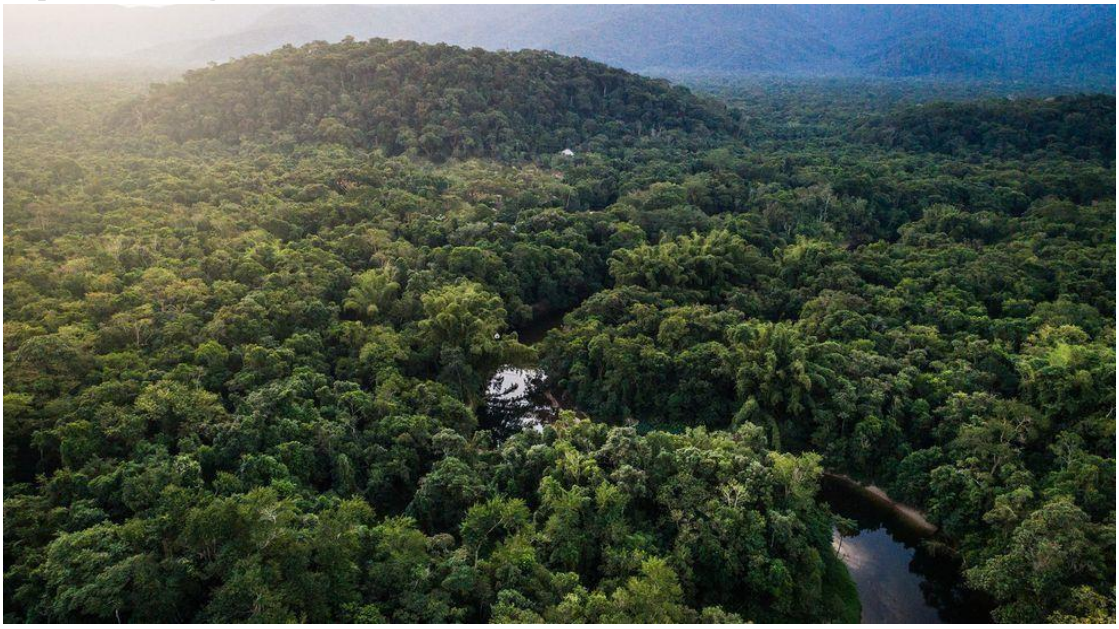
Team id	PNT2022TMID41770
TOPIC	UAT Initiation and Design

What is user acceptance testing and how is it different from quality assurance?



User Acceptance Testing (UAT) checks whether a product is the right one for the end users. It has other names, e.g., *end-user testing*, *operational*, *application*, *beta testing,* or *validation* but they describe the same thing. In quality assurance, it's important to distinguish between *validation* and *verification*.

Verification refers to general QA processes aimed at testing the technical aspects of a product to ensure it actually
works. **Validation** (or user acceptance testing) is conducted to make sure that the product corresponds with business requirements and can be used*Validation and verification activities in terms of overall product testing*

Validation activity can be divided into two types of testing.

Alpha testing is the initial stage of acceptance testing, typically performed by internal testers, to ensure that the product functions correctly and meets business requirements.

Beta testing, the second type of acceptance testing, aims at meeting user acceptance criteria. UAT can be performed by

- the actual users of an existing product,
- users of a previous version of a product,
- stakeholders involved in the development of the product, and/or
- business analysts as end-user specialists.

This enables the development team to fix most of the usability problems, bugs, and unexpected issues concerning functionality, system design, business requirements, etc.

Play

Acceptance criteria explained

Why do you actually need UAT?

The main purpose of acceptance testing is to validate that the product corresponds with the needs of users (defined at the product discovery stage) and is ready for launch. According to an Origsoft survey on UAT usage, over 75 percent of respondents said they conduct multiple cycles of end-user testing with 57 percent claiming the poor quality of the product as a reason.

So, here are the main reasons why UAT is important and should be a part of your development.

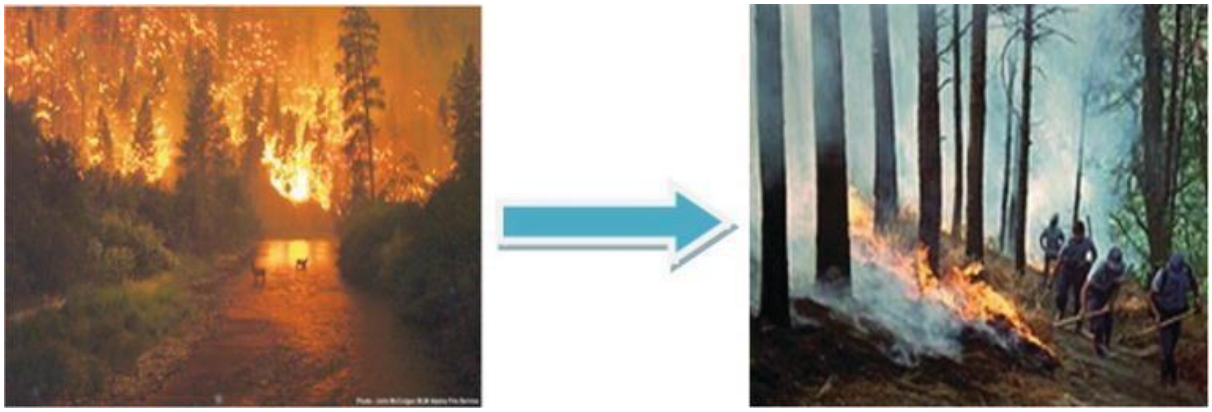
Ensure correspondence with business requirements. As we already mentioned, UAT is done to verify that the product operates in the realworld circumstances as required and allows end users to solve targeted problems. If you skip UAT, you

might miss out on some important flaws or system malfunctions that will inevitably cause user dissatisfaction.

Adjust initial requirements. Sometimes, as end users test the product, they can come up with some valuable thoughts on how to improve the tested software. Getting such feedback will allow you to adjust your requirements to get a result that will be more useful for your customers.

Avoid losses. First, it's cheaper to fix the product in the early stages of development, so finding flaws due to UAT will allow your development team to improve the product much more easily (that mostly concerns the Agile model though. Read on for more details). Second, we all know stories about product failures because of poor functionality and usability. UAT provides you with real-world user feedback and makes it far less likely to have losses caused by an unsuccessful product launch.

In any case, UAT requires organization and preparation work to make it effective. If you want to ensure your product's validity, consider the following steps in conducting user acceptance testing.



UAT key stages

1. Analyze product requirements and define key deliverables

Analyzing product requirements is the first step of UAT planning. The primary source of input information would be the software requirements specification as it includes the complete scope of business and functional requirements.

Business requirements are the high-level objectives of your organization that communicate business needs. Those may sound like “customers should be able to use multiple payment methods.”

Functional requirements bridge a technical solution with the business requirements. So, the functional requirement would sound like “implement PayPal, Visa and Mastercard, Payoneer payment gateways.”

The overview of these requirements will tell you exactly what you should test, whether the implemented solutions work for the users and solve problems for the business. Functional requirements can be translated into test cases, considering the success criteria of business requirements. And that will help you form an overall testing strategy.

Consider engaging your business analysts, QA engineers, or product owners for requirement analysis.

The final planning stage is creating technical documentation for the UAT process. Here, you document your testing strategy, rules, test scenarios/cases, standards, etc. The following sections describe the documentation used in user acceptance testing.

User acceptance testing deliverables

UAT test plan. Creating a UAT test plan will help you keep everybody aligned with the same objectives and vision. The main document, it includes all the information concerning what will be tested, by whom, and how. To cover all the organizational and processual aspects of UAT, you have to detail the testing strategy and entry/exit criteria.

End-user testing strategy. The strategy outlines the product you are testing, the purpose of user-acceptance testing, types of tests, and objectives. Your testing strategy should cover such information as

- product description,
- testing objectives,
- testing scope, • standards,
- testing types,,
- testers/roles
- process curators (managers),
- reviewers,
- reporting standards, and
- outcomes.

Entry criteria. These are the conditions that establish that the software is ready to be tested. They are set at the earliest stage of planning by the development team, QA, business analysts, and stakeholders.

Exit or acceptance criteria. These are the conditions that dictate that the software is valid for the users. Matching acceptance criteria would be the final stage of your UAT.

Test scenarios. Test scenarios are hypothetical situations that users may encounter when interacting with your product. Their aim is to guide your testers through possible system usage problems.

Basically, a test scenario should convey a simple idea of what will be tested. An example of a scenario is “check shopping cart functionality.” Each user scenario is connected with one or two requirements or user stories. They are written to validate that the system is usable, checking the end-to-end operations with real data.

To write good test scenarios for user acceptance testing, consider involving end users in approval to include all the possible use cases, both common and uncommon. Also, consider writing them in plain language, avoiding complicated phrasing or overly techy explanations.

Test cases. A test case is a set of specific actions that are taken to test and verify a particular system behavior, feature, or functionality. Test cases are more detailed units that have to correspond with all the test scenarios. Most often you will convert your user stories and business use cases to write efficient test cases. Examples of test cases are:

1. Check unregistered user adding the product into the shopping cart.
2. Check shopping cart filtering.
3. Check the “continue shopping” button.

Test cases are efficient when there is a clear purpose stated, and the user is able to understand what they should do to complete it. The user guide to a test case may look like this:

1. Open the application.
2. Add any product to a shopping cart.
3. Authentication is not needed. 4. Proceed to the shopping cart.

You may also include expected results in the test case, so that the user is aware of what is going to happen:

1. The product will appear in a shopping cart.
2. The system will ask you to authorize as a registered user.

Reporting standards. Define how a report should look and what information an end user should provide.

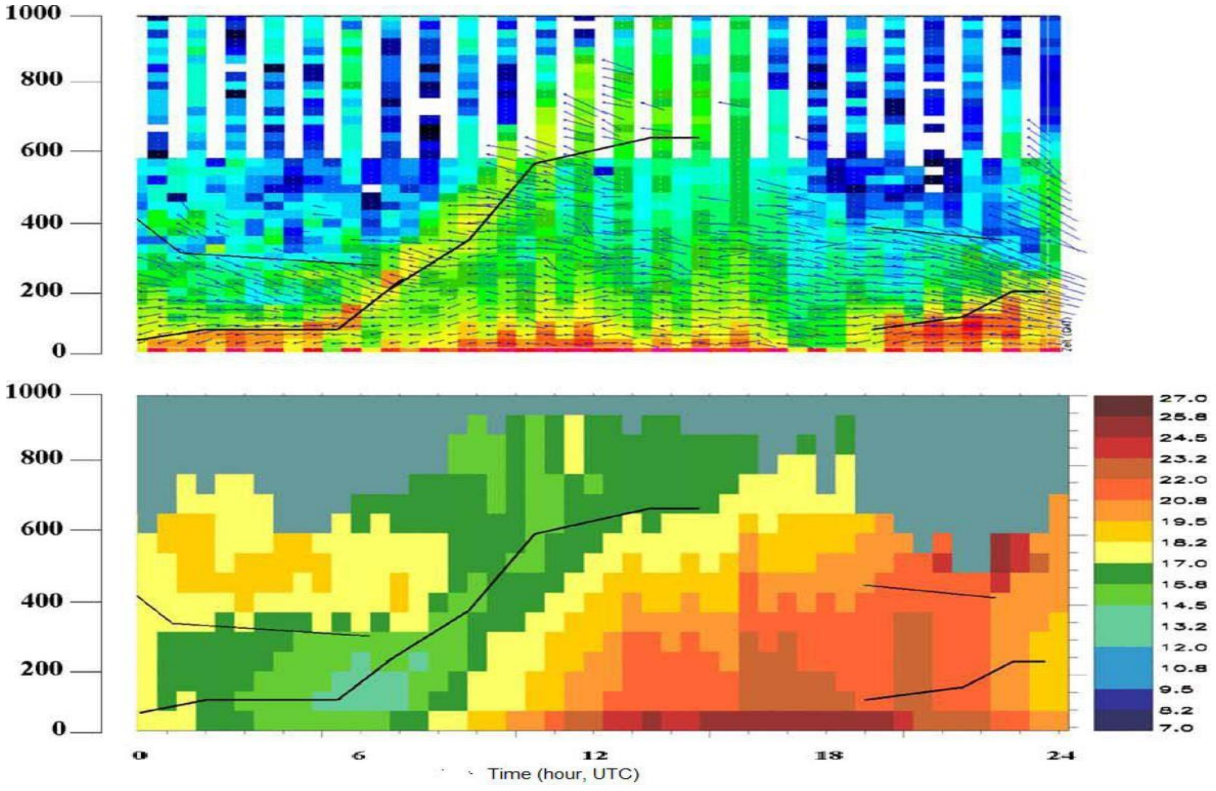
Test reports. These accumulate documented output data when the test is completed. Depending on the testing standards and testing scenario, various information can be included in reports. But typically in UAT, QA teams will require only a sign-off from the tester. A sign-off is just a confirmation that the test is successful and it corresponds to the user's criteria.

At the end of UAT, deliverables provided can be used by QA engineers or a UAT manager to extract valuable data and communicate results to the development team.

Traditionally, quality assurance engineers will be responsible for processing end-user feedback. The results of tests, bug reports, and fail/pass records are provided to developers to ensure constant communication between different parts of the team. Based on the enduser feedback, the QA team can also provide software quality metrics to measure progress in terms of UAT.

User acceptance testing templates

We've mentioned a few important documents that have to be created for proper UAT planning and execution. There are different ways to write them, but here are some templates that may come in handy.



-

Test plan templates: Test Plan template by Coley Consulting, sfsu template (downloadable link), or iiba template (downloadable link)

- Test scenario template
- Test report template

2. Choose the time and form of end-user testing

Acceptance testing may take place at different stages of the project, depending on the methodology you are using, but typically it's conducted at the end of the development cycle before release. As two of the most popular project management methodologies in software development are Waterfall and Agile, we will look at the user acceptance testing process within those two models.

Acceptance testing in the Waterfall model

To dive deeper into the details, we need to quickly recap what a Waterfall model is. It's a traditional project management methodology based on a step-by-step development of the product.

The stages don't intersect, meaning there is no simultaneous designing and design testing, or development and testing. The whole process is strictly documented and intended to deliver a fully functional application at the end of development without iterations.*User acceptance stage within the Waterfall mo*User-acceptance testing in Waterfall takes place at the final stage of development, right before the launch.

It can be conducted only after the system is considered code and function ready, after achieving the following benchmarks.

Product business requirements have been met.

-
- The code base is finished.
- QA activities (system, integration, unit testing) have been completed.
- Bugs revealed during the QA stage have been fixed.
- Minor visual issues are in an acceptable range.
- User-acceptance environment (UAT manager, tools for testing, test scenarios, etc.) is created.

In the Waterfall model, user-acceptance testing is the definitive point that demonstrates software readiness. If a product meets user acceptance criteria, it means the product is ready for production. UAT activities, in that case, are for completing the system check, its functionality, usability, and bugs. But still the primary goal is to ensure that the product corresponds with the initial requirements and enduser needs.

User acceptance in Agile methodologies

The Agile model of software development isn't as straightforward as Waterfall. It's based on iterating each development stage until the product reaches the required quality and functionality. Iterations of each phase allow for highly flexible development and dynamic change in requirements, as Agile doesn't focus on creating much documentation. And that allows the development team to quickly respond to changing requirements from the customer.

*User acceptance testing in Agile mod*The image shows the Agile product development cycle with iterations. You can conduct user acceptance testing at each stage of the project to ensure product validity. The main difference between UAT in Waterfall and in Agile is that UAT is carried out multiple times (often within each iteration) and its results may impact the initial requirements, as it delivers instant feedback on what works the best.

The checkpoints for starting end-user testing in an Agile project are formed business requirements,

- UX/system documentation,

-
- testing material (interactive mock-ups, high-fidelity prototypes, demos), and
- user-acceptance environment.

In Agile, UAT is an integral part of overall testing activities, so it may take different forms and use different tools. For example, these can be tests on functional and nonfunctional requirements or early-stage testing to validate assumptions made during the planning stage. At the end of each iteration, acceptance testing produces deliverables that are used to modify requirements, system architecture, UX style guides, etc.

3. Recruit users and form UAT team

As we mentioned earlier, testers can be recruited from your existing user base. Depending on the project specifics, those can be subject matter experts, real-world users of the product, stakeholders, business analysts, product owner, or the customer. You can also use crowdsourcing platforms to search for testers or hire a freelance user-testing specialist.

Consider creating a social media message or even a landing page to attract an audience. Your potential testers should not necessarily be tech savvy or familiar with software testing processes. However, the ones who already have or will use your product (or maybe a similar one) will be good candidates for your UAT since in that case you can avoid deep onboarding and QA team involvement.

4. Implement end-user testing tools and onboard testers

Of course, there are specific instruments on the market that are designed for end-user testing. Most popular tools offer testing management functionality like reporting, task overviews, and testing documentation templates. Here are some examples of software that can be used to support your UAT activities.

Usersnap is a popular platform for providing visual feedback on the tested software and web-based applications. Basically, it's a tool that allows users to mark the bugs right on the screen, leave comments and suggestions, and share the feedback. There are a lot of similar instruments such as Userback and UserTesting.

FitNesse is an open source, wiki powered framework for acceptance test automation. It allows all stakeholders to easily create, edit, and run tests, creating early feedback. Users enter specially formatted inputs to automatically generate tests that are immediately run by the system. Then the output is returned and highlighted depending on whether it matches the expected result or not. This collaboration platform has a mild learning curve and is popular among Agile teams.

Bugwolf is another instrument for conducting UAT. Besides testing environment and bug reporting, it offers gamification and competition features to motivate and engage users. You will also find useful built-in payment options if you are going to conduct end-user testing online.

Well known project management tools like Jira or Trello also have functionality for conducting UAT.

Testing dashboard in SpiraTest

5. Create user acceptance environment and run training

To get the most out of end-user testing, start with the training. Your testers and UAT manager are responsible for that. Consider structuring your training process to include the following aspects.

- Introduce users to the testing process and its objectives.
- Train users to use tools for end-user testing if you are going to use them.
- Provide them with reporting standards and guidelines.
- Ensure users understand test cases properly, providing support if needed.
- Supply them with access to the testing environment.

Most often end-user testing can be done on the user's side, meaning you won't have to supply your testers with the hardware. The whole process can also be done online. More complicated projects or confidential data may require gathering a dedicated team of user testers at your office. It's also important to appoint a manager who will provide documentation, tools, and support.

6. Run the tests

Once you have your test scenarios and test cases, you are good to go with the tests. To support your end users during the process and get the required results, deliver a clear understanding of what actions each test case requires. Keep in mind that your users are not professional testers. During the test, be sure to provide real or close to real data to the users, avoiding sample content or dummy buttons. Any misconception may get them stuck at the test case.

Another important aspect is having your developers ready to fix anything that goes wrong. Your testing environment can shut down or there can be errors preventing users from testing. Users should be able to access required functionality at each stage of testing, whether it's an interactive design or a functional app, to allow them to perform each test case included in the test plan.

7. Collect output information and analyze it

During your UAT activities, you will get tons of data from the testers. Your QA team will have to analyze it. The data is collected via user reports submitted manually or via a specific tool. Additionally, you can conduct interviews with

separate users to get more insight about the test cases they performed and what they think of them.

To evaluate system readiness, consider measuring the percentage of tests passed/failed/fixed.

Test tracking dashboard in Panaya

There are also a few more points that should be considered:

System stability. Stability can be determined by the number of unexpected errors met during the UAT.

Coverage of testing. Coverage is measured by the number of test scenarios/cases written and their ratio to the overall finished tests. You may also match your UAT testing results with the user journey map to understand which part of functionality was left untested.

Usability of the system. This can be calculated by the number of tests not passed because the user didn't find a way to do it. But the overall UX is tested during usability testing, which is conducted as a separate activity.

Contract/requirement compliance. Requirement compliance is checked after all the end-user tests are finished. It ensures that the software build still corresponds to the initial requirements/contract scope, even after changes brought by user acceptance.

8. Fix bugs, retest, and sign-off

After executing UAT, all defects have to be documented with relevant comments and passed to the development team. They have to make adjustments to the code to address the issues revealed by UAT.

Once you fix the bugs, retest them to make sure everything is working properly. When the acceptance criteria are reached and approved by reviewers, the final acceptance decision is made about productionreadiness of the product.

UAT team roles

As we mentioned earlier, UAT testing is different from other QA activities because it's performed not only by tech specialists; it's also important to engage actual end-users in this process. Involving QA professionals and business analysts is also necessary, as is close collaboration with the project manager and development team.

The responsibilities of the UAT team can differ depending on the company and project needs, but here is an example of the role distribution you can consider.

Business program manager. This is the person that coordinates and oversees the entire project, aligning it with business objectives. Before the UAT stage, the program manager should generate the program delivery plan and business requirements document to support the testing activities. He/she is also responsible for reviewing and approving the test plan and test strategy.

During UAT, the program manager monitors test activities execution and ensures completion on schedule and budget. Afterwards, he/she reviews the test report and decides on deployment to production.

UAT test lead/manager. The test lead's responsibility is to accurately plan and organize the UAT. For that, usually close cooperation with the project manager is required.

The test lead gathers and analyzes all the business and functional requirements that are then used to develop the necessary documentation, i.e., test strategy, test plan, test scenarios, etc. Additionally, at the preparation stage, he/she works with the test team, assigning test scenarios to team members and organizing training to make sure testers understand the UAT procedure. The test lead also prepares and manages the necessary resources and loads essential test data in test tools.

Throughout the UAT, the lead coordinates the execution of test cases, making sure all test results are documented. He/she also tracks test progress, collects metrics, and creates/maintains a test report.

UAT test team members. The main task of the test team is to execute tests in accordance with the provided schedule and instructions.
Testers should create test logs and report on defects and incidents. They also typically participate in retesting activities (if needed).

The **project manager,** as the person responsible for successful project completion, has to monitor testing activities, provide organizational support, and report on progress. He/she would also act as a mediator between the testing team, developers, customer, and any other possible stakeholders.

UAT checklist

Summing up the UAT guidelines we presented above, we've developed a checklist to help you organize your testing activities and not miss out on anything important.

Initiating the UAT project.

1. Verify with your development team that all product components are ready for testing. Document any issues that could not be addressed before the UAT.
2. Identify the key stakeholders.
3. Choose a team leader responsible for the project, including paperwork.
4. Discuss and agree on the project structure, UAT team, and UAT documentation.
5. Thoroughly discuss the testing procedures and create an initial UAT plan.

Planning UAT.

1. Create your UAT team and make sure you have testers from each market segment and/or each group of stakeholders. Be certain all participation-related documentation is complete and signed (nondisclosure, participation agreement, etc.).
2. Communicate the testing strategy and schedule to the team. Make sure every member understands the roles, procedures, and responsibilities.

3. Make sure all business requirements are captured and communicated to the UAT team.
4. Discuss and agree on entry and exit criteria.
5. Prepare all the business documentation: test plan, test scenarios, test cases, etc.
6. Communicate the business objectives and acceptance/exit criteria of the system.
7. Agree on reporting standards.
8. Conduct the needed training on the system and auxiliary tools. Make sure the testers understand how to report incidents.
9. Gather and prepare all the necessary resources for UAT activities. Book space if needed.
10. Prepare and test the environment, test management tools, devices, servers, feedback channels, issue tracking, content delivery, etc.
11. Make sure you have all the logins, that security access has been set up, and test data has been loaded.

Executing UAT.

1. Monitor how procedures are carried out and make sure reports are submitted timely and accurately.
2. Create and maintain the test summary report.

Post-UAT activities.

1. Analyze the output information by measuring the percentage of tests that passed/failed as well as categorizing defects by severity.
2. Identify status against acceptance criteria.