Testing Guidelines for NCR

At the heart of the testing framework is the notion of Test Driven Development (TDD). Test definition, execution and evaluation play an integral role in every part of the project life cycle. On the one hand this can be thought of as a comprehensive quality assurance process that minimizes project risk and cost by emphasizing defect prevention over inspection. However, it's really much more than that. The process fosters better communication between the project team and its customers, minimizes the development of unnecessary functionality and provides the foundation for continuous integration and iterative development. Thus, it contributes to project scope management, project risk management, project schedule management, project cost management and project communications management.

The following provides a brief overview of the different levels of testing and where they fit in the software development life cycle. In general, we recommend the following for all levels of testing:

1. Perform the testing activity as early as possible in the development life cycle
2. Automate as much as possible – Although the cost of automation may seem high, it is typically more than offset by the dramatically reduced cost and increased utility of regression testing. If the cost of maintenance of automated test scripts is high, this is typically due to immature automated testing expertise or use of tools that are not fit for purpose. Always investigate these potential root causes before promoting manual testing over automated testing.

Unit Level Testing

Unit Testing

Automated "white-box" tests are written by developers to test code at a very low level of granularity. Developers use unit tests to verify that their code works as designed and to assess unanticipated side effects. As described in more detail later, the continuous build process also leverages the unit test suite. ThoughtWorks encourages the use of the "XUnit" family of tools (JUnit – Java, Test::Unit – Ruby, nUnit - .NET, http::Unit, phpUnit etc.).

http://martinfowler.com/bliki/Xunit.html

Functional Testing

Functional Testing

Functional testing validates that the System meets the functional requirements as defined in a Story. This type of testing is focused on narrow system functionality (typically at the Story level). This is primarily performed by Test Engineers in collaboration with BAs using tools like JBehave, Cucumber, Selenium etc.

Organizations such as NCR often rely on commercial testing tools such as QTP, WinRunner, SilkTest, etc. Although ThoughtWorks have experienced success using such tools, we typically prefer open source alternatives as they tend to be easier to adapt to individual project needs and generate easier to maintain test scripts.

Integration Testing

Integration testing validates the proper functioning of a system in conjunction with other related systems. This should be performed whenever required system functionality requires multiple distinct systems working together. This type of testing is used to verify that the system works as specified at the module, environment and interface level.

End-to-End Testing

End-to-end tests exercise fully integrated system functionality. It is applicable in a situation where the desired business functionality is accomplished by multiple related systems.

Regression Testing

Regression testing is performed to validate that unexpected side effects have not been introduced during development. The most common method is to re-run the tests that have been previously executed and validate the desired functionality of the system under test. As mentioned earlier, maximum automation of test scripts dramatically reduces the cost and increases the utility of regression testing.

User Acceptance Testing

User Acceptance Tests are “black-box” tests defined and performed by actual users or user representatives. They validate system behavior by performing typical and non-typical system activities. These tests are ideally defined prior to the start of an iteration. This clarifies the meaning of the stated requirements, provides a clearly defined definition of success and keeps business people (or representatives) actively engaged in the project. UAT can be both an iterative process for acceptance of specific features as well as a distinct phase within the development process.

NFR, Operations and Compliance Testing

Data Validation Testing

This type of testing is performed to validate that system data supports the functionality developed for the system. Data validation testing is necessary in any scenario where the application (1) changes the state of a data store used by other systems or (2) retrieves data from a source, transforms it, and updates a target database. Different flavors of data validation testing includes:

Data migration - validating that system data can be moved from a source data store to a target data store, typically focuses on one-time or batch-oriented data movement.

Live Data Testing - data validation activity focusing on production data using actual live data where possible and sanitized data when restrictions apply.

Scalability and Performance Testing

These are automated tests used to measure system response under stressful conditions such as large numbers of simultaneous users or high transaction volumes. We advocate performing this type of testing as early as possible in the project life cycle in an effort to minimize the impact of invalid architecture assumptions. Third-party tools such as JMeter and LoadRunner are often used for this type of testing. However, custom tools may be required early in the project because the third-party tools typically require a minimum level of completed functionality.

PLEASE NOTE – A typical excuse for delaying scalability and performance testing is that the target functionality is not available for testing early in the project life cycle. We still advocate early and often scalability and performance testing because such testing can more easily identify the source of degradation at the time that it is introduced. In other words, early scalability and performance testing should focus more on trends rather than adherence to requirements.

Reliability Testing

In some cases, it is required to validate that the system meets the expectations in terms of fault tolerance. The possibilities are:

Failover / Recovery Testing - validate that the system displays expected environmental fault tolerance behavior.

Disaster Recovery Testing - validate that the system can be restored to a usable state after environmental disaster.