as a class of tools and expanded their recommendation to included test management tools (e.g., forecast tools for retest resources). The standards group recommended that a standard on software acceptance should address tool requirements. One tool type that had priority in discussions was a tracking tool, either for tracing requirements and test documentation, or for tracking test progress and results. All the groups preferred a highly-integrated tool set, but as emphasized by the research group, more research is needed before an effective integrated tool set will reach the marketplace.

The management of software acceptance begins with the requirements for both custom and off-shelf purchases. Initial management includes planning based on various analyses (e.g, risk, cost/benefit). Risk concerns were voiced strongly as a primary responsibility of management in determining the amount of resources to devote to software acceptance. Communication was cited as an important means of alleviating frustrations usually encountered during software acceptance. Good management practice includes incorporating practices that lead to the effective communication of management decisions to the technical staff, communication among the technical staff members, and communication from the technical staff to management. Management also provides support by monitoring and controlling resources. In particular, resources (e.g., tools, training, staff time) must be made available so as to ensure the timely completion of software acceptance.

Problems in software acceptance testing experienced by some workshop participants frequently found solutions from discussion of other participants' more successful experiences in similar situations. End users who perform software acceptance tests were more successful when they had standardized internal procedures, adequate training, and software engineering practices in place for testing. The consensus of all the groups is that it is management's task to ensure the existence of this type of environment for end user software acceptance testing. Careful selection of automation may also contribute to successes; however, both research and development are needed to create adequate tools for software acceptance.

While most groups agreed that more deliverables should be evaluated for acceptance, and that the evidence that acceptance criteria have been met may come in varied forms, all groups had difficulty quantifying criteria. Research in human factors is important in almost all areas (e.g., for transforming requirements to usable forms, for understanding and quantifying maintainability, for acceptable user documentation). Identification and involvement of the end user when the requirements are defined is also an important step toward improving the software acceptance process.

The life-cycle approach to software acceptance includes both custom and off-shelf software, but generally from different perspectives. In both cases, development testing may contribute to the confidence that the software meets its requirements. For custom software, the purchaser should be tracking the development testing as performed by the developer and as required by the contract. For off-shelf software, the vendor should be employing good software engineering practices, including software testing during development. Purchasers will want to know how the testing was done. Finally more research is needed to understand how results of development testing can be used as part of the acceptance process for off-shelf software.

For acceptance of off-shelf software, vendors and customers are both involved but perhaps serially rather than in parallel as for custom software. Thus the life-cycle acceptance process is different. Vendors should follow the acceptance practices during development as though they were their own customers. Customers must then match existing software characteristics of a product to their needs. The customers need the confidence that the product meets its advertised characteristics. One approach to off-shelf testing is to standardize the basic specifications for a generic off-shelf software product or accept a *de facto* standard for a specific product type. Then conformance tests could be executed to demonstrate conformance to that standard by products from different vendors. If a standard for a product type did not exist, users would use other test procedures (e.g., benchmarking). When off-shelf products are to be tailored for a customer, the recommended practices for acceptance of custom software should be implemented. Whether the off-shelf product is accepted as is, or is modified to customers' needs, an important component is its integration into an existing hardware and software environment. This integration is easier to quantify than other acceptance criteria. Hence the development of guidelines to aid software acceptance of off-shelf software products is feasible.

Given the state of research and development in software acceptance, the best approach is to develop a guide, not a standard, for software acceptance. A guide would provide options. It would describe the types of information needed for planning and managing the acceptance of software at critical points in the lifecycle. Evaluation methods and procedures, and the automated aids to support them, have not matured sufficiently to be standardized for all software.

5.0 REFERENCES

[DOD2167]

"Military Standard Defense System Software Development," DOD-MIL-STD 2167, Washington, D.C., 1985.

[FIPS38]

"Guidelines for Documentation of Computer Programs and Automated Data Systems," National Bureau of Standards FIPS PUB 38, 1976.

[FIPS64]

"Guidelines for Documentation of Computer Programs and Automated Data Systems for the Initiation Phase," National Bureau of Standards FIPS PUB 64, 1979.

[FIPS99]

"Guideline: A Framework for the Comparison of Software Development Tools," National Bureau of Standards FIPS Pub 99, 1983.

[FIPS101]

"Guideline for Lifecycle Validation, Verification, and Testing of Computer Software," Software," National Bureau of Standards FIPS PUB 101, 1983.

[FIPS105]

"Guideline for Software Documentation Management," National Bureau of Standards FIPS PUB 105, 1984.

[FIPS106]

"Guideline on Software Maintenance," National Bureau of Standards FIPS PUB 106, 1984.

[IE729]

"IEEE Standard Glossary of Software Engineering Terminology," ANSI/IEEE Std. 729-1983, The Institute for Electrical and Electronics Engineers, Inc., 345 East 47th St., New York, NY 10017.

[IE730]

"IEEE Standard for Software Quality Assurance Plans," ANSI/IEEE Std. 730-1984, The Institute for Electrical and Electronics Engineers, Inc., 345 East 47th St., New York, NY 10017.

[IE829]

"IEEE Standard for Software Test Documentation," ANSI/IEEE Std.829-1983, The Institute for Electrical and Electronics Engineers, Inc., 345 East 47th St., New York, NY 10017.

[IE1012]

"IEEE Standard for Software Verification and Validation Plans," IEEE Std.1012, The Institute for Electrical and Electronics Engineers, Inc., 345 East 47th St., New York, NY 10017.

[NBS3407]

Wallace, Dolores R., "An Experiment in Software Acceptance Testing," NBSIR 86-3407, National Bureau of Standards, Gaithersburg, MD 20899, July, 1986.

[NBS136]

Wallace, Dolores R., "An Overview of Computer Software Acceptance Testing," NBS SP 500-136, National Bureau of Standards, Gaithersburg, MD 20899, February, 1986.

[RADC]
Bowen, Thomas P., et al, "Specifications of Software Quality Attributes," RADC-TR-85-37, 3 vols., Rome Air Development Center, Griffiss AFB, NY 13441-5700, February, 1985.

6.0 APPENDIX. ATTENDEES

W. Richards Adrion, National Science Foundation
Barbara Audrey, US Army
Steve Barnum, Department of Commerce
John Bielski, ASCI
John Biggie, Dept. of Health and Human Services
Peggy Brouse, Mitre Corporation
Nander Brown, Nander Brown & Company
Robert B. Bunting, Department of Education
Lee Burke, Treasury Department
John Cherniavsky, Georgetown University; National Bureau of Standards
Lori Clarke, University of Massachusetts, Amherst
Alison Cleary, Data General
Al Coblentz, Applied Info Develop Inc.
John Cochran, General Services Administration
Vincent Dell'Orto, Internal Revenue Service
Bill Dupras, NESTAR Systems
Charles Eater, Veterans Administration
Walter Ellis, IBM Federal Systems Division
Doris Fairbanks, Internal Revenue Service
William H. Farr, Naval Surface Weapons Center
Richard Fath, Federal Communications Commission
Jean-Philippe Favreau, National Bureau of Standards
Sheila Frankel, National Bureau of Standards
Lela Franklin, Central Intelligence Agency
Charlotte A. Gallagher, Data Systems, Inc.
David Gelperin, Software Quality Engineering
Al Grau, Social Security Administration
Keith L. Hatfield, International Bureau of Software Testing
Harry Heffernan, Government Computer News
Stephen J. Hirsch, National Security Agency
Mei-Cheng Hu, IBM/FSD
Millie Ingels, Department of Commerce
Marvin L. Kelliebrew, Department of Commerce
Rick Kuhn, National Bureau of Standards
Cyr Linonis, Bureau of the Census
Phil Marriott, Computer Technology Assoc.
Roger Martin, National Bureau of Standards
Ronnie Martin, Georgia Institute of Technology
Gene McDowell, Department of Commerce/NOAA
Walter D. Medley Jr., Government Services Administration
Susan Menke, Federal Times
Robert Munsey, Bureau of the Census
A. J. Neumann, National Bureau of Standards
Joe Nichols, Small Business Administration
Wilma Osborne, National Bureau of Standards
Tom Ostrand, Siemens Corporation

Mark Palmer, National Bureau of Standards
N. Bea Parker, Bureau of the Census
Betty Paul, SAIC
Jim Petro, Social Security Administration
John Rachac, Sperry Corporation
Jerry Raveling, Sperry Corporation
Donna Reale, Internal Revenue Service
Samuel T. Redwine, Jr., Institute for Defense Analyses
Brenda Rigg, Pansophic Systems Inc.
Ron Rosh, Internal Revenue Service
George Sauer, Aspen Systems Corp.
Dan Schneider, U.S. Dept. of Justice
David M. Siefert, NCR
Mike Slingluff, Federal Home Loan Mortgage Co.
Louis Smith, Bell Communications Research
Tom Stengle, NASA Goddard Space Flight Center
John Sullivan, Nuclear Regulatory Agency
Keiji Tasaki, NASA Goddard Space Flight Center
Marvin L. Thomas, U.S. Army
Henry Utz, Department of Commerce
Dolores Wallace, National Bureau of Standards
Elaine Weyuker, Courant Institute of Mathematical Sciences, NYU
Annette Winward, American Automobile Association
William Wong, National Bureau of Standards
Steven Zeil, University of Massachusetts, Amherst

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. NBS/SP-500/146	2. Performing Organ. Report No.	3. Publication Date March 1987

4. TITLE AND SUBTITLE

Computer Science and Technology:
Report on the NBS Software Acceptance Test Workshop, April 1-2, 1986

5. AUTHOR(S)

Dolores R. Wallace
John C. Cherniavsky

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234
Gaithersburg, MD 20899

7. Contract/Grant No.

8. Type of Report & Period Covered

Final

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
GAITHERSBURG, MD 20899

10. SUPPLEMENTARY NOTES

Library of Congress Catalog Card Number 87-619806

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

This document is a report on the Software Acceptance Test Workshop held at the National Bureau of Standards, April 1-2, 1986. The workshop consisted of eight sessions divided over two days. The topics of the first day's sessions were acceptance testing of off-shelf software, test case selection techniques, automated support for software acceptance testing, and software acceptance criteria. The topics of the second day's sessions were the management of software acceptance testing, standardization issues in software acceptance testing, research areas for software acceptance testing, and the state of practice in software acceptance testing. This report describes the charges given to all of the sessions, highlights of discussions from each of the sessions, and the conclusions of the workshop. This report is intended for those who purchase, market, develop or maintain software and for those who are responsible for software acceptance testing.

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

automated tools; custom software; management; off-shelf software; research; software acceptance criteria; software acceptance testing; standardization; test case selection; test planning; test practices.

13. AVAILABILITY

☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☒ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

☐ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

50

15. Price

ANNOUNCEMENT OF NEW PUBLICATIONS ON COMPUTER SCIENCE & TECHNOLOGY

Superintendent of Documents,
Government Printing Office,
Washington, DC 20402

Dear Sir:

Please add my name to the announcement list of new publications to be issued in the series: National Bureau of Standards Special Publication 500-.

Name _____

Company _____

Address _____

City _____ State _____ Zip Code _____

(Notification key N-503)

NBS Technical Publications

Periodical

Journal of Research—The Journal of Research of the National Bureau of Standards reports NBS research and development in those disciplines of the physical and engineering sciences in which the Bureau is active. These include physics, chemistry, engineering, mathematics, and computer sciences. Papers cover a broad range of subjects, with major emphasis on measurement methodology and the basic technology underlying standardization. Also included from time to time are survey articles on topics closely related to the Bureau's technical and scientific programs. Issued six times a year.

Nonperiodicals

Monographs—Major contributions to the technical literature on various subjects related to the Bureau's scientific and technical activities.

Handbooks—Recommended codes of engineering and industrial practice (including safety codes) developed in cooperation with interested industries, professional organizations, and regulatory bodies.

Special Publications—Include proceedings of conferences sponsored by NBS, NBS annual reports, and other special publications appropriate to this grouping such as wall charts, pocket cards, and bibliographies.

Applied Mathematics Series—Mathematical tables, manuals, and studies of special interest to physicists, engineers, chemists, biologists, mathematicians, computer programmers, and others engaged in scientific and technical work.

National Standard Reference Data Series—Provides quantitative data on the physical and chemical properties of materials, compiled from the world's literature and critically evaluated. Developed under a worldwide program coordinated by NBS under the authority of the National Standard Data Act (Public Law 90-396). NOTE: The Journal of Physical and Chemical Reference Data (JPCRD) is published quarterly for NBS by the American Chemical Society (ACS) and the American Institute of Physics (AIP). Subscriptions, reprints, and supplements are available from ACS, 1155 Sixteenth St., NW, Washington, DC 20056.

Building Science Series—Disseminates technical information developed at the Bureau on building materials, components, systems, and whole structures. The series presents research results, test methods, and performance criteria related to the structural and environmental functions and the durability and safety characteristics of building elements and systems.

Technical Notes—Studies or reports which are complete in themselves but restrictive in their treatment of a subject. Analogous to monographs but not so comprehensive in scope or definitive in treatment of the subject area. Often serve as a vehicle for final reports of work performed at NBS under the sponsorship of other government agencies.

Voluntary Product Standards—Developed under procedures published by the Department of Commerce in Part 10, Title 15, of the Code of Federal Regulations. The standards establish nationally recognized requirements for products, and provide all concerned interests with a basis for common understanding of the characteristics of the products. NBS administers this program as a supplement to the activities of the private sector standardizing organizations.

Consumer Information Series—Practical information, based on NBS research and experience, covering areas of interest to the consumer. Easily understandable language and illustrations provide useful background knowledge for shopping in today's technological marketplace.

Order the **above** *NBS publications from: Superintendent of Documents, Government Printing Office, Washington, DC 20402.*

Order the **following** *NBS publications—FIPS and NBSIR's—from the National Technical Information Service, Springfield, VA 22161.*

Federal Information Processing Standards Publications (FIPS PUB)—Publications in this series collectively constitute the Federal Information Processing Standards Register. The Register serves as the official source of information in the Federal Government regarding standards issued by NBS pursuant to the Federal Property and Administrative Services Act of 1949 as amended, Public Law 89-306 (79 Stat. 1127), and as implemented by Executive Order 11717 (38 FR 12315, dated May 11, 1973) and Part 6 of Title 15 CFR (Code of Federal Regulations).

NBS Interagency Reports (NBSIR)—A special series of interim or final reports on work performed by NBS for outside sponsors (both government and non-government). In general, initial distribution is handled by the sponsor; public distribution is by the National Technical Information Service, Springfield, VA 22161, in paper copy or microfiche form.

U.S. Department of Commerce
National Bureau of Standards
Gaithersburg, MD 20899