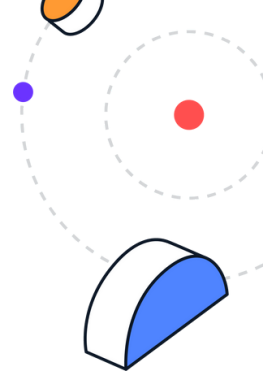




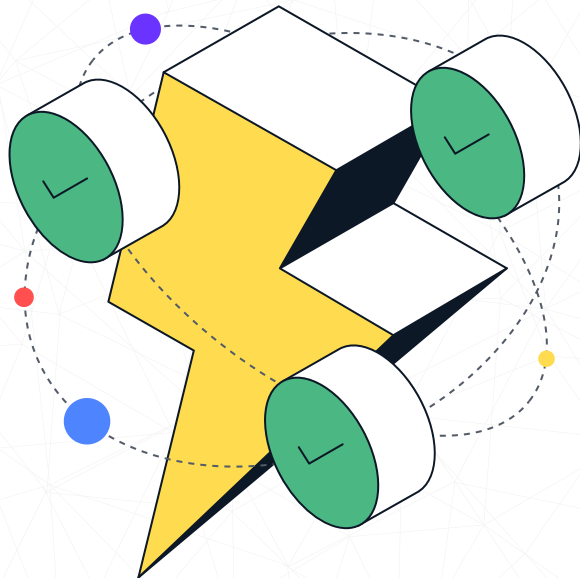
The Ultimate Guide to UAT: User Acceptance Testing

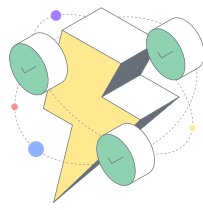




CONTENTS

Introduction	1
Scoping	2
Strategy	3
Planning	6
Preparation and initiation	7
Detailed plan	10
Execute the plan	11
Review and manage	13
Repeat	14
Retrospective	15
Conclusion	16





01

INTRODUCTION:

Organizing, managing, and executing any type of testing or testing project is substantial and can be daunting if you are not used to it.

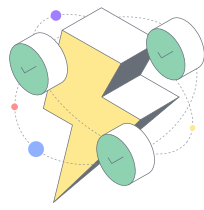
When it comes to User Acceptance Testing (UAT), as opposed to testing in a dedicated QA team, the challenges change, and there are other factors to consider. The work involved, and the related challenges that face the UAT team and managers, are often not well understood by IT management or business user management. Thus, there can be a mismatch in expectations leading to frustration and less than ideal outcomes.

UAT inevitably trends towards the end of any software delivery project and sometimes is not even considered by those with a more development focus, to be part of software delivery. Both factors can have negative influences on the UAT phase and the outcome. In the first instance, time pressures may be increased by approaching deadlines set long ago, and in the second, any development, and to a degree, QA teams who have been involved up to this point will be less interested as they have already 'moved on.'

Equally, the importance of UAT, and the considerable cost, are often not appreciated by everyone. But those who understand its value know that without useful input from the users, acceptance of the changes, or new system, module, etc., will be affected. Even worse, residual errors may cause costly disruption and bad feelings. How often has the expression "but it is your system" been used accusingly by various parties? In most cases, the cost associated with UAT of business systems will be significantly greater than any other phase or type of testing. So it is worth investing in, making it as efficient and as successful as possible.

The goal for any UAT phase of a software delivery project is to support a smooth implementation of the change. This means flushing out any residual errors, ensuring the business processes work smoothly and efficiently, that the users know and are comfortable with their part in these revised processes. Also, it should be remembered that the project's commercial goals need to be achieved, such as business operation and efficiency is improved, or a capability is enabled which did not previously exist.

This document aims to look at all the possible activities that might apply in a UAT project for completeness and transparency and help put a framework in place to give the project the best chance of success to maximize positive outcomes from each activity and the project overall.



02

We have tried to document all the major aspects of a UAT project that should be considered. This does not mean to say they all need to be addressed in every project. This is primarily a list of things we think should be considered to obtain the best outcome. Pick the ones that work for you in your situation

SCOPING:

1) Work with product teams and management to determine timescales

While UAT may be an event towards the end of a project, it is still wise to start early. Engage with the teams involved in earlier phases to make sure that UAT is on their radar and considered in their plans. They might not have even considered that there is a possibility that some defects will be identified and will require fixing.

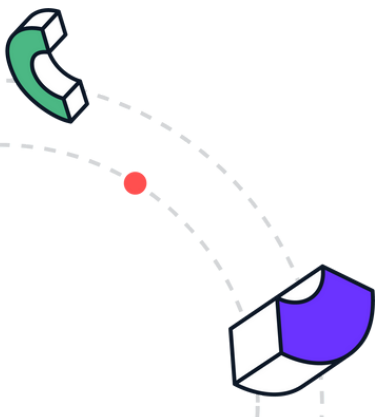
In an agile environment, the end may not be that far from the start, and acceptance may occur in stages within a single sprint or iteration.

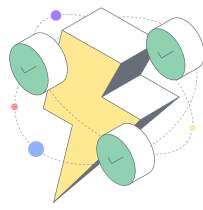
At the same time, you need to think about where resources will come from and what the overall strategy will be. You may need to stake your claim firmly and early. This is also an excellent time to get a handle on why the project is being carried out in business terms; there must be a business benefit somewhere, and this needs to be writ large in peoples' minds because if this is not achieved, the project will not be a success. This headline may be critical in getting and maintaining everyone's attention and commitment.

- What does the whole project look like?
- What are the business goals for the project?
- What are the previous phases?
- Who are the other parties involved?
- What methodology is being used, and where does UAT fit in?
- What is the strategy and content for previous test phases? What can be re-used from these?
- What allowances have been made for issue/bug refactoring?
- Planned go-live dates and why.
- The extent of the changes.
- Believability that the previous milestones before UAT will be met.

2) Analyze requirements and changes

- Significance of change.
- The scale of change.
- Systems affected.
- Business teams affected and the number of users.
- Existing assets affected or which can be used.
- Vendor input and advice from previous clients.





03

3) Work with product teams and management to determine the impact of changes and business risk

Scale and impact can be closely related but considering that it is possible to have a very small change that is so significant, the effect is disproportionate. You may need to work with systems architects and analysts to discover the impact's scope and allow for this in your plan for UAT.

- Systems impact
- Business impact, how vital are the affected systems?
- IT team impact
- The risk to business because of undetected errors
- The risk if timescales are not met
- Scope of testing required

STRATEGY:

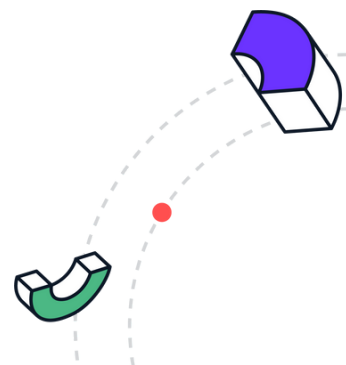
4) Refer to test policies

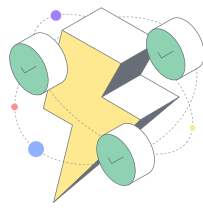
Different organizations have different cultures and methods. Some of these may have developed from historic need and others from a more dynamic approach. It does not mean, however, that existing policies are going to be perfect for this situation. However, if you are going to adapt or vary from them for a good reason, you will need to make this clear early on. If they exist, then the points therein will have been put there for a reason which might be more than 'cultural', so take a good, considered look. If they don't yet exist, then you have a golden opportunity to write them for future guidance and best practice for your organization.

- If you have an organization policy for UAT, does it work for this?
- Does it need altering?
- Do they need writing?
- If not going to follow, then why not, and who agrees or otherwise?
- How is the project risk affected by any policy change?

5) Determine the test strategy

- Do you need a fully test case-driven, documented approach?
- How much exploratory testing is appropriate?
- What resources will be used, and in what way?
- Who will write test cases?
- Level of detail needed vs. SME knowledge
- Are detailed test scripts required?





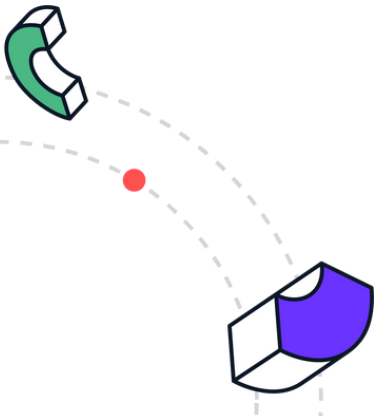
04

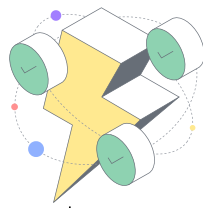
- Who will carry out the tests?
- How will they do them?
- How many rounds of testing will be required?
- Will it all be manual?
- Exploratory testing phase?
- Can you build in automation as you go?
- How will you determine readiness? What metrics will guide you in this, and how will you gather these along the way

6) Determine which tools to use

There is no doubt that appropriate technology can make life easier, and maybe you already have this in-house. There are 3 or 4 key areas to consider.

- A platform to manage the assets of the projects combined with communication, resources assignment, and scheduling. Fundamentally a test management solution, but it needs to be in a format that supports the users' needs and the project. For example, depending on the approach you take, you may have UAT 'issues' distinct from 'defects. You may have 'feedback' which can be positive, negative, or neutral. You may wish to include surveys and questionnaires to gauge user confidence and buy-in. You might also be taking a 'crowd testing' approach to resourcing rather than pulling on specific people for allocated tasks. Whatever your approach, the Test Management platform needs to be designed to support the style of UAT being planned and the needs of the users.
- One of the most time-consuming and frustrating aspects of performing UAT is producing test evidence and documentation. Users quickly get fed up with alt + PrtScn + paste into a Word document or spreadsheet. Plus, that does not tell the full story, and when the question is asked, "What exactly did you do?" that slow and inefficient approach will struggle to answer it correctly. It is a 'must-have' technology that captures user actions unobtrusively and accurately, with little or no effort, especially if you are planning on any exploratory testing. If you don't provide this, you are not treating the users or the persons that must deal with the feedback with any respect. Inexpensive solutions exist for this at the price of a cup of coffee a day. The availability of a good capture solution may also take the pressure off the level of detail required in test cases or test scripts because minute instructions are less important if you know what happened.





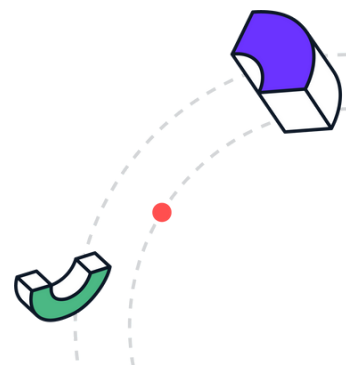
05

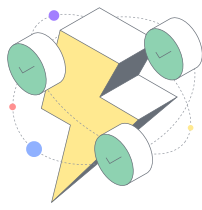


- Automation. UAT does require user involvement. That involvement is immensely beneficial for ownership, process optimization, and adoption. However, if the users have become a little jaded after the third cycle of creating 20 new suppliers, you will not be surprised. This would be a clear case of an opportunity to learn about what that business process is and to automate, whether for this project or to help with future change. If you don't look for opportunities to automate some aspects now, you are consigning all future testing to a time-consuming manual approach.
- Test Data management. Testing needs test environments and ideally consistent data to go with it. This is especially true for successful automation, but if your UAT scripts all start with "First find a Purchase Order which has not had goods received yet," you are slowing the process down and making the result less predictable and therefore harder to validate.

Factors to consider in your choice of supporting technology

- How will you organize test assets?
 - Requirements
 - Test Cases
 - Test Scripts
 - Test Results
 - Feedback
 - Issue Reports
 - Defects
 - Surveys
- How will you ensure traceability for test coverage?
- Will you allocate test tasks to teams or individuals?
- How will users know what to do and how to do it?
- Monitoring progress? Key metrics?
- How will you capture test results?
- Can you automate some aspects?
 - If so, what solution fits your team, skills, timescales, approach, and objectives?
 - Can you use existing tools, or do you need to acquire them?
- How will you create consistent test environments ideally without a lot of disk space and time involved?
- How will you communicate with product teams, business users, stakeholders for buy-in?





06

PLANNING:

The time has come to start to map it out, month by month, week by week, or whatever time unit suits you at this stage. Keep it loose and rough at this stage as it will change, and flex as other factors influence it. You might be best off doing this as a presentation as a critical challenge will be to get agreement and buy-in from stakeholders.

7) Outline the test plan

- Pull the above elements into a schedule in weekly segments.
- How realistic is it?
- Where are the compromises?
- Consider areas of risk.
- Engage with product teams, business users, stakeholders for buy-in.

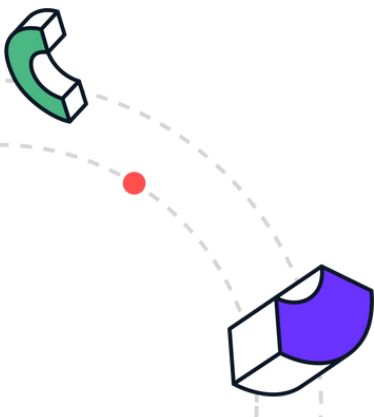
8) Determine the resource needs

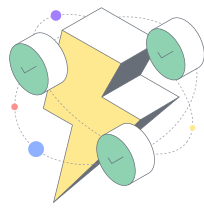
- What resources are needed to deliver the plan?
 - Location
 - Skills
 - Availability
 - Can they be supplemented?
- Engage with product teams, business users, stakeholders for buy-in.

9) Assess the role and impact of interfaces and other systems

No system is an island these days in this world of connectivity. Almost certainly, any meaningful application for which you are planning UAT will have connections to other applications, internally and possibly externally. This complicates the challenge, extending the coverage and nature of test cases, resources you may need, and test environments.

- Most businesses operate on many integrated solutions, each specialized in its own field.
- Does this widen the business teams to be involved?
- Do you have linked test environments to support this end-to-end testing?
- Consider the impact on test cases and test data.





07

PREPARATION AND INITIATION

There are many things to consider in the development of the plan. Hopefully, the following covers most of them, and with a bit of luck, more than you need. Use it as a checklist and adapt to your situation.

10) Understand and capture business processes

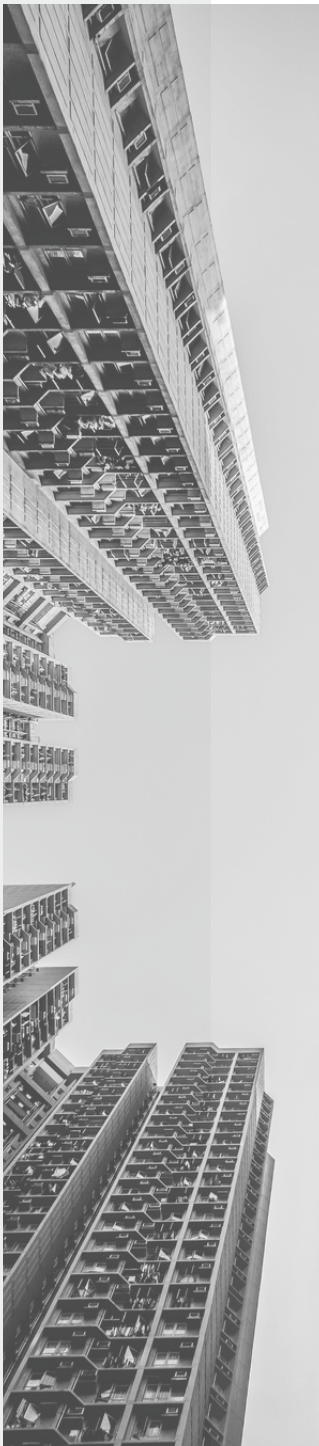
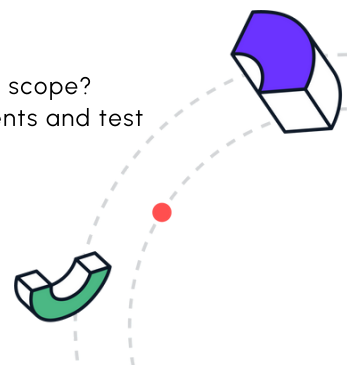
- You need to know how the business operates to be able to test a change in that area. Is this information available in a documented form or as knowledge in key users?
- It may be that these processes are going to change, and testing will be in a conference room pilot or model office environment, in which case you need to know the business inputs, outputs, and outcomes. These will surely have been mapped out by now if part of a new implementation, but you will need access to that to plan the testing.
- You might want to document these as use cases using the classic structure. Still, that format does not lend itself well to multiple scenarios, so you need to support any process definition with the required data variations – both for input and outputs. Spreadsheets are great for this (but not for managing the rest of the work!)
- If the project is largely revisions & additions to an existing system, you might be better off capturing existing production processes if that knowledge does not already exist. If so, use capture technology to document the 'As-is' business processes. You can use this to create text style instruction test cases or even how-to videos, which users can refer to along with the data scenarios. You will need to organize these assets, and your UAT management solution should provide the repository and ability to organize them.

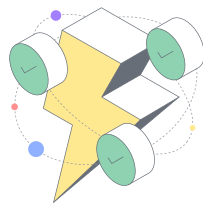
11) Create or adapt test cases

There are several important factors to consider when looking at the content of test cases and the first of these is coverage. In other words, have you identified all the processes and scenarios that need to be tested? This analysis is critical, and it may require considerable input from business users to create them in the first instance. Some organizations delegate this task entirely to the users, which might be the best way. Still, if you do, you will probably need some inspection and reconciliation to ensure that you have achieved the necessary level of coverage.

12) Check coverage

- Do the test cases provide full coverage of the test scope?
- Maintain traceability between changes/requirements and test cases





08

13) The level of detail in your test cases may vary depending on several factors

It may be possible to perform effective UAT with very lightweight test cases, but this will depend on the level of system knowledge in the people performing the tests. If you don't need the detail, there is no reason to put it in the test case. Historically detailed test cases and scripts also helped define what the tester was doing so any issues could be described in the context of that series of events. But these days, that is not necessary, provided you use capture technology which shows precisely what the user was doing and enables them to document their concerns and feedback.

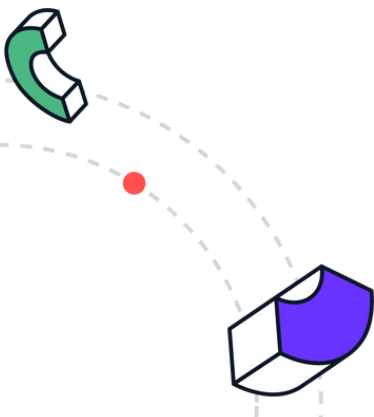
Whatever the approach, these test cases have become valuable assets that we want to re-use, and hence they need to be stored and organized in a way that supports that.

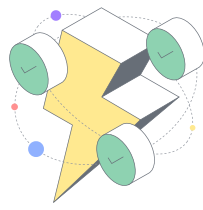
- Organize test case repository and classification.
- Depending on who is preparing test cases, what help and guidance do they need?
- Amend existing test cases if available to match changes/requirements.
- Create test cases where needed.
- Consider the sequence and dependencies.

14) Create test scripts

The same principles as outlined for test cases apply to any scripts which take instructions to the next level. Coverage and level of detail are essential in relation to the skills and knowledge of the persons executing the tests. Again, using a purpose-designed repository that makes scripts accessible, maintainable, referenceable, and re-usable is fundamental to efficiency and avoiding future frustration.

- Create a standard script template if there is not an existing one.
- Consider if detailed scripts for each test case are required.
- Base scripts on captured existing processes, if available.
- The UAT management platform should support a script repository and linked to test cases.
- Consider what format to hold the scripts in.
- Are training aids required to assist in script execution?





09

15) Determine test data needs and strategy

There are two aspects of data that need to be initially considered:

- What data is needed to be processed
- What supporting data is required

That means considering where the required data will come from, and this is inevitably based on what data is already there. This data needs to be treated as an asset. You don't want to have to rebuild it or recreate it every time you go through a test cycle, although that is sometimes a typical use of automation – to load data needed for subsequent tests. But if you can avoid this by diligent use of technology and virtualized environments, this will save considerable time and effort.

- Plan where and how to provide test environments(s).
- Can virtual environments be used?
- Consider the backup and reset strategy for the data.
- What supporting technology is required?
- You will probably need to coordinate with IT, infrastructure teams.

16) Test the test scripts

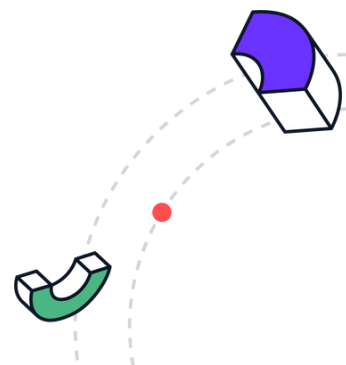
If you have created test scripts to be given to the users, you will need to check that they are workable and understandable for at least some of them. If the users have created them, you probably won't need to.

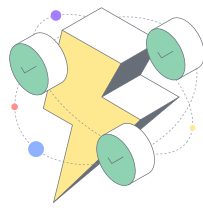
- Do you need to check the content, format?
- Liaise with users to check their needs and the script content or check what they have developed.

17) Risk and priority assessment

There may come a situation where some corners must be cut, perhaps because of time or resource pressures. In an ideal world, this does not happen, but in real life, it happens all too often, and at times organizations have paid the cost of not having thoroughly tested changes or having fully engaged and informed users. But if such a situation is approaching, it may make it more palatable if the critical tests have been carried out successfully and only lower-risk ones remain. Know what category they fall into may help release and planning decisions.

- Which tests are the most important?
- Are there any that could be skipped if needed?





10

DETAILED PLAN:

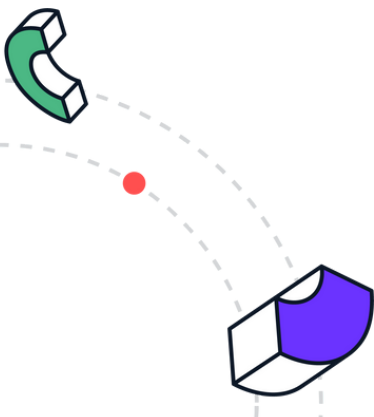
The nature of the strategy you are taking will determine the level of detail required in the plan. If you are allocating test tasks at an individual or perhaps department level, you will need a granular plan which combines the task, the test case, the person, and the timescales. Suppose you are adopting a looser 'crowd' style strategy. In that case, the allocation of individual tasks becomes less of an issue in the first instance, but the monitoring of who has done what (and what has not been done yet) becomes more important.

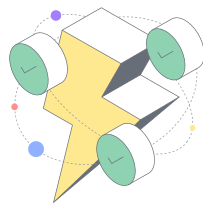
The crowd approach has some advantages because it is more dynamic, resource availability determines the progress, and there is greater visibility of the whole project, which can invoke greater accountability. Broadly it involves making the test cases available to anyone who has the time and skills to carry them out and then monitoring the results, perhaps with a pre-determined execution and pass level. One downside is that timescales may be less predictable and enforceable, so you will want to make sure this approach if used, suits the project.

18) Collaborate with the user department heads for resource access

Who is going to do what and when?

- Publish goals, the approach, and the timescales.
- Determine skills, names, availability of seconded resources.
- Obtain commitment from their management.
- Establish communication method and frequency.
- Use a central platform to enhance visibility and remove the reporting burden.
- Plan the sequence and dependencies of tests.
- Allocate time, resources, and areas for exploratory testing.
- Consider Timescales.
- Determine resources.
- Plan communication and access to the central platform.





19) Exploratory testing

Exploratory testing should be considered as part of the whole quality mix. The value may depend on many factors, not least the influence you have over the final users. The less influence, the more exploratory testing you should build in. For example, suppose you have a small group of in-house users who have a very dedicated job and will follow the correct process religiously. In that case, you may not need to be concerned about much exploratory testing. However, suppose the application is to be rolled out publicly and contains sensitive or valuable information. In that case, you can be sure there are people out there who will try to break it, and it is much better you do that internally first. You may already have included this in earlier QA phases of testing, but why not let your users run free also?

20) Communicate the plan to managers and individuals

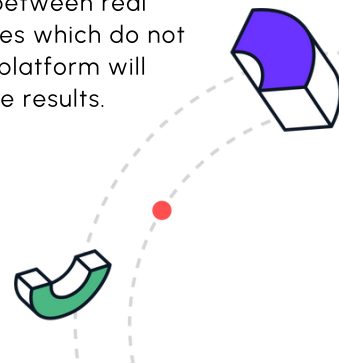
- Hold a briefing session as a group, ideally
- Inform individual business units
- Consider how they will be notified of changes and progress.
- Provide access to central dashboards and metrics.

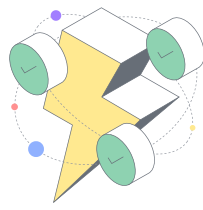
EXECUTE THE PLAN:

Depending on the team's level of skills and knowledge, it may be beneficial to have workshops to get the team up to speed and flush out any concerns. They will need to understand how the process works and want the expectations of them are.

It is essential to set the standards you need for feedback. It may be that you are only interested in reports of issues, and there may be requirements for that information. Using a capture tool is essential to make this practical and to support users. It should capture inputs and screenshots against the target applications with the ability for the user to add remarks, feedback, and issue information. It is more common now for UAT managers to seek feedback beyond just issue reports. Encouraging this does help improve engagement and ownership, reducing the possibilities for conflict with development and configuration teams.

Introducing surveys is also a great way of increasing user engagement providing the opportunity for a broader perspective on acceptance and readiness. It also helps highlight the differences between real broadly accepted concerns versus more individual ones which do not have broad support. Ideally, your UAT management platform will support the concept of user surveys and sharing of the results.





12

21) Train users

- Teach the testing approach you want.
- How to access the central system.
- How to access work allocation.
- How to execute tasks and capture test steps.
- Feedback – mechanism, and standards.

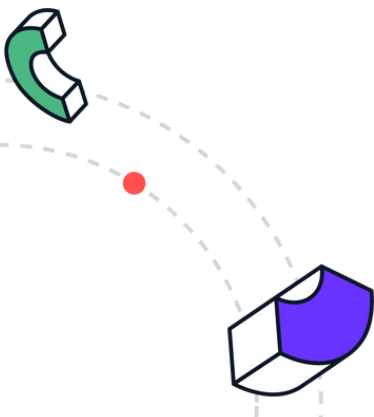
22) Execute the plan and monitor

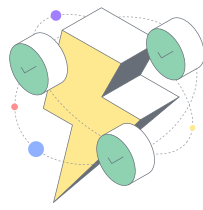
- Ensure the central system is updated.
- Assess progress and rate of progress
- Deal with issues and concerns.
- Maintain team motivation.
- Ensure required results are achieved.
- Encourage good feedback.
- Communicate status with business users and stakeholders.
- Monitor completion rates.
- Monitor forecast completion.
- Monitor level of issues reported.

23) Triage test results

Whereas a full-time QA team may be able to identify and raise defects correctly, it is often preferable with users testing to distinguish between issues and defects, combined with a triage process to determine which issues should be promoted to defects. This can avoid swamping development teams with incorrect or duplicated defect reports but makes it easy for users to report what they find. To be effective, sound evidence and audit trails of the test result are needed, and this really requires comprehensive and easy-to-use capture technology.

- Review feedback and issue reports.
- Examine the collected evidence.
- Update central system with new issue status.
- Convert issues to defects to report to development/configuration teams if needed
- Document and communicate defects to the product team.
- Provide a feedback loop to users of issue status.





13

REVIEW AND MANAGE:

In a perfect world, tests will have been developed to provide 100% coverage of the areas and scenarios needing testing. These will all have been executed correctly, and all passed, even if it has taken a few cycles. At the same time, users will not have significant questions or concerns and will have provided positive feedback about the release's quality and readiness. Everyone agrees that the changes are positive and will achieve the project goals.

Reaching this comfortable position is largely dependent on all the factors you have considered and implemented before this point. You will also need visibility of the project metrics to determine when this status is approaching and eventually arrives. This data needs to have been collected as the project has progressed and cannot be realistically constructed in retrospect, so it needs to be baked into your management platform and methods from the start.

24) Assess overall status and readiness

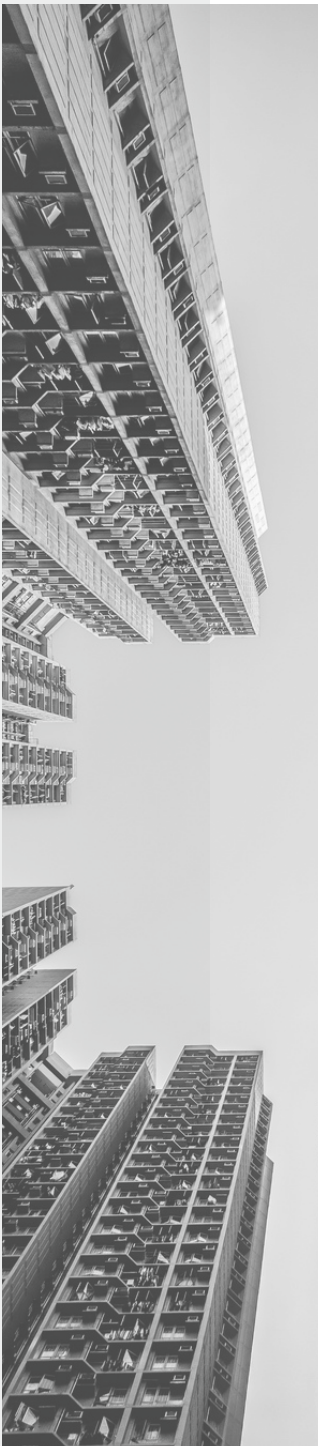
- The test passes vs. failures and outstanding tests.
- Outstanding issues.
- Outstanding defects.
- Progress against your key metrics.
- User surveys and content.
- Communicate status to stakeholders.

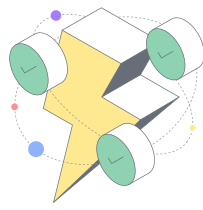
25) Automation build strategy

Automation of tests can play a valuable part in UAT. It provides the most value in areas that have already 'passed' user inspection but will need re-testing or re-execution either as part of a subsequent cycle in this project or as part of a future release cycle where it forms the basis of regression testing. It can also provide value in data loading to support testing activities. Some people consider that UAT should only ever be manual; eyes and hands-on from the users as part of building their confidence in the delivered system.

Taken too far, this might be considered a 'punishment'! After all, if during three rounds of testing, it has been necessary to create 20 new suppliers and 200 new purchase orders that have been working perfectly so far, users will not give this repeated task the same enthusiasm as they did initially. If automated, they can still validate and accept the results without the hard labor, and valuable test assets have been created for future use.

But for automation to be successful, it needs to be relatively easy to develop, maintain, and understand. Thankfully, the days of complex coded automation solutions dependent on specialist automation engineers are over. By delegating some repetitive aspects of the testing scope to automation time is freed up for the users, it might just mean less effort or re-directed to time spent on exploratory testing.





14

- Can passed manual tests be converted to automation?
- Where is the most significant gain?
- Is the process knowledge available, captured?
- What future requirements for regression testing can be addressed now?
- Consider intensive, repetitive manual tests or tasks (such as data loading) for early automation.
- Ensure you have selected an automation solution to match team skills and rate of progress.
- Are there configuration and environment tasks that will need to be repeated when moved to production, which can be captured in automation for consistency?

REPEAT:

If your UAT, Conference Room Pilot (CRP), or Model Office Testing (MOT) project has been completed successfully with only one cycle, you are very lucky, or very good, or both! It is most likely that several cycles will be needed, perhaps with reducing effort and coverage as the testing progresses and confidence in some areas solidifies. But this is also a time to be careful as spotlighting only the areas of latest concern may allow errors to sneak past in previously cleared tests that have not been repeated. This is where automation of these areas can pay dividends, providing confidence without the hard work.

Having got successfully through the first cycle of testing, the remaining ones will be easier, the plan will be shorter, and the team will be up to speed. But you will still need to plan again and leverage what you have done before, and having a good platform already in place will pay dividends.

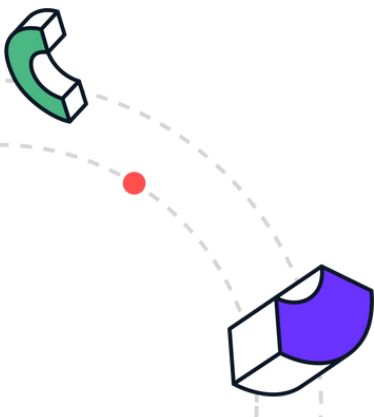
26) Replan for testing

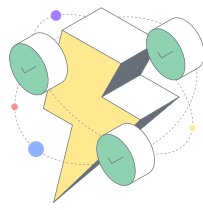
- Approach to re-testing?
- Only failures?
- Regression testing – Whole, partial, manual, none?
- Exploratory testing? The more, the better.
- A data environment matching the test needs.

27) Apply automation to areas not needing manual testing

- Re-baseline scripts to include changes.
- Review results with users.

28) Repeat EXECUTE and REVIEW & MANAGE sections





15

RETROSPECTIVE:

There will always be things that could have gone better, and this is the time to reflect on these and improve. Hopefully, it is also a time to celebrate a successful implementation with the probably large numbers of people involved, and it is good to do so for many reasons. It will also help make future UAT projects more appealing, and to the user community, you will need to call on.

29) Analyze and critique the project

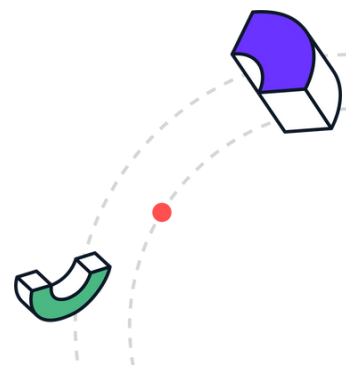
- What was the level of issues found, and was it acceptable?
- Similarly, for the level of defects.
- Where was time lost or gained?
- How effective was the communication?
- How was the quality of the delivered results perceived?
- Consider the level of questions and concerns after/during go-live
- Who were the stars of the show – praise where due.

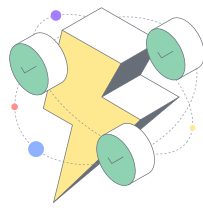
30) Organize assets from the plan and results for future use

- Collate assets and organize them for future use.
 - Test cases.
 - Test Scripts.
 - Automated Scripts.
 - Creation of automated regression?
 - Data.

31) Adapt test policies if necessary

- What strategic changes would have created a better framework?





16

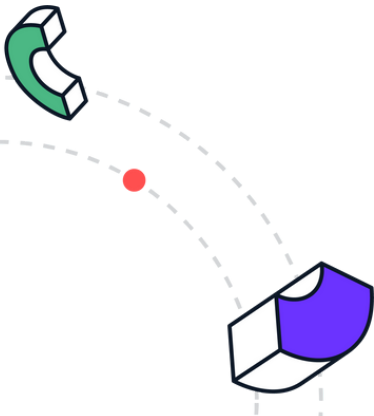
CONCLUSION:

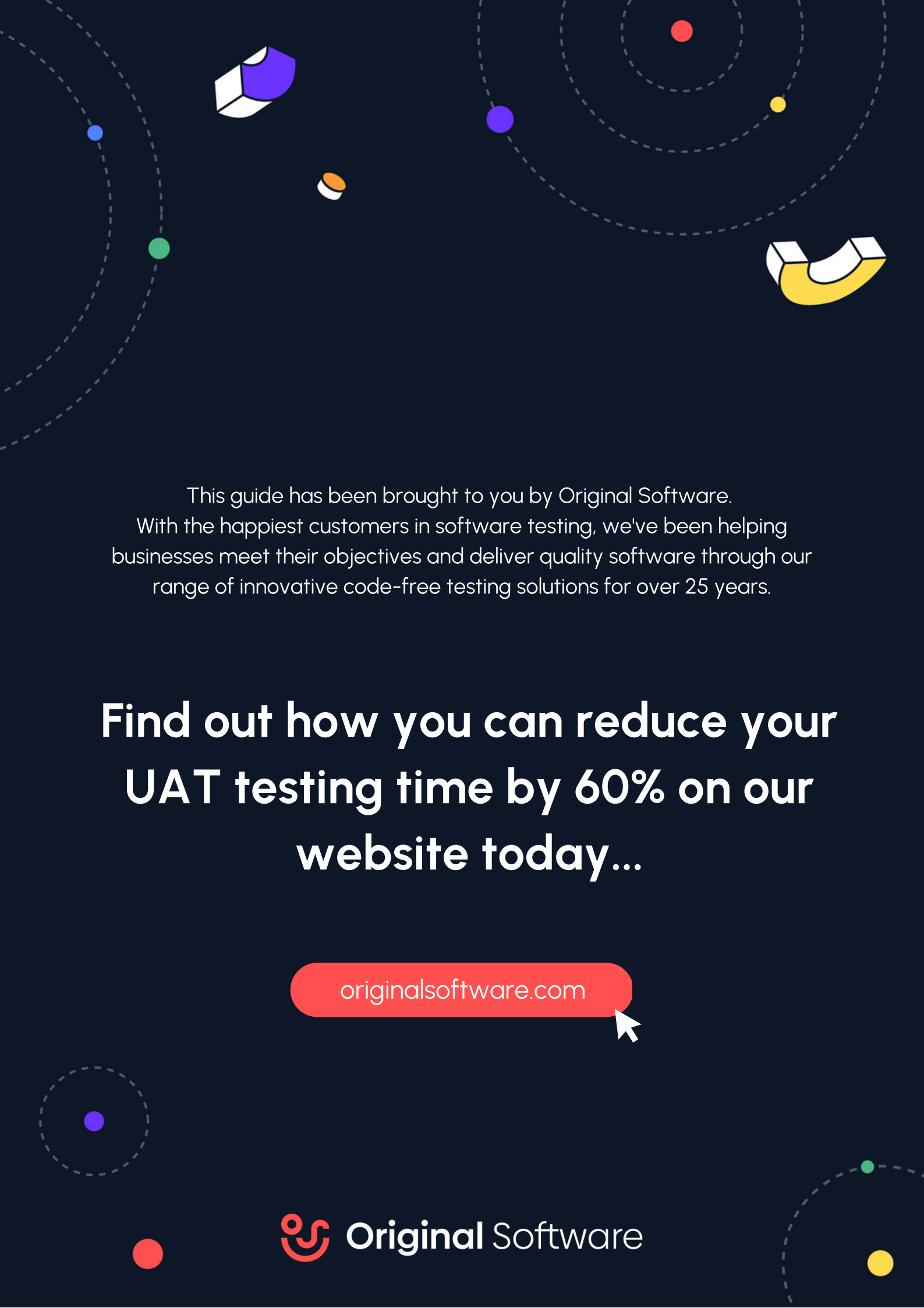
The nature of User Acceptance Testing is bound to vary according to the project's needs, the business, the users, and the overall risk. In some cases, it is the only form of testing an organization will experience, but even if it is not, UAT is the most expensive type of testing that can be performed.

Many Quality Assurance theorists and academics will be unhappy that cost is even mentioned in the context of testing. They will consider it irrelevant compared to the project's goals and will argue this by making the comparison to the cost of not testing—a fair point. However, the cost is a real factor and a real constraint for many projects, and given that UAT is the most costly type of testing that can be performed, it is worth considering how the cost can be controlled and reduced. Thankfully, a cost reduction is a by-product of other more inspiring objectives and is achieved by efficiency, re-use, improving communication, and better testing approaches that drive out more issues in less time, with less effort.

In the end, delivering an issue-free system, which the users know how to use and gain benefit from, will be the most valuable factor in the project's success.

This guide has been created to help identify many of the topics and considerations which should be examined when looking to achieve the above goals. The use of technology is a recurring theme because it plays a valuable, indeed essential, part in making the improvements (compared to a manual process) to enable these goals to be achieved. In some organizations which are too IT and development-focused, testing is deemed to be only worthy of attention within the IT team, and once in the hands of the business users, it can be forgotten. Perhaps because by then, it is someone else's budget, but the reality is that it impacts the business as a whole by increasing cost and reducing project success if not done well. The business case for investment in improving this process is usually easy when considering the amount of time that can be saved and reducing issues that hit production systems. No organization wants to be the next 'Software testing failure' headline, and UAT is a crucial component to maximize the quality of the delivered solution and stay out of the news for the wrong reasons.





This guide has been brought to you by Original Software.
With the happiest customers in software testing, we've been helping
businesses meet their objectives and deliver quality software through our
range of innovative code-free testing solutions for over 25 years.

**Find out how you can reduce your
UAT testing time by 60% on our
website today...**

originalsoftware.com

 **Original Software**