WHITE PAPER

User Acceptance Testing Best Practices for Scientific Informatics Solutions

Avoiding the catastrophe: "Software developed by design but not as desired by the user"

Kalleid

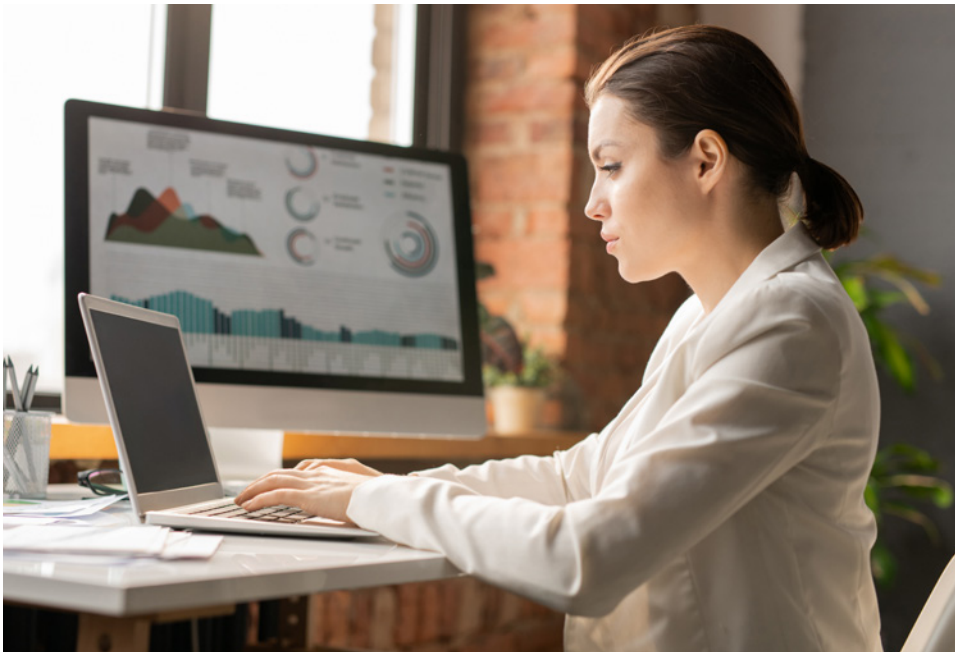
CONTENTS:

Kalleid

WHITE PAPER

User Acceptance Testing Best Practices for Scientific Informatics Solutions

Avoiding the catastrophe: "Software developed by design but not as desired by the user"



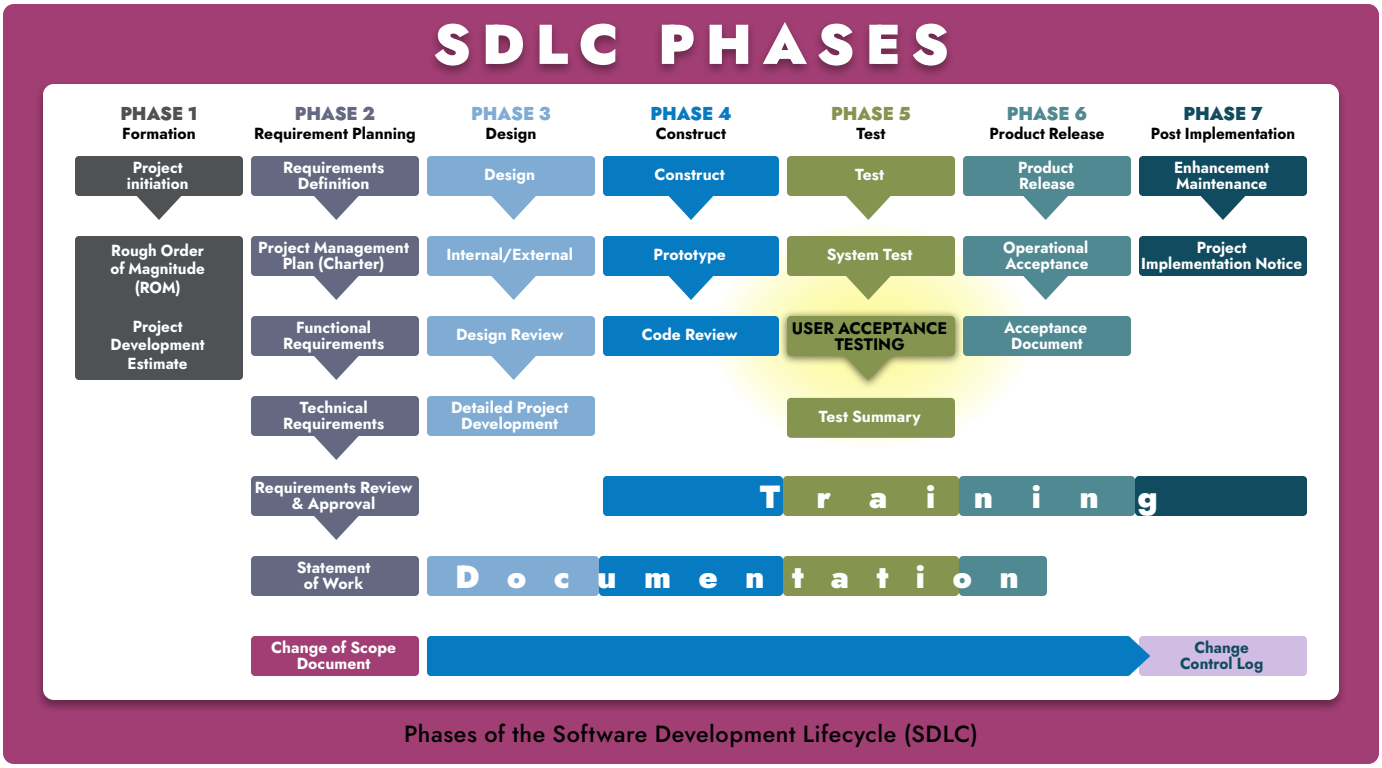
IT projects in scientific organizations typically require large investments of money, resources, and time. A laboratory information management system (LIMS) implementation, for example, can require skilled personnel hundreds of hours to complete and cost a company hundreds of thousands or even millions of dollars. As such, it is critically important for any organization to get a software implementation right the first time.

Unfortunately, IT projects are notorious for failure. According to a recent survey of over 3,000 project managers in a diverse sampling of industries, 14% of IT projects completed over the previous 12 months were deemed total failures, 31% did not meet the original goals and business intent of the project, 43 percent exceeded their initial budgets, and 49 percent were late. IT projects can fail due to a wide range of factors, a few of which include:

- Insufficient or inexperienced project personnel
- Selecting and implementing the wrong system for your organization
- Mediocre project management
- Lack of management buy in
- A focus on cost at the expense of business value
- Lack of effective business analysis, resulting in inaccurate requirements
- Misalignment of project goals and business goals
- Inaccurate cost estimates
- Shifting organizational priorities
- Lack of user engagement and/or input

"When users are not properly engaged during the development and testing of a new system, they tend to revert to pre-deployment systems and methodologies post go-live."

This last factor, user engagement, is an important driver of project success. When users are not properly engaged during the development and testing of a new system, they tend to revert to pre-deployment systems and methodologies post go-live. Or worse, they use the tools incorrectly or not as intended, resulting in workflow confusion and inconsistency in data capture.



Phases of the Software Development Lifecycle (SDLC)

There are many different types of testing that are performed during the software development lifecycle (SDLC) to verify that software is functioning as required prior to system release. Of these, user acceptance testing (UAT) is critical to the overall success of the IT project, as it verifies that the software performs in a manner that is acceptable to the user. UAT gives users the final say in software approval, assuring that the software was developed as desired and aligned to the workflow needs. This is an important precursor to ensuring user adoption and compliance with the solution.

Organizations which make the mistake of skipping UAT testing, or conducting it hastily, often experience financial or operational loses, along with post release rework. In this white paper, we will discuss UAT best practices and provide an example of an effective UAT script.

"Functional tests are important, but you'll never know how your system will behave in the real world unless it is tested by real users. UAT is typically the last type of testing done before the product or application is released to the production environment"

What is User Acceptance Testing?

A wide variety of testing typically occurs during the SDLC to verify that a software system is functioning properly prior to release – functional testing, performance testing, integration testing, security testing, regression testing, etc. Most of this testing falls under the banner of "system testing" and is done by the software development team to find and fix bugs and also to ensure that the software meets the functional requirements that were developed in collaboration with the business. But what if developers misinterpret the requirements, or the requirements are poorly defined and do not properly capture the customer's needs?

Functional tests are important, but you'll never know how your system will behave in the real world unless it is tested by real users. User acceptance testing is typically the last type of testing done before the product or application is released to the production environment. In **UAT,** software is tested by the real users at their premises.

Unlike other forms of testing, UAT is not focused on finding and fixing software bugs (although any bugs found will certainly be communicated back to the developers). The main purpose of UAT is for business users to perform a final check to confirm that the developers have properly interpreted and implemented the functional requirements prior to system rollout.

The user-centric approach of UAT makes it one of the most important phases of testing or validating computer software. UAT serves to ensure the system does what is intended from a user perspective, which is critical to ensure users are happy with and will ultimately adopt the system. In addition, UAT activities often help to clarify requirements and may also unearth additional requirements that the business sponsor may not have thought of prior to seeing the system. UAT can also help the business sponsor discover problems in aspects of the application that were not tested by the development team.

The timing of UAT can vary slightly depending on whether a waterfall or agile development methodology is used for the project. In the waterfall (sequential) approach, UAT is the final stage of testing done by users to confirm that the system meets requirements prior to go-live. In an agile (iterative) approach, system design and testing take place in short sprints with system functionality becoming available incrementally at the end of each sprint. In this case, light UATs are conducted before rolling out functionality at the end of each sprint.

User Acceptance Testing Best Practices

Given the importance of UAT for clarifying requirements and ensuring user adoption, it is essential for companies to plan and execute it effectively. A best practice methodology should be followed to ensure successful UAT. Key aspects of this methodology include:

Conduct UAT design alongside business analysis phase – Effective requirements development through proper business analysis (BA) is the first step and foundation of every successful IT project. Good BA engages users along with other business stakeholders in shaping future state workflows and developing requirements (both functional and non-functional). This helps users buy into the future state being implemented because they are consulted throughout the process, thus ensuring adoption.

UAT development should begin as soon as requirements are defined, with the scripts being written in concert with users to make sure they accurately reflect business processes. It's a good idea to write your UAT scripts as soon as possible in the SDLC so you don't end up having to build a massive test database right before UAT is conducted.

Create a good UAT team – The UAT team should ideally be composed of users in every group or area of an organization so that every user role can be tested. An assessment should thus be performed to identify all system users and their different roles prior to creating the UAT team. That said, testing every role in a large company with a large application is usually not feasible. In this case, it is important to perform a risk-based assessment to identify and prioritize those roles involved in critical business objectives for testing. Key roles to consider:

- **UAT testers** – testers should be familiar with the new requirements that the system is intended to fulfill and have the time set aside to conduct a thorough testing process.
- **UAT owner** (i.e., project manager) – the UAT owner is responsible for updating the business sponsor on the status of the tests, collaborating with them on decisions, and managing the work for the testers. The UAT owner should ideally have good knowledge of the software system being tested.
- **Business owner** – the business owner is responsible for guiding the UAT owner in testing for the requirements. This person also owns the responsibility for managing any change control items that come out of UAT.

Create a UAT plan – A testing plan should be developed that includes all the relevant information necessary for conducting UAT – dates, testers, team roles and responsibilities, testing environment, timeframe for building the testing environment, scope of testing, assumptions and constraints, testing strategy (e.g., in person or self-paced testing), communication protocols, entry criteria, acceptance criteria, templates, etc. This plan should be created during the requirements solicitation and refinement process.

Create a proper testing environment – UAT should be conducted in a physical space that allows software QA team members, the project manager, and users to work through the test cases together. Additionally, UAT should be conducted in a virtual environment that matches the exact configuration of the production environment (e.g., a virtual laboratory environment) so that users can test real-world cases. Under no circumstances should UAT be done in the development environment.

Create good documentation – Proper documentation is essential for UAT. In addition to the UAT plan, documentation for UAT should include:

- **Test outcomes** – testers should be provided with a template used to document all outcomes, comments, issues and statuses during UAT.
- **Requirements for sign-off** – the business owner should define and document a set of requirements for moving from UAT to production environment. Test cases should be designated as high, medium and low based on their impact to the business. Issues found in anything other than a low impact test case should be corrected prior to going live with the system.
- **UAT completion report** – this report should contain a traceability matrix detailing test results along with next steps and recommendations.
- **Change control process** – a standard process should be documented and followed for any issues that need to be corrected before go-live.

Writing Effective UAT Scripts

Effective test scripts are the cornerstone of successful UAT. Before test scripts can be written, however, detailed user stories and acceptance criteria need to be developed, as these provide the foundational information necessary to write the scripts.

Develop user personas. User personas define the user for whom the software is being designed. You should understand some basic descriptors about your audience, including motivations, desires, current pain points and future state wishes. The persona helps paint a picture of the user at the center of the workflow. The personas participate in one or more user stories.

Develop user stories that tie back to business needs. To keep users at the center of the conversation, a set of real-world use cases (user stories) covering all essential user roles should be developed during the requirements development process. A central feature of Agile software development frameworks, user stories are deliberately simplified versions of requirements that clearly articulate how a specific software feature delivers value to the end user. User stories are usually expressed in the following format:

As a < type of user >, I want < some outcome> so that < some reason >

"To keep users at the center of the conversation, a set of real-world use cases covering all essential user roles should be developed during the requirements development process."

Here's an example of a user story: As an analytical chemist, I want to be able to capture purification data and spectra in my electronic laboratory notebook so I can record the characterization data for the samples and share the information with colleagues.

Define acceptance criteria (AC). As a user story does not contain enough information to develop effective UAT scripts, a set of concise acceptance criteria should be developed for each user story. Acceptance criteria provide the details of the user story that will form the basis of writing the UAT scripts. User stories define requirements for a software feature or functionality, whereas AC define the conditions that have to be met by the system for the user story or the requirement to be satisfied.

Each user story may have up to 3 or 4 acceptance criteria associated with it which help define the scope of the user story. In order to make sure all scenarios for a given user story are captured (including error scenarios), AC are best developed collaboratively along with the user stories in a Requirements Workshop attended by business analysts, business stakeholders, UAT owner, and members of the software QA team.

The agile approach to software development typically uses a scenario-oriented approach for describing acceptance criteria in a *Given/When/Then* format:

- **GIVEN** (the context/initial conditions)
- **WHEN** (an action performed)
- **THEN** (the expected result should occur)

Acceptance criteria will be used during UAT to determine if a particular feature meets user requirements and is performing at an acceptable level for business users. AC must be written in a way that allows users to grade them on a pass/fail basis, with the collective grades of all the AC in UAT informing the business decision on whether to accept and deploy the system.

It is important to note that much care needs to be taken when developing AC, because the AC determine whether the software will be accepted or sent back to the developers for rework. This is the part of the UAT process where trade-offs are considered and compromise happens. A few important questions to ask when defining AC include:

- How many defects and bugs in the system are we willing to tolerate for release to the public?
- What will happen if the system is delayed?
- What are the damages if the system is delayed?

Collaborate with users to write effective UAT scripts. After the user stories and associated acceptance criteria have been defined, it is time to write the test scripts. Authors of the testing scripts sometimes rely on functional specifications to build the content, but this is shortsighted. As described above, UAT scripts should be written in concert with the users in the early stages of the project. The users should ensure that the scripts provide adequate coverage of the requirements and that the steps are executed in a way that reflects the business process.

Too often, UAT scripts are not reviewed by the business users prior to execution, resulting in errors or issues that could have been prevented through proper training or refactoring the scripts. If users are not engaged in reviewing the UAT script at all, they may execute them, only to find that the way the system was configured or designed does not suit the business process. By the time you reach the stage of UAT execution, it may be too late to address changes, or worse, your testing results can derail the deployment altogether.

A few additional tips for writing effective UAT scripts include:

- **Make test cases easy to do** – Put yourself in the shoes of the tester when writing test scripts and make sure the scripts are concise and clear.
- **Remove bias from your statements** – Using biased language can affect the results of the test. Consider the difference between "now see if you can find ..." and "now find ..." The former phrasing implies the task will be difficult, and this may convince the user that it is.
- **Dry run or practice the scripts with real users prior to formal execution.** This is a critical step for ensuring your scripts are realistic and provide adequate coverage of the business process.

Key Elements of the UAT Script

UAT scripts should be written in sufficient detail for testers to have all the information they need readily accessible to execute the test case step by step. The scripts should be written in a template that includes the following:

- **Test Case Identifier** – The test case number
- **Objective** – What functionality will this test case verify?
- **Prerequisites** – Are there any prerequisites that need to be satisfied before testing can commence?
- **Special Instructions** – Are there any special instructions necessary to execute this test case (e.g., login as this user Role)?
- **Acceptance criteria** – List each acceptance criteria associated with this test case
- **Summary of results** – Was the acceptance criteria met (yes or no)?
- **Test date** – Date the test was executed
- **Tested by** – List of user(s) executing the test
- **Signature** – Signatures of users who executed the test case
- **Scripts** – For each step in the test case, provide the following:
  - Step ID
  - Instructions
  - Expected results
  - Actual results
  - Pass/fail
  - User comments

"UAT scripts should be written in concert with the users in the early stages of the project. The users should ensure that the scripts provide adequate coverage of the requirements and that the steps are executed in a way that reflects the busiess process."

After the UAT is done, the UAT Owner and Business Owner should collaborate to compile the data collected, evaluate the test results, and write the UAT Completion Report detailing recommendations and next steps.

Conclusion

In this white paper, we provided a comprehensive methodology that serves to ensure success in UAT activities. User acceptance testing should not be taken lightly, as it is an important safeguard to ensure developers have properly interpreted and implemented the functional requirements. Like many other aspects of software development, UAT is fundamentally about making sure the requirements are right the first time. By confirming that the software system/application works for the user, UAT helps promote user adoption and ultimately the overall business value of the software.

In the rush to get a software product out the door and into the hands of users, many companies make the mistake of rushing or skimping on UAT activities. Given the business and operational risks involved in releasing a new or updated application into your organizational workflows, any time and resources spent on an effective UAT process prior to system go-live is a wise investment. While software developers prefer to identify and fix problems in earlier phases of testing, finding issues in UAT is still much better (i.e., cheaper) than after the software has gone live.

About Kalleid: Kalleid, Inc. is a boutique IT consulting firm that has served the scientific community since 2014. We work across the value chain in R&D, clinical and quality areas to deliver support services for software implementations in highly complex, multi-site organizations. At Kalleid, we understand that people are at the center of any successful business transformation and providing both business analysis and software testing/validation services is an important part to our integrated approach to IT projects. Our business analysts and testing/validation experts are seasoned professionals offering insights backed by 10+ years of experience who understand both the technologies being implemented and how to work with people and scientists. If you are interested in exploring how Kalleid business analysis and testing/validation services can benefit your organization, please don't hesitate to contact us today. Visit **kalleid.com** to learn more.

[1]"Success rates rise – transforming the high cost of low performance," Project Management Institute, 2017 Pulse of the Profession Survey. Available at: https://www.pmi.org/-/media/pmi/documents/public/pdf/learning/thought-leadership/pulse/pulse-of-the-profession-2017.pdf?sc_lang_temp=en