Oracle Insurance Policy Administration System Quality Assurance Testing Methodology

An Oracle White Paper
August 2008

ORACLE

Oracle Insurance Policy Administration System Quality Assurance Testing Methodology

Oracle Insurance Policy Administration System Quality Assurance Testing Methodology

INTRODUCTION

Quality assurance (QA) testing involves assessing the operation of a system or application under controlled conditions (normal and abnormal) and evaluating the results. QA testing intentionally attempts to make things go wrong to determine if the system is operating according to product design and development specifications and user expectations. The testing process also verifies that the correct results occur under normal conditions.

The objectives of the QA testing methodology are

- Establish a consistent testing practice
- Provide common tools and templates
- Create reusable testware
- Ensure delivery of a high-quality system

The Oracle insurance policy administration system's QA testing methodology provides project managers, QA managers, QA testers, developers, and customers a framework for consistent, efficient, and successful software testing. This white paper describes the recommended QA testing methodology. Note that because this testing methodology is generic, adaptation to specific projects is required.

SUMMARY OF THE TESTING LIFECYCLE

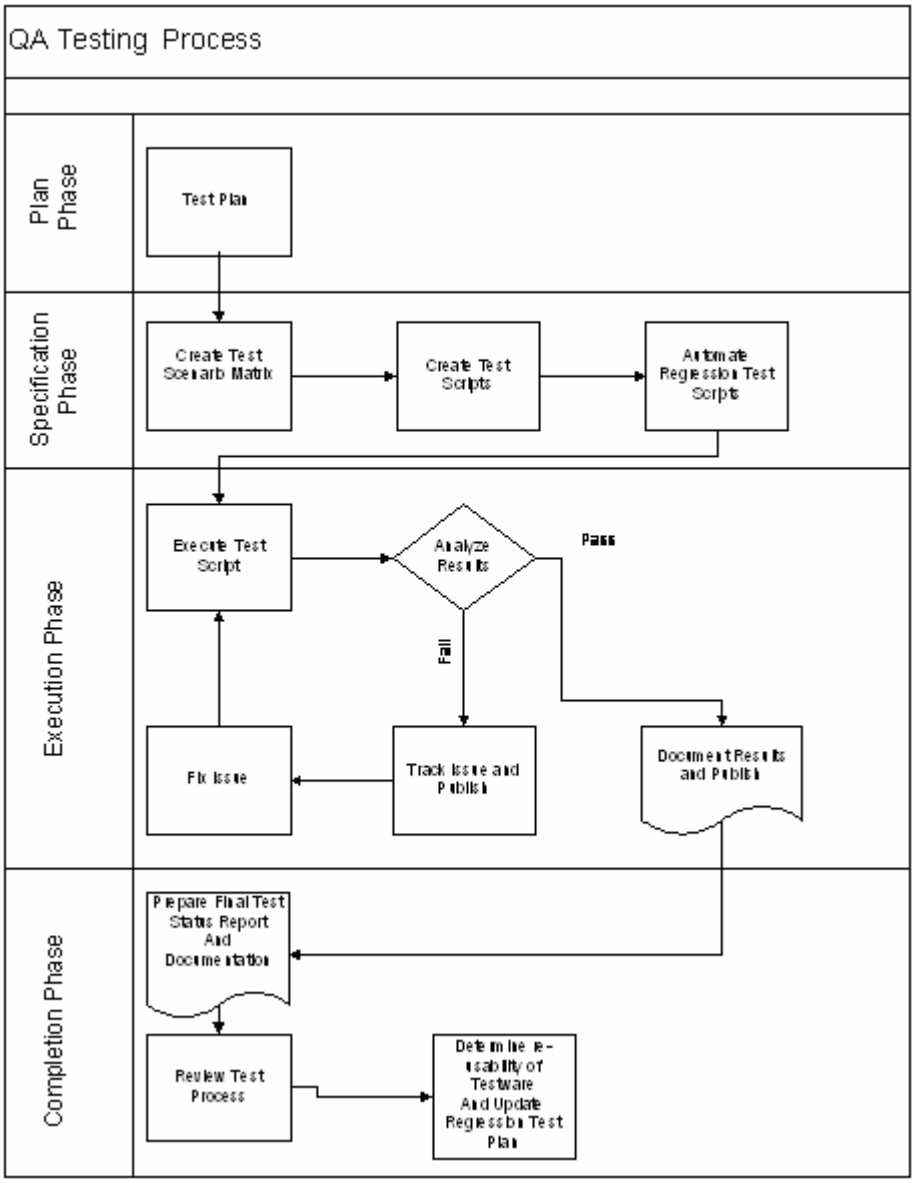
Oracle's insurance policy administration QA testing lifecycle is based on the Test Management Approach (TMap)[1] and includes four phases, each one comprising various supporting deliverables:

- Plan phase
  - Develop test plan
- Specification phase

[1] Martin Pol et al, *Testen Volgens TMap* (*Testing According to TMap*), Tutein Nolthenius, 's-Hertogenbosch (1995), cited in Tim Koomen and Martin Pol, *Test Process Improvement: A Practical Step-by-Step Guide to Structured Testing* (Boston, MA: Addison-Wesley Professional, 1999).

    - Develop test case matrix
    - Develop test case script
    - Automate regression test cases
- Execution phase
    - Execute test scripts
    - Report and track issues
    - Report testing progress
- Completion phase
    - Create release report: identify unresolved issues, determine potential risks
    - Review test process: identify successful areas, identify areas for improvement
    - Determine testware reusability: update master regression test plans and script

The four phases are used for all projects; however, not all the steps in each phase are needed and some phases can take place simultaneously. The figure illustrates the project testing lifecycle; each box represents a QA process.



The Oracle insurance policy administration system QA testing lifecycle

PLAN PHASE

The primary activity of the planning phase is developing the test plan. The test plan outlines how the system will be tested, the resources needed to test the system, the responsibilities of each resource, what tools will be used, and when the test activities are scheduled to take place.

Test Plan

The test plan is the first document QA prepares for any project. The product and functional specifications as well as the use case documents are used in preparation of the test plan. The product specification outlines the product requirements

whereas the functional specification outlines how the new features will be implemented. The test plan is a living document that needs to be maintained throughout the project.

The test plan consists of the following categories:

- Project introduction
  - Provides an overview of the project
  - Tests objectives
- Scope of the test effort
  - Lists the areas of the system that will be affected (and tested) and the new functionality provided by the change
  - Illustrates the size of the testing effort and the key areas that will be affected by the change, for example, reports, interfaces, data entry screens
- Test strategy
  - Lists what types of testing will be performed and gives definitions for each (see "Glossary" for types and definitions)
  - Prioritizes testing; lists any testing that will not be performed and associated risks
- Environment requirements
  - Outlines the environmental requirements (hardware and software) needed to create and update the test environment(s)
  - Lists the roles and responsibilities for environment setup and maintenance
- Test schedule
  - Lists the schedule for each test method (can include time estimate and costs); provides the location of the document if the schedule is created in another application (for example, Microsoft Project, Microsoft Excel spreadsheet)
- Control procedures
  - Lists the frequency of testing status meetings and required attendees
  - Describes the method of reporting, prioritizing, assigning, and communicating issues to the development staff; can be handled through Microsoft Word or Excel documents or through an online defect tracking system
  - Lists the frequency and the method of communicating issues to the entire team
  - Lists how change requests will be addressed

- Functions to be tested
  - Lists the functionality that will be tested (usually links back to use cases)
- Resources and responsibilities
  - Lists the resources needed to complete the testing effort and the size of the team and their responsibilities
- Deliverables
  - Lists the project deliverables (including writing the test cases and setting up the test environment), the group or person responsible, and the target completion date
- Suspension criteria
  - Lists the situations that could warrant suspension of testing and the person(s) authorized to make that determination; only critical items that either physically prevent testing or result in wasted effort are able to halt testing
- Resumption criteria
  - Lists the conditions that must exist to resume testing once it has been suspended
- Dependencies
  - Lists dependencies that could affect testing, such as staff needs and hardware needs
- Risks
  - Lists potential project risks that could arise (such as an aggressive testing schedule or project scope change) and possible work-arounds
- Tools
  - Lists any tools that will be used to assist with testing (for example, automated testing tools or online defect tracking system), who will be responsible for them, who will have access to them, and how they will be used
- Documentation
  - Lists the type of documentation that will be created during the test process
- Approvals
  - Includes the list of names required for sign-off

SPECIFICATION PHASE

The primary activities of the specification phase are test case scenario development and test script development. The test case scenarios are based on the product

specifications or use cases. The test scripts are derived from the test case scenarios. As the testers create the test case scenarios, they also build the individual test script.

Test Case Scenario Matrix

This document is used to outline each test scenario. Each scenario should reference only one business requirement. The matrix consists of the following fields:

- **Item #.** Sequential number used to identify the scenario.
- **Functionality/Use Case**. Links the scenario to the required functionality or use case.
- **Description.** Description of the scenario that needs to be tested.
- **Tester**. Person who is responsible for testing the scenario.
- **Test Case #**. Cross-reference back to the actual script that includes this scenario.
- **Pass/Fail**. Indicates if the scenario failed when tested.
- **Issue #**. The number of the item that failed.

Test Case Scripts

This document is used to define the functionality being tested as well as outline the testing steps. The test scripts are derived from the test case scenarios. Although the matrix only allows for one "test condition" or scenario per item, the items can be combined where appropriate and executed in the same script. The test script should be written in such detail that someone other than the author could easily execute the script.

The test case script typically contains the following information:

Introduction

- Provides an overview of the document
- Lists any definitions, acronyms, or abbreviations needed to properly interpret the document
- Lists any documents that will be referenced in the test script
- Lists any output that will be created or referenced to verify the accuracy of results.

Testing Environment

- Documents the environment in which the test is being run
- Documents setup information
- Lists any parameters or flags that need to be set to successfully execute the test

Test Case and Script

- Provides a description of the case and lists the test case scenarios that are being addressed
- Provides any preconditions needed for the case
- Outlines in detail step-by-step instructions to execute the script
- Details the expected results
- Details the values needed to create other conditions or scenarios for the case

Automating Test Cases

Because of the iterative nature of the Oracle insurance policy administration system's development process, currently only regression tests are automated. This could be expanded to functional areas that can be executed during functional testing of base code changes.

The test scripts that are created are supplied to the automation team. The team creates scripts and spreadsheets accessed by the testing tool (HP QuickTest Professional) to enter test cases and verify expected results.

EXECUTION PHASE

The primary activities of the execution phase are executing the test case scripts, documenting results, reporting, prioritizing, tracking, assigning, and retesting issues.

Executing the Test Case Script

- The tester executes the test script.
- The tester analyzes the results and verifies the following:
  - The system is behaving as expected without any errors.
  - Data was stored and produced correctly, calculations were performed according to specifications, and so on.
  - Any applicable correspondence or report is produced and data is correct.
  - Any applicable "spawned" transaction is produced.

Documenting the Test Results

The tester documents findings on the test script and if necessary creates an issue report.

Reporting Issues

Issue reports are written to notify the project team of areas that are not functioning according to the business requirements or product development standards. The report needs to communicate what problem was encountered and how the system

should function. The issues can be tracked using a defect tracking application or manually using Word or Excel.

The report should contain the following items:

- **Issue number.** Number used to identify the issue.
- **Reported by.** Name of the tester who found the issue.
- **Status.** The status of the issue, for example, submitted, assigned, in progress, duplicate, withdrawn, fixed, in QA, closed.
- **Date.** Date the issue was found.
- **Build.** The build version in which the issue was found.
- **Functional area.** The functional area of the system being tested.
- **Priority.** The severity of the issue: critical, high, medium, low.
- **Test script.** The ID number of the test script being tested.
- **Developer.** The name of the developer assigned to the issue.
- **Description.** Detailed description of the problem, what the tester was doing (including step-by-step instructions to recreate the problem), and any error messages. The tester should do as much analysis as possible to determine what caused the problem and how to recreate it.
- **Developer's notes.** The developer completes this section once assigned.
- **Tester's notes.** The tester completes this section once the fix has been delivered to QA.

The issue reports are forwarded to one member of the team and compiled into an issues log—useful for notifying the team of outstanding issues. The more-detailed issue report is forwarded to the developer assigned to resolve the issue and the tester assigned to test the fix.

Reviewing the Testing Status and Issues

The QA team prepares and distributes a weekly status report to the product development team. This report describes

- Critical issues that were encountered that require management intervention
- Areas of concern that management should be made aware of
- Key accomplishments for the week
- The next week's objectives, listing any outstanding objectives that were not completed in the current week and why and objectives for the coming week
- Testing metrics

The team meets weekly to discuss the testing status and to review and prioritize outstanding issues. The QA lead person determines the frequency of this meeting.

Retesting Issues

Once an issue has been resolved and the code has been moved into the test environment (per code migration procedures), the tester re-executes the test script. Then, depending on the outcome, one of the following actions is taken:

- If the issue has been resolved, the issue report is updated and the issue is closed.
- If the issue is still occurring, the issue status is changed to "Failed Retest" and the issue is reassigned to the developer.
- If a new issue is discovered during testing, a new issue report is created.

COMPLETION PHASE

The primary activity in the completion phase is to identify any unresolved issues, potential risks, and work-arounds and evaluate the test process.

Identifying Unresolved Issues

Any issues that are not resolved before implementation need to be identified. The project team determines if any issues can prevent the product rollout and identifies potential risks and work-arounds. A formal meeting is scheduled to discuss those issues. In addition to containing regular status report information, this report contains the following:

- Unresolved open issues and associated risks
- Deferred issues that will be addressed in another release
- Pending issues that require some decision or additional information before being addressed
- Fixed issues that have been fixed but not yet verified by QA

Reviewing the Test Process

The project team then meets to review the test process. The purpose of this meeting is to

- Review how well the process was followed during the project cycle and identify areas for improvement
- Determine the reusable testware

CONCLUSION

Software applications that operate virtually error free and according to design require rigorous testing. The Oracle insurance policy administration system's QA testing methodology described in this white paper provides project managers, QA managers, QA testers, developers, and customers a framework for consistent, efficient, and successful software testing and aids in delivering the highest-quality software applications to the market.

GLOSSARY

The following terms are typically used during a QA software testing lifecycle.

Acceptance testing: Final testing based on specifications of the end user or customer, or based on use by end users and customers over some limited period of time.

Alpha testing: Testing an application when development is nearing completion; minor design changes could still be made as a result of such testing.

Beta testing: Testing an application when development and testing are essentially completed and final problems need to be found before final release.

Black box testing: Tests that are not based on any knowledge of internal design or code; the tests are based on requirements and functionality.

Evaluation: The evaluation and inspection of the various intermediate products or processes in the system development cycle.

Expected results: The predetermined conditions and values the system is expected to produce. This includes screen images, reports, and update files.

Functional testing: Black box testing that is based on overall requirements specifications—covers all combined parts of a system.

Guerilla testing: Testing that intentionally tries to break the system, for example, entering data out of order or entering illogical values.

Incremental integration testing: Continuous testing of an application as new functionality is added; requires that various aspects of an application's functionality be independent enough to work separately before all parts of the program are completed.

Install/uninstall testing: Testing of full, partial, or upgrade installation and uninstallation process.

Integration test: Testing of combined parts of an application to determine if they function together correctly.

Issue: The difference found between the expected and the actual results, a defect in the code, or a change needed to satisfy a business requirement.

Load testing: Testing an application under heavy loads, such as testing of a Web site under a range of loads to determine at what point the system's response time degrades or fails.

Performance testing: Testing that ensures that the application responds in the time limit set by the user.

Quality assurance: All planned and systematic actions needed to provide adequate confidence that a product or service will satisfy given requirements for quality.

Recovery testing: Testing a system to determine how well it recovers from crashes, hardware failures, and other catastrophic problems.

Regression testing: Retesting after fixes or modifications of the software or its environment.

Sanity testing: Typically an initial testing effort done to determine if a new software version is performing well enough to be accepted for a major testing effort.

Security testing: Testing how well the system protects against unauthorized internal or external access, willful damage, and so on.

Static testing: Physical examination of work product (document or code) for agreement with governing specifications and standards without executing the code (example code review).

Stress test: A test that ensures the system will respond appropriately with many users and activities happening simultaneously.

Test case: A description of a test to be executed, with focus on a specific aim.

Test plan: A document describing the scope, approach, resources, and schedule of intended test activities.

Test process: The collection of tools, techniques, and working methods used to perform a test.

Test script: A description of the execution of a sequence of related actions and checks related to test cases, and a description of how the testing will be done.

Testware: All test documents (such as test specifications, test scripts, a description of test infrastructure, and so on) produced during the test process. Reuse of this document is required for maintenance purposes and therefore must be portable and maintainable.

Unit testing: The smallest unit of code that can be tested for particular functions or code modules; typically performed by the developer (not the tester).

Usability testing: Testing for "user friendliness," or how easily and intuitively a customer can use the application or product.

User acceptance testing (UAT): Testing done by user groups to determine if the software is satisfactory to an end user or customer.

White box testing: Tests based on knowledge of the internal logic of an application's code.

Oracle Insurance Policy Administration System Quality Assurance Testing Methodology
August 2008

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com