NIST Special Publication 500-180

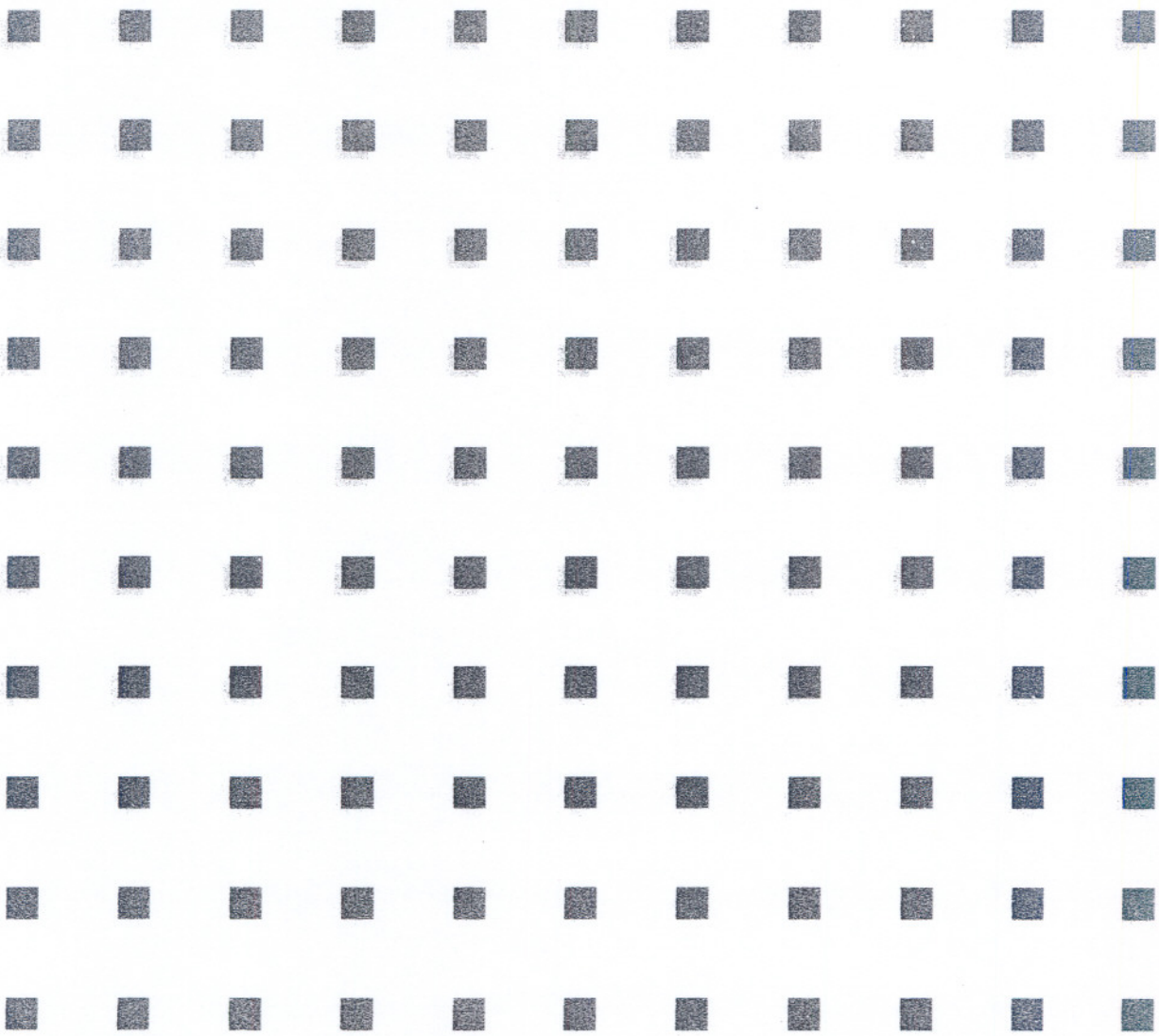
Computer Systems Technology

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology

NIST

Guide to Software Acceptance

Dolores R. Wallace
John C. Cherniavsky



NIST Special Publication 500-180

Guide to Software Acceptance

Dolores R. Wallace
John C. Cherniavsky*

National Computer Systems Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899

*Professor of Computer Science
Georgetown University
Washington, DC 20057

April 1990

U.S. DEPARTMENT OF COMMERCE
Robert A. Mosbacher, Secretary
NATIONAL INSTITUTE OF STANDARDS
AND TECHNOLOGY
John W. Lyons, Director

Reports on Computer Systems Technology

The National Institute of Standards and Technology (NIST) (formerly the National Bureau of Standards) has a unique responsibility for computer systems technology within the Federal government. NIST's National Computer Systems Laboratory (NCSL) develops standards and guidelines, provides technical assistance, and conducts research for computers and related telecommunications systems to achieve more effective utilization of Federal information technology resources. NCSL's responsibilities include development of technical, management, physical, and administrative standards and guidelines for the cost-effective security and privacy of sensitive unclassified information processed in Federal computers. NCSL assists agencies in developing security plans and in improving computer security awareness training. This Special Publication 500 series reports NCSL research and guidelines to Federal agencies as well as to organizations in industry, government, and academia.

National Institute of Standards and Technology Special Publication 500-180
Natl. Inst. Stand. Technol. Spec. Publ. 500-180, 42 pages (Apr. 1990)
CODEN: NSPUE2

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1990

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

ABSTRACT

Software acceptance is a life cycle process which includes acceptance of interim and final software products for both new and maintained software systems. This guide assists buyers in understanding acceptance issues relative to a basic life cycle model and some of its variants. The guide identifies six categories (functionality, performance, interface quality, overall quality, security, and safety) for which acceptance criteria must be defined. The guide identifies issues to be considered when in establishing acceptance criteria. Finally the guide directs managers in planning and implementing a software acceptance program, with emphasis on the final software acceptance testing.

KEYWORDS: acceptance categories; acceptance criteria; life cycle models; software acceptance; software acceptance plan; software acceptance testing; software product assurance; software quality

PREFACE

The Guide to Software Acceptance is intended to assist buyers and developers in preparing for software acceptance. It addresses software acceptance of interim, as well as final, software products. This Guide:

- defines software acceptance;
- establishes procedures for acceptance of interim and final products;
- establishes software acceptance as a life cycle process for both new and evolving software systems;
- clarifies the role of software acceptance testing as part of software acceptance; and
- identifies generic criteria for products to be presented for final acceptance.

The Government increasingly relies on software to administer services, to manage its programs, and to control products which its personnel use. Industrial users rely on software systems for inclusion in their products or to aid in designing and building the products, to administer their services, and to manage their companies. The ability to compete in a world economy requires reliable software systems. Improvement in both Government's and industry's software practices improves the ability to support the nation's economy and to compete in a highly competitive world market.

The use of this guide is recommended for the managers and technical staffs of organizations that provide or purchase software systems or that perform software product assurance activities. Software product assurance organizations typically provide services for software quality assurance, software configuration management, and software verification, validation, and testing (VV&T).

The use of this guide is also recommended to the management of companies who produce off-the-shelf software as their business product. These managers are responsible for releasing their companies' software products to the marketplace. By applying software acceptance during the development of their products, they may contribute to their companies' success which depends upon the quality and timeliness of the developed software.

CONTENTS

TABLES

1. INTRODUCTION

Computers and their associated software control services that affect life, property, and the national defense. Software (referred to as software system or system) also supplies services that allow companies to function in highly competitive business environments. The systems must exhibit a high level of reliability if a business is to compete in the world-wide marketplace. In spite of substantial resources invested in system development and maintenance, delivered systems often do not meet requirements for operating capability, reliability, overall software quality, security, and software safety [1].

Software acceptance testing at delivery is usually the final opportunity for the buyer to examine the software and to seek redress from the developer for insufficient or incorrect software. Frequently the software acceptance test period is the only time the buyer is involved in acceptance and the only opportunity the buyer has to identify deficiencies in a critical† software system. This exposes the buyer to the considerable risk that a needed system will never operate reliably (because of inadequate quality control during development). To reduce the risk of problems arising at delivery or during operation, the buyer must become involved with software acceptance early in the acquisition process.

Software acceptance requires procedures for identifying acceptance criteria for interim life cycle products and for accepting them. Final acceptance not only acknowledges that the entire software product adequately meets the buyer's requirements but also acknowledges that the process of development was adequate. As a life cycle process, software acceptance enables:

- early detection of software problems (and time for the buyer to plan for possible late delivery);
- preparation of appropriate test facilities; and
- early consideration of the user's needs during software development.

Based on the results of a Software Acceptance Test Workshop [2], this guide presents information on software acceptance testing [3] and applies its fundamental concepts to the larger process of software acceptance. This process may need to be modified for several reasons, including:

- The user's organization may not be prepared to expend the effort for a complete acceptance activity.
- The project for which this guide is to be used may already be in progress.
- Off-the-shelf software or other reusable software may be a major component of the software system.

While a software acceptance program requires a major effort on the part of the accepting organization, the organization may not be able to expend this effort

† The term *critical* conveys the meaning of economic or social catastrophe (e.g., loss of life) and in this guide also conveys the meaning of strategically vital to an agency's or company's long term economic welfare.

because of insufficient staff, time, or experience. A possible solution is to contract a third party to develop and manage an acceptance program. A variation is to require developers to provide data from their software product assurance activities; this approach depends on the developer's performance on previous software system projects and on the scope of the current project. The accepting organization remains responsible for acceptance decisions when oversight is delegated.

While the approach to software acceptance defined in this guide assumes the buyer has built acceptance considerations into the project from its beginning, it can be adapted to situations in which a project has not been started with an acceptance program and has encountered difficulties (e.g., missed deadlines or poor performance on a demonstration test). This may require new contractual agreements, especially to obtain information concerning the requirements, design, and software assurance activities of the developer. The buyer may have to adjust schedules to allow time for construction of acceptance tests.

Existing guidelines on selecting third party software and on software acceptance testing should be applied to off-the-shelf software and other reusable software prior to acceptance of the software system [4,3]. This guide also applies for the adaptation of off-the-shelf packages to a buyer's environment.

The software system acceptance activities described in this guide can be used in any development environment. The guide describes how differences in development environments and life cycle models may affect software acceptance activities and concerns. Life cycle models and environments include a basic model developed from the waterfall model [5,6,7]†, the spiral model [8], the evolutionary model [9], prototyping [10], and application generators [11,12].

This document provides a framework from which managers may identify their requirements for software acceptance, including:

- special issues dependent on methodology and project features;
- products for acceptance;
- acceptance criteria;
- acceptance reviews;
- acceptance testing; and
- acceptance plan, including the acceptance test plan.

1.1. Terminology

The term *accepted* and its variants are used in several contexts. Accepted means that interim and final software products are examined to determine whether they meet specific criteria. If they do, then they have passed acceptance

† The description of the waterfall model was presented by W. W. Royce in "Managing the Development of Large Software Systems: Concepts and Techniques" at IEEE WESCON, August 1970, reprinted in *IEEE TUTORIAL: Software Engineering Project Management* and in *Proceedings of the Eighth International Conference on Software Engineering.* The references in this guide describe variants appearing in Federal guidelines and standards.

and may be accepted. The term *acceptance decision* refers to the decision made after a software product has been evaluated; the decision may include rejection, partial acceptance, or total acceptance. Within the context of *acceptance test* activities, contractual acceptance of a software product means payment or release from further development tasks, and implies that the developer has satisfied the buyer that the completed system is performing correctly. Some operational testing may occur as part of this process.

Software system refers to the software, with all associated documentation and support tools, operating in its intended environment. In many cases (e.g., weapons systems), testing of the complete system in the intended environment may be beyond the scope of the software testing addressed in this guide. The size of the software systems considered in this document range from a few thousand to many millions of lines of instructions. This document does not address the initial selection and certification issues of off-the-shelf software for use in larger systems or as stand-alone systems; these issues are discussed in [4].

In this guide the terms organization, developer, buyer, user, and manager refer to specific participants in the software development and acceptance process. An *organization* is an individual or a group of individuals responsible for some aspect of software development, maintenance, product assurance, or acceptance. The *developer* is the organization responsible for the development of the software. The *buyer* is the organization representing those who need and may use the system and those who are associated with the contractual procurement of the system. (The *buyer* may be the organization responsible for marketing a product.) The *user* is the organization that will be using the software system. The *manager* is the organization responsible for acquiring and accepting the software. When the buyer and manager are different, the buyer has final accountability for the acceptance decision.

Buyer and user may be the same organization or there may be a separation between procurement and actual users. The user and developer may be the same (e.g., systems software developed by the systems staff). The users may be personnel with expert or non-expert skills related to software evaluation. The buyer may be the contracting officer. The persons determining the quality of the software may be systems analysts who will not actually use the procured system. The manager may be representatives of the user or experts hired as *system integrators* or other contractual experts. System integrators are independent organizations who are hired to manage the complete software acquisition; this task may include responsibility for overseeing the software acceptance process. This guide does not assign specific organizations for various product assurance activities; each project establishes those relationships according to project needs. This guide encourages the buyer to ensure user involvement throughout the software acceptance process.

2. SOFTWARE ACCEPTANCE

Software acceptance is an incremental process of approving or rejecting software systems during development or maintenance†, according to how well the software satisfies pre-defined criteria. Acceptance decisions occur at pre-specified times at which processes, support tools, interim documentation, segments of the software, and finally the total software system must meet pre-defined criteria for acceptance. Subsequent changes to the software may affect previously accepted elements. The final acceptance decision occurs with verification that the delivered documentation is adequate and consistent with the executable system and that the complete software system meets all buyer requirements. This decision is usually based on *software acceptance testing:* "formal testing conducted to determine whether a software system satisfies its acceptance criteria and to enable the buyer to determine whether to accept the system" [3]. Formal final software acceptance testing must occur at the end of the development process. It consists of tests to determine whether the developed system meets predetermined functionality, performance, quality, and interface criteria. Criteria for security or safety may be mandated legally or by the nature of the system.

Software acceptance also provides development managers a tool for monitoring development of quality software. Developers may apply measurement criteria to results from the product assurance activities (e.g., configuration management, quality assurance, and VV&T) to support decisions regarding the progress of software development activities.

2.1. Acceptance Reviews

Software acceptance is specified in a formal plan during the earliest activity of the software life cycle. The software acceptance plan identifies products for acceptance, the specific acceptance criteria, acceptance reviews, and acceptance testing throughout the entire life cycle.

Acceptance decisions need a framework in which to operate; items such as contracts, acceptance criteria, and formal mechanisms are part of this framework. The software acceptance must state or refer to specific criteria that products must meet in order to be accepted. A principal means of acceptance in the development of critical software systems (e.g., weapon systems, defense information networks, robotic control software) is a periodic review of interim software documentation and other software products [13].

† In this guide, for the purpose of software acceptance the activities of software maintenance are assumed to share the properties of software development. "Development" and "developer" include "maintenance" and "maintainer."

Table 1. Titles of requirements documents

- Functional Requirements Description; Data Requirements Specification [7]
- Functional Description [14]
- Software Requirements Specification [15, 16]
- Software Requirements Specification; Interface Requirements Specification [6]
- Software Requirements Specification; Information System Product Specification; External Interface Requirements Data Item Description (DID) [17]

A disciplined software acceptance program for software of any type may use acceptance reviews as a formal mechanism. When the acceptance decision requires change, another review is necessary to ensure that the required changes have been implemented and properly configured and that any affected pieces pass acceptance. For large or complicated projects several reviews may be necessary during the development of a single product. This guide does not specify the specific names of products for review because, as illustrated in table 1, the formal names of products vary according to the life cycle and standards applicable to a project. At the least, reviews should occur for software requirements, software designs, source code, system integration, and acceptance testing [18].

2.2. Software Products for Acceptance

While software acceptance activities apply to all products of the software system, scheduled activities may vary according to the type of product. Examples of product types are listed in table 2. The guidance in this document applies primarily to development products but may sometimes apply to other product types. Acceptance activities for off-the-shelf software, support software, and environment products occur early in the system planning activities. For selection and acceptance of many support products, (e.g., whether a specific design tool is appropriate), the issues addressed in chapter 5 of this guide and in [4] must be considered. Tools acquired to support software acceptance testing must pass acceptance. Guidelines of [4] apply to off-the-shelf products that will be used as stand-alone systems and to products considered for reuse. Acceptance activities identified in this guide apply to *changes* to off-the-shelf packages and reusable software and their integration into the software system.

Table 2. Generic software products for acceptance

Product Type	Examples
• Development Products	Software Requirements, Maintenance Manual
• Process Products	Development Schedules, Acceptance Criteria
• Support Products	Design Tools, Application Generators
• Environment Products	Operating Systems, Databases
• Assurance Products	Test Plan, Test Case Generators

The buyer organization may accept software products produced by the buyer or other organizations for project management, software product assurance activities, and software acceptance. Acceptance activities may also apply to the receipt of documents, the documentation of software product assurance activities by the developer and other organizations, intermediate product testing (usually supplied and performed by the developer), results or summaries of intermediate product testing, software quality criteria measures of acceptance, the results or summaries of evaluating the software against the criteria, and the final acceptance test plans. The software acceptance plan describes the options available for acceptance decisions and the documentation requirements for accepting a product that has not met all of the acceptance criteria.

2.3. Buyer's Role in Software Acceptance

Accountability for software acceptance belongs to the buyer, whose responsibilities are listed in table 3. The buyer may delegate some responsibilities to an acceptance manager who may be a user, a systems integrator, a developer, or some other third party (e.g., V&V). The buyer must be actively involved in defining the type of information required, evaluating that information, and deciding at various points in the development activities if the products are ready for progression to the next activity. Preparation activities, as indicated in table 4, enable the buyer to fulfill software acceptance responsibilities.

Table 3. Buyer responsibilities

- Ensure user involvement in developing system requirements and acceptance criteria.
- Identify interim and final products for acceptance, their acceptance criteria, and schedule.
- Plan how and by whom each acceptance activity will be performed.
- Plan resources for providing information on which to base acceptance decisions.
- Schedule adequate time for buyer staff to receive and examine products and evaluations prior to acceptance review.
- Prepare the acceptance plan.
- Respond to the analyses of all project entities before accepting or rejecting.
- Approve the various interim software products against quantified criteria at interim points.
- Perform the final acceptance activities, including formal acceptance testing, at delivery.
- Make an acceptance decision for each product.

The buyer (through the acceptance manager) identifies the software products and the criteria for software acceptance. Acceptance of the system requirements must involve examining data from all means appropriate to the project (e.g., requirements verification, prototyping) to ensure that they adequately represent the user's needs. Acceptance criteria derived from these requirements quantitatively describe the functional, interface, performance, and quality measurements that the software must satisfy. Depending on the specific project, security and safety criteria may be necessary.

While the user or a third party may perform many of the planning and acceptance test activities, the buyer's technical role, manager, is non-trivial. The

buyer must ensure that all planning activities and documentation are complete. The buyer must either perform the acceptance tests or observe the tests.

The acceptance manager is responsible for ensuring test procedures have been completely defined and implemented and that all resources are ready for the acceptance testing. In the case of final acceptance testing, the tests should be performed in the operational environment. If that is impossible, contingencies must be built into the acceptance agreement to ensure that the developer is responsible for deficiencies discovered in the operational environment.

Final acceptance of software based on software acceptance testing usually means that the contract and project are completed, with the exception of any caveats or contingencies at acceptance. Final payment for the software occurs and the developer has no further development obligations. (Of course there may be maintenance agreements but these are a separate issue.)

Table 4. Buyer preparation for acceptance

- Acquire full knowledge of the application for which the system is intended.
- Become fully acquainted with the application as it is currently accomplished by the buyer's organization.
- Identify similar applications in other organizations.
- Understand the risks and benefits of the software development methodology that is to be used in building the software system.
- Fully understand the consequences of adding new functions to enhance an existing system.

3. SOFTWARE DEVELOPMENT STAGES AND ACCEPTANCE

This guide uses the basic model of software development stages of FIPSPUB132 [19] as a basis for software acceptance. Variations which either modify the basic model to incorporate more of the process of software development or to incorporate new technology to improve software development are addressed. Table 5 lists the stages of the basic model. Table 6 describes several variations of the basic model and their characteristics. Discussion of variations which concern the management of the software development process is outside the scope of this guide.

In the basic model, software development is partitioned into distinct stages (table 5). The end of each stage is an interim acceptance point. For large projects, there may be several acceptance points within a stage. The variations of the basic model also generate acceptance documentation at similar points. The variations differ in the mechanism that creates the documentation and in the timing of the acceptance points. For example, rapid prototyping, while possibly delaying the development of a requirements document, may lead to a requirements specification or may evolve directly into a system that accurately reflects a user's needs [9].

Table 5. The basic model stages

- Concept Stage
- Requirements Stage
- Design Stage
- Coding Stage
- Integration and Test Stage
- Final Acceptance Test Stage
- Maintenance and Enhancement Stage

Variations on the basic model incorporate risk management strategies. The spiral variant captures uncertain technology (go/no go decisions) and explicitly allows for changes in early stages. Variants using incremental or evolutionary development reflect constrained technology or resources [9]. Development starts with a simple system and adds functionality with each successive version. If the functionality is planned, the variant is incremental; if the functionality is not

completely known at the beginning of system construction, it is evolutionary. Both are referred to as evolutionary. The prototyping variant fully integrates the buyer's requirements into the development of requirements specifications and is especially important if the user interface is a major component of the system. Variants that emphasize the reuse of software focus on functionality (and cost) concerns. Variants using fourth generation languages increase productivity through more expressive languages which often require less code than third generation languages. These variants may also be combined. Rapid prototyping, for example, is very useful in the evolutionary variant.

Table 6. Basic model variations

Variation	Description
Prototype	Software development with prototyping defines requirements by providing programs that simulate functionality [10]. A system that uses prototyping for the entire software development refines the programs into a fully working system.
Fourth Generation	Fourth generation languages are very high level languages for specific application domains [11,12]. Their defining characteristic is a major reduction in the amount of code to be written. Performance issues may arise [20,21].
Evolutionary	The evolutionary variant explicitly addresses the development of successive versions or enhancements, with considerable reuse of requirement and design documents [9]. It may require extensive use of rapid prototyping, reverification and regression testing.
Spiral	The spiral variant explicitly includes acceptance points at stages of the basic model where decisions, with documented rationale, are made whether or not to continue development [8]. The decisions are frequently technology dependent.
Reusable	Software reuse either reuses generic software, where interfaces must be constructed to other parts of the system, or is primarily a modification process where generic software is modified in constructing the software system. This includes modification of off-the-shelf software [22].

Each variant in developing a software system may require a different focus in the acceptance process; several are indicated in table 7. The stages in the basic model define points at which products can be examined for acceptance. The variants produce similar products, but the acceptance procedures may be somewhat different. In all cases the acceptance plan must provide for iteration throughout the development process and in the acceptance activities.

A development process that emphasizes early system definition (e.g., prototyping) requires that the buyer be especially active during the requirements definition. A development process using fourth generation languages requires vigilance in the acceptance of the fourth generation language because of the trust that will be placed in the functionality that is automatically generated. Care in designing the system for performance may be needed, with performance testing of the resultant software system. The use of the evolutionary variant requires the production of robust regression tests and some measure of system extensibility. The use of the spiral variant must provide for tests for changing requirements and specifications; configuration control is particularly important. When system development takes advantage of extensive software reuse, interface testing and performance testing of generic software are important.

Table 7. Acceptance with basic model and variants

Model or Variant	*Sampling of Acceptance Concerns*
• Basic	Acceptance at all stages of the basic model.
• Prototype	Special attention towards the performance of the prototyped system.
• Fourth Generation	Certification of the fourth generation development system. Performance of the final system.
• Evolutionary	Testing for extensibility and regression testing.
• Spiral	Quantifiable tests concerning project feasibility.
• Reusable	Performance and interface testing.

4. SOFTWARE ACCEPTANCE ACTIVITIES AND CRITERIA

Acceptance decisions for software products usually occur at, or as a result of, major reviews when products and activities are completed (e.g., the requirements document, the architectural design, integration testing, acceptance testing). The buyer and acceptance manager evaluate the products and other product information from the developer and other organizations. The information includes results of software product assurance activities which are matched against acceptance criteria. The acceptance criteria specify the values that a product must meet for acceptance (e.g., a performance requirement that a function must be executed within one second).

Sometimes buyers may schedule reviews of partial products, acknowledging that documents or products have been received and tentatively accepting them until all evaluation data have been analyzed. For example, the buyer may accept a draft of the user manual at the requirements or design review and accept the final version at installation or final acceptance testing of the system. Demonstrations (e.g., prototype) may sometimes serve as an acceptance activity. Because some products may not be fully accepted, procedures for iteration of preceding development activities and for acceptance of the changes may need to be established. Buyers must follow configuration control procedures, even for the occasional times that products are evaluated at some time other than a formal review. At the minimum, an acceptance activity must be scheduled after system or operational testing or upon installation of off-the-shelf software, depending upon the nature of the software system.

Typical acceptance decisions include:

- Changes are required and accepted before progression to the next activity.
- Some changes must be made and accepted before further development of that section of the product, but other changes may be made and accepted at the next major review.
- Progress may continue and changes are to be accepted at next review.
- No changes are required and progress may continue.

While the goal is to achieve and accept only perfect software, more likely some criteria will not be completely satisfied for each product. The buyer may choose to accept software with unsatisfied criteria. The buyer must establish values in advance for individual requirements and for collections of requirements (e.g., design description may have no more than five module descriptions with missing information on constraints). Many requirements of the latter category are quantitative requirements for quality attributes.

The remainder of this section describes the steps for establishing and using acceptance criteria as outlined in table 8.

Table 8. Establishing and using acceptance criteria

- Identify system requirements for categories of acceptance requirements.
- Determine the system criticality and that of its components.
- Identify quantifiable measures of system requirements.
- Establish acceptance criteria for each measure for each product.
- Collect and analyze evaluation data.

4.1. Categories for Acceptance

Acceptance requirements that a system must meet can be divided into six categories as listed in table 9. Requirements for functionality relate to the functions that the system must execute. Requirements for performance relate to operational requirements such as time or resource constraints. Interface quality requirements relate to any interface (e.g., human/machine, module/module). Overall software quality requirements are those that specify limits for factors or attributes such as reliability, testability, correctness, usability [23, 24, 25, 26, 27, 28, 29, 30, 31]. The criterion that a requirements document may have no more than five statements with missing information is an example of quantifying the quality factor of completeness. Security requirements relate to authorized access of system resources and to process integrity. Software is frequently used in systems whose failure could result in personal injury or death, that is, systems whose failure may affect public health and welfare. Software safety requirements are necessary when injury or death may occur as a result of system failure.

Table 9. Categories of acceptance requirements

- Functionality
- Performance
- Interface Quality
- Overall Software Quality
- Security
- Software Safety

Documentation for software acceptance requirements is not always organized according to these six categories. For example, in the past security and software safety have been addressed as software quality attributes. The Computer Security Act [32] requires all Federal agencies to identify computer systems that contain sensitive information. For the identified systems explicit requirements dealing with security are appropriate. For systems where safety is a concern, separate requirements for safety are important. Regardless of the organization of the software requirements documentation, buyers must consider these six categories of acceptance criteria.

4.2. System Criticality

The criticality of a system is important in determining quantitative acceptance criteria. The buyer should determine the degree of criticality of the requirements in the six categories. By definition, all safety criteria are critical and by law, certain security requirements are critical [32]. Some typical factors affecting criticality include [33,34]:

- importance of system to agency or industry;
- consequence of failure;
- complexity of the project;
- technology risk; and
- complexity of the user environment.

Products, or pieces of products, with critical requirements do not qualify for acceptance if they do not satisfy their acceptance criteria. A product with failed noncritical requirements may qualify for acceptance depending upon quantitative

acceptance criteria for quality factors. Clearly if a product fails a substantial number of noncritical requirements, the quality of the product is questionable.

Examples in tables 10 and 11 illustrate how criticality changes according to project features. Table 10 lists some critical features for a software controlled control system in an airplane while table 11 lists some critical and non-critical features for a payroll system. In this example, the acceptance criteria for the airplane control system will be more rigorous than many criteria for the batch payroll system.

Table 10. Airplane control system criticality

- Functionality is critical to mission accomplishment.
- Performance is critical for control system response times.
- Interface quality is critical for flight ergonomics.
- Overall software quality is critical for some attributes but not all.
- Security is critical due to the possibility of sabotage.
- Software safety is critical because failure may cause injury or death.

Another example of different acceptance criteria for a similar performance requirement demonstrates the need to identify critical features for each software system. Two systems, a courthouse docket system and an embedded weapons control system, have a requirement for a screen to display information 1 second after the system receives it. The response of the software program for a courthouse docket clerk that displayed information 5 seconds after receipt under a full system load, but operated within 1 second for standard system load, is a noncritical performance failure. A 5-second response for a screen display for a weapons control software system contained in a airplane under any circumstances would be a critical failure if the 5-second delay meant the possible destruction of the airplane. In the case of the courthouse system, the acceptance criterion may allow acceptance if the system fails the performance requirement only under a full system load. For the airplane, the acceptance criterion would require rejection for any failure to meet the performance requirement.

Table 11. Batch payroll system criticality

- Numeric functionality is critical; other functions may be less important.
- Performance is noncritical unless the payroll cannot be met.
- Interface quality may not be critical at the ergonomic level.
- Overall software quality may be critical for maintainability.
- Security is critical due to the possibility of fraud.
- Software safety is probably not an issue.

4.3. Acceptance Criteria

The buyer has the responsibility of ensuring that functional, performance, software safety, security, and interface requirements contain numeric criteria. The buyer must also ensure that quality requirements are quantified, especially in the collective sense of assessing the acceptability of a product. The buyer must be careful in writing the contract. The buyer believes that the worst acceptable system is being defined; the developer uses the same criteria as the definition of the best system that will be produced for the buyer. Similarly a contract with absolute values for some criteria rather than a range of acceptable values could result in an expensive system or a system that would never satisfy the acceptance criteria. Monetary penalties may be effectively used to ensure that the failure of noncritical criteria is kept to a minimum.

Table 12 identifies some acceptance issues for each of the six generic categories with respect to the basic model stages. For a specific software system, buyers must examine their projects' characteristics and criticality in order to develop expanded tables of issues and concerns for acceptance of that software system. Some of the issues may change according to the part of development for which criteria are being defined. For example, for requirements the quality "testability" may mean that requirements are expressed in quantified specifications but for design and code "testability" may mean that test cases can be automatically developed. Successful development of acceptance criteria tables may lead to their adoption as a baseline for future acceptance products.

After the issues and concerns have been identified, the buyer must establish acceptance criteria, both for individual elements of a product and for summary information. These criteria should be the acceptable numeric values or ranges of values. The buyer should compare the established acceptable values against the number of problems presented at acceptance time. For example, if the number of inconsistent requirements exceeds the acceptance criterion, then the requirements document should be rejected. At that time, the established procedures for iteration and change control go into effect.

Table 12. Acceptance issues by category

Category	Sampling of Acceptance Issues
Functionality	Document and code consistency internally and between stages. Traceability of functionality. Adequate verification of logic. Functional evaluation and testing. Preservation of functionality in the operating environment.
Performance	Feasibility analysis of performance requirements. Correct simulation and instrumentation tools. Performance analysis in the operating environment.
Interface Quality	Interface documentation. Interface complexity. Interface and integration test plans. Interface ergonomics. Operational environment interface testing.
Overall Software Quality	Quantification of quality measures. Criteria for acceptance of all software products. Adequacy of documentation and software system development standards. Quality criteria for operational testing.
Security	Security requirements identification. Security test plans. Formal verification of security [35].
Software Safety	Identification of safety requirements. Fault tree construction and tracing. Elimination of development methods or technology inappropriate for safety-critical systems. Incorporation of safety interlocks and fail safe code to prevent and recover from potentially unsafe states.

4.4. Collection and Analysis of Data

Once the acceptance criteria are established, the buyer may be dependent on others to supply the evaluation information for the project's products. Even though the data are provided by others, the buyer has the responsibility of ensuring that the data are correct and provide appropriate information. The buyer has the ultimate responsibility for acceptance.

While error-free software is desirable, experience indicates that for large systems it is unattainable using current methods of software development. Instead, it is necessary to determine appropriate statistics to use in determining acceptability. It is outside the scope of this document to describe statistical methods and reliability models; information on these topics may be found in several documents [36, 37, 38, 39, 40]. The historical profile of discovery of errors throughout the software life cycle for similar systems may give some evidence of the number of errors that are likely remaining. This also applies to final acceptance testing where the criterion for acceptance may be the mean time between failures or the volume of errors.

The determination of what data to collect, how to collect it, and the analysis of that data to determine whether a quality metric has been satisfied is a difficult, though necessary, task. Since many metrics are simple counts of discrepancies or percentage calculations (e.g., 3 errors per thousand lines of code), many metric calculations can be automated. Whenever possible this should be done. Otherwise the manager will be inundated with data and will not be able to make a reasoned acceptance decision. Modern technology (e.g., application generators, project management tools) may provide mechanics for data collection and analyses. Some tools, especially application generators, have underlying databases and provide statistical functions. The Software Acceptance Test Workshop [2] recommended a minimum set of tools for software acceptance testing.

5. SOFTWARE ACCEPTANCE TESTING

While some software acceptance activities may include testing of pieces of the software, formal software acceptance testing is the point in the development life cycle that if the buyer accepts the software, then a contractual requirement between the buyer and seller has been met. Final software acceptance testing is the last opportunity for the buyer to examine the software for functional, interface, performance, security, software safety, and quality features, prior to the final acceptance review. The system at this time must include the delivered software, all user documentation, and final versions of other software deliverables. The review of software acceptance testing results is often the final step in the software acceptance process. Major differences exist in the responsibilities of buyers for acceptance of interim products and software acceptance testing. A contrast of key points from an overview of software acceptance testing [3] against other acceptance activities indicates how the buyers are involved at a detailed technical level for software acceptance testing. These key points include planning and administrative responsibilities, objectives, approach, location and automation requirements, staff responsibilities and documentation for software acceptance testing.

5.1. Planning and Administration

In software acceptance, buyers accept interim and final software products based on how well those products meet established acceptance criteria. For many acceptance activities, buyers approve the plans for development and software product assurance activities while relying on others to implement those plans and deliver the information (the products and evaluations of products) for their acceptance reviews. In contrast, two exceptions, prototyping and software acceptance testing, involve the buyer in administering the technical activities to prepare for acceptance. Sometimes users evaluate prototypes at various stages of development for acceptance and progression to the next stage; in these circumstances, prototyping should be conducted as rigorously as acceptance testing. In any case, the buyers have a direct role in:

- the planning and administration of software acceptance testing;
- the execution of the software acceptance tests; and
- the review of the test results to determine acceptance or rejection.

The planning and administration include:

- facility requirements;
- test documentation requirements; and
- staffing requirements.

These issues are addressed in more detail in table 13.

Table 13. Final acceptance management activities

- Planning and making arrangements for facilities, special equipment, and other resources.
- Scheduling personnel time for training and for acceptance testing.
- Planning for automation to facilitate test documentation, execution, and analysis.
- Planning for training for users and anyone involved with operating the system during acceptance testing.
- Establishing criteria for each acceptance test.
- Assigning responsibility for
  - Acceptance stop and restart decisions,
  - Designing tests, establishing test cases and detailed test procedures,
  - Administering, executing, analyzing acceptance tests.

5.2. Objectives

A primary objective of software product assurance is to locate errors and potential trouble spots in the software products. The initial analysis of requirements, which includes verification that they will result in a product satisfying the users, is important because the requirements are the basis for acceptance criteria. Later software product assurance activities do not focus on direct demonstration that the system will operate as the user expects. These activities concentrate on ensuring that each successive development product is consistent with previous products and that the requirements will be met. Acceptance testing provides a final opportunity to observe system operation relative to the users' needs. The primary objective of formal acceptance testing is to demonstrate that the implemented software system satisfies the user's requirements for the software system. If the earlier acceptance activities were successfully followed, then the acceptance testing phase should be little more than a formality. Users must be involved to meet this primary objective. They are the most knowledgeable about the current and new methods and practices of their environment which the system must satisfy.

5.3. Location

Usually data for accepting interim software products are developed and collected at the developer's site or at the site of independent organizations for software product assurance activities. Planning and providing the facilities and equipment for development and software product assurance activities is usually the responsibility of other organizations, with approval by the buyer. The buyers rarely provide more than a meeting room for reviews or the capability for a demonstration of interim software. However, software acceptance testing usually

occurs at the installation site. Some exceptions (e.g., missile defense systems, transoceanic communications systems) require unique simulation or other testing methods which are outside the scope of this document. Facilities needed for software acceptance testing may include more than the delivered system and support software and hardware necessary for its operation in the production environment. The need to check out all peripheral equipment used for testing and to ensure staff familiarity with this equipment is frequently overlooked. Planning for special rooms and the equipment needs for test execution, collection, analysis and verification of test results is the responsibility of the buyers. For all the acceptance activities, the use of automation may be necessary to manage all the information and in the case of acceptance testing, to generate the information [2].

5.4. Approach

The complete set of system requirements and acceptance criteria form the basis for determining the overall approach to acceptance testing and the specific testing and examination methods. Features of the installation site and the software system affect how the software acceptance testing will be done. Unique arrangements are necessary when the software cannot be completely installed and executed in a live environment (e.g., missile defense systems). Multiple configurations may have to be distributed at several installation sites. The set of test cases may not be identical for each site; configuration management of test documentation requires special attention.

When the new system is a replacement for one already in use, buyers must assure the integrity of their business operations while placing the replacement into operation. For example, the old system and the new system are used in parallel until complete functionality has been verified. In some cases the task may take several months to ensure that a complete business or accounting cycle has occurred. This concern will influence the approach to software acceptance testing. Often, a new computer is delivered shortly before the application software is presented for acceptance testing. Advance planning is necessary to ensure that operations staff have been trained sufficiently so that acceptance testing of applications residing in the new computer can proceed without interruption.

5.5. Staff Responsibilities

Buyers are involved in establishing how acceptance testing will be done, even when a third party is contracted for software acceptance testing. At the minimum their facilities and staff will be involved in the actual testing. The buyers must ensure that the users create scenarios of how they perform their functions and how the software system is expected to be used. Even when the system provides a new capability, only the users can provide the information necessary for constructing scenarios that will be implemented during operation of the system. The buyer must provide additional managerial time for monitoring the process from its beginning. The buyer must allow time for developing and reviewing the test documentation, whether developed by a third party or by the users.

5.6. Acceptance Test Documentation

Software acceptance testing, like other testing of the system, must be documented carefully, with traceability of test cases to the system requirements and the acceptance criteria. Several guidelines provide information for test plans, test designs and cases, and test procedures [5, 7, 19, 41]. Each of these documents contains specific types of information. Together they form the basis for thorough, controlled acceptance testing.

The test plan, prepared along with the complete software acceptance plan, identifies requirements that must be addressed such as those listed in table 14.

Table 14. Test plan requirements

- The organizations and their responsibilities for software acceptance testing.
- Identification of methods for traceability of requirements to test cases.
- Administration of the process.
- Completeness of test cases and test procedures.
- Descriptions of error reporting and error analysis techniques.
- Location, testing approach, facilities, equipment, and training.
- Acquisition of special purpose testing equipment and software.
- Cost estimation of testing.

When a third party is involved, the acceptance plan establishes the interrelationships of all involved parties and a review schedule for materials developed by the third party. Planning for staff time to develop and review test documentation is often overlooked; when this happens, the project increases its risks of having inadequate acceptance testing. The plan should be updated as the project progresses, both to reflect changes in project status and to complete definition of tasks that depend on evolving information.

The acceptance test plan must be cited or included in the overall software acceptance plan. For long term projects, the acceptance test plan may be separate but for short term projects, the acceptance plan may include the details

for software acceptance testing. It is reasonable to keep the two plans in one document if either of the two following events is likely to occur:

- acceptance test planning is left until late in the development; or
- acceptance planning and implementation are ignored while preparing for acceptance tests.

The users of the software system must be involved in defining the test designs, at the minimum approving scenarios which an independent organization may have designed. The users must identify most frequently used functions, most difficult functions to execute from a user perspective, and other features of the system that are essential to its successful operation. Table 15 identifies typical contents of test designs and cases. Table 16 lists typical information to be included in the software acceptance test procedures.

Table 15. Test design information

- The design of the tests.
- The objectives and constraints for each test.
- The traceability of test designs and cases to system requirements.
- The supporting tools required for each test.
- The inputs and expected outputs of each test case.
- The specification of initialization and stopping conditions.
- The extent to which interfaces are tested.
- The acceptance criteria for the tests.

Table 16. Test procedure information

- The association of each procedure to the appropriate test designs and cases.
- The location and scheduling of the testing.
- The identification of required pretest procedures.
- The availability of peripheral support items (e.g., printers, modems).
- The detailed procedures for execution of each test.
- The procedures for the collection of test results and problem resolution.

The goals of the test documentation are to provide all necessary information to the tester at the time of execution and to enable a test to be easily repeated under the same conditions.

6. SOFTWARE ACCEPTANCE PLANNING AND MANAGEMENT

Managers responsible for software acceptance must ensure that the results of software acceptance activities demonstrate whether contractual requirements meet buyer needs, and whether the delivered software system meets the contractual requirements. Software acceptance managers apply elements of traditional management (e.g., planning and organizing, monitoring and controlling, providing support, performing cost-benefit and risk analyses) to managing the contractual process of acquiring software. Software acceptance managers use their technical knowledge of the proposed software system, of risks associated with its development and maintenance, and of its expected use to establish the criteria for acceptance. If an acceptance manager does not have the required knowledge or technical skills for establishing the requirements of the acceptance plan, then a technical staff person should assist the manager during the planning period, and perhaps for the entire project!

6.1. The Software Acceptance Plan

The first step in effective software acceptance is the simultaneous development of a software acceptance plan, general project plans and contractual requirements. This will ensure that user needs are represented correctly and completely in the contractual requirements. This may involve only the user and acceptance manager organizations and may be completed before the contract for development is awarded; approval must be as rigorous as for other acceptance activities. Further, this simultaneous development will provide an overview of the acceptance activities to ensure that resources for them are included in the project plans. The initial plan may not be complete and may contain estimates which will need to be changed as more complete project information becomes available.

Acceptance managers define the objectives of the acceptance activities and a plan for meeting them. The contractual requirements, knowledge of how the software system is expected to be used in the operational environment, and knowledge of risks associated with the project's life cycle approach provide a basis for determining the acceptance objectives. Because most of this information may be provided by users, initial planning sessions may be interactive between acceptance mangers and users to assure that all parties fully understand what the acceptance criteria should be. After the initial software acceptance plan has been prepared, reviewed and approved, the acceptance manager is responsible for implementing the plan and for assuring that the acceptance objectives are met. The plan may have to be revised for this assurance to occur.

Table 17 provides examples of information which should be included in a software acceptance plan. The first section provides an overview of the software development or maintenance project. Then there are major sections for the management responsibilities and for administrative matters. The plan has an overview section describing the technical program for software acceptance. Details for each software acceptance activity or review appear in separate sections as a supplement to the overview.

Table 17. Acceptance plan contents

Project Description	Type of system; life cycle methodology; user community of delivered system; major tasks system must satisfy; major external interfaces of the system; expected normal usage; potential misuse; risks; constraints; standards and practices.
Management Responsibilities	Organization and responsibilities for acceptance activities; resource and schedule requirements; facility requirements; requirements for automated support, special data, training; standards, practices and conventions; updates and reviews of acceptance plans and related products.
Administrative Procedures	Anomaly reports; change control; record keeping; communication between developer and manager organizations.
Acceptance Description	Objectives for entire project; summary of acceptance criteria; major acceptance activities and reviews; information requirements; types of acceptance decisions; responsibility for acceptance decisions.
Each Acceptance Review	Products for acceptance; objectives for each review; acceptance criteria; source of additional information about each product; acceptance requirements; general approach; test and examination techniques and required automated support.
Final Acceptance Testing	Test plan and acceptance criteria; test cases and procedures; test results and analyses; facilities; tool acquisition and checkout; staff.

6.2. Project Description

The project description provides information on the project parameters which are binding on software acceptance. Project information in the software acceptance plan identifies the purpose of the software system to be accepted and its relationship to any existing software already in operation, the external interfaces the system must satisfy within the operating environment (e.g., with other computer systems, with users), and major functions the system must satisfy. Two other types of information in the plan's project description will help acceptance managers plan acceptance criteria. One concerns the development activities and the other concerns the intended operation of the system. The plan should describe the nature of the development (e.g., a completely new system, a major enhancement, error corrections and changes due to new regulations or

algorithms) and the assignment of organizations for development and software product assurance activities.

The software acceptance description in turn establishes the relationship of the acceptance activities to the project and establishes the necessary elements for performing software acceptance activities. From this information and descriptions of the life cycle methodology and automated support, acceptance planners may perform the following activities:

- determine potential weak spots in the software;
- plan acceptance activities for the product forms likely to be produced;
- establish acceptance reviews based on interim products from all involved organizations; and
- plan facilities and estimate schedules for software acceptance testing at delivery of the system.

6.3. Management

The management section of the plan identifies how the software acceptance will be managed. It identifies the role of each organization in software acceptance and specific responsibilities for each acceptance procedure, including developing and updating the acceptance plan. This section of the plan identifies the facilities, software and hardware, and training requirements. It identifies any risks or constraints associated with the project and with the acceptance activities. The plan presents the resource requirements in terms of finances, staff, and schedule. Contingencies should be addressed.

6.4. Administration

The administration section of the plan identifies the procedures that the staff will execute as they perform acceptance activities. Some of the procedures that need to be explained are the following:

- conduct of reviews;
- management and control of project data (e.g., version control on incoming products for acceptance and on any documents generated by the software acceptance staff);
- handling of deviations from the plan; reporting of any anomalies or problems;
- tracking resolution of anomalies; and
- communication between software acceptance staff and other organizations involved with the project.

6.5. Software Acceptance Description

The overview of software acceptance establishes the objectives that the software acceptance activities will be designed to meet, and the general acceptance criteria the software system must meet. The overview identifies the interim and final products for acceptance and the technical activities and procedures which will be used during acceptance. The overview identifies the types,

sources, and form of information needed for acceptance activities. The overview identifies who will be responsible for acceptance decisions and the types of decisions that are allowable. Usually the acceptance activities for software products involve evaluation of information from development or product assurance activities and possibly some interactive effort (e.g., prototyping) by the users. Then some decision is made regarding acceptance of the products. The final acceptance review may use all of the previous information and focus attention on the results of the software acceptance testing. The acceptance overview identifies acceptance criteria which are the top level features the system must satisfy. Criteria at a detailed level for individual products are described in the plan section with their respective acceptance procedure description.

6.6. Software Acceptance Activities

The software acceptance plan provides a section for each acceptance procedure with specific information about the objectives for each activity. The plan may include a description of the products to be accepted through that activity, the criteria the products must meet, the evaluation method to be used, the source of other information that may be used to judge the product (e.g., verification results or integration test results), and any facility or software requirements for the acceptance activity. Product descriptions identify the source of the product and the form in which it will be presented.

6.7. Software Acceptance Testing

The most extensive period for implementing any given portion of the plan is likely to be the acceptance testing. The acceptance plan may either include the acceptance test plan, or may cite it as a separate document. When a separate document is used, the acceptance plan should reference it and provide at least an overview of the approach, resources, and estimated time for the acceptance testing. The initial acceptance test plan should be ready with the acceptance plan. Other test documentation (including test designs, cases and procedures) may be prepared separately from the acceptance plans. In fact, their preparation is a continuing process throughout the software project. It is important for the acceptance management to plan for review and approval of all acceptance test documentation. An important part of readiness for acceptance testing is monitoring preparation of facilities, ensuring equipment used only for the acceptance tests is available and checked out, and checking that testers are trained to use all parts of the system.

7. SUMMARY

Software acceptance is a contractual process with buyers and developers identifying products and criteria for the acceptance of software systems. Some software may have to pass acceptance before the software requirements have been fully specified. Examples include:

- software used to support the development of the system;
- software for operating the system; and
- existing software for incorporation into the system.

Developers agree to acceptance criteria which the buyer has developed. The buyer must define the acceptance criteria based on the system requirements for functionality, performance, interface quality, overall software quality, security, and software safety. Other project characteristics such as the specific methodology (or variant) must be considered in defining the acceptance criteria. The buyer bases acceptance decisions on analyses and reviews of the products and of results from software product assurance activities.

The buyer must plan and manage the software acceptance program carefully to assure that adequate resources are available for the software acceptance activities. The buyer must include detailed planning for software acceptance testing in the early planning for the entire software acceptance program. The procedures in managing software acceptance enable all those involved in the software project to focus on the system requirements and how well the evolving system is satisfying the requirements. Software acceptance requires buyer's resources and commitment from the beginning of the project. As an interactive process especially involving the user, its completion will result in delivered software that offers its users the services they require.

REFERENCES

[1] Rothfeder, Jeffrey, "Using the Law to Rein in Computer Runaways: More Unhappy Buyers Are Taking Systems Suppliers to Court," *Business Week,* April 3, 1989.

[2] Wallace, Dolores R., and John C. Cherniavsky, "Report on the NBS Software Acceptance Test Workshop April 1-2, 1986," Natl. Bur. Stand. (U.S.) Spec. Publ. 500-146; 1987 March.

[3] Wallace, Dolores R., "An Overview of Software Acceptance Testing," Natl. Bur. Stand. (U.S.) Spec. Publ. 500-136; 1986 February.

[4] Frankel, Sheila, "Guidance on Software Package Selection," Natl. Bur. Stand. (U.S.) Spec. Publ. 500-144; 1986 November.

[5] "Guideline for Lifecycle Validation, Verification, and Testing of Computer Software," Federal Information Processing Standards Publication 101, FIPSPUB101, Natl. Bur. Stand. (U.S.) 1983 June.

[6] DOD-STD-2167A Military Standard Defense System Software Development, AMSC No. 4327, Department of Defense, Washington DC, February 29, 1988.

[7] "Guidelines for Documentation of Computer Programs and Automated Systems," Federal Information Processing Standards Publication 38, FIPSPUB38, Natl. Bur. Stand. (U.S.) 1976 February.

[8] Boehm, Barry W., "A Spiral Model of Software Development and Enhancement," *COMPUTER,* IEEE Computer Society, May 1988, Vol. 21, No. 5, pp. 61-72.

[9] Davis, A. M., E. H. Bersoff, and E. R. Comer, "A Strategy for Comparing Alternative Software Development Life Cycle Models," *IEEE Transactions in Software Engineering,* Vol. 14, No. 10, October 1988, pp. 1453-1461.

[10] Fisher, Gary E., "Application Software Prototyping and Fourth Generation Languages," Natl. Bur. Stand. (U.S.) Spec. Publ. 500-148; 1987 May.

[11] Fisher, Gary E., "A Functional Model for Fourth Generation Languages," Natl. Bur. Stand. (U.S.) Spec. Publ. 500-138; 1986 June.

[12] Gray, Martha Mulford, "Guide to the Selection and use of Fourth Generation Languages," Natl. Bur. Stand. (U.S.) Spec. Publ. 500-143; 1986 September.

[13] Wallace, Dolores R., and Roger U. Fujii, "Software Verification and Validation: An Overview," *IEEE Software,* May 1989, pp. 10-17.

[14] DOD-STD-7935A Military Standard DOD Automated Information Systems (AIS) Documentation Standards, Department of Defense, Washington, DC, 31 October 1988.

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

[16] ANSI / IEEE Std. 830-1984, "Guide to Software Requirements Specifications," The Institute for Electrical and Electronics Engineers, Inc., 345 West 47th St., New York, NY 10017, 1984.

[17] Information System Life-Cycle and Documentation Standards and Management Plan Documentation and Data Item Descriptions (DID); Release 4.3, February 1989, NASA Headquarters, Washington, DC.

[18] Wallace, Dolores R., and Roger U. Fujii, "Software Verification and Validation: Its Role in Computer Assurance and Its Relationship with Software Project Management Standards," Natl. Inst. Stand. Technol. Spec. Publ. 500-165; 1989 May.

[19] "Guideline for Software Verification and Validation Plans," Federal Information Processing Standards Publication 132, FIPSPUB132, Natl. Bur. Stand. (U.S.) 1987 November.

[20] Misra, Santosh K., and Paul J. Jalics, "Third-Generation versus Fourth-Generation Software Development," *IEEE Software,* IEEE Computer Society, Los Alamitos, CA, July, 1988, pp 8-14.

[21] Babcock, C., "New Jersey Motorists in Software Jam," *Computerworld,* September 30, 1985, pp 1,6.

[22] Wong, William, "Management Guide to Software Reuse," Natl. Bur. Stand. (U.S.) Spec. Publ. 500-143; 1988 April.

[23] Cavano, Joseph P., and James A. McCall, "A Framework for the Measurement of Software Quality," *The Proceedings of the ACM Software Quality Assurance Workshop,* November 1978, pp 133-179; reprinted in Chow, Tsun S., *Tutorial Software Quality Assurance,* IEEE Computer Society.

[24] Birrell, N. D., and M. A. Ould, *A Practical Handbook for Software Development,* Cambridge University Press, New York, NY 1985.

[25] Bowen, Thomas P., *et al* , "Software Quality Measurement for Distributed Systems," Rome Air Development Center Air Force Systems Command, Griffiss Air Force Base, NY 13441-5700, RADC-TR-83-175, Vols. I-III, July 1983.

[38] IEEE Std. 982.1 - 1988, A Standard Dictionary of Measures to Produce Reliable Software, The Institute for Electrical and Electronics Engineers, Inc., 345 West 47th St., New York, NY 10017, 1984.

[39] IEEE Std. 982.2 - 1988, Guide for the Use of Standard Dictionary of Measures to Produce Reliable Software, The Institute for Electrical and Electronics Engineers, Inc., 345 West 47th St., New York, NY 10017, 1984.

[40] Musa, John D., Anthony Iannini, and Kazuhira Okumoto, *Software Reliability: Measurement, Prediction, Application,* McGraw Hill Book Company, 1987.

[41] ANSI/ IEEE Std. 829-1984, "Standard for Software Test Documentation," The Institute for Electrical and Electronics Engineers, Inc., 345 West 47th St., New York, NY 10017, 1983.

[26] Bowen, Thomas P., *et al* , "Specification of Quality Attributes," Rome Air Development Center, Air Force Systems Command, Griffiss Air Force Base, NY 13441-5700, RADC-TR-85-37, Vols. I-III, February 1985.

[27] Cavano, Joseph P., and James A. McCall, "A Framework for the Measurement of Software Quality," *The Proceedings of the ACM Software Quality Workshop,* November 1978, pp. 133-139; reprinted in *Tutorial: Software Quality Assurance: A Practical Approach,* ed. Tsun S. Chow, IEEE Computer Society Press, 1985.

[28] Dobbins, James H., and Robert D. Buck, "The Cost of Software Quality," and Dobbins, James H., "Inspections as an Up-Front Quality Technique," *Handbook of Software Quality Assurance,* Eds, C. Gordon Schulmeyer and James I. McManus, Van Nostrand Reinhold Company, Inc., 1987.

[29] IEEE P1061, "Draft Standard for a Software Quality Metrics Methodology," The Institute for Electrical and Electronics Engineers, Inc., 345 West 47th St., New York, NY 10017, 1984.

[30] Pressman, Roger S., *Software Engineering: A Practitioner's Approach,* McGraw-Hill Book Company, New York, NY 1982.

[31] Watts, Richard, *Measuring Software Quality,* NCC Publications, The National Computing Center Limited, Oxford Road, Manchester M1 7ED, England, 1987.

[32] Computer Security Act of 1987, Public Law 100-235, 100th United States Congress, Washington, DC, January 8, 1988.

[33] AFSC / AFLCP 800-5, Software Independent Verification and Validation, Air Force Systems Command and Air Force Logistics Command, Washington DC May 1988.

[34] Hankinson, Allen L., "Computer Assurance: Security, Safety, and Economics," *Proceedings of the Fourth Annual Conference on Computer Assurance,* June 19-23, 1989, National Institute of Standards and Technology, Gaithersburg, MD, The Institute for Electrical and Electronics Engineers, Inc., 345 West 47th St., New York, NY 10017, 1989.

[35] Bell, D. E., and L. J. LaPadula, *Secure Computer Systems: Unified Exposition and Multics Interpretation,* MTR-2997 Rev. 1, Mitre Corp., Bedford, Mass., March 1979.

[36] Cho, C. K., *An Introduction to Software Quality Control,* John Wiley, 1980.

[37] Cho, Chin-Kuei, *Quality Programming - Developing and Testing Software with Statistical Quality Control,* John Wiley, 1987.