**1.1. What is testing?**

A procedure for critical evaluation or a means of determining the presence, quality, or genuineness of something is termed as testing. In general, testing is finding out how well something works.

**What is Software?**  
Software is a set of instructions to perform some task. Software is used in many applications of the real world. Some of the examples are  
•    Application software, such as word processors  
•    Firmware in an embedded system  
•    Middleware, which controls and co-ordinates distributed systems  
•    System software such as operating systems  
•    Video Games  
•    Websites

Software is both a product and a vehicle for delivering a product (information). It is not manufactured but engineered. As time goes, the software deteriorates during changing conditions and requirements.

All of these software applications need to run without any error and provide a quality service to the user of the application. In this regard the software has to be tested for its accuracyand correctness.

**Introduction to Software Testing**

Software Testing is the process which ensures that the system or application is free from the inconsistency or unusual behaviour and working as per the expected behaviour or requirements. It is a critical element in the software development life cycle and has the potential to save time and money by identifying problems early and to improve customer satisfaction by delivering a more defect-free product.

Software Testing can be defined in simple words as “Performing Verification and Validation of the Software Product” for its correctness and accuracy of working.

**IEEE Definition**  
Software testing is the process of analysing a software item to detect the differences between existing and required conditions (that is, bugs) and  
to evaluate the features of the software item.

**Importance of Software Testing**

“A clever person solves a problem. A wise person avoids it.”   
Albert Einstein

**Why is testing necessary?**

•    Complexity of software makes it error prone  
•    Untested or weakly tested software can cause losses in real life.  
•    To learn about reliability of software  
•    To stay in business

Software testing answers questions that development and code reviews cannot.  
•    Does it really work as expected?  
•    Does it meet the users’ requirements?   
•    Is it what the users expect?  
•    Do the users like it?   
•    Is it compatible with our other systems?   
•    How does it perform?   
•    How does it scale when more users are added?   
•    Which areas need more work?   
•    Is it ready for release?

Examples of Importance of Software Testing.

•    ISRO calls off GSLV launch after fuel leak.

Because of the final flight test ISRO could identify the problem and called off the launch and thus saved millions and millions of RS  
<<<http://www.thehindu.com/news/national/isro-calls-off-gslv-launch-after-fuel-leak/article5038402.ece>>>

•    NASA Mars Polar Lander

NASA Mars Polar Lander – 1999 – Malfunction due to anunexpected setting of a single data bit

•    Disney’s Lion King

Disney’s Lion King – 1994 – 1995- The software did not work on the most common systems that the public had.

**When to Start / Stop Testing**

**1. When to Start Testing**

An early start to testing reduces the cost, time to rework and error free software that is delivered to the client.   
In Software Development Life Cycle (SDLC) testing can be started from the Requirements Gathering phase and lasts till the deployment of the software.

It also depends on the development model that is being used.

Example: In Water fall model formal testing is conducted in the Testing phase.

                And in Incremental model, testing is performed at the end of every

                increment/iteration and at the end the whole application is tested.

**2. When to Stop Testing**

It is difficult to determine when to stop testing, as testing is a never ending process.  
It is difficult to claim that any software is 100% tested.

Following are the aspects which should be considered to stop the testing:

•    Testing Deadlines.  
•    Completion of test case execution.   
•    Completion of Functional and code coverage to a certain point.   
•    Bug rate falls below a certain level and no high priority bugs are identified.   
•    Management decision.

**1.2. Software Failure**

**What is a Software Failure**

Software failure is a term used to label a malfunction, a flaw, a failure or a fault in a computer program that produces incorrect results. Most software failures arise from errors made by people in a program's design.

Effect of an Error

An error or mistake leads to a defect which can in turn cause the software failure.

                                     Error     --------->     Defect    ------------->         Failure

**Causes of software Failure**

Some of the causes making software to fail are:

* Environmental conditions

          – The environment set up for testing the various versions of the code being unavailable on time.

* Hardware Failures

          – Crash of the dependent hardware, loss of versions can impact the code quality

* Lack of skilled resources

– Lack of knowledge both Technical and Business can lead to critical, expensive errors. In order to avoid this, companies should seek people with specialized skills, even though they may come at a higher price.

* Inadequate reviews / walk-through

– The defects might go unnoticed and cause the failure.

* Changing Requirements

– Frequent changes to requirements impacts the quality of code and in turn the software.

* Time constraints to meet up the deadlines

          – Rushing up to meet the deadlines leaves many defects being  injected into the code.

**Impact of Software Failure**

Software failure can lead to:

* Loss of money
* Loss of time
* Loss of business reputation
* Injury
* Death

**Software Failure Examples**

* In February 2003 the U.S. Treasury Department mailed 50,000 Social Security checks without a beneficiary name. A spokesperson said that the missing names were due to a software program maintenance error.
* In October 1999 the $125 million NASA Mars Climate Orbiter—an interplanetary weather satellite—was lost in space due to a data conversion error. Investigators discovered that software on the spacecraft performed certain calculations in English units (yards) when it should have used metric units (meters).
* In June 1996 the first flight of the European Space Agency's Ariane 5 rocket failed shortly after launching, resulting in an uninsured loss of $500,000,000. The disaster was traced to the lack of exception handling for a floating-point error when a 64-bit integer was converted to a 16-bit signed integer.

**Keeping Software Under control**

* To prevent the software failure, it has to be thoroughly tested before it go live.
* Incorrect testing or insufficient testing causes a software to fail.
* More and better testing is needed for a defect free software, but it isnot quite as simple to achieve.
* Each and every part of the software has to be tested.
* With large and complex systems it will never be possible to testeverything exhaustively.
* There is a greater probability of a failure in the large and complexsystem and the impact of the failure is also greater.

What we test, and how much we test is related to risk involved. Greater risk implies more and better testing is required.

Testing is done to deliver a quality product. Here quality refers whether the product is meeting the user's requirements.

**Resource Triangle**

The items at the corner of the triangle are time, money and quality. These three affect one another, and also influence the features that are or are not included in the delivered software. If we need to deliver a system faster (i.e. in less time), it will usually cost more.

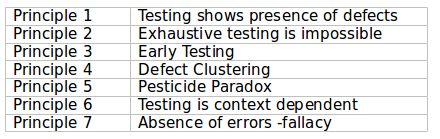
* Testing is to ensure that key functional and nonfunctional requirements are examined before any defects are reported to the development team.
* Testing cannot directly remove defects, nor can it directly enhance quality.
* By reporting defects it makes their removal possible and so contributes to the enhanced quality of the system.
* In addition, the systematic coverage of a software product to be measured.
* Testing is one component in the overall quality assurance activity that seeks to ensure that systems enter service without defects that can lead to serious failures.

**1.3. General Principles of Testing**

**Seven Principles of Testing**

To test the entire possible combinations of project, EXECUTION TIME &COSTS will rise exponentially. Hence, the testing principle states that EXHAUSTIVE( In this type of testing we try to check the output given by the software by entering all the possible inputs, in fact we use all the permutations and combinations of the inputs.) testing is not possible. Instead we need optimal amount of testing based on the risk assessment of the application where the principles will play a major role.

**Principles Of Testing**



Software testing is an extremely creative and intellectually challenging task. When testing follows the principles, the creative element of test design and execution rivals any of the preceding software development steps.

**Principle 1: Testing shows presence of defects**

* Testing can show that defects are present, but cannot prove that there are no defects. Even after testing the application or product thoroughly we cannot say that the product is 100% defect free.
* Testing always reduces the number of undiscovered defects remaining in the software but even if no defects are found, it is not a proof of correctness.

**Principle 2: Exhaustive testing is impossible**

* Testing everything including all combinations of inputs and preconditions is not possible. So, instead of doing the exhaustive testing we can use risks and priorities to focus testing efforts
* Example:In an application in one screen there are 15 input fields, each having 5 possible values, then to test all the valid combinations you would need 30,517,578,125 (515) tests.
* The above Example is very unlikely that the project timescales would would allow for this number of tests. So, accessing and managing risk is one of the most important activities and reason for testing in any project.

**Principle 3: Early Testing**

* The sooner we start the testing activities the better we can utilize the available time. As soon as the initial products, such the requirement or design documents are available, we can start testing.
* It is common for the testing phase to get squeezed at the end of the development life-cycle, i.e. when development has finished, so by starting testing early, we can prepare testing for each level of the development life-cycle
* Another important point about early testing is that when defects are found earlier in the life-cycle, they are much easier and cheaper to fix.

**Principle 4: Defect Clustering**

* During testing, it can be observed that most of the reported defects are related to small number of modules within a system. i.e. small number of modules contain most of the defects in the system.
* A small number of modules contains most of the defects discovered during pre-release testing or shows the most operational failures.
* By identifying and focusing on these clusters, testers can efficiently test the sensitive areas while concurrently testing the remaining “non-sensitive” areas.

**Principle 5: Pesticide Paradox**

* The same kind of tests are repeated again and again, eventually the same set of test cases will no longer be able to find any new bugs. This is called “Pesticide Paradox”.
* To overcome this “Pesticide Paradox”, it is really very important to review the test cases regularly and new and different tests need to be written to exercise different parts of the software or system to potentially find more defects.
* Anytime a fault is fixed or a new functionality added, we need to do regression testing to make sure the new changed software has not broken any other part of the software.
* However, those regression test cases also need to change to reflect the changes made in the software to be applicable and hopefully fine new defects.

**Principle 6: Testing is context dependent**

* Testing is basically context dependent. Different kinds of sites are tested differently.
* For example, a software application in a medical device needs more testing than a games software. More importantly a medical device software requires risk based testing, be compliant with medical industry regulators and possibly specific test design techniques.
* The same tests should not be applied across the board because different software products have varying requirements, functions and purposes

**Principle 7: Absence of errors –fallacy**

* A test that finds no errors is different than concluding that the software is error-free. It should be assumed that all software contains some faults, even if said faults are hidden.
* When the built system is unusable and does not fulfill the user’s needs and expectations then finding and fixing defects does not help.

**1.4. Software Quality**

**Quality**

There exist several definitions of software quality. Also, one quality attribute might be more important to a user than another. In any case, software quality is a multidimensional quantity and is measurable.

Quality from the

* Customer’s Viewpoint - Fitness for use, or other customer needs
* Producer’s Viewpoint - Meeting requirements

Quality has two elements

* QA - Quality Assurance
* QC - Quality Control

**Quality Assurance**

* Quality assurance activities are work process oriented.
* They measure the process, identify deficiencies, and suggest improvements.
* The direct results of these activities are changes to the process.
* These changes can range from better compliance with the process to entirely new processes.
* The output of quality control activities is often the input to quality assurance activities.
* Audits are an example of a QA activity which looks at whether and how the process is being followed. The end result may be suggested improvements or better compliance with the process.

**Quality Control**

What is Quality Control?

* Quality control activities are work product oriented.
* They measure the product, identify deficiencies, and suggest improvements.
* The direct results of these activities are changes to the product.
* These can range from single-line code changes to completely reworking a product from design.
* They evaluate the product, identify weaknesses and suggest improvements.
* Testing and reviews are examples of QC activities since they usually result in changes to the product, not the process.
* QC activities are often the starting point for quality assurance (QA) activities.