The Fundamental Testing Process in Practical or Company environment

Ms. Palak Khanna
Assistant Professor
Department of Computer Applications,
Chandigarh Group of Colleges
Landran, Mohali (Punjab), India

Ms. Navpreet Kaur,
Assistant Professor
Department of Computer Applications
Chandigarh Group of Colleges
Landran, Mohali (Punjab), India

Abstract
The paper focuses on the practical process of software testing. Those who just get out of college and start searching for jobs have this curiosity "How would be the actual working environment in the companies?" This paper intends to highlight the basic practices that are followed in all working environments. Starting from SRS, then testing a small unit to finally a complete system involves various levels.
Keywords: *Levels of Testing, Unit Testing, Integration Testing, System Testing, Acceptance Testing*

Introduction

Each product development starts with development of Software requirement Specifications (SRS) and then the product goes into the development phase. Each product is broken into multiple smaller modules/phases so as to facilitate the development process and easily track the process and above all reduce the risk failures during the development process. Development of the product also happens in different levels starting from creation of components, Components integration, and then complete system development. Testing starts from testing a simple screen, sub-module, module, combination of modules, and then finally testing the complete system. For each component/sub-component also called feature / Sub-feature being developed, testing can be performed. Thus for each development level, there is corresponding test level.

Levels of Testing:

1. **Unit testing:-**Unit testing focuses on the verification effort of the smallest unit of software design - the unit. The units are identified at the detailed design phase of the software development life cycle, and the unit testing can be conducted in parallel phase for multiple units.

Five aspects are tested under unit testing considerations

- The module interphase is tested to ensure that information properly flows in and out of the unit program under test.
- The local data structure is examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithm's execution.
- Boundary conditions are tested to ensure that the module operates properly at boundaries established to limit or restrict processing.

- All independent paths (basis paths) through the control structure are exercised to ensure that all statements in a module have been executed at least once.
- And finally, all error-handling path are tested.

Unit Test Coverage Goals:

Path coverage: Path coverage technique is to verify weather each of the possible paths in each of the function has executed properly. A path is a set of branches of possible flow. Since loop introduces unbounded number of paths, the path coverage technique employs a test that consider only a limited number of looping possibilities.

Statement coverage: The statement coverage technique requires that every statement in the program is to be evoked at least once. It verifies coverage at high level rather than decision execution or Boolean expressions. The advantage of this is that measure can be applied directly to the object code & does not require processing source code.

Decision (Logic/Branch) coverage: The decision coverage test technique seeks to identify the percentage of all possible decision outcomes that have been considered by a suit of test procedures. It requires that every point of entry & exit in the software program be invoked at least once. It also requires that all possible conditions for a decision in the program be exercised at least once.

Condition coverage: This technique seeks to verify the accuracy of true or false outcome of each Boolean sub expression. This technique employs tests that measure the sub expressions independently.

Multiple-condition coverage: This technique covers the different conditions which are inter-related.

The integral parts covered under unit testing will be: Active server page (ASP) that invokes the ATL component (which in turn can use C++ classes).The actual interaction of the component with the persistent store or database or database tables driver for the unit testing of a unit belonging to a particular component or subsystem depends on the component alone. Wherever user interface is available, UI called from a web browser will initiate the testing process. If UI is not available then appropriate driver (code in C++ as an example) will be developed for testing.

Unit testing would also include testing inter-unit functionality within a component. This will consist of two different units belonging to same component interacting with each other. The functionality of such units will be tested with separate unit test(S).
Each unit of functionality will be tested for the following considerations:
Type: This validation ensures that the value corresponding to the type of field(only) should be entered i.e. if any is created for alphanumeric characters, then the user should not be allowed to

input any value other than alphanumeric characters.

Presence: This validation ensures all mandatory fields should be present, they should also be mandated by database by making the column NOT NULL.

Size: This validation ensures the size limit for a float or variable character string input from the user not to exceed the size allowed by the database for the respective column.

Validation: This is for any other business validation that should be applied to a specific field or for a field that is dependent on another field (E.G.: Range validation like body temperature should not exceed 106 degree Celsius), duplicate check etc.

GUI based: in case the unit is UI based, GUI related consistency check like font sizes, background color, window sizes, and message & error boxes will be checked.

Integration testing

After unit testing, modules shall be assembled or integrated to form the complete software package as indicated by the high level design. Integration testing is a systematic technique for verifying the software structure and sequence of execution while conducting test to uncover errors associated with interfacing.

Black-box test case design techniques are the most prevalent during integration, although limited amount of white box texting may be used to ensure coverage of major control paths. Integration testing is sub-divided as follows:

i) **Top-down integration testing:** Top-down integration is an incremental approach for the construction of a program structure. Modules are integrated by moving downward through the hierarchy, beginning with the main control module (main program). Modules subordinate to the main control module are incorporated into the structure either by depth-first approach or by breadth-first approach.

ii) **Bottom-up Integration testing**: Bottom-up integration testing, as its name implies begins construction and testing with atomic modules (i.e., modules at the lowest level in the program structure). Since modules are integrated from bottom up, processing required for modules sub-ordinate to a given level is always available and the need for stubs is eliminated.

iii) **Integration testing for OO projects:**

Thread Based Testing: Thread based testing follows an execution thread through object to ensure that classes collaborate correctly. It involves:

- Set of class required to respond to one input or even for system are identified. Each thread is integrated and tested individually.
- Regression test is applied to ensure that no side effects occurs.

Use based testing

Use based testing evaluates the system in layers. The common practice is to employ the use cases to drive the validation process. It involves:

- Initially independent classes (i.e., classes that use very few other classes) are integrated and tested.
- Followed by the dependent classes that use independent classes. Here dependent classes with a layered approach are used.
- Followed by testing next layer of (dependent) classes that use independent classes

This sequence is repeated by adding and testing next layer of dependent classes until entire system is tested.

Integration testing for web application

Collaboration diagrams, screen and report layouts are matched to OOAD (object-oriented Analysis and design) and associated class integration test case report is generated.

2. System testing

After the software has been integrated (constructed), sets of high order tests shall be conducted. System testing verifies that all elements mesh properly and the overall system function/performance is achieved. The purpose of system testing is to fully exercise the computer-based system. The aim is to verify all the system elements and validate its conformance against SRS.

Acceptance testing

When the customize software is built for one customer, a series of acceptance tests are conducted to enable the customer to validate all the requirements. Acceptance tests are conducted at the development site or at the customer site depending upon the requirements and mutually agreed principles. Acceptance testing may be conducted either by the customer depending on the type of project & the contractual agreement. A series of acceptance tests are conducted to enable the customer to validate all requirements as per user requirement document (URD).Alpha and beta (or field) testing are also forms of acceptance testing. Developers or stakeholders of the product often want to get feedback from potential or existing customers in their market before selling the product commercially.

Alpha Testingis performed at the developing organization's site.

Beta testing, or field testing, is performed by people at their own locations using customer data. Both are performed by potential customers, and not the developer of the product. Acceptance testing assesses the system's readiness for testing, development and use, and is not necessarily the final level of testing. For example, a large-scale system integration test may come after the acceptance test for a system.

Summary: The various levels of testing involved in practice are- unit testing that focuses verification effort on the smallest unit of software design. Integration testing that is a systematic

technique for verifying the software structure and sequence of execution while conducting tests to uncover errors associated with interfacing. System testing that aims to verify that all system elements validate conformance against SRS. Acceptance tests that are conducted at the development site or at the customer site depending upon the requirement and mutually agreed principles.

References

[1] http://istqbexamcertification.com/what-is-fundamental-test-process-in-software-testing/

[2] http://www.softwaretestinghelp.com/

[3] http://www.testingexcellence.com/fundamental-test-process-software-testing/

[4] Introduction to SQA & Testing Vol. I by BTES

[5] Lessons Learned in Software Testing, by C. Kaner, J. Bach, and B. Pettichord

[6] Testing Computer Software, by C. Kaner, J. Falk, and H. Nguyen