The Challenges of UAT

© Geoff Horne
Managing Director
iSQA
October 2002

What Is UAT and Why Is It Important?

User Acceptance Testing (UAT) sits on the end of the tail of the software development and implementation lifecycle. It is the very last process affected by changes to specification, functionality, and delivery and it's focus and inherent nature requires different strategies to other phases of testing.

In the view of many experts, UAT is the single most important stage in the software development process, the last chance to ensure software is fit for purpose before releasing it for widespread use. It is usually a time of complexity compounded by stress and pressure brought on by deadlines.

UAT is one of the most difficult stages to manage and, often, the least appreciated by senior management. In our view, it is also, the least well-documented and structured of any the software testing processes.

Every software application, from the simplest report to the most complex multi-million dollar, real-time mission critical system, requires UAT. Failure to perform effective UAT will almost certainly result in end users experiencing problems with the software and the cost of fixing issues at this stage can be many times higher than if found and fixed before release. And sometimes the "cost" of sending Granny a zillion dollar bill cannot be measured in tangible terms - especially if the media is alerted!

In UAT, the software is tested for compliance with statutory regulations or organisational business processes and rules, and to observe how it will behave under operational conditions. The emphasis switches from finding defects to ensuring that the software is suitable for the task it is being implemented for. It is often also the first time non-development staff will work with a new system and the first time development staff will interact with non-technical staff.

Against a background of management pressure to achieve delivery dates or fulfil promises to a Board or customers, the scope for poor communication, misunderstandings, and corner cutting is substantial. Therefore, we believe professional UAT management is key to the successful completion of a software implementation.

It is also a common assumption made by many organisations that if an off-the-shelf software package is implemented, that no testing is required as it is expected, and quite rightly so, that the developer or vendor has adequately tested and generally quality-assured their product.

Whilst this sometimes proves not to be the case, any initiatives on this count carried out by the developer can really only be measured against a functional specification of the product and not against the specific business requirements of each individual customer. After all, the main body of knowledge and practical expertise in these processes lies with the intended users of the software.

There have been many well-documented and publicised software implementation disasters. If one is to examine the reasons behind some of these, one of the primary causes, along with poor project management, is the failure to verify the application against the business requirements.

There are few certainties in life apart from death and taxes, but one of them is that the need for effective UAT is vital and will grow in importance and scale over the next decades.

What Should Be Covered During UAT?

UAT determines the degree to which delivered software meets agreed user requirement specifications. It confirms whether the software supports the existing business processes and any new initiatives expected as part of the delivery. User requirements may evolve during implementation as the early benefits and short-comings of a new system begin to be realised. Sometimes this will require software enhancements to be made and therefore test cases and plans should cover regression testing on future software releases.

Even if software passes the vendor's functional testing, it must still be tested to see how it will perform in the business environment before being released for general use. Failure to conduct effective UAT will almost certainly result in problems being found after release. If this occurs, developers and testers are open to criticism about their professionalism and public embarrassment can be caused to the organisation and it's senior management. In addition, the costs of fixing problems at this stage can be many hundred times higher than if they we uncovered earlier in the lifecycle.

During UAT, the way the software is intended to perform and behave upon release for general use is assessed. This includes the:

- Accuracy and utility of user documentation and procedures.
- Quality and accuracy of data being produced.
- Layout and content of reports and audit trails.
- Release and installation procedures.
- Configuration management issues.

A prerequisite, therefore, is that the application being handed over for UAT is complete according to the agreed specification, and that it has been thoroughly and effectively system tested by the vendor. This is often not the case, and a great deal of UAT resource and time can be spent finding functional errors that should have been discovered and fixed during system testing.

UAT addresses the broadest scope of requirements and therefore must cover the following areas:

- **Functional Requirements -** ensuring all business functions are performed as defined and satisfy user's needs.

- **Operational Requirements -** ensuring requirements for data capture, processing, distribution and archiving are met.

- **Performance Requirements -** ensuring performance criteria such as response times, transaction throughput and simultaneous processing are achieved.

- **Interface Requirements -** ensuring all business systems linked to the software system in UAT pass and receive data or control as defined in the requirements specification.

In addition, the following areas are often included within a UAT exercise:

- **Security Requirements -** ensuring that the application is properly configured to prevent unauthorised or inappropriate access.

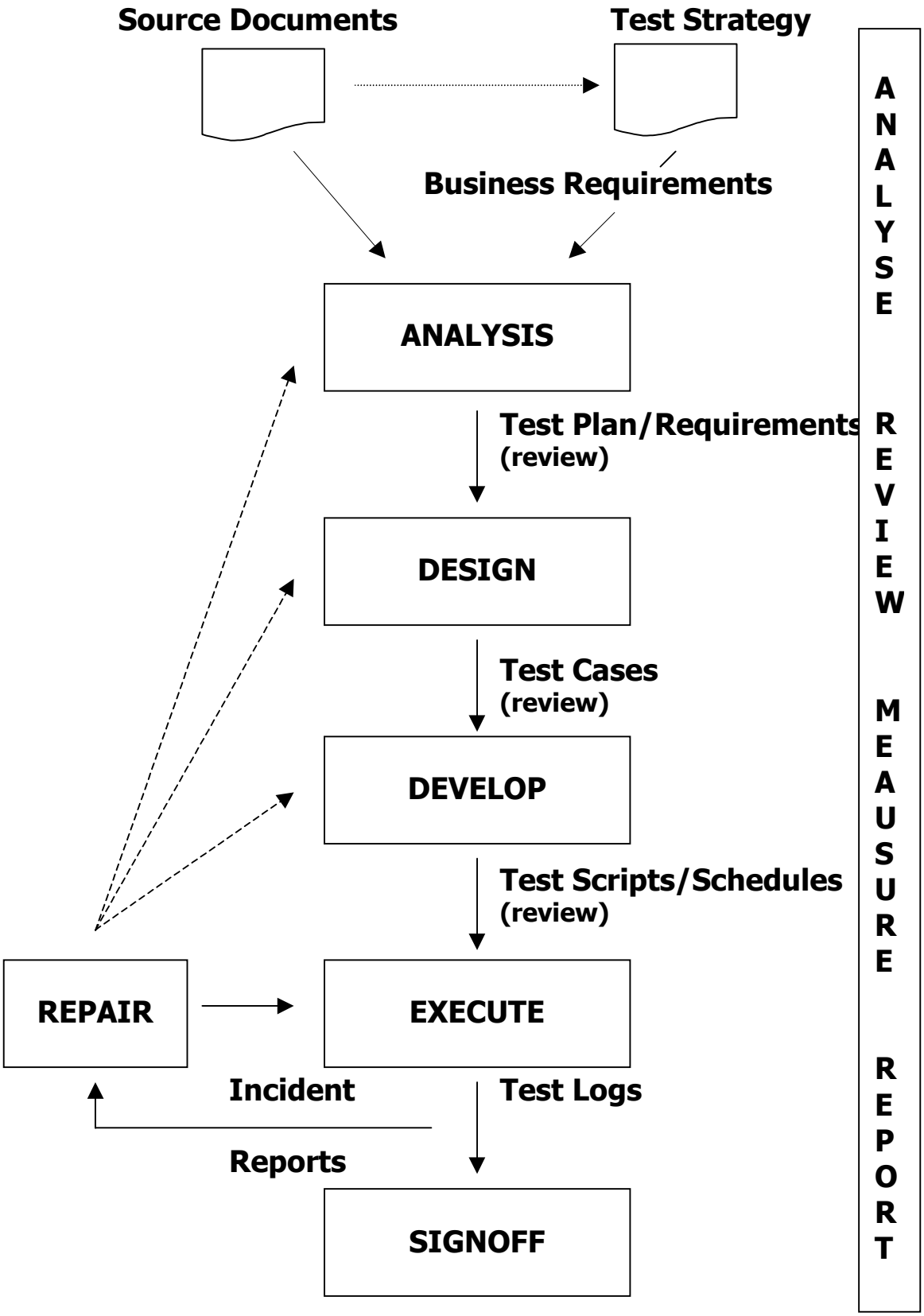
- **Disaster Recovery Requirements -** ensuring that in the event of the application being abnormally terminated, that it can be reset and re-established without losing or corrupting data.

- **Data Migration Requirements -** ensuring that any legacy data has been correctly and accurately converted and migrated to the new system including the ability of the application to carry out day-to-day functions on that data.

What Are the Steps Within UAT?

iSQA offers testing consultancy and management services however we are firmly of the belief that execution UAT at least, as its title suggests, is the responsibility of the users - those people within the business who will be using the software for their day-to-day tasks or their representatives eg. expert users, secondees, or temporary staff identified as part of the UAT team. However users within a business are not always able to conduct the full UAT lifecycle themselves, which is to be expected as they are primarily business and operational people and not professional IT or QA practitioners.

The UAT lifecycle can be broken down as follows:



Before starting UAT, those responsible for executing the tests need to be trained in how to test. This is rarely recognised as being important, but in our experience it is one area in which a small investment in time and effort can pay great dividends in terms of reducing wasted work. Good software testing requires a disciplined approach by testers to find out what doesn't work.

Developing a Test Strategy

The intent of an overall Test Strategy is specify how the UAT will be carried out by delineating high-level testing procedures and outlining general test coverage.

Requirements Analysis/Test Planning

The aim of a UAT Plan is to identify the essential elements of the software to be tested, with any wider coverage being considered desirable or a bonus. With larger or extremely complex software, total test coverage may not be possible or practicable. Whilst it may be possible to calculate all the possible combinations and permutations of test conditions, many of these will not actually occur in practice or not be significant enough to warrant testing time and effort.

Therefore, it is essential to identify vital and significant business scenarios as the testing priority and, if time and resource permits, to extend that testing subset to less vital or significant areas. Ideally, UAT planning should begin during the requirements definition stage of the project. At this stage, a set of use cases is defined as part of determining what functionality and features the new software should provide.

Successful UAT planning is thorough and avoids unsatisfactory resolutions such as short cutting UAT or drafting too many people into the UAT Team at too late a stage.

Test Design/Test Planning

Suitable test design enables the requirements to be put into a form whereby the application is broken down into testable chunks. Requirements are usually at a high-level and the design process seeks to further refine these into tangible processes that can be specified as test cases.

While a Test plan can be delivered prior to identifying test cases, a test schedule can only be finalised once all test cases are known.

Test Development

Once each test case has been quantified, it can be broken down into three parts

- Pre-requisites - what data and other set up is required to run the test
- Test Steps - what actions within the application should the user take to run the test
- Verification - how will each test be verified that it has actually worked

The above three areas can be contained within a test script which specifies to a level of detail defined in the Test Strategy, the steps the tester is required to take to execute each test case.

Once all the tests have been developed they can be organised into schedules ready for execution.

Test Execution

It isn't sufficient just to find an error, the tester must also record the conditions prior to starting a test, the actions taken, and what results occurred. The tester must produce physical evidence, for example screen prints, and be able to repeat the problem. Otherwise, the developer charged with fixing the problem will not understand it, not be able to repeat it, and reject it as a problem. The test will then have to be repeated and recorded properly, a waste of everyone's time and effort.

To ensure adequate control on the clearance of errors and to improve management forecasting of UAT completion, each incident must be recorded separately by the tester using the official incident

logging and tracking system. Depending on the size and complexity of the software under test, the number of incidents reported can range from a few dozen to several thousand.

Managing Incidents

As well as logging problems found, testers will come across incidents of clarification and enhancement. Ambiguities in specifications are quite common and may not be discovered until UAT. These are clarified and resolved between the teams involved and agreed to be either a software problem to be cleared now or an enhancement to be provided in some future release.

An effective incident tracking system such as TestTrackPro allows incidents to be described, prioritised and tracked in fine detail, to attach screen prints, and to classify and categorise incidents in the manner most useful for a particular project or stage of a project. TestTrackPro provides excellent query and report building facilities as well as status and situation reports, and statistics down to individual tester level.

These features enable the Test Manager to identify the software functions and staff causing problems, to understand where the bottlenecks are, and to provide better forecasts of Estimated Time to Go (ETG) to UAT completion.

The development team clears errors, and supplies a new release incorporating bug fixes and enhancements of the software to the UAT Team. This process is repeated until all reported incidents are resolved to the commissioning organisation's satisfaction.

Management of the Test Process

As mentioned earlier, the scope for poor communication, misunderstandings, and corner cutting during UAT is substantial. Therefore, we believe professional UAT management is key to the successful completion of software development and implementation and we suggest review of the next Section to understand the potential challenges in more detail.

Common Difficulties With UAT

Software design and production is an extremely difficult process. Systems analysts and programmers are clever, dedicated people. They take a pride in their work but they are nearly always hard pressed eg.

Software developers have to:

- Interpret user requirements
- Design a software system to meet those requirements
- Develop the software code according to those requirements
- Achieve performance targets
- Work in a tightly-controlled development environment
- Conform to quality management rules
- Satisfy deadlines

Developers require formidable technical skills and mental ability. However, they focus on developing a technically sound and good quality software system. These technical people do not have the business knowledge and organisational understanding to be able to prove the software system operates effectively in the wider business environment in which it will be used.

Users, on the other hand, have this business knowledge, but are less computer literate and technical than developers. Although a small permanent UAT Team may exist, expert users drawn from various parts of the organisation supplement permanent staff when a new software system is developed. These people may have never tested software before or may be unskilled in the new software systems. Project Managers drafted in to be Test Managers, might be very good at their normal job in the organisation, but the requirements of User Acceptance Testing management are very different from anything they have experienced before.

The Great Divide

The types of testing done by developers and the UAT team are different. Unit and system testing aim to prove that module code is correct and that the integrated software modules work to specification and hypothetical data is often used to create test cases. UAT, however, aims not just to prove that the software functions correctly, but that it is fit to go into general operational use. Privileged access to live data and real life cases mean that it is more likely that they will to be used for UAT.

Also, systems analysts, in particular, often find it difficult to understand the tester's problems in using the software. The systems analyst is very familiar with the new software, having designed it after all, whereas using the software is a brand new and sometimes daunting experience for the tester. The relationship between the systems analyst and the tester can be empathetic and helpful but more likely than not it can range from technical condescension to dismissiveness on the part of the systems analyst. Often communication is more difficult if it is conducted remotely via written reports, emails, or faxes rather than face-to-face or on the telephone.

In addition to revealing problems with the basic functionality of the software, UAT frequently reveals omissions and errors in the original requirements specification. There are a variety of reasons:

- The user may be unclear about what the software system should do, and developers don't know enough about the user's business processes to point out any deficiencies.

- The developers understand what the user wants and knows the requirements specification is incomplete or incorrect, but doesn't query the discrepancy. Very occasionally unscrupulous developers use this situation to minimise their workload and cut costs when working to a tight schedule. Failure to meet requirements may also produce further

revenue-earning enhancements for the developers once the software system is operational.

- Developers simply misunderstand the user's requirements.

These problems are minimised if users define their requirements:

- As use cases and results to be produced by the software system.
- By reference to a prototype of the software system produced by developers and maintained in parallel with changes to the requirements.

Patiently, and in collaboration, with developers allowing plenty of time to prepare, review, and revise the requirements.

Inexperienced Testers

A common failure is to believe that anyone can do testing and often temporary staff with no experience of the industry are recruited. A detailed training programme is required, and the UAT Process must be very explicit if testing is to be effective. If the UAT Process is not detailed enough, testers will make uneducated guesses, get them wrong, and testing is ineffective. Tests will be repeated over and over again with loss of time and increased UAT costs.

Testers need close supervision and control. Otherwise, their work rate declines as tests are carried out incorrectly or duplicated. Bottlenecks in assigning work to testers, retrieving test cases from the system, or printing test cases for checking, lead to work shortages and hence to idle testers.

Inconsistent Reviewers

Expert users of the software system can be seconded from other parts of the organisation to check the work of the testers. They may be supervisors, team leaders, or sales managers with staff management responsibilities. These reviewers are very experienced in the results being generated by the existing business system, and are, therefore, ideally placed to check the output from the new software system.

However, they may differ in their interpretation of the results of UAT, and a consensus between reviewers about what constitutes a successful test result must be achieved. If this does not occur, tests may be passed by one reviewer, which another reviewer would fail. This, in turn, creates doubt about the correctness of completed tests, undermining management confidence at best, or at worst resulting in potentially catastrophic failures of the live system.

There is, also, a real danger of conflicting priorities arising between their normal departmental work and UAT, with UAT coming off second-best. This can make planning and resource management problematical.

Unmotivated UAT Team

The longer the time seconded staff spend in UAT, the more difficult it is for them to remain motivated. This is particularly true if little or no progress is being made, and the same tests are being run again and again with no successful conclusion.

Expert users are in the invidious position of comparing the old business system that probably worked very well with the new one that doesn’t. They can, therefore, become very negative and antipathetic about the software system in UAT. Moreover, the longer they stay away from their normal duties, responsibilities, and staff, the more likely they are to become de-motivated.

A well-motivated UAT Team will produce better results than a demotivated one. Motivated UAT Team members are more positive and proactive and look for ways of improving testing.

Compliance Issues

In some industries, for example Financial Services, compliance with government regulations is mandatory. Specialist compliance officers are employed to ensure the best advice is given to customers. These compliance officers sometimes do not work concurrently with the UAT Team. Compliance officers may, therefore, reject a complete UAT programme after the event. If at all possible, compliance officers should be involved in test planning, and be party to authorising the UAT Plan before testing commences.

Poor Release Control

For effective testing, all testers and reviewers should use the same release of software. Although this may sound obvious, often testers test using different software versions. In addition, testers can often test in the wrong environment and there needs to be processes in place to ensure that the "live" environment is not logged into for testing!

Unsuitable Test Hardware

Testing on a hardware configuration different to that intended for live operation can give false functional and performance results.

Poor Planning

Planning is crucial to UAT, but the time and resources required are frequently under-estimated. The complex logic of most software systems makes creating effective tests difficult, time-consuming, and imperfect. Theoretical data is used when live data is available, and even when live data is used, sampling techniques are inadequate and do not represent a good cross-section of business cases.

A good UAT Plan is the foundation for UAT, subsequent regression testing, and future test automation.

Incomplete Test Scripts

Test scripts are not produced in enough detail to make it easy for testers to enter test case data. Incomplete test scripts are worse than no scripts as they are open to misinterpretation by testers. This leads to a waste of time and effort as the test results are unusable, the test script usually has to be expanded, and the test rerun. Inadequate numbering, identification, descriptions, and poor management of test scripts, test suites, and test libraries leads to duplication and confusion.

Test Records Lacking

When testers retain inadequate proof, there is no evidence of testing being done. This is not acceptable in an ISO9001 environment or to internal or external auditors. Instead of raising an error report for every error found running a test case, error reports are lumped together making test management difficult. Without comprehensive test records, good judgements about the progress or quality of testing are impossible. Ultimately, management confidence in the quality of UAT determines whether the software system goes live.

Test Management Shortcomings

A casual approach to test management leads to corner cutting, inadequate testing, or no testing being done. Therefore, the Test Manager must be in complete control. However, this is often not the case. Typical management shortcomings are:

- **Out-of-date technical specifications -** if UAT is based on out-of-date technical specifications the integrity and effectiveness of testing is jeopardised. Technical specifications must be up-to-date and authorised before UAT planning starts.

- **Critical business functions not understood -** UAT must ensure that business functions work properly. If UAT fails to prove critical business functions, the software system goes into live with public embarrassment and increased costs.

- **No clear UAT objectives established -** UAT objectives should be clear and unequivocal. If they are not, testers work with no co-ordination or cohesion. UAT becomes subjective rather than objective and test coverage is less than complete.

- **Alternative UAT strategies not identified -** if UAT objectives are not established, often no UAT strategies are devised either. Some thought must go into determining alternative approaches to UAT, so direction and resources can be switched quickly because original plans do not work out.

- **UAT Process inadequately defined -** to be effective and to achieve UAT objectives, testers need to understand the UAT Process. A clear statement of who does what, where, when, how and, preferably, why must be prepared, documented, communicated to, and understood by the UAT Team.

- **Process measurement not in place -** the key to taking effective action is to measure what is happening. Only by measuring can bottlenecks in the UAT Process be isolated and corrective action taken. Only by measuring can managers get a clear view of progress towards completion.

- **Poor process controls -** measurement provides statistics and pegs in the ground, but without interpretation and translation into decisive management action they are of limited use. If part of the UAT Process is found to be deficient, changes should be made. In practice, theoretical plans are adapted to reflect the real world.

- **Hands-off management style -** there is no substitute for hands-on, rigorous, and reactive test management. Left to themselves, testers do their best but may make inappropriate decisions. They become demotivated if managers do not react effectively to suggestions made by the testers. Motivation, enthusiasm, and direction are the key to successful testing and these stem from strong hands-on test management.

- **Conflict of interest -** it is too easy for a customer to opt out of the responsibility for UAT by permitting the application vendor or system integrator/implementor to manage and/or perform UAT. UAT is the customer's last opportunity to verify the application before live operation and if UAT is forced onto the vendor's/integrator's agenda or timetable, then the Test Manager's ability to remain objective and quality-focused is compromised.

Increasingly, organisations hire consultants to develop a UAT Strategy. Often they produce very detailed and professional-looking documents that are fine in theory but of limited use in practice. A good methodology is rooted in actual experience with many different organisations. It is simple common sense distilled and structured to make UAT of direct benefit to any company.

As mentioned elsewhere in this document, iSQA is experienced in the consultancy and management of UAT projects with the added benefit of having performed many such projects for many different companies. We understand the challenges of this discipline and have successfully verified many applications against business requirements.