What Is User Acceptance Testing (UAT): A Complete Guide

Learn what is User Acceptance Testing (UAT), along with its definition, types, steps, and examples:

My rule number one when trying to understand a new concept is that: **the name is always going to be relevant and mostly a literal meaning** (in the technical context).

Finding out what that is, will give an initial understanding of it and help me to get started with.

Let us put this concept to test.

What Is User Acceptance Testing?

We know what testing is, acceptance means approval or agreement. The user in the context of a software product is either the consumer of the software or the person who requested it to be built for him/her (client).

So, following my rule – the definition will be:

User Acceptance Testing (UAT), also known as beta or end-user testing, is defined as testing the software by the user or client to determine whether it can be accepted or not. *This is the final testing performed once the functional, system and regression testing are completed.*

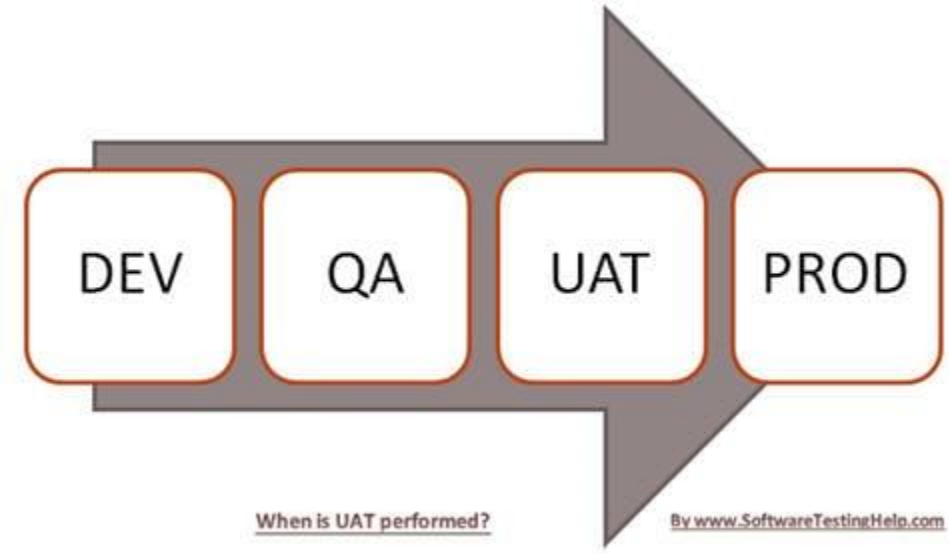
The main purpose of this testing is to validate the software against the business requirements. This validation is carried out by the end users who are familiar with the business requirements.

UAT, alpha and beta testing are different types of acceptance testing.

As user acceptance test is the last testing that is carried out before the software goes live, obviously this is the last chance for the customer to test the software and measure if it is fit for the purpose.

When is it Performed?

This is typically the last step before the product goes live or before the delivery of the product is accepted. This is performed after the product itself is thoroughly tested (i.e after system testing).



Who Performs UAT?

Users or client – This could be either someone who is buying a product (in the case of commercial software) or someone who has had a software custom built through a software service provider or the end user if the software is made available to them ahead of the time and when their feedback is sought out.

The team can be comprised of beta testers or the customer should select UAT members internally from every group of the organization so that each and every user role can be tested accordingly.

Need for UAT

Developers and functional testers are technical people who validate the software against the functional specifications. They interpret the requirements according to their knowledge and develop/test the software (here is the importance of domain knowledge).

This software is complete according to the functional specifications but there are some business requirements and processes that are known only to the end users are either missed to communicate or misinterpreted.

This testing plays an important role in validating if all the business requirements are fulfilled or not before releasing the software for market use. Use of live data and real use cases make this testing an important part of the release cycle.

Many businesses who suffered big losses due to post-release issues know the importance of a successful User Acceptance Test. The cost of fixing the defects after release is many times greater than fixing it before.

Is UAT really necessary?

After performing loads of system, integration and regression testing one would wonder about the necessity of this testing. Actually speaking, this is the most important phase of the project as this is the time at which the users who are actually going to use the system would validate the system for its fit to purpose.

UAT is a test phase that largely depends on the perspective of the end users and the domain knowledge of a department that represents the end users.

As a matter of fact, it would really be helpful to the business teams, if they were involved in the project quite early, so that they can provide their views and contributions that would help in effective usage of the system in the real world.

User Acceptance Testing Process

The easiest way to understand this process is to think of this as an autonomous testing project – which means, it will have the plan, design and the execution phases.

The following are the pre-requisites before the planning phase begins:

#1) Gather the key Acceptance Criteria

In simple terms, Acceptance criteria is a list of things that are going to get evaluated before accepting the product.

These could be of 2 types:

(i) Application Functionality or Business Related

Ideally, all the key business functionality should get validated, but due to various reasons, including time, it is not practical to do it all. Therefore, a meeting or two with the client or the users who are going to be involved in this testing can give us an idea on how much testing is going to be involved and what aspects are going to be tested.

(ii) Contractual – We are not going to go into this and the involvement of the QA team in all this is almost nothing. The initial contract that gets drawn up even before the SDLC begins is reviewed and an agreement is reached upon whether all the aspects of the contract have been delivered or not.

We are going to focus only on the application functionality.

#2) Define the scope of QA involvement.

QA team's role is one of the following:

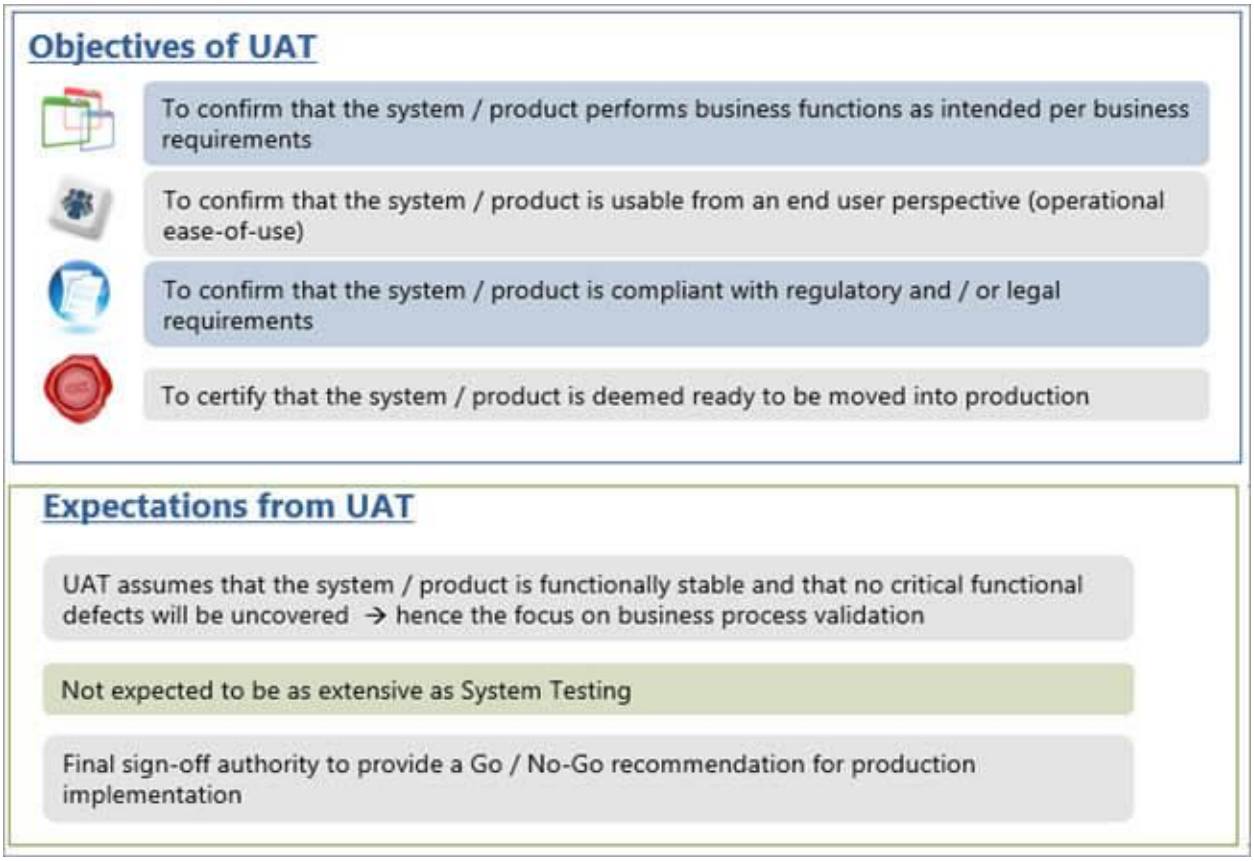
(i) No Involvement – This is very rare.

(ii) Assist in this testing – Most common. In this case, our involvement could be training the UAT users on how to use the application and be on standby during this testing to make sure that we can help the users in case of any difficulty. Or in some cases, in addition to being on standby and assisting, we might share their responses and record the results or log bugs etc., while the users perform the actual testing.

(iii) Perform UAT and present Results – If this is the case, the users will point the areas of the AUT that they want to evaluate and the evaluation itself is performed by the QA team. Once done, the results are presented to the clients/users and they will make a decision on whether the results that they have in hand are sufficient or not and in accordance with their expectations in order to accept the AUT. The decision is never that of the QA team.

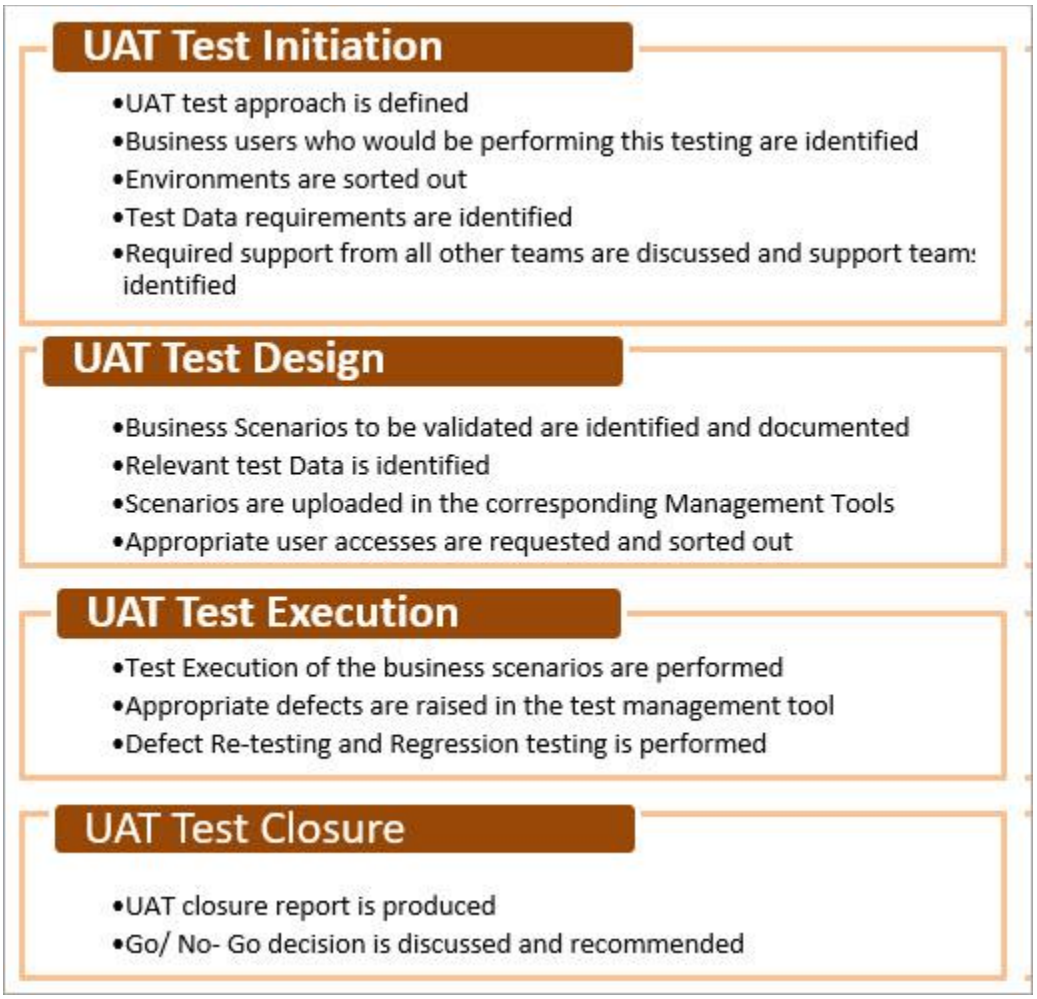
Depending on the case on hand, we decide which approach is best.

The primary Objectives and expectations:



Usually, UAT is undertaken by a Subject Matter Expert (SME) and /or a business user, who might be the owner or the customer of a system under test. Similar to the System testing phase, the UAT phase also encompasses religious phases before it is brought to closure.

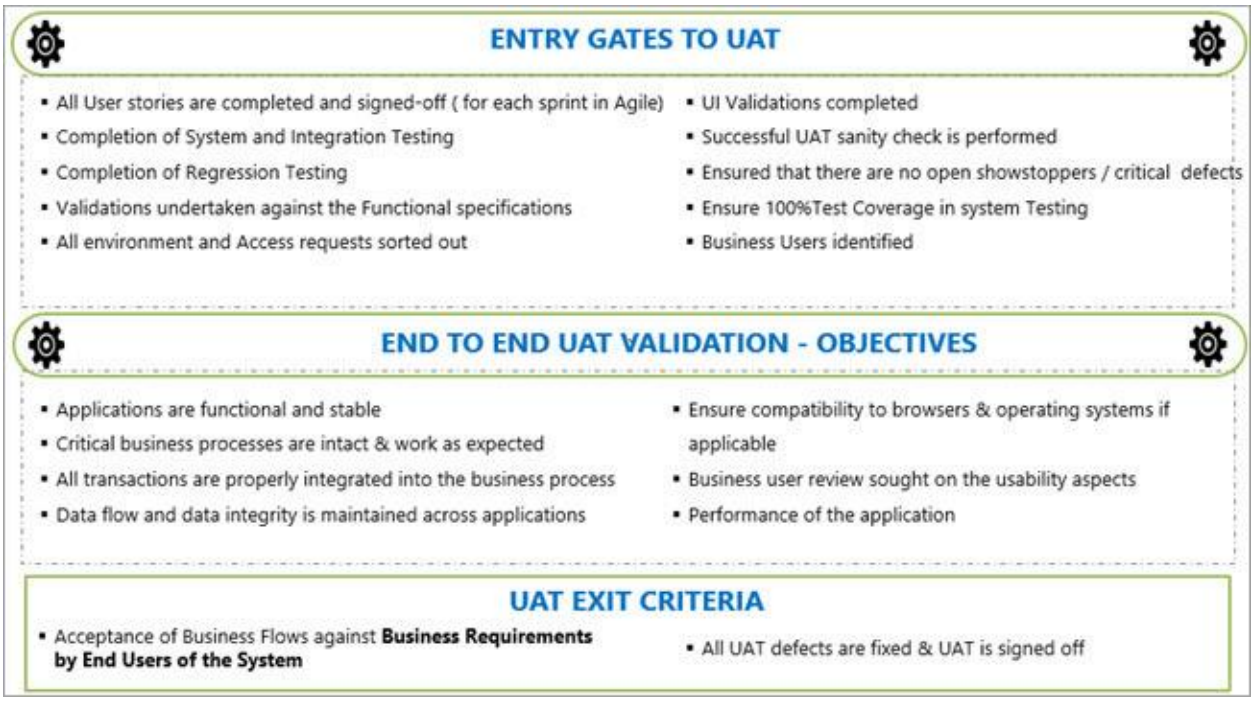
Key activities of each UAT phase is defined below:



UAT Governance

Similar to system testing, effective governance is enforced for UAT to ensure that strong quality gates along with the defined Entry and Exit criteria (provided below **).

** Please note that it is just a guidance. This could be modified based on the project needs and requirements.



UAT Test Planning

The process is almost the same as with the regular test plan in the system phase.

The most common approach followed in most of the projects is to plan for both system and UAT testing phases together. For more information on UAT test plan along with a sample, please check out the attached test plan document's UAT sections.

User Acceptance Test Plan

(This is the same that you would find on our site for the QA training series as well).

Click on the below image and scroll down to find the test plan document sample in various formats. In that template check the UAT section.

The dates, environment, actors(who), communication protocols, roles and responsibilities, templates, results and their analysis process, entry-exit criteria – all of this and anything else that is relevant will be found in the UAT test plan.

Whether the QA team is participating, partially participating or not participating at all in this test, it is our job to plan this phase and make sure that everything is taken into consideration.

=> Here is a User Acceptance Test Plan Sample Document

UAT Design

The gathered acceptance criteria from the users are used in this step. Samples could look as shown below.

(These are excerpts from CSTE CBOK. This is one of the best references available about this testing.)

User Acceptance Testing Template:

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 has been met.

Based on the criteria, we (QA team) give them the users a list of UAT test cases. These test cases are not different from our regular system test cases. They are just a subset as we test all of the application as opposed, just to the key functional areas.

In addition to these, the data, templates to record test results, administrative procedures, defect logging mechanism, etc., has to be in place before we move to the next phase.

Test Execution

Usually, when possible, this testing happens in a conference or a war room sort of a set up where the users, PM, QA team representatives all sit together for a day or two and work through all the acceptance test cases.

Or in case of the QA team performing the tests, we run the test cases on the AUT.

Once all the tests are run and the results are in hand, the **Acceptance Decision** is made. This is also called the **Go/No-Go decision**. If the users are satisfied it's a Go, or else it's a No-go.

Reaching the acceptance decision is typically the end of this phase.

Tools & Methodologies

Typically, the type of software tools that are used during this testing phase is similar to the tools used while performing functional testing.

Tools:

As this phase involves validating the complete end to end flows of the application, it might be difficult to have one tool to automate this validation completely. However, to some extent, we would be able to leverage the automated scripts developed during system testing.

Similar to system testing, users would also use test management and defect management tool like QC, JIRA etc. These tools can be configured to cumulate data for the User Acceptance phase.

Methodologies:

Though conventional methodologies such as specific business users performing UAT of the product is still relevant, in a truly global world like today, User Acceptance Testing sometimes has to involve varied customers across countries based on the product.

For Example, an e-commerce website would be used by customers across the globe. In scenarios like this, crowd testing would be the best viable option.

Crowd testing is a methodology where people from all over the world can participate and validate the usage of the product and give suggestions and recommendations.

Crowd testing platforms are built and are being used by many organizations now. A website or a product which needs to be crowd tested is hosted in the platform and the customers can nominate themselves to do the validation. The feedbacks provided are then analyzed and prioritized.

Crowd Testing methodology is proving to be more effective as the pulse of the customer across the globe can be easily understood.

UAT in Agile Environment

The agile environment is more dynamic in nature. In an agile world, business users will be involved throughout the project sprints and the project would be enhanced based on the feedback loops from them.

At the beginning of the project, business users would be the key stakeholders to provide requirement thereby updating the product backlog. During the end of each sprint, business users would participate in the sprint demo and would be available for providing any feedbacks.

Moreover, a UAT phase would be planned before the sprint completion where the business users would do their validations.

The feedbacks which are received during sprint demo and sprint UAT, are collated and added back to the product backlog which is constantly reviewed and prioritized. Thus in an agile world, the business users are more close to the project and they evaluate the same for its use more frequently unlike the traditional waterfall projects.

UAT Team – Roles & Responsibilities

A Typical UAT organization would have the following Roles and responsibilities. The UAT team would be supported by the project manager, development and testing teams based on their needs.

Roles	Responsibilities	Deliverables
Business Programme Manager	• Create and maintain Programme Delivery plan • Review and Approve UAT Test Strategy and Plan • Ensure the successful	• Programme progress report • Weekly status report

	completion of the programme on schedule and budget • Liaise with IT programme Manager and monitor the progress of the programme • Work closely with business operations team and equip them for Day 1 operation • Sign-off Business Requirement Document • Review the e-learning course content	
UAT Test Manager	• Crete UAT Strategy • Ensure effective collaboration between IT and Business BA and PMO • Participate in requirements walkthrough meetings • Review Effort Estimation, Test Plan • Ensure Requirement Traceability • Drive metrics collection to quantify the benefits derived out of the updated testing methodology, tools and environment usage	• Master Test Strategy • Review & approve Test Scenarios • Review & approve Test Cases • Review & Approve Requirement Traceability Matrix • Weekly Status report
UAT Test Lead & Team	• Verify & Validate Business Requirement against Business process • Estimation for UAT • Create & Execute UAT test Plan • Participate in requirement JAD session • Prepare test scenarios, test cases and test data based on Business Process • Maintain Traceability • Execute test cases and prepare test logs • Report defects in test management tool and manage them throughout their lifecycle • Produce UAT End of test report	• Test Log • Weekly Status Report • Defect Report • Test Execution Metrics • Test Summary Report • Archived Reusable Test artifacts

	• Provide Business Readiness Support and Live proving	

7 Challenges Of UAT And Mitigation Plan

It doesn't matter if you are a part of a billion-dollar release or a startup team, you should overcome all these challenges for delivering successful software for the end user.

#1) Environment setup and deployment process:

Carrying out this test in the same environment used by the functional test team will certainly end up overlooking the real-world use cases. Also, crucial testing activities like performance testing can't be carried out on a test environment with incomplete test data.

Separate production like environment should be set up for this test.

Once the UAT environment is separated from the test environment, you need to control the release cycle effectively. Uncontrolled release cycle may lead to different software versions on test and UAT environment. Valuable acceptance test time is wasted when the software is not tested on the latest version.

Meanwhile, the time required for issue tracking on incorrect software version is high.

#2) Test Planning:

This testing should be planned with a clear acceptance test plan in the requirement analysis and design phase.

In strategy planning, the set of real-world use cases should be identified for execution. It is very important to define the test objectives for this testing as a complete test execution is not possible for large application in this testing phase. Testing should be carried out by prioritizing critical business objectives first.

This testing is carried out at the end of the testing cycle. Obviously, it is the most critical period for the software release. Delay in any of the previous stages of development and testing will eat up the UAT time.

Improper test planning, in worst cases, leads to an overlap between the system testing and UAT. Due to less time and pressure to meet deadlines, the software is deployed to this environment even if functional testing is not completed. The core goals of this testing can't be achieved in such situations.

The UAT test plan should be prepared and communicated to the team well before beginning this test. This will help them for test planning, writing test cases & test scripts and creating a UAT environment.

#3) Handling new business requirements as incidents/defects:

Ambiguities in requirements get caught in the UAT phase. UAT testers find issues arising due to ambiguous requirements (by looking at the complete UI which wasn't available during requirement gathering phase) and log it as a defect.

The customer expects these to be fixed in the current release without considering the time for the change requests. If a timely decision is not taken by the project management on these last-minute changes, then this could lead to the release failure.

#4) Unskilled testers or testers without business knowledge:

When there is no permanent team, the company selects UAT staff from various internal departments.

Even if the staff is well familiar with the business needs, or if they are not trained for the new requirements that are being developed, they can't perform effective UAT. Also, a non-technical business team might face many technical difficulties in executing the test cases.

Meanwhile, assigning testers at the end of the UAT cycle do not add any value to the project. Little time to train the UAT staff can significantly increase the chances of UAT success.

#5) Improper Communication Channel:

Communication between remote development, testing, and UAT team is more difficult. Email communication is often very difficult when you have an offshore tech team. A small ambiguity in incident reports can delay its fix for a day.

Proper planning and effective communication are critical to effective team collaboration. Project teams should use a web-based tool to log defects and questions. This will help to distribute the workload evenly and avoid reporting duplicate issues.

#6) Asking Functional test team to perform this testing:

There is no worse situation than asking the functional test team to perform UAT.

Customers offload their responsibility to the test team due to lack of resources. The whole purpose of this testing gets compromised in such cases. Once the software goes live, the end users will quickly spot the issues which are not considered as real-world scenarios by the functional testers.

Solution to this is to assign this testing to the dedicated and skilled testers having business knowledge.

#7) The Blame Game

Sometimes business users just try to find reasons to reject the software. It might be their selfdom to show how superior they are or blame the development and testing team to get respect in the business team. This is very rare but happens in teams with internal politics.

It's very difficult to deal with such situations. However, building a positive relationship with the business team would definitely help to avoid the blame game.

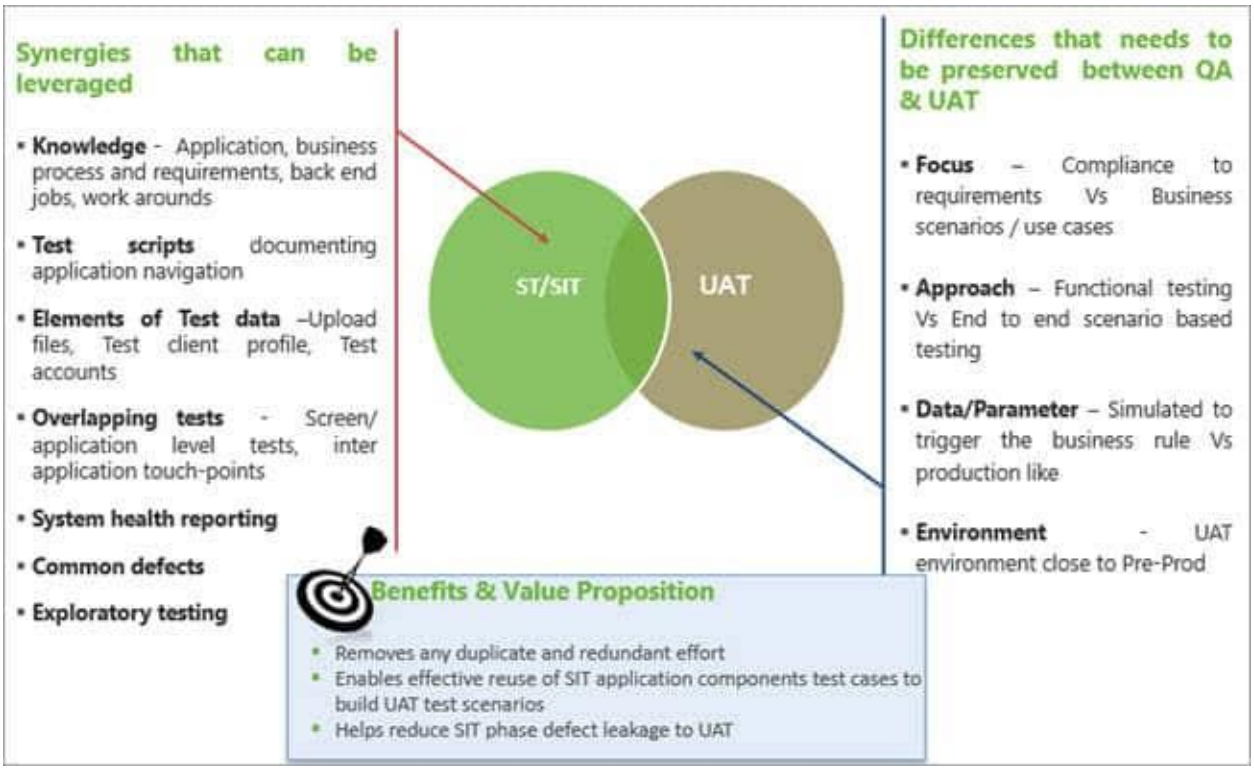
I hope these guidelines will certainly help you to execute a successful user acceptance plan by overcoming various challenges. Proper planning, communication, execution, and motivated team are the keys to successful user acceptance testing.

System Testing Vs User Acceptance Testing

Involvement of the testing team starts quite early in the project right from the requirement analysis phase.

All through the project life cycle, some kind of validation is performed for the project, i.e. Static testing, Unit testing, System testing, integration testing, an end to end testing or regression testing. This leaves us to understand better about the testing performed in the UAT phase and how different it is from the other testing performed earlier.

Though we see the differences in SIT and UAT, it is important that we leverage synergies but still maintain the independence between both the phases which would enable faster time to market.



Concluding Points

#1) UAT is not about the pages, fields or buttons. The underlying **assumption** even before this test begins is that all that basic stuff is tested and is working fine. God forbid, the users find a bug as basic as that – it is a very bad news for the QA team. :(

#2) This testing is about the entity that is the primary element in the business.

Let me give you an Example: If the AUT is a ticketing system, the UAT is not going to be about, searching the menu that opens a page etc. It is about the tickets and their reservation, the states that it can take, its journey through the system, etc.

Another **Example,** if the site is a car dealership site, then the focus is on the "car and its sales" and not really the site. Hence, the core business is what is verified and validated and who is better to do it than the business owners. That's why this testing makes the most sense when the customer is involved to a major extent.

#3) UAT is also a form of testing at its core which means **that there is a good chance of identifying some bugs at this phase too**. It sometimes happens. Aside from the fact that it is a major escalation on the QA team, the UAT bugs usually mean a meeting to sit and discuss how to handle them as following this testing there is usually no time to fix and retest.

The decision would be either to:

- Push the go-live date, fix the issue first and then move on.
- Leave the bug as it is.
- Consider it as a part of the change request for future releases.

#4) UAT is classified as Alpha and Beta testing, but that classification is not that important in the context of typical software development projects in a service-based industry.

- **Alpha testing** is when UAT is carried out in the software builder's environment and is more significant in the context of commercial off the shelf software.
- **Beta testing is when the UAT is carried out in the production environment or the client's environment. This is more common for customer-facing applications. The users here are the actual customers like you and me in this context.**

#5) Most of the times in a regular software development project, UAT is carried out in the QA environment if there is no staging or UAT environment.

In short, the best way to find out if your product is acceptable and fit for purpose is to actually put it in front of the users.

Organizations are getting into the Agile way of delivering, business users are getting more involved and the projects are being enhanced and delivered via feedback loops. All being done, User Acceptance phase is considered as the gate for getting into implementation and production.