BEST PRACTICE
Acceptance Test Planning

The acceptance test plan basically follows the practices used for developing an overall system test plan. Specifically this section will address:

- Acceptance Criteria
- Acceptance Test Plan
- Use Case Test Data

Acceptance Criteria

The user must assign the criteria the software must meet to be deemed acceptable. Ideally, this is included in the software requirements specifications. In preparation for developing the acceptance criteria, the user should:

- Acquire full knowledge of the application for which the system is intended.
- Become fully acquainted with the application as it is currently implemented by the user's organization.
- Understand the risks and benefits of the development methodology that is to be used in correcting the software system.
- Fully understand the consequences of adding new functions to enhance the system.

Acceptance requirements that a system must meet can be divided into these four categories:

- Functionality Requirements
  These requirements relate to the business rules that the system must execute.
- Performance Requirements
  These requirements relate to operational aspects, such as time or resource constraints.
- Interface Quality Requirements
  These requirements relate to connections from one component to another component of processing (e.g., human-machine, machine-module).
- Overall Software Quality Requirements
  These requirements specify limits for factors or attributes such as reliability, testability, correctness, and usability.

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. Assessing the criticality of a system is important in determining quantitative acceptance criteria. The user should determine the degree of criticality of the requirements by the above acceptance requirements categories.

By definition, all safety criteria are critical; and by law, certain security requirements are critical. Some typical factors affecting criticality include:

- Importance of the system to organization or industry
- Consequence of failure
- Complexity of the project
- Technology risk
- 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.

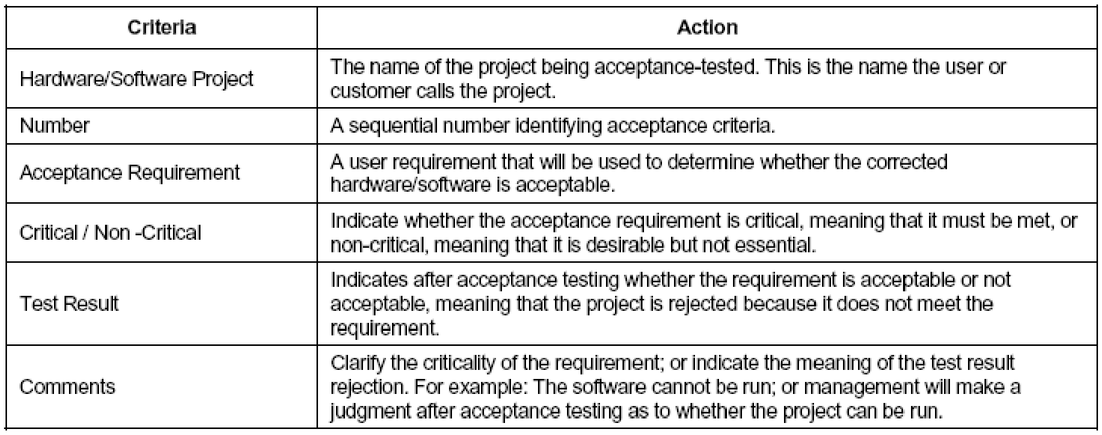
The user has the responsibility of ensuring that acceptance criteria contain pass or fail criteria. The acceptance tester should approach testing assuming that the least acceptable corrections have been made; while the developer believes the corrected system is fully acceptable. Similarly, a contract with what could be interpreted as a range of acceptable values could result in a corrected system that might never satisfy the user's interpretation of the acceptance criteria.

For specific software systems, users must examine their projects' characteristics and criticality in order to develop expanded lists of acceptance criteria for those software systems. Some of the criteria may change according to the phase of correction for which criteria are being defined. For example, for requirements, the "testability" quality may mean that test cases can be developed automatically.

The user must also establish acceptance criteria for individual elements of a product. 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 criteria, then the requirements document should be rejected. At that time, the established procedures for iteration and change control go into effect.

Table 21 explains what users need to provide to document the acceptance criteria. It should be prepared for each hardware or software project within the overall project, where the acceptance criteria requirements should be listed and uniquely numbered for control purposes.

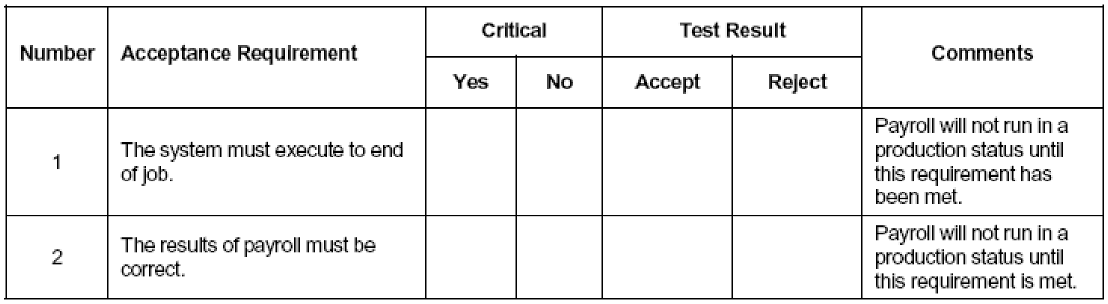
Table 21. Example of Required Information to Document Acceptance Criteria



Criteria	Action
Hardware/Software Project	The name of the project being acceptance-tested. This is the name the user or customer calls the project.
Number	A sequential number identifying acceptance criteria.
Acceptance Requirement	A user requirement that will be used to determine whether the corrected hardware/software is acceptable.
Critical / Non -Critical	Indicate whether the acceptance requirement is critical, meaning that it must be met, or non-critical, meaning that it is desirable but not essential.
Test Result	Indicates after acceptance testing whether the requirement is acceptable or not acceptable, meaning that the project is rejected because it does not meet the requirement.
Comments	Clarify the criticality of the requirement; or indicate the meaning of the test result rejection. For example: The software cannot be run; or management will make a judgment after acceptance testing as to whether the project can be run.

After defining the acceptance criteria, determine whether meeting the criteria is critical to the success of the system. As shown in Table 22, this is indicated by placing a checkmark in the Yes or No columns under Critical. Note that if an acceptance criterion were critical, then the system would not be accepted if that requirement has not been met.

Table 22. Acceptance Criteria Example



Number	Acceptance Requirement	Critical		Test Result		Comments
		Yes	No	Accept	Reject	
1	The system must execute to end of job.					Payroll will not run in a production status until this requirement has been met.
2	The results of payroll must be correct.					Payroll will not run in a production status until this requirement is met.

Table 22 is an example of two acceptance criteria for a payroll project. It shows that both acceptance criteria are critical for the project. The Test Result column is blank, indicating the test has not yet been performed. The Comments column reports that the payroll system will not run unless these two critical requirements are met.

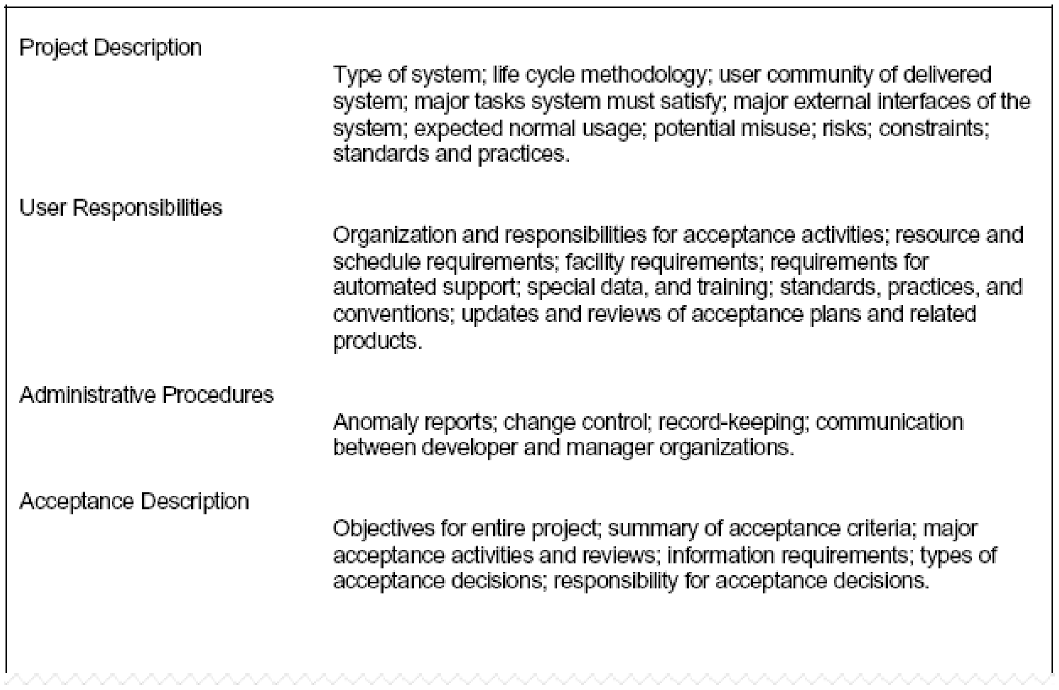
Acceptance Test Plan

The first step to achieve software acceptance is the simultaneous development of a Software Acceptance Plan, general project plans, and software requirements to ensure that user needs are represented correctly and completely. This simultaneous development will provide an overview of the acceptance activities, to ensure that resources for them are included in the project plans. Note
that the initial plan may not be complete and may contain estimates that 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. Knowing how the software system is intended to be used in the operational environment and the risks associated with the project's life cycle provide a basis for determining these acceptance objectives. Because users may provide most of this information, initial planning sessions may be interactive between acceptance managers 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 assuring that the plan's objectives are met. It may have to be revised before this assurance is warranted.

Figure 83 lists examples of information that should be included in a Software Acceptance Plan. The first section of the plan is an overview of the software development or maintenance project, followed by major sections for management responsibilities and administrative matters. The plan's overview section describes the technical program for software acceptance. Details for each software acceptance activity or review appear in separate sections as supplements to the overview.



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.
User Responsibilities	Organization and responsibilities for acceptance activities; resource and schedule requirements; facility requirements; requirements for automated support; special data, and 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.

Figure 83. Acceptance Test Plan Table of Contents

The plan must include the techniques and tools that will be utilized in acceptance testing. Normally, testers will use the organization's current testing tools, which should be oriented toward specific testing techniques.

Two categories of testing techniques can be used in acceptance testing: structural and functional. Remember that acceptance testing must be viewed in its broadest context; not the minimal testing that some users perform after the information system professionals have concluded their testing.

The functional testing techniques help ensure that the requirements/specifications are properly satisfied by the software system. Functional testing is not concerned with how processing occurs, but with the results of processes.

Structural testing ensures sufficient checking of the implementation of the function by finding test data that will force sufficient coverage of the structured presence in the implemented software. It evaluates all aspects of this structure to verify that the structure is sound.

Use Case Test Data

A *use case* is a test case which represents how the software will be used in operation. A use case is built on a business transaction and can be test data or a test script. Unit testing will attempt to determine whether there are any variances between unit specifications and the unit as it is executed. Integration testing will attempt to determine if there is a variance between specified integration and actual integration. System testing will validate that the system, when assembled, will perform as specified. The test cases and scripts used for these three levels of testing are focused more on the components of the software than business transactions.

Many software testers do not have an adequate knowledge of the business to create business or use cases for test purposes. For example, an online data entry clerk may need to go to more than one source to gather the information needed to enter a business transaction. For example, they may need to look up the credit history of a customer prior to approving that order. While the software tester would be primarily concerned with the single system being developed, the user is concerned with the totality of events that lead to a business transaction being entered into the software.

When use cases are developed by clerical people intimately familiar with the system, they tend to know the type of problems that are typical in the business system. Thus, they can simulate those unusual events through use cases that may not be developed during normal system testing.

An individual use case consists of:

- Preconditions that set the stage for the series of events that should occur for the use case
- Post-conditions that state the expected outcomes of the above process
- Sequential narrative of the execution of the use case

References

Guide – CSTE Common Body Of Knowledge, V6.1