Acceptance testing

In engineering and its various subdisciplines, **acceptance testing** is a test conducted to determine if the requirements of a specification or contract are met. It may involve chemical tests, physical tests, or performance tests.

Acceptance testing of an aircraft catapult

In systems engineering it may involve black-box testing performed on a system (for example: a piece of software, lots of manufactured mechanical parts, or batches of chemical products) prior to its delivery.[1]

In software testing the ISTQB defines *acceptance testing* as:

> Formal testing with respect to user needs, requirements, and business processes conducted to determine whether a system satisfies the acceptance criteria [2] and to enable the user, customers or other authorized entity to determine whether to accept the system.
>
> — Standard Glossary of Terms used in Software Testing[3]:2

Six of the primary mirrors of the James Webb Space Telescopebeing prepared for acceptance testing

Acceptance testing is also known as user acceptance testing (UAT), end-user testing, operational acceptance testing (OAT), acceptance-test-driven development (ATTD) or field (acceptance) testing. Acceptance criteria are the criteria that a system or component must satisfy in order to be accepted by a user, customer, or other authorized entity.[4]

A smoke test may be used as an acceptance test prior to introducing a build of software to the main testing process.

Contents

Overview

Testing is a set of activities conducted to facilitate discovery and/or evaluation of properties of one or more items under test.[5] Each individual test, known as a test case, exercises a set of predefined test activities, developed to drive the execution of the test item to meet test objectives; including correct implementation, error identification, quality verification and other valued detail.[5] The test environment is usually designed to be identical, or as close as possible, to the anticipated production environment. It includes all facilities, hardware, software, firmware, procedures and/or documentation intended for or used to perform the testing of software.[5]

UAT and OAT test cases are ideally derived in collaboration with business customers, business analysts, testers, and developers. It's essential that these tests include both business logic tests as well as operational environment conditions. The business customers (product owners) are the primary stakeholders of these tests. As the test conditions successfully achieve their acceptance criteria, the stakeholders are reassured the development is progressing in the right direction.[6]

- User acceptance test (UAT) criteria (in agile software development) are usually created by business customers and expressed in a business domain language. These are high-level tests to verify the completeness of a user story or stories 'played' during any sprint/iteration.
- Operational acceptance test (OAT) criteria (regardless if using agile, iterative or sequential development) are defined in terms of functional and non-functional requirements; covering key quality attributes of functional stability, portability and reliability.

Process

The acceptance test suite may need to be performed multiple times, as all of the test cases may not be executed within a single test iteration.[7]

The acceptance test suite is run using predefined acceptance test procedures to direct the testers which data to use, the step-by-step processes to follow and the expected result following execution. The actual results are retained for comparison with the expected results.[7] If the actual results match the expected results for each test case, the test case is said to pass. If the quantity of non-passing test cases does not breach the project's predetermined threshold, the test suite is said to pass. If it does, the system may either be rejected or accepted on conditions previously agreed between the sponsor and the manufacturer.

The anticipated result of a successful test execution:

- test cases are executed, using predetermined data
- actual results are recorded
- actual and expected results are compared, and
- test results are determined.

The objective is to provide confidence that the developed product meets both the functional and non-functional requirements. The purpose of conducting acceptance testing is that once completed, and provided the acceptance criteria are met, it is expected the sponsors will sign-off on the product development/enhancement as satisfying the defined requirements (previously agreed between business and product provider/developer).

User acceptance testing

User acceptance testing (UAT) consists of a process of verifying that a solution works for the user.[8] It is not system testing (ensuring software does not crash and meets documented requirements), but rather ensures that the solution will work for the user (i.e., tests that the user accepts the solution); software vendors often refer to this as "Beta testing".

This testing should be undertaken by a subject-matter expert (SME), preferably the owner or client of the solution under test, and provide a summary of the findings for confirmation to proceed after trial or review. In software development, UAT as one of the final stages of a project often occurs before a client or customer accepts the new system. Users of the system perform tests in line with what would occur in real-life scenarios.[9]

It is important that the materials given to the tester be similar to the materials that the end user will have. Testers should be given real-life scenarios such as the three most common or difficult tasks that the users they represent will undertake.

The UAT acts as a final verification of the required business functionality and proper functioning of the system, emulating real-world conditions on behalf of the paying client or a specific large customer. If the software works as required and without issues during normal use, one can reasonably extrapolate the same level of stability in production.[10]

User tests, usually performed by clients or by end-users, do not normally focus on identifying simple cosmetic problems such as spelling errors, nor on showstopper defects, such as software crashes; testers and developers identify and fix these issues during earlier unit testing, integration testing, and system testing phases.

UAT should be executed against test scenarios. Test scenarios usually differ from System or Functional test cases in that they represent a "player" or "user" journey. The broad nature of the test scenario ensures that the focus is on the journey and not on technical or system-specific details, staying away from "click-by-click" test steps to allow for a variance in users' behaviour. Test scenarios can be broken down into logical "days", which are usually where the actor (player/customer/operator) or system (backoffice, front end) changes.

In industry, a common UAT is a factory acceptance test (FAT). This test takes place before installation of the equipment. Most of the time testers not only check that the equipment meets the specification, but also that it is fully functional. A FAT usually includes a check of completeness, a verification against contractual requirements, a proof of functionality (either by simulation or a conventional function test) and a final inspection.[11][12]

The results of these tests give clients confidence in how the system will perform in production. There may also be legal or contractual requirements for acceptance of the system.

Operational acceptance testing

Operational acceptance testing (OAT) is used to conduct operational readiness (pre-release) of a product, service or system as part of a quality management system. OAT is a common type of non-functional software testing, used mainly in software development and software maintenance projects. This type of testing focuses on the operational readiness of the system to be supported, and/or to become part of the production environment.

Acceptance testing in extreme programming

Acceptance testing is a term used in agile software development methodologies, particularly extreme programming, referring to the functional testing of a user story by the software development team during the implementation phase.[13]

The customer specifies scenarios to test when a user story has been correctly implemented. A story can have one or many acceptance tests, whatever it takes to ensure the functionality works. Acceptance tests are black-box system tests. Each acceptance test represents some expected result from the system. Customers are responsible for verifying the correctness of the acceptance tests and reviewing test scores to decide which failed tests are of highest priority. Acceptance tests are also used as regression tests prior to a production release. A user story is not considered complete until it has passed its acceptance tests. This means that new acceptance tests must be created for each iteration or the development team will report zero progress.[14]

Types of acceptance testing

Typical types of acceptance testing include the following

User acceptance testing
: This may include factory acceptance testing (FAT), i.e. the testing done by a vendor before the product or system is moved to its destination site, after which site acceptance testing (SAT) may be performed by the users at the site.[15]

Operational acceptance testing
: Also known as operational readiness testing, this refers to the checking done to a system to ensure that processes and procedures are in place to allow the system to be used and maintained. This may include checks done to back-up facilities, procedures for disaster recovery, training for end users, maintenance procedures, and security procedures.

Contract and regulation acceptance testing
: In contract acceptance testing, a system is tested against acceptance criteria as documented in a contract, before the system is accepted. In regulation acceptance testing, a system is tested to ensure it meets governmental, legal and safety standards.

Factory acceptance testing

Acceptance testing conducted at the site at which the product is developed and performed by employees of the supplier organization, to determine whether a component or system satisfies the requirements, normally including hardware as well as software.[16]

Alpha and beta testing
: Alpha testing takes place at developers' sites, and involves testing of the operational system by internal staff, before it is released to external customers. Beta testing takes place at customers' sites, and involves testing by a group of customers who use the system at their own locations and provide feedback, before the system is released to other customers. The latter is often called "field testing".

List of acceptance-testing frameworks

- Concordion, Specification by example (SbE) framework
  - Concordion.NET, acceptance testing in .NET
- Cucumber, a behavior-driven development (BDD) acceptance test framework
  - Capybara, Acceptance test framework for Ruby web applications
  - Behat, BDD acceptance framework for PHP
  - Lettuce, BDD acceptance framework for Python
- Fabasoft app.test for automated acceptance tests
- Framework for Integrated Test (Fit)
  - FitNesse, a fork of Fit
- iMacros
- ItsNat Java Ajax web framework with built-in, server based, functional web testing capabilities.
- Mocha, a popular web acceptance test framework based on Javascript and Node.js
- Ranorex
- Robot Framework
- Selenium
- Specification by example (Specs2)
- Watir
- Gauge (software)

See also

- Acceptance sampling
- Conference room pilot
- Development stage
- Dynamic testing
- Engineering validation test
- Grey box testing
- Test-driven development
- White box testing
- Functional testing (manufacturing)

References

1. Black, Rex (August 2009). *Managing the Testing Process: Practical Tools and Techniques for Managing Hardware and Software Testing*. Hoboken, NJ: Wiley. ISBN 0-470-40415-9.
2. "acceptance criteria". Innolution, LLC. June 10, 2019.
3. "Standard Glossary of Terms used in Software Testing, Version 3.2: All Terms" (PDF). ISTQB. Retrieved February 20, 2019.
4. *ISO/IEC/IEEE International Standard - Systems and software engineering*. ISO/IEC/IEEE. 2010. pp. vol., no., pp.1–418.
5. *ISO/IEC/IEEE 29119-1-2013 Software and Systems Engineering - Software Testing - Part 1- Concepts and Definitions*. ISO. 2013. Retrieved October 14, 2014.
6. *ISO/IEC/IEEE DIS 29119-4 Software and Systems Engineering - Software Testing - Part 4- Test Techniques*. ISO. 2013. Retrieved October 14, 2014.
7. *ISO/IEC/IEEE 29119-2-2013 Software and Systems Engineering - Software Testing - Part 2- Test Processes*. ISO. 2013. Retrieved May 21, 2014.
8. Cimperman, Rob (2006). *UAT Defined: A Guide to Practical User Acceptance Testing*. Pearson Education. pp. Chapter 2. ISBN 9780132702621.
9. Goethem, Brian Hambling, Pauline van (2013). *User acceptance testing : a step-by-step guide*. BCS Learning & Development Limited. ISBN 9781780171678.
10. Pusuluri, Nageshwar Rao (2006). *Software Testing Concepts And Tools*. Dreamtech Press. p. 62. ISBN 9788177227123.
11. "Factory Acceptance Test (FAT)". Tuv.com. Archived from the original on February 4, 2013. Retrieved September 18, 2012.
12. "Factory Acceptance Test". Inspection-for-industry.com. Retrieved September 18, 2012.
13. "Introduction to Acceptance/Customer Tests as Requirements Artifacts". *agilemodeling.com*. Agile Modeling. Retrieved December 9, 2013.
14. Don Wells. "Acceptance Tests". Extremeprogramming.org. Retrieved September 20, 2011.
15. Prasad, Durga (March 29, 2012). "The Difference Between a FAT and a SAT". *Kneat.com*. Retrieved July 27, 2016.
16. "ISTQB Standard glossary of terms used in Software Testing". Retrieved March 15, 2019.

Further reading

- Hambling, Brian; van Goethem, Pauline (2013). *User Acceptance Testing: A Step by Step Guide*. Swindon: BCS Learning and Development Ltd. ISBN 978-1-78017-167-8.

External links