**INTRODUCTION TO TESTING**

**Introduction to Testing:**

Testing is a process, which reveals errors in the program. It is the major quality measure employed during software development. During software development. During testing, the program is executed with a set of test cases and the output of the program for the test cases is evaluated to determine if the program is performing as it is expected to perform.

**TESTING IN STRATEGIES**

In order to make sure that the system does not have errors, the different levels of testing strategies that are applied at differing phases of software development are:

**Unit Testing:**

Unit Testing is done on individual modules as they are completed and become executable. It is confined only to the designer's requirements.

**Each module can be tested using the following two Strategies:**

**Black Box Testing:**

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing has been uses to find errors in the following categories:

* Incorrect or missing functions
* Interface errors
* Errors in data structure or external database access
* Performance errors
* Initialization and termination errors.

In this testing only the output is checked for correctness. The logical flow of the data is not checked.

**White Box testing:**

In this the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It has been uses to generate the test cases in the following cases:

* Guarantee that all independent paths have been Executed.
* Execute all logical decisions on their true and false Sides.
* Execute all loops at their boundaries and within their operational bounds
* Execute internal data structures to ensure their validity.

**Integrating Testing :**

Integration testing ensures that software and subsystems work together a whole. It tests the interface of all the modules to make sure that the modules behave properly when integrated together.

**System Testing :**

Involves in-house testing of the entire system before delivery to the user. It's aim is to satisfy the user the system meets all requirements of the client's specifications.

**Acceptance Testing :**

It is a pre-delivery testing in which entire system is tested at client's site on real world data to find errors.

#### Test Approach :

**Testing can be done in two ways:**

* Bottom up approach
* Top down approach

**Bottom up Approach:**

Testing can be performed starting from smallest and lowest level modules and proceeding one at a time. For each module in bottom up testing a short program executes the module and provides the needed data so that the module is asked to perform the way it will when embedded with in the larger system. When bottom level modules are tested attention turns to those on the next level that use the lower level ones they are tested individually and then linked with the previously examined lower level modules.

**Top down approach:**

This type of testing starts from upper level modules. Since the detailed activities usually performed in the lower level routines are not provided stubs are written. A stub is a module shell called by upper level module and that when reached properly will return a message to the calling module indicating that proper interaction occurred. No attempt is made to verify the correctness of the lower level module.

**Validation:**

The system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled. In case of erroneous input corresponding error messages are displayed

Testing is a process used to help identify the correctness, completeness and quality of developed computer software. With that in mind, testing can never completely establish the correctness of computer software.

The quality of the application can and normally does vary widely from system to system but some of the common quality attributes include reliability, stability, portability, maintainability and usability. Refer to the ISO standard ISO 9126 for a more complete list of attributes and criteria.

Testing helps is verifying and Validating if the Software is working as it is intended to be working. Things involve using Static and Dynamic methodologies to Test the application.

* 1. Testing is a process of executing a program with the intent of finding an error.
  2. A good test case is one that has a high probability of finding an as yet undiscovered error.
  3. A successful test is one that uncovers an as yet undiscovered error.

Testing should systematically uncover different classes of errors in a minimum amount of time and with a minimum amount of effort. A secondary benefit of testing is that it demonstrates that the software appears to be working as stated in the specifications.

The data collected through testing can also provide an indication of the software's reliability and quality. But, testing cannot show the absence of defect, it can only show that software defects are present.

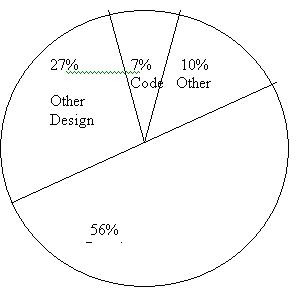
# Testing Start Process

Testing early in the life cycle reduces the errors. Test deliverables are associated with every phase of development. The goal of Software Tester is to find bugs, find them as early as possible, and make them sure they are fixed.

The number one cause of Software bugs is the Specification. There are several reasons specifications are the largest bug producer.

In many instances a Spec simply isn’t written. Other reasons may be that the spec isn’t thorough enough, it’s constantly changing, or it’s not communicated well to the entire team. Planning software is vitally important. If it’s not done correctly bugs will be created.

The next largest source of bugs is the Design, That’s where the programmers lay the plan for their Software. Compare it to an architect creating the blue print for the building, Bugs occur here for the same reason they occur in the specification. It’s rushed, changed, or not well communicated.



Coding errors may be more familiar to you if you are a programmer. Typically these can be traced to the Software complexity, poor documentation, schedule pressure or just plain dump mistakes. It’s important to note that many bugs that appear on the surface to be programming errors can really be traced to specification.

The other category is the catch-all for what is left. Some bugs can blamed for false positives, conditions that were thought to be bugs but really weren’t.

There may be duplicate bugs, multiple ones that resulted from the square root cause. Some bugs can be traced to Testing errors.

Costs: The costs re logarithmic- that is, they increase tenfold as time increases. A bug found and fixed during the early stages when the specification is being written might cost next to nothing, or 10 cents in our example. The same bug, if not found until the software is coded and tested, might cost $1 to $10. If a customer finds it, the cost would easily top $100.

# Testing Stop Process

This can be difficult to determine. Many modern software applications are so complex, and run in such as interdependent environment, that complete testing can never be done. "When to stop testing" is one of the most difficult questions to a test engineer. Common factors in deciding when to stop are:

* Deadlines (release deadlines, testing deadlines.)
* Test cases completed with certain percentages passed
* Test budget depleted
* Coverage of code/functionality/requirements reaches a specified point
* The rate at which Bugs can be found is too small
* Beta or Alpha Testing period ends
* The risk in the project is under acceptable limit.

Practically, we feel that the decision of stopping testing is based on the level of the risk acceptable to the management. As testing is a never ending process we can never assume that 100 % testing has been done, we can only minimize the risk of shipping the product to client with X testing done. The risk can be measured by Risk analysis but for small duration / low budget / low resources project, risk can be deduced by simply: -

* Measuring Test Coverage.
* Number of test cycles.
* Number of high priority bugs.

**8.1Test Plan**

Test plan consist of following points:

1. Title of the Project: “Air Freight Pursue”
2. Objective of the document: - In this test plan we are covering the activities and functionality of different modules and their sub modules. In this document we covering what kind of test cases should described.
3. Scope of the document: in this document, in each phase, what are going to do and how are going to do. In this section we are mentioning which requirements are testing. Take one module i.e. user login module.

For user module the client requirements are:

* + User name must be not less than 4 character and not greater than 10.
  + Username must start with character, not with digits.
  + Username should not contain any special character.
  + Password field should fill with at lest 4character.

The above requirements are only for login module and other functional requirements can declared in this section. The requirements which are declared in this section it may depend on other requirements on different module.

1. Objective of Testing: the main objective of testing in this application is to chances of preventing the defects on the client environment.
2. Brief Description of the application:-
3. Critical Functionalities: - in this section we are discussing Key roles & Causes for success of application.
4. Test Data Requirements and Collection:-

In this section we are collecting the requirements for the application. From the different resources.

Such as

* Collecting from the client.
* Refereeing the existing applications which are similar to current one’s

8) Training Requirements:-

Training requirements are focused in 2 areas

1. Technology – in this we are discussing about the new technology in the market and scope of that technologies in the future.
2. Domain- in this we are discussing about the Existing personal Training knowledge.

9) Resource Requirements: - In this we discussing about the required resources for the application such

* + Employees
  + Software Licenses
  + Bridge Number

10) Scheduling: - In this section we will be discussing about Staring Date and End Date of the application are given.

11) Input Criteria: - there are different criteria conditions i.e. Unit Testing, Release Note & Installation.

12) Exit Criteria: - In this only evaluation documents only accepted.

13) Risk Analysis: - In this we are converse about Risk Analysis such as

* Risk on Resources.
* Risk on TimeLine.
* Risks are identified by Preparing Solution Plan.

**8.2 Types of Tests Carried:-**

In our application we are performing the Gray Box testing (white and black).

Test case for Login Form:

|  |  |  |
| --- | --- | --- |
| Steps | Descriptions | Executed Result |
| 1. | Launch the Application | Application Should be Launched. |
| 2 | Enter Valid Username & Password In respective fields | Data should be entered in fields. |
| 3 | Click on Login” Button | Field should be validating. If not show error Message |