



**USER ACCEPTANCE TESTING (UAT) PROCESS
VERSION 1.0**

MARCH 3, 2008

Information and Technology Management Branch

❖ IM / IT Standards & Guidelines ❖

1.	INTRODUCTION.....	3
2.	DOCUMENT PURPOSE	3
3.	INTENDED AUDIENCE	3
4.	USER ACCEPTANCE TESTING (UAT) PROCESS	3
	4.1 UAT TEST CASES WRITING APPROACH	4
	4.1.1 Requirements-based test cases.....	4
	4.1.2 Business process-based test cases.....	4
	4.1.3 Data-driven test cases.....	4
	4.2 WHEN TO WRITE UAT TEST CASES.....	5
	4.3 WHEN TO RUN UAT TEST CASES	5
	4.4 WHEN TO STOP TESTING	6
	4.5 UAT STANDARDS AND GUIDELINES	6
	4.6 UAT ROLES AND RESPONSIBILITIES	7
	4.7 UAT CHECKPOINTS IN THE MINISTRY ADE-SDLC	8
	4.7.1 New Application development.....	8
	4.7.2 Maintenance of existing applications.....	8
	4.7.3 Infrastructure Migration Projects.....	8
	4.7.4 Emergency Fix.....	9
	4.8 UAT TESTING AUTOMATION OPTIONS	9

1. Introduction

User Acceptance Testing (UAT) does three things for a software development project :

- They measure how compliant the system is with the business requirements.
- They expose functionality/business logic problems that unit testing and system testing have missed out since unit testing and system testing do not focus much on functionality/business logic testing.
- They provide a means of determining how “done” the system is.

In reality, by the time the UAT starts, contracts have been signed and money has been spent. So, user acceptance testing is not an "accept or reject" proposition any more. UAT is rather more about finding gaps between “how the completed system works” and “how business operational processes are performed”. Also, UAT is generally considered to be a “validation process” rather than a “verification process”. “Validation” determines if something works as intended in the user's environment and meets their needs. “Verification” determines if something has been built according to specifications.

2. Document Purpose

The purpose of this document is to define and describe a process for User Acceptance Testing (UAT) of Ministry’s applications. The document also describes UAT roles and responsibilities, check points etc.

3. Intended Audience

The main intended audiences for this document are the clients, Business Analysts and other UAT testers who need to know about UAT standards of Ministry Applications. Also, The AMS service provider and IMG staff who are involved in the application promotion and change management processes would find this document useful.

4. User Acceptance Testing (UAT) Process

It is a common belief that well-documented Business Requirements are the basis for UAT and exhaustive requirements specification would be of great help during UAT. In reality, exhaustive requirements specifications are impossible. Even if exhaustive specifications were possible, they do not guarantee that the system will do what users want. The exhaustive requirements specifications cannot be validated directly. *Someone still has to translate those requirements specifications into use cases and test cases. These test cases are executed either by the dedicated UAT testers or clients themselves.*

A well-defined User Acceptance Test (UAT) process addresses these issues. The clients get a second independent perspective of testing to compensate for the errors and gaps that

could be present in requirements. Well-defined acceptance tests can evolve as the system functionality evolves. Acceptance tests can be validated directly – if an acceptance test passes, the system meets the user requirements documented in that test. Finally, a well-defined UAT process together with a feasible amount of test automation would yield productive and effective acceptance test results.

The following sections describe the Ministry ADE recommendations on UAT process.

4.1 UAT test cases writing approach

There are three common approaches to writing UAT test cases :

- Requirements-based test cases writing
- Business process (workflow) (or, user scenario) based test cases writing
- Data driven test cases writing

4.1.1 Requirements-based test cases

Writing test cases based on the business requirements that were documented earlier through BRD is a traditional and most common approach to test cases writing. The approach helps capture test scenarios directly from the BRD. However, if test cases are written only based on the business requirements and the BRD, then there are risks of carrying over the defects or missing things from the requirements to the test cases. Also, there are chances that the requirements themselves are incomplete or incorrectly specified or outdated and in such case, the test cases would also be incorrect or outdated.

4.1.2 Business process-based test cases

One of the most critical concerns in any system deployment is that the system will not work immediately after deployment to support the user's workflow or business processes. Business process validation through "Business process-based test cases" helps ensure that business processes are supported by the application or system that was delivered. The downside of adopting only Business process-based test cases is that data related testing is missed out.

4.1.3 Data-driven test cases

Data driven test cases are typically created as extensions to other types of acceptance test cases described earlier. Data that is external to the functional test cases is loaded and used to extend the test cases. As an example, if a form with some fields is going through acceptance testing, then the tester has to enter value for each form-field and then do the testing. Using a Data-driven test case, the tester can enter multiple sets of form-field values and store each set as a separate test case OR all multiple sets of form-field values

as a single test case for future replay and execution of the test case. Data-driven test cases are typically automated through test tools. Data-driven testing also means that scripts read data from an external storage site, for example, from a file or database, rather than use values hard-coded in the script. Such a separation makes tests logically simpler. A script containing several sets of hard-coded values can be rather difficult to modify. Normally, the storages hold both input and verifying data. The advantages of data driven test is that it can be repeated many times allowing the users to focus on the more important process testing. Also, the data driven testing includes verifications of applications input data with the data stored in the database.

The downside of the Data-driven test case is that it focuses heavily on the data side and misses out on the process and business requirements side.

Recommendations

ADE process recommends that no single approach is likely to be adequate on its own. A combination of the three approaches discussed above is recommended to be adopted for writing effective acceptance test cases.

4.2 When to write UAT test cases

Business people should write acceptance tests much before the development (coding) of the system is completed. Writing test cases early in the system development minimizes gaps between “what is expected of the system” and “what the system is delivering”. Also, the test cases should not be written too early in the development, since there should be enough information about what is being tested in order to write effective test cases.

Recommendations

ADE process recommends that UAT test case writing should not be the last step in the system development. UAT test cases should be written just after the completion and approval of requirements, data models and application models - but much before the development is completed.

A typical recommended scenario is to write Test Strategy and Test Planning and get it approved soon after the Business Requirements are approved. Test cases and test data preparation should be appropriately timed to be written in the development cycle without risking implementation timing.

4.3 When to run UAT test cases

Business processes and requirements and their timely validation should be paramount throughout the development process. It is a common practice to run the UAT test cases at

the end of the system development – that is after the system is moved into UAT environment. This practice poses the risk of detecting major system functionality problems too late in the development. A “configurable frequency” should be set for running the UAT test cases and tests should be able to run automatically at the configurable frequency, and also manually as needed. Once the tests have been written, they should be run frequently.

Recommendations

ADE process recommends that UAT test case should not be run only at the end of system development. Rather, they should be run frequently and at a configurable frequency and also manually as needed.

In situations where UAT is time consuming and not feasible for frequent and repeat runs, the following recommendation is applicable: UAT may be separated into testing of individual modules or module groups and the larger system testing at the end. Testing of modules or module groups should be done as and when the modules are migrated into UAT.

4.4 When to stop testing

The three important resources for testing, “people, time and money,” are limited for any project. It impossible to do testing for every possible test scenario. So, the testing process should cover the testing of important system functionality (on which the project justification is based) that can be achieved within the available time and budget.

Recommendations

ADE process recommends that “when to stop testing” is a judgment call and the amount of testing to be done should be evaluated based upon the amount of business functionality that the system must deliver as per project agreements and justifications.

4.5 UAT Standards and Guidelines

Ministry ADE process recommends the following guidelines for UAT Testing of Ministry Applications :

- The final UAT Testing must always be done only in the Ministry UAT environment.
- Prior to commencing of UAT testing, there should be confirmation from development teams that Unit Testing and System Testing have been successfully completed as per the ADE standards and all the units and the whole system have passed all the test cases outlined in the Unit Test and System test plans.

- Always create a UAT Test plan (a set of test cases arranged in the sequence of chronological execution) for the system before starting the actual testing.
- UAT Test Plan must be created before starting the system development.
- Always create UAT Test data for the system before starting the actual testing.
- To minimize the number of test cases, combine test cases into one if they test the same feature.
- Test cases should describe the *functionality (scenario) being tested, input, expected result, actual result, pass/fail status and rectification strategy for the problems discovered, test run date and time, name of the person/role who ran the test.*
- Use already existing/pre-established test cases wherever possible.
- Conduct the tests as specified in the test cases.
- Re-run the necessary tests after every change to the system.
- If any of the actual results do not agree with the expected results, the developer fixes the code, conducts unit tests and system test again and UAT testers (or clients) will rerun the specific UAT test cases.
- Update the UAT Test Plan if it was discovered to be incorrect or out of date.
- Where possible, create scripts to automate the execution of the test case.
- Use live or representative data as test data as much as possible in order to provide realistic test scenarios.

Recommendations

Ministry ADE process requires that UAT testing is done on the application before the request for promotion of the whole application to Production is considered. To this effect, the Business/Client Services is required to submit the following to the Ministry along with the request for promotion :

- **UAT test results or log files outlining test case name, functionality (scenario) being tested, input, expected result, actual result, pass/fail status, rectification strategy for the problems discovered, test run date and time, name of the person/role who ran the test.**
- **Confirmation that UAT was successful and the system has passed all the UAT test cases**

Ministry ADE process recommends preserving and storing all UAT Test cases and UAT Test data for re-running purposes.

4.6 UAT Roles and Responsibilities

The Ministry ADE process recommends that the whole process of UAT Testing (which must include UAT Test Plan creation, Test cases design, Test execution, Test results review and action/communication to development team is the responsibility of the

Business area (Clients) or a dedicated UAT team, with support from the Client Services team.

Fixing of problems found in UAT is the responsibility of the development teams or Ministry IMG, depending on whether the problem is a development issue or an infrastructure issue.

Promoting of successful system from UAT to Production is the responsibility of the IMG/WTS/CGI, subject to confirmation received from Business/Client Services on the success of UAT testing.

The Ministry ADE process recommends that all UAT Test cases are preserved and saved in a readable format and at an accessible location, so that the test cases could be re-used for re-running of test cases during application changes, infrastructure migrations etc without having to re-write the test cases.

4.7 UAT Checkpoints in the Ministry ADE-SDLC

The Ministry ADE process recommends the UAT testing be performed for all Ministry applications (regardless of category, roadmap and technology) at the following checkpoints in the Application's life cycle :

4.7.1 New Application development

All new applications must ensure that full UAT testing of the application is done as per the UAT standards defined by the Ministry ADE process. UAT results and log files must be submitted along with the request for promotion of the application from UAT to Production environment. Ministry IMG staff at their discretion may do a random auditing of the test results and log files prior to promoting the application to Production environment.

4.7.2 Maintenance of existing applications

During the implementation of post-production changes to the Applications, specific UAT test cases must be rerun and UAT results and log files must be submitted along with the request for promotion of the application from UAT to Production environment. Ministry IMG staff at their discretion may do a random auditing of the test results and log files prior to promoting the application to Production environment.

4.7.3 Infrastructure Migration Projects

UAT retesting is a mandatory task during infrastructure migration projects such as 10g upgrade project. UAT retesting should be done on the whole migrated system.

4.7.4 Emergency Fix

UAT Testing is mandatory during Emergency Fix. Emergency fix must not be applied directly to the Production environment. It must be applied first in the Emergency Fix environment (EFX). Specific UAT test cases must be rerun in the EFX environment before promotion of the system from EFX to Production. Ministry ADE recommends that applying the fix should also be done as a post-mortem exercise progressively in all other environments too : DEV → TEST → UAT. After applying the fix, the three levels of testing (unit testing, system testing and the UAT testing) should be performed in the respective environments before making the progressive promotion of application from DEV → TEST → UAT.

4.8 UAT Testing automation Options

Traditionally UAT is performed manually. The advantage of performing UAT manually is that clients need to see what the software actually looks like and how it performs. Automation takes this perspective away from the clients. The downside of manual testing is that certain mundane repetitive testing tasks could be highly time-consuming. Also, when the UAT test cases are to be rerun for multiple releases of the application at different times, the automation of UAT would be beneficial.

The possibility of procuring and implementing end-to-end testing tools is being explored. Most end-to-end testing tools support automation of some of the testing tasks such as *record and playback keyboard strokes* as the testing is performed. The end-to-end testing tools are also likely to support other types of testing as well, such as System Testing, Stress Testing etc.