What is user acceptance testing (UAT)?

User acceptance testing (UAT), also called *application testing* or *end-user testing*, is a phase of software development in which the software is tested in the real world by its intended audience. UAT is often the last phase of the software testing process and is performed before the tested software is released to its intended market. The goal of UAT is to ensure software can handle real-world tasks and perform up to development specifications.

In UAT, users are given the opportunity to interact with the software before its official release to see if any features have been overlooked or if it contains any bugs. UAT can be done in-house with volunteers, by paid test subjects using the software or by making the test version available for download as a free trial. The results from the early testers are forwarded to the developers, who make final changes before releasing the software commercially.

What is the purpose of UAT?

User acceptance testing validates the testing done at the end of the development cycle. It is typically completed after unit testing, quality assurance, system testing and integration testing. The software may undergo other testing phases and be completely functional but might still not meet its requirements if it is not well received by its intended users. This can happen if software requirements were not clearly defined to the developers, if certain modifications made during development changed the scope of the project or if the software just was not ready to be tested in a dynamic, real-world environment. Overall, UAT safeguards against faulty, ineffective or unfinished software products being released.

To be effective, UAT should be thorough and reflect user requirements, while also identifying potential problems not yet detected in previous tests. Without UAT, tested software may be released with bugs or a lack of a clearly defined goal for end users. These issues can be costly and potentially damaging to the software vendor's reputation.

Who performs UAT?

End users normally perform user acceptance testing. They are the most effective group to test software in this form because they know exactly how the software will be used on a daily basis and what changes need to be made to be suitable for this day-to-day use.

Internal functional experts also play a role in UAT, as they help shape UAT cycles and test management, as well as interpret the results.

Types of UAT

Multiple types of software tests qualify as user acceptance testing. These tests include the following:

- **Beta testing.** The software is given to groups of end users who evaluate it for its intended purpose and provide feedback to developers for improvements.
- **Black box testing.** An end user tests specific software functions without seeing the internal code.
- **Operational acceptance testing.** The focus is on predefined workflow for the software and operational readiness, such as product compatibility, reliability and stability.

How to perform UAT

The number of steps involved in a user acceptance test may differ, depending on how granularly the team wants to define each step in the process. For the most part, however, these steps commonly include the following:

1. **Plan.** The business requirements, time frame and strategies for UAT are outlined.
2. **Identify and create real-world test scenarios.** These test scenarios should cover as many functional cases as possible that end users may face.
3. **Select the testing team.** Developers can decide whether to have only a few end users test the software or to open up testing to more participants by offering a free trial over the web. End users should have knowledge of the business and how to detect and report issues.
4. **Test and document.** The end users begin testing the software, logging any potential bugs or other issues. All bugs should be recorded in a bug tracker with notes on how to reproduce the errors.
5. **Update code, retest and sign off.** The development team adjusts the code based on test results -- resolving any bugs or making suggested changes -- and then retests. Once the software meets the users' criteria, the tester signs off on the changes.

Challenges of UAT

Some of the possible challenges or downfalls of user acceptance testing include the following:

- **Poor test planning.** Because UAT is the last stage of the software development lifecycle, any delays in previous stages mean less time and more pressure to complete this stage faster. Better planning should be done for both UAT and software development, and the proper development time should be allotted to each.
- **Bad choice of UAT users.** If UAT testers are not properly trained, they may not know how to properly submit bugs or reports. This can cause the organization to be unaware of the different bugs or how to replicate them. UAT testers should be properly trained.
- **Testing environment and deployment.** Using the same environment that was used with functional testing and system testing could lead to software dependencies in that particular environment. Organizations should use a different environment for UAT.
- **Communication gaps.** Gaps in communication between UAT and testing teams could cause delays or problems with reporting of bugs or testing scenarios. Teams need to ensure they have good planning and communication processes in place.

UAT best practices

Some best practices of user acceptance testing include the following:

- **Gather information.** The correct data must be collected, including the process being tested, the actions that must be taken for tests and a set of guidelines for selecting test data.
- **Properly identify the target audience.** This helps identify UAT users who know what to look for and how to provide useful feedback.
- **Understand the project scope.** Specific processes may not need to be tested, so data can be collected from only the processes needed.
- **Design.** Different testing steps can be assigned to different users. Test cases should also be detailed and specify procedures, expected results and conditions a tester may need to verify.
- **Confirm business objectives.** Once the testing is done and bugs are resolved, a sign-off confirmation should be in place to indicate that changes meet business requirements.

Learn how to streamline UAT with processes from Agile.

This was last updated in March 2022

When you hear the term “software testing,” do you think about one particular type of test — such as or — or do you immediately start visualizing the complex, interconnected web of test types and techniques that comprise the broad world of software testing?

Most experienced developers understand that software testing isn’t a singular approach, although, in the broadest sense, it refers to a collection of tests and evaluations that aim to determine whether a software application works as it should and if it can be expected to continue working as it should in real-world use scenarios. Basically, software testing aims to ensure that all the gears are churning smoothly and work together like a well-oiled machine.

approaches to software testing, all of which are equally important in reaching a realistic conclusion to the pressing questions facing developers and testers:

- Does the application as a whole work?
- Do all features function as expected?
- Can the application withstand the demands of a heavy load?
- Are there security vulnerabilities that could put users at risk?
- Is the application reasonably easy to use, or will users find it a pain in the a$$?
- And others

Still, it's not a simple matter of running a few tests and getting the green light. There's a process to thorough software testing, which entails

, ensuring that you're covering the right features and functions,