# What is Software Testing?

Software testing is a process of executing a program or application with the intent of finding the [**software bugs**](http://tryqa.com/what-is-defect-or-bugs-or-faults-in-software-testing/).

* It can also be stated as the **process of**[**validating**](http://tryqa.com/what-is-validation-in-software-testing-or-what-is-software-validation/)**and**[**verifying**](http://tryqa.com/what-is-verification-in-software-testing-or-what-is-software-verification/)that a software program or application or product:
  + Meets the business and technical requirements that guided it’s design and development
  + Works as expected
  + Can be implemented with the same characteristic.

Let’s break the definition of [**Software testing**](http://tryqa.com/what-is-a-software-testing/) into the following parts:

**1)  Process:**Testing is a process rather than a single activity.

**2)**  **All Life Cycle Activities:** Testing is a process that’s take place throughout the [**Software Development Life Cycle (SDLC)**](http://tryqa.com/what-are-the-software-development-life-cycle-sdlc-phases/).

* The process of designing tests early in the life cycle can help to prevent defects from being introduced in the code. Sometimes it’s referred as **“verifying the test basis via the**[**test design**](http://tryqa.com/what-is-test-design-or-how-to-specify-test-cases/)**”**.
* The **test basis** includes documents such as the requirements and design specifications.

**3)**  [**Static Testing**](http://tryqa.com/what-is-static-testing/)**:**  It can test and find defects without executing code. Static Testing is done during verification process. This testing includes reviewing of the documents (including source code) and static analysis. This is useful and cost effective way of testing.  For example: reviewing, [**walkthrough**](http://tryqa.com/what-is-walkthrough-in-software-testing/), [**inspection**](http://tryqa.com/what-is-inspection-in-software-testing/), etc.

**4)**  [**Dynamic Testing**](http://tryqa.com/what-is-dynamic-testing-technique/)**:**  In dynamic testing the software code is executed to demonstrate the result of running tests. It’s done during validation process. For example: [**unit testing**](http://tryqa.com/what-is-unit-testing/),[**integration testing**](http://tryqa.com/what-is-integration-testing/), [**system testing**](http://tryqa.com/what-is-system-testing/), etc.

**5)**[**Planning**](http://tryqa.com/what-is-the-purpose-and-importance-of-test-plans/)**:**  We need to plan as what we want to do. We control the test activities, we report on testing progress and the status of the software under test.

**6)  Preparation:**  We need to choose what testing we will do, by selecting test conditions and [**designing test cases**](http://tryqa.com/what-is-test-design-or-how-to-specify-test-cases).

**7)  Evaluation:**  During evaluation we must check the results and evaluate the software under test and the completion criteria, which helps us to decide whether we have finished testing and whether the software product has passed the tests.

**8)  Software products and related work products:**  Along with the testing of code the testing of requirement and design specifications and also the related documents like operation, user and training material is equally important.

# Why is software testing necessary?

[**Software Testing**](http://tryqa.com/what-is-a-software-testing/) is necessary because we all make mistakes. Some of those mistakes are unimportant, but some of them are expensive or dangerous. We need to check everything and anything we produce because things can always go wrong – [**humans make mistakes all the time**](http://tryqa.com/when-do-defects-in-software-testing-arise/).

Since we assume that our work may have mistakes, hence we all need to check our own work. However some mistakes come from bad assumptions and blind spots, so we might make the same mistakes when we check our own work as we made when we did it. So we may not notice the flaws in what we have done.

Ideally, we should get someone else to check our work because another person is more likely to spot the flaws.

There are several reasons which clearly tells us as why Software Testing is important and what are the major things that we should consider while testing of any product or application.

Software testing is very important because of the following reasons:

1. Software testing is really required to point out the [**defects**](http://tryqa.com/what-is-defect-or-bugs-or-faults-in-software-testing/)and errors that were made during the [**development phases**](http://tryqa.com/what-are-the-software-development-life-cycle-sdlc-phases/).
   * Example: Programmers may make a mistake during the implementation of the software. There could be many reasons for this like lack of experience of the programmer, lack of knowledge of the programming language, insufficient experience in the domain, incorrect implementation of the algorithm due to complex logic or simply human error.
2. It’s essential since it makes sure that the customer finds the organization reliable and their satisfaction in the application is maintained.
   * If the customer does not find the testing organization reliable or is not satisfied with the quality of the deliverable, then they may switch to a competitor organization.
   * Sometimes contracts may also include monetary penalties with respect to the timeline and quality of the product. In such cases, if proper software testing may also prevent monetary losses.
3. It is very important to ensure the Quality of the product. Quality product delivered to the customers helps in gaining their confidence. (Know more about [**Software Quality**](http://tryqa.com/what-is-software-quality/))
   * As explained in the previous point, delivering good quality product on time builds the customers confidence in the team and the organization.
4. Testing is necessary in order to provide the facilities to the customers like the delivery of high quality product or software application which requires lower maintenance cost and hence results into more accurate, consistent and reliable results.
   * High quality product typically has fewer defects and requires lesser maintenance effort, which in turn means reduced costs.
5. Testing is required for an effective performance of software application or product.
6. It’s important to ensure that the application should not result into any [**failures**](http://tryqa.com/what-is-a-failure-in-software-testing/)because it can be very expensive in the future or in the later stages of the development.
   * Proper testing ensures that bugs and issues are detected early in the life cycle of the product or application.
   * If defects related to requirements or design are detected late in the life cyle, it can be very expensive to fix them since this might require redesign, re-implementation and retesting of the application.
7. It’s required to stay in the business.
   * Users are not inclined to use software that has bugs. They may not adopt a software if they are not happy with the stability of the application.
   * In case of a product organization or startup which has only one product, poor quality of software may result in lack of adoption of the product and this may result in losses which the business may not recover from.

# What are software testing objectives and purpose?

[**Software Testing**](http://tryqa.com/) has different goals and objectives.The major objectives of Software testing are as follows:

* [**Finding defects**](http://tryqa.com/what-is-defect-or-bugs-or-faults-in-software-testing/) which may get created by the programmer while developing the software.
* Gaining confidence in and providing information about the level of [**quality**](http://tryqa.com/what-is-software-quality/).
* To prevent defects.
* To make sure that the end result meets the business and user requirements.
* To ensure that it satisfies the BRS that is Business Requirement Specification and SRS that is System Requirement Specifications.
* To gain the confidence of the customers by providing them a quality product.

Software testing helps in finalizing the software application or product against business and user requirements. It is very important to have good test coverage in order to test the software application completely and make it sure that it’s performing well and as per the specifications.

While determining the [**test coverage**](http://tryqa.com/what-is-test-coverage-in-software-testing-its-advantages-and-disadvantages/) the test cases should be designed well with maximum possibilities of finding the errors or bugs. The [**test cases**](http://tryqa.com/test-case/) should be very effective. This objective can be measured by the number of defects reported per test cases. Higher the number of the defects reported the more effective are the test cases.

Once the delivery is made to the end users or the customers they should be able to operate it without any complaints. In order to make this happen the tester should know as how the customers are going to use this product and accordingly they should write down the test scenarios and design the test cases. This will help a lot in fulfilling all the customer’s requirements.

Software testing makes sure that the testing is being done properly and hence the system is ready for use. Good coverage means that the testing has been done to cover the various areas like functionality of the application, [**compatibility**](http://tryqa.com/what-is-compatibility-testing-in-software/)of the application with the OS, hardware and different types of browsers, [**performance testing**](http://tryqa.com/what-is-performance-testing-in-software/) to test the performance of the application and [**load testing**](http://tryqa.com/what-is-load-testing-in-software/) to make sure that the system is reliable and should not crash or there should not be any blocking issues. It also determines that the application can be deployed easily to the machine and without any resistance. Hence the application is easy to install, learn and use.

# What is Defect or bugs or faults in software testing?

**Definition:**A **defect is an error or a bug**, in the application which is created. A programmer while designing and building the software can make mistakes or error. These mistakes or errors mean that there are flaws in the software. These are called defects.

* When actual result deviates from the expected result while testing a software application or product then it results into a defect. Hence, any deviation from the specification mentioned in the product functional specification document is a defect. In different organizations it’s called differently like bug, issue, incidents or problem.
* When the result of the software application or product does not meet with the end user expectations or the software requirements then it results into a Bug or Defect. These defects or bugs occur because of an error in logic or in coding which results into the [**failure**](http://tryqa.com/what-is-a-failure-in-software-testing/)or unpredicted or unanticipated results.

**Additional Information about Defects / Bugs:**

While testing a software application or product if large number of defects are found then it’s called Buggy.

When a tester finds a bug or defect it’s required to convey the same to the developers. Thus they report bugs  with the detail steps and are called as Bug Reports, issue report, problem report, etc.

This Defect report or Bug report consists of the following information:

* **Defect ID** – Every bug or defect has it’s unique identification number
* **Defect Description** – This includes the abstract of the issue.
* **Product Version** – This includes the product version of the application in which the defect is found.
* **Detail Steps** – This includes the detailed steps of the issue with the screenshots attached so that developers can recreate it.
* **Date Raised** – This includes the Date when the bug is reported
* **Reported By** – This includes the details of the tester who reported the bug like Name and ID
* **Status** – This field includes the Status of the defect like New, Assigned, Open, [**Retest**](http://tryqa.com/what-is-retesting/), [**Verification**](http://tryqa.com/what-is-verification-in-software-testing-or-what-is-software-verification/), Closed, Failed, Deferred, etc.
* **Fixed by** – This field includes the details of the developer who fixed it like Name and ID
* **Date Closed** – This includes the Date when the bug is closed
* **Severity –**Based on the severity (Critical, Major or Minor) it tells us about impact of the defect or bug in the software application
* **Priority –**Based on the Priority set (High/Medium/Low) the order of fixing the defect can be made. (Know more about [**Severity and Priority**](http://tryqa.com/what-is-the-difference-between-severity-and-priority/))

# What is a Failure in software testing?

If under certain environment and situation defects in the application or product get executed then the system will produce the wrong results causing a failure.

Not all [**defects**](http://tryqa.com/what-is-defect-or-bugs-or-faults-in-software-testing/) result in failures, some may stay inactive in the code and we may never notice them. Example:  Defects in dead code will never result in failures.

It is not just defects that give rise to failure. Failures can also be caused because of the other reasons also like:

* Because of the environmental conditions as well like a radiation burst, a strong magnetic field, electronic field or pollution could cause faults in hardware or firmware. Those faults might prevent or change the execution of software.
* [**Failures may also arise**](http://tryqa.com/from-where-do-defects-and-failures-in-software-testing-arise/) because of human error in interacting with the software, perhaps a wrong input value being entered or an output being misinterpreted.
* Finally failures may also be caused by someone deliberately trying to cause a failure in the system.

**Difference between Error, Defect and Failure in software testing:**

**Error:**The mistakes made by programmer is known as an ‘Error’.  This could happen because of the following reasons:

–           Because of some confusion in understanding the functionality of the software

–           Because of some miscalculation of the values

–           Because of misinterpretation of any value, etc.

**Defect:**The bugs introduced by programmer inside the code are known as a defect. This can happen because of some programatical mistakes.

**Failure:**If under certain circumstances these defects get executed by the tester during the testing then it results into the failure which is known as software failure.

Few points that are important to know:

* When tester is executing a test he/she may observe some difference in the behavior of the feature or functionality, but this not because of the failure. This may happen because of the wrong test data entered, tester may not be aware of the feature or functionality or because of the bad environment. Because of these reasons incidents are reported. They are known as incident report. The condition or situation which requires further analysis or clarification is known as incident. To deal with the incidents the programmer need to to the analysis that whether this incident has occurred because of the failure or not.
* It’s not necessary that defects or bugs introduced in the product are only by the software. To understand it further let’s take an example. A bug or defect can also be introduced by a business analyst. Defects present in the specifications like requirements specification and design specifications can be detected during the [**reviews**](http://tryqa.com/what-are-the-types-of-review/). When the defect or bug is caught during the review cannot result into failure because the software has not yet been executed.
* These defects or bugs are reported not to blame the developers or any people but to judge the quality of the product. The quality of product is of utmost importance. To gain the confidence of the customers it’s very important to deliver the quality product on time.

# From where do defects and failures in software testing arise?

Defects and failures basically arise from:

* [**Errors in the specification**](http://tryqa.com/what-is-black-box-specification-based-also-known-as-behavioral-testing-techniques/), design and implementation of the software and system
* Errors in use of the system
* Environmental conditions
* Intentional damage
* Potential consequences of earlier errors

[**Errors in the specification**](http://tryqa.com/what-is-black-box-specification-based-also-known-as-behavioral-testing-techniques/)**and design of the software:**

Specification is basically a written document which describes the functional and non – functional aspects of the software by using prose and pictures. For testing specifications there is no need of having code. Without having code we can test the specifications. About 55% of all the bugs present in the product are because of the mistakes present in the specification. Hence testing the specifications can save lots of time and the cost in future or in later stages of the product.

**Errors in use of the system:**

Errors in use of the system or product or application may arise because of the following reasons:

–          Inadequate knowledge of the product or the software to the tester. The tester may not be aware of the functionalities of the product and hence while testing the product there might be some defects or failures.

–          Lack of the understanding of the functionalities by the developer. It may also happen that the developers may not have understood the functionalities of the product or application properly. Based on their understanding the feature they will develop may not match with the specifications. Hence this may result into the defect or failure.

**Environmental conditions:**

Because of the wrong setup of the testing environment testers may report the defects or failures. As per the recent surveys it has been observed that about 40% of the tester’s time is consumed because of the environment issues and this has a great impact on quality and productivity. Hence proper test environments are required for quality and on time delivery of the product to the customers.

**Intentional damage:**

The defects and failures reported by the testers while testing the product or the application may arise because of the intentional damage.

Consider an example where an application is not secure and does not check for SQL Injections. During security testing, testers can inject SQL commands that may result in the application data or database being corrupted. In this case the intentional damage would have been caused and reported by the testers.

If this issue is not caught, it could be exploited by hackers who could also inflict intentional damage.

**Potential consequences of earlier errors:**

Errors found in the earlier stages of the development reduce our cost of production. Hence it’s very important to find the error at the earlier stage. This could be done by reviewing the specification documents or by walkthrough. The downward flow of the defect will increase the cost of production.

# When do defects in software testing arise?

Because of the following reasons the software [**defects**](http://tryqa.com/what-is-defect-or-bugs-or-faults-in-software-testing/)arise:

– The person using the software application or product may not have enough knowledge of the product.

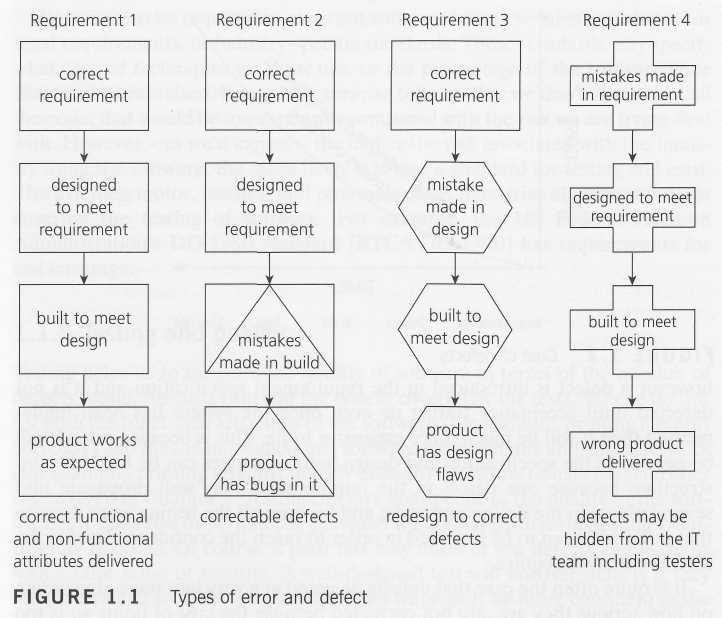
– Maybe the software is used in the wrong way which leads to the defects or [**failures**](http://tryqa.com/what-is-a-failure-in-software-testing/).

– The developers may have coded incorrectly and there can be defects present in the design.

– Incorrect setup of the testing environments.

To know when defects in [**software testing**](http://tryqa.com/what-is-a-software-testing/) arise, let us take a small example with a diagram as given below.

We can see that **Requirement 1** is implemented correctly – we understood the customer’s requirement, designed correctly to meet that requirement, built correctly to meet the design, and so deliver that requirement with the right attributes: [**functionally**](http://tryqa.com/what-is-functional-testing-testing-of-functions-in-software/), it does what it is supposed to do and it also has the right [**non-functional**](http://tryqa.com/what-is-non-functional-testing-testing-of-software-product-characteristics/) attributes, so it is fast enough, easy to understand and so on.

With the other requirements, errors have been made at different stages. **Requirement 2** is fine until the software is coded, when we make some mistakes and introduce defects. Probably, these are easily spotted and corrected during testing, because we can see the product does not meet its design specification.

The defects introduced in **Requirement 3** are harder to deal with; we built exactly what we were told to but unfortunately the designer made some mistakes so there are defects in the design. Unless we check against the requirements definition, we will not spot those defects during testing. When we do notice them they will be hard to fix because design changes will be required.

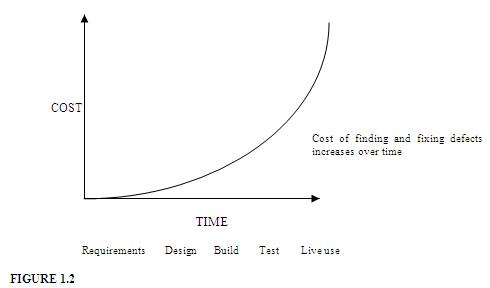
The defects in **Requirement 4** were introduced during the definition of the requirements; the   product has been designed and built to meet that flawed requirements definition. If we test the product meets its requirements and design, it will pass its tests but may be rejected by the user or customer. Defects reported by the customer in acceptance test or live use can be very costly. Unfortunately, requirements and design defects are not rare; assessments of thousands of projects have shown that defects introduced during requirements and design make up close to half of the total number of defects.

# What is the cost of defects in software testing?

The cost of [**defects**](http://tryqa.com/what-is-defect-or-bugs-or-faults-in-software-testing/)can be measured by the impact of the defects and [**when we find them**](http://tryqa.com/from-where-do-defects-and-failures-in-software-testing-arise/). Earlier the defect is found lesser is the cost of defect. For example if error is found in the requirement specifications during requirements gathering and analysis, then it is somewhat cheap to fix it. The correction to the requirement specification can be done and then it can be re-issued. In the same way when defect or error is found in the design during design review then the design can be corrected and it can be re-issued. But if the error is not caught in the specifications and is not found till the [**user acceptance**](http://tryqa.com/what-is-acceptance-testing/) then the cost to fix those errors or defects will be way too expensive.

If the error is made and the consequent defect is detected in the [**requirements phase**](http://tryqa.com/what-are-the-software-development-life-cycle-phases/) then it is relatively cheap to fix it.

Similarly if a requirement specification error is made and the consequent [**defect**](http://tryqa.com/what-is-defect-or-bugs-or-faults-in-software-testing/) is found in the **design phase** then the design can be corrected and reissued with relatively little expense.

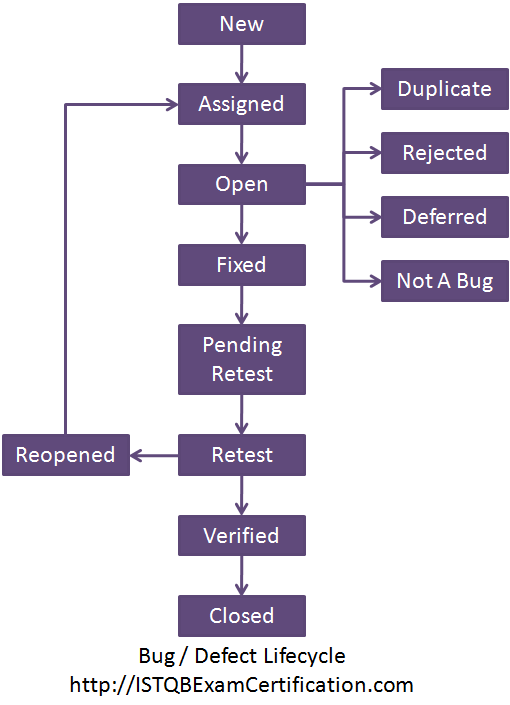
**[](http://tryqa.com/wp-content/uploads/2011/12/cost-of-defects.jpg)**

The same applies for **construction phase**. If however, a defect is introduced in the requirement specification and it is not detected until [**acceptance testing**](http://tryqa.com/what-is-acceptance-testing/) or even once the system has been implemented then it will be much more expensive to fix. This is because rework will be needed in the specification and design before changes can be made in construction; because one defect in the requirements may well propagate into several places in the design and code; and because all the testing work done-to that point will need to be repeated in order to reach the confidence level in the software that we require.

It is quite often the case that defects detected at a very late stage, depending on how serious they are, are not corrected because the cost of doing so is too expensive.

# What is a Defect Life Cycle or a Bug lifecycle in software testing?

**Defect life cycle** is a cycle which a defect goes through during its lifetime. It starts when defect is found and ends when a defect is closed, after ensuring it’s not reproduced. [**Defect life cycle**](http://tryqa.com/what-is-a-defect-life-cycle/) is related to the bug found during testing.

The bug has different states in the Life Cycle. The Life cycle of the bug can be shown diagrammatically as follows:

[**Bug or defect**](http://tryqa.com/what-is-defect-or-bugs-or-faults-in-software-testing/)**life cycle includes following steps or status:**

1. **New:**  When a defect is logged and posted for the first time. It’s state is given as new.
2. **Assigned:**  After the tester has posted the bug, the lead of the tester approves that the bug is genuine and he assigns the bug to corresponding developer and the developer team. It’s state given as assigned.
3. **Open:** At  this state the developer has started analyzing and working on the defect fix.
4. **Fixed:** When developer makes necessary code changes and verifies the changes then he/she can make bug status as ‘Fixed’ and the bug is passed to testing team.
5. **Pending retest:**  After fixing the defect the developer has given that particular code for retesting to the tester. Here the testing is pending on the testers end. Hence its status is pending retest.
6. [**Retest**](http://tryqa.com/what-is-retesting/)**:**  At this stage the tester do the retesting of the changed code which developer has given to him to check whether the defect got fixed or not.
7. [**Verified**](http://tryqa.com/what-is-verification-in-software-testing-or-what-is-software-verification/)**:** The tester tests the bug again after it got fixed by the developer. If the bug is not present in the software, he approves that the bug is fixed and changes the status to “verified”.
8. **Reopen:** If the bug still exists even after the bug is fixed by the developer, the tester changes the status to “reopened”. The bug goes through the life cycle once again.
9. **Closed:** Once the bug is fixed, it is tested by the tester. If the tester feels that the bug no longer exists in the software, he changes the status of the bug to “closed”. This state means that the bug is fixed, tested and approved.
10. **Duplicate:** If the bug is repeated twice or the two bugs mention the same concept of the bug, then one bug status is changed to “duplicate**“.**
11. **Rejected:** If the developer feels that the bug is not genuine, he rejects the bug. Then the state of the bug is changed to “rejected”.
12. **Deferred:** The bug, changed to deferred state means the bug is expected to be fixed in next releases. The reasons for changing the bug to this state have many factors. Some of them are [**priority**](http://tryqa.com/what-is-the-difference-between-severity-and-priority/)of the bug may be low, lack of time for the release or the bug may not have major effect on the software.
13. **Not a bug:**  The state given as “Not a bug” if there is no change in the functionality of the application. For an example: If customer asks for some change in the look and feel of the application like change of colour of some text then it is not a bug but just some change in the look of the application.

# What is the difference between Severity and Priority?

There are two key things in defects of the [**software testing**](http://tryqa.com/what-is-a-software-testing/). They are:

1)     Severity

2)     Priority

What is the difference between Severity and Priority?

**1)  Severity**:

It is the extent to which the [**defect**](http://tryqa.com/what-is-defect-or-bugs-or-faults-in-software-testing/) can affect the software. In other words it defines the impact that a given defect has on the system.**For example:** If an application or web page crashes when a remote link is clicked, in this case clicking the remote link by an user is rare but the impact of  application crashing is severe. So the severity is high but priority is low.

Severity can be of following types:

* **Critical:**The defect that results in the termination of the complete system or one or more component of the system and causes extensive corruption of the data. The failed function is unusable and there is no acceptable alternative method to achieve the required results then the severity will be stated as critical.
* **Major:**The defect that results in the termination of the complete system or one or more component of the system and causes extensive corruption of the data. The failed function is unusable but there exists an acceptable alternative method to achieve the required results then the severity will be stated as major.
* **Moderate:**The defect that does not result in the termination, but causes the system to produce incorrect, incomplete or inconsistent results then the severity will be stated as moderate.
* **Minor:**The defect that does not result in the termination and does not damage the [**usability**](http://tryqa.com/what-is-usability-testing-in-software-and-its-benifits-to-end-user/)of the system and the desired results can be easily obtained by working around the defects then the severity is stated as minor.
* **Cosmetic:**The defect that is related to the enhancement of the system where the changes are related to the look and field of the application then the severity is stated as cosmetic.

**2)  Priority**:

Priority defines the order in which we should resolve a defect. Should   we fix it now, or can it wait? This priority status is set by the tester to the developer mentioning the time frame to fix the defect. If high priority is mentioned then the developer has to fix it at the earliest. The priority status is set based on the customer requirements.**For example:**If the company name is misspelled in the home page of the website, then the priority is high and severity is low to fix it.

Priority can be of following types:

* **Low:**The defect is an irritant which should be repaired, but repair can be deferred until after more serious defect have been fixed.
* **Medium:**The defect should be resolved in the normal course of development activities. It can wait until a new build or version is created.
* **High:**The defect must be resolved as soon as possible because the defect is affecting the application or the product severely. The system cannot be used until the  repair has been done.

**Few very important scenarios related to the severity and priority which are asked during the interview:**

**High Priority & High Severity**: An error which occurs on the basic functionality of the application and will not allow the user to use the system. (Eg. A site maintaining the student details, on saving record if it, doesn’t allow to save the record then this is high priority and high severity bug.)

**High Priority & Low Severity:** The spelling mistakes that happens on the cover page or heading or title of an application.

**High Severity & Low Priority:** An error which occurs on the functionality of the application (for which there is no workaround) and will not allow the user to use the system but on click of link which is rarely used by the end user.

**Low Priority and Low Severity:** Any cosmetic or spelling issues which is within a paragraph or in the report (Not on cover page, heading, title).

# What are the principles of testing?

**Principles of Testing**– There are seven principles of [**testing**](http://tryqa.com/what-is-a-software-testing/). They are as follows:

**1) Testing shows presence of defects:** Testing can show the [**defects**](http://tryqa.com/what-is-defect-or-bugs-or-faults-in-software-testing/) 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.

**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**](http://tryqa.com/what-is-risk-in-software-testing/)and [**priorities**](http://tryqa.com/what-is-the-difference-between-severity-and-priority/) to focus testing efforts. For 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. This is very unlikely that the project timescales 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.

**3) Early testing:** In the [**software development life cycle**](http://tryqa.com/what-are-the-software-development-life-cycle-phases/) testing activities should start as early as possible and should be focused on defined objectives.

**4) Defect clustering:** A small number of modules contains most of the defects discovered during [**pre-release testing**](http://tryqa.com/what-is-alpha-testing/) or shows the most operational failures.

**5) Pesticide paradox:** If the same kinds of tests are repeated again and again, eventually the same set of test cases will no longer be able to find any new bugs. 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.

**6) Testing is context dependent:** Testing is basically context dependent. Different kinds of sites are tested differently. For example, safety – critical software is tested differently from an e-commerce site.

**7) Absence – of – errors fallacy:** If the system built is unusable and does not fulfil the user’s needs and expectations then finding and fixing defects does not help.

# What is fundamental test process in software testing?

[**Testing**](http://tryqa.com/what-is-a-software-testing/) is a process rather than a single activity. This process starts from test planning then designing [**test cases**](http://tryqa.com/test-case/), preparing for execution and evaluating status till the test closure. So, we can divide the activities within the fundamental test process into the following basic steps:

1)    Planning and Control  
2)    Analysis and Design  
3)    Implementation and Execution  
4)    Evaluating exit criteria and Reporting  
5)    Test Closure activities

**1)    Planning and Control:**

[**Test planning**](http://tryqa.com/what-is-the-purpose-and-importance-of-test-plans/) has following major tasks:  
**i.**  To determine the scope and [**risks**](http://tryqa.com/what-is-risk-in-software-testing/)and identify the objectives of testing.  
**ii**. To determine the test approach.  
**iii.** To implement the test policy and/or the [**test strategy**](http://tryqa.com/what-are-the-test-approaches-or-strategies-in-software-testing/). (Test strategy is an outline that describes the testing portion of the [**software development cycle**](http://tryqa.com/what-are-the-software-development-life-cycle-sdlc-phases/). It is created to inform PM, testers and developers about some key issues of the testing process. This includes the testing objectives, method of testing, total time and resources required for the project and the testing environments.).  
**iv.** To determine the required test resources like people, test environments, PCs, etc.  
**v.** To schedule test analysis and design tasks, test implementation, execution and evaluation.  
**vi**. To determine the **Exit criteria**we need to set criteria such as **Coverage criteria.** (Coverage criteria are the percentage of statements in the software that must be executed during testing. This will help us track whether we are completing test activities correctly. They will show us which tasks and checks we must complete for a particular   level of testing before we can say that testing is finished.)

**Test control**has the following major tasks:  
**i.**  To measure and analyze the results of reviews and testing.  
**ii.**  To monitor and document progress,[**test coverage**](http://tryqa.com/what-is-test-coverage-in-software-testing-its-advantages-and-disadvantages/) and exit criteria.  
**iii**.  To provide information on testing.  
**iv.**  To initiate corrective actions.  
**v.**   To make decisions.

**2)  Analysis and Design:**

[**Test analysis**](http://tryqa.com/what-is-test-analysis-or-how-to-identify-the-test-conditions/)**and**[**Test Design**](http://tryqa.com/what-is-test-design-technique/)has the following major tasks:  
i.   To review the **test basis.** (The test basis is the information we need in order to start the test analysis and   create our own test cases. Basically it’s a documentation on which test cases are based, such as requirements, design specifications, product risk analysis, architecture and interfaces. We can use the test basis documents to understand what the system should do once built.)  
**ii.**   To identify test conditions.  
**iii.**  To design the tests.  
**iv**.  To evaluate testability of the requirements and system.  
v.  To design the test environment set-up and identify and required infrastructure and tools.

**3)  Implementation and Execution:**  
During test implementation and execution, we take the test conditions into **test cases**and procedures and other **testware** such as scripts for automation, the test environment and any other test infrastructure. (Test cases is a set of conditions under which a tester will determine whether an   application is working correctly or not.)  
(Testware is a term for all utilities that serve in combination for testing a software like scripts, the test environment and any other test infrastructure for later reuse.)

**Test implementation** has the following major task:  
**i.**  To develop and prioritize our test cases by using techniques and create **test data**for those tests. (In order to test a software application you need to enter some data for testing most of the features. Any such specifically identified data which is used in tests is known as test data.)  
We also write some instructions for carrying out the tests which is known as **test procedures.**  
We may also need to automate some tests using [**test harness**](http://tryqa.com/what-is-test-harness-unit-test-framework-tools-in-software-testing/) and automated tests scripts. (A test harness is a collection of software and test data for testing a program unit by running it under different conditions and monitoring its behavior and outputs.)  
**ii.** To create test suites from the test cases for efficient test execution.  
(Test suite is a collection of test cases that are used to test a software program   to show that it has some specified set of behaviours. A test suite often contains detailed instructions and information for each collection of test cases on the system configuration to be used during testing. Test suites are used to group similar test cases together.)  
**iii.** To implement and verify the environment.

**Test execution** has the following major task:  
**i.** To execute test suites and individual test cases following the test procedures.  
**ii.** To re-execute the tests that previously failed in order to confirm a fix. This is known as **confirmation testing or**[**re-testing**](http://tryqa.com/what-is-retesting/)**.**  
**iii.** To log the outcome of the test execution and record the identities and versions of the software under tests. The **test log** is used for the audit trial. (A test log is nothing but, what are the test cases that we executed, in what order we executed, who executed that test cases and what is the status of the test case (pass/fail). These descriptions are documented and called as test log.).  
**iv.** To Compare actual results with expected results.  
**v.** Where there are differences between actual and expected results, it report discrepancies as Incidents.

**4)  Evaluating Exit criteria and Reporting:**  
Based on the risk assessment of the project we will set the criteria for each test level against which we will measure the “enough testing”. These criteria vary from project to project and are known as **exit criteria**.  
Exit criteria come into picture, when:  
— Maximum test cases are executed with certain pass percentage.  
— Bug rate falls below certain level.  
— When achieved the deadlines.

**Evaluating exit criteria** has the following major tasks:  
i.  To check the test logs against the exit criteria specified in test planning.  
ii.  To assess if more test are needed or if the exit criteria specified should be changed.  
iii.  To write a test summary report for stakeholders.

**5)  Test Closure activities:**  
Test closure activities are done when software is delivered. The testing can be closed for the other reasons also like:

* When all the information has been gathered which are needed for the testing.
* When a project is cancelled.
* When some target is achieved.
* When a maintenance release or update is done.

**Test closure activities** have the following major tasks:  
i.  To check which planned deliverables are actually delivered and to ensure that all incident reports have been resolved.  
ii. To finalize and archive testware such as scripts, test environments, etc. for later reuse.  
iii. To handover the testware to the maintenance organization. They will give support to the software.  
iv To evaluate how the testing went and learn lessons for future releases and projects.

# What is Software Quality?

Quality software is reasonably [**bug or defect**](http://tryqa.com/what-is-defect-or-bugs-or-faults-in-software-testing/) free, delivered on time and within budget, meets requirements and/or expectations, and is maintainable.

ISO 8402-1986 standard defines quality as  “the totality of features and characteristics of a product or service that bears its ability to satisfy stated or implied needs.”

Key aspects of quality for the customer include:

* Good design – looks and style
* Good functionality – it does the job well
* Reliable – acceptable level of breakdowns or failure
* Consistency
* Durable – lasts as long as it should
* Good after sales service
* Value for money

**Good design – looks and style:**

It is very important to have a good design. The application or product should meet all the requirement specifications and at the same time it should be user friendly. The customers are basically attracted by the good looks and style of the application. The right color combinations, font size and the styling of the texts and buttons are very important.

**Good functionality – it does the job well:**

Along with the good looks of the application or the product it’s very important that the functionality should be intact. All the features and their functionality should work as expected. There should not be any deviation in the actual result and the expected result.

**Reliable – acceptable level of breakdowns or failure:**

After we have tested for all the features and their functionalities it also very important that the application or product should be reliable. For example: There is an application of saving the students records. This application should save all the students records and should not fail after entering 100 records. This is called reliability.

**Consistency:**

The software should have consistency across the application or product. Single software can be multi dimensional. It is very important that all the different dimensions should behave in a consistent manner.

**Durable – lasts as long as it should:**

The software should be durable. For example if software is being used for a year and the number of data has exceed 5000 records then it should not fail if number of records increases. The software product or application should continue to behave in the same way without any functional breaks.

**Good after sales service:**

Once the product is shipped to the customers then maintenance comes into the picture. It is very important to provide good sales services to keep the customers happy and satisfied. For example if after using the product for six months the customer realizes to make some changes to the application then those changes should be done as fast as possible and should be delivered to the customers on time with quality.

**Value for money:**

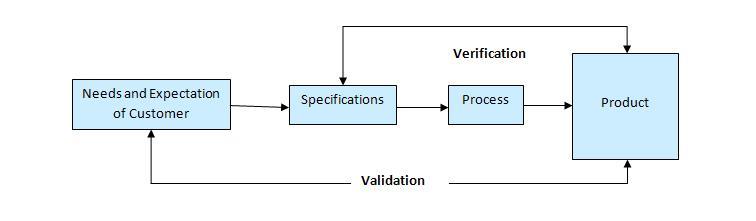
It’s always important to deliver the product to the customers which have value for money. The product should meet the requirement specifications. It should work as expected, should be user friendly. We should provide good services to the customers. Other than the features mentioned in the requirement specifications some additional functionality could be given to the customers which they might not have thought of. These additional functionalities should make their product more user friendly and easy to use. This also adds value for money.

**Chapter 2. Testing throughout the testing lifecycle**

What is Verification in software testing? or What is software verification?

**Verification** makes sure that the product is designed to deliver all functionality to the customer.

* Verification is done at the starting of the development process. It includes [**reviews**](http://tryqa.com/what-are-the-types-of-review/) and meetings, [**walk-throughs**](http://tryqa.com/what-is-walkthrough-in-software-testing/), [**inspection**](http://tryqa.com/what-is-inspection-in-software-testing/), etc. to evaluate documents, plans, code, requirements and specifications.
* Suppose you are building a table. Here the verification is about checking all the parts of the table, whether all the four legs are of correct size or not. If one leg of table is not of the right size it will imbalance the end product. Similar behavior is also noticed in case of the software product or application. If any feature of software product or application is not up to the mark or if any [**defect**](http://tryqa.com/what-is-defect-or-bugs-or-faults-in-software-testing/) is found then it will [**result into the failure**](http://tryqa.com/from-where-do-defects-and-failures-in-software-testing-arise/) of the end product. Hence, verification is very important. It takes place at the starting of the development process.

* Software verification and validation
* It answers the questions like: **Am I building the product right?**
* Am I accessing the data right (in the right place; in the right way).
* It is a Low level activity
* Performed during development on key artifacts, like walkthroughs, reviews and inspections, mentor feedback, training, checklists and standards.
* Demonstration of consistency, completeness, and correctness of the software at each stage and between each stage of the development life cycle.

According to the [**Capability Maturity Model (CMM)**](http://tryqa.com/what-is-cmm-capability-maturity-model-what-are-cmm-levels/) we can also define verification as the process of evaluating software to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase. [IEEE-STD-610].

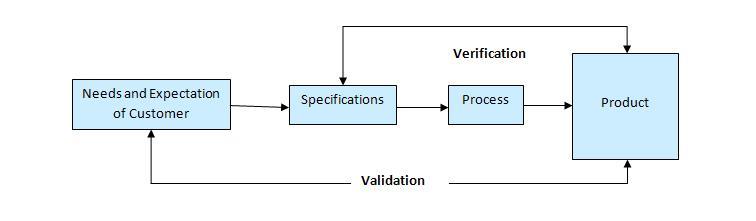
**Advantages of Software Verification :**

1. Verification helps in lowering down the count of the defect in the later stages of development.
2. Verifying the product at the starting phase of the development will help in understanding the product in a better way.
3. It reduces the chances of failures in the software application or product.
4. It helps in building the product as per the customer specifications and needs.

# What is Validation in software testing? or What is software validation?

**Validation is determining** if the system complies with the requirements and performs functions for which it is intended and meets the organization’s goals and user needs.

* Validation is done at the end of the development process and takes place after [**verifications**](http://tryqa.com/what-is-verification-in-software-testing-or-what-is-software-verification/) are completed.
* It answers the question like: **Am I building the right product?**
* Am I accessing the right data (in terms of the data required to satisfy the requirement).
* It is a High level activity.
* Performed after a work product is produced against established criteria ensuring that the product integrates correctly into the environment.
* Determination of correctness of the final software product by a development project with respect to the user needs and requirements.

Software verification and validation

According to the [**Capability Maturity Model (CMM)**](http://tryqa.com/what-is-cmm-capability-maturity-model-what-are-cmm-levels/) we can also define validation as The process of evaluating software during or at the end of the development process to determine whether it satisfies specified requirements. [IEEE-STD-610].

A product can pass while verification, as it is done on the paper and no running or functional application is required. But, when same points which were verified on the paper is actually developed then the running application or product can fail while validation. This may happen because when a product or application is build as per the specification but these specifications are not up to the mark hence they fail to address the user requirements.

**Advantages of Validation:**

1. During verification if some [**defects**](http://tryqa.com/what-is-defect-or-bugs-or-faults-in-software-testing/) are missed then during validation process it can be caught as failures.
2. If during verification some specification is misunderstood and development had happened then during validation process while executing that functionality the difference between the actual result and expected result can be understood.
3. Validation is done during testing like feature testing, integration testing, system testing, load testing, compatibility testing, stress testing, etc.
4. Validation helps in building the right product as per the customer’s requirement and helps in satisfying their needs.

Validation is basically done by the testers during the testing. While validating the product if some deviation is found in the actual result from the expected result then a bug is reported or an [**incident is raised**](http://tryqa.com/what-is-an-incident-in-software-testing/). Not all incidents are bugs. But all bugs are incidents. Incidents can also be of type ‘Question’ where the functionality is not clear to the tester.

Hence, validation helps in unfolding the exact functionality of the features and helps the testers to understand the product in much better way. It helps in making the product more user friendly.

# What is Capability Maturity Model (CMM)? What are CMM Levels?

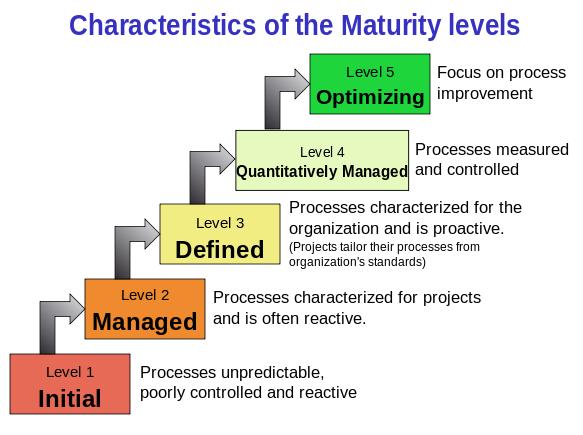
Capability Maturity Model is a bench-mark for measuring the maturity of an organization’s software process. It is a methodology used to develop and refine an organization’s [**software development**](http://tryqa.com/what-are-the-software-development-models/) process. CMM can be used to assess an organization against a scale of five process maturity levels based on certain Key Process Areas (KPA). It describes the maturity of the company based upon the project the company is dealing with and the clients. Each level ranks the organization according to its standardization of processes in the subject area being assessed.

A maturity model provides:

* A place to start
* The benefit of a community’s prior experiences
* A common language and a shared vision
* A framework for prioritizing actions
* A way to define what improvement means for your organization

In CMMI models with a staged representation, there are five maturity levels designated by the numbers 1 through 5 as shown below:

1. Initial
2. Managed
3. Defined
4. Quantitatively Managed
5. Optimizing

Maturity levels consist of a predefined set of process areas. The maturity levels are measured by the achievement of the **specific** and **generic goals** that apply to each predefined set of process areas. The following sections describe the characteristics of each maturity level in detail.

**Maturity Level 1 – Initial:***Company has no standard process for software development. Nor does it have a project-tracking system that enables developers to predict costs or finish dates with any accuracy.*

In detail we can describe it as given below:

* At maturity level 1, processes are usually ad hoc and chaotic.
* The organization usually does not provide a stable environment. Success in these organizations depends on the competence and [**heroics of the people in the organization**](http://tryqa.com/what-are-the-roles-and-responsibilities-of-a-tester/) and not on the use of proven processes.
* Maturity level 1 organizations often produce products and services that work but company has no standard process for software development. Nor does it have a project-tracking system that enables developers to predict costs or finish dates with any accuracy.
* Maturity level 1 organizations are characterized by a tendency to over commit, abandon processes in the time of crisis, and not be able to repeat their past successes.

**Maturity Level 2 – Managed:***Company has installed basic software management processes and controls. But there is no consistency or coordination among different groups.*

In detail we can describe it as given below:

* At maturity level 2, an organization has achieved all the **specific**and **generic goals** of the maturity level 2 process areas. In other words, the projects of the organization have ensured that [**requirements are managed**](http://tryqa.com/what-is-requirements-management-tools/) and that [**processes are planned**](http://tryqa.com/what-is-the-purpose-and-importance-of-test-plans/), performed, measured, and controlled.
* The process discipline reflected by maturity level 2 helps to ensure that existing practices are retained during times of stress. When these practices are in place, projects are performed and managed according to their documented plans.
* At maturity level 2, requirements, processes, work products, and services are managed. The status of the work products and the delivery of services are visible to management at defined points.
* Commitments are established among relevant stakeholders and are revised as needed. Work products are reviewed with stakeholders and are controlled.
* The work products and services satisfy their specified requirements, standards, and objectives.

**Maturity Level 3 – Defined:***Company has pulled together a standard set of processes and controls for the entire organization so that developers can move between projects more easily and customers can begin to get consistency from different groups.*

In detail we can describe it as given below:

* At maturity level 3, an organization has achieved all the **specific**and **generic goals.**
* At maturity level 3, processes are well characterized and understood, and are described in standards, procedures, tools, and methods.
* A critical distinction between maturity level 2 and maturity level 3 is the scope of standards, process descriptions, and procedures. At maturity level 2, the standards, process descriptions, and procedures may be quite different in each specific instance of the process (for example, on a particular project). At maturity level 3, the standards, process descriptions, and procedures for a project are tailored from the organization’s set of standard processes to suit a particular project or organizational unit.
* The organization’s set of standard processes includes the processes addressed at maturity level 2 and maturity level 3. As a result, the processes that are performed across the organization are consistent except for the differences allowed by the tailoring guidelines.
* Another critical distinction is that at maturity level 3, processes are typically described in more detail and more rigorously than at maturity level 2.
* At maturity level 3, processes are managed more proactively using an understanding of the interrelationships of the process activities and detailed measures of the process, its work products, and its services.

**Maturity Level 4 – Quantitatively Managed:***In addition to implementing standard processes, company has installed systems to measure the quality of those processes across all projects.*

In detail we can describe it as given below:

* At maturity level 4, an organization has achieved all the **specific goals**of the process areas assigned to maturity levels 2, 3, and 4 and the **generic goals** assigned to maturity levels 2 and 3.
* At maturity level 4 Sub-processes are selected that significantly contribute to overall process performance. These selected sub-processes are controlled using statistical and other quantitative techniques.
* Quantitative objectives for quality and process performance are established and used as criteria in managing processes. Quantitative objectives are based on the needs of the customer, end users, organization, and process implementers. Quality and process performance are understood in statistical terms and are managed throughout the life of the processes.
* For these processes, detailed measures of process performance are collected and statistically analyzed. Special causes of process variation are identified and, where appropriate, the sources of special causes are corrected to prevent future occurrences.
* Quality and process performance measures are incorporated into the organizations measurement repository to support fact-based decision making in the future.
* A critical distinction between maturity level 3 and maturity level 4 is the predictability of process performance. At maturity level 4, the performance of processes is controlled using statistical and other quantitative techniques, and is quantitatively predictable. At maturity level 3, processes are only qualitatively predictable.

**Maturity Level 5 – Optimizing:***Company has accomplished all of the above and can now begin to see patterns in performance over time, so it can tweak its processes in order to improve productivity and reduce defects in software development across the entire organization.*

In detail we can describe it as given below:

* At maturity level 5, an organization has achieved all the **specific goals** of the process areas assigned to maturity levels 2, 3, 4, and 5 and the **generic goals** assigned to maturity levels 2 and 3.
* Processes are continually improved based on a quantitative understanding of the common causes of variation inherent in processes.
* Maturity level 5 focuses on continually improving process performance through both incremental and innovative technological improvements.
* Quantitative process-improvement objectives for the organization are established, continually revised to reflect changing business objectives, and used as criteria in managing process improvement.
* The effects of deployed process improvements are measured and evaluated against the quantitative process-improvement objectives. Both the defined processes and the organization’s set of standard processes are targets of measurable improvement activities.
* Optimizing processes that are agile and innovative depends on the participation of an empowered workforce aligned with the business values and objectives of the organization.
* The organization’s ability to rapidly respond to changes and opportunities is enhanced by finding ways to accelerate and share learning. Improvement of the processes is inherently part of everybody’s role, resulting in a cycle of continual improvement.
* A critical distinction between maturity level 4 and maturity level 5 is the type of process variation addressed. At maturity level 4, processes are concerned with addressing special causes of process variation and providing statistical predictability of the results. Though processes may produce predictable results, the results may be insufficient to achieve the established objectives. At maturity level 5, processes are concerned with addressing common causes of process variation and changing the process (that is, shifting the mean of the process performance) to improve process performance (while maintaining statistical predictability) to achieve the established quantitative process-improvement objectives.

# What are the Software Development Life Cycle (SDLC) phases?

There are various software development approaches defined and designed which are used/employed during development process of software, these approaches are also referred as “Software Development Process Models” (e.g. [**Waterfall model**](http://tryqa.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/), [**incremental model**](http://tryqa.com/what-is-incremental-model-advantages-disadvantages-and-when-to-use-it/), [**V-model**](http://tryqa.com/what-is-v-model-advantages-disadvantages-and-when-to-use-it/), [**iterative model**](http://tryqa.com/what-is-iterative-model-advantages-disadvantages-and-when-to-use-it/), [**RAD model**](http://tryqa.com/what-is-rad-model-advantages-disadvantages-and-when-to-use-it/), [**Agile model**](http://tryqa.com/what-is-agile-model-advantages-disadvantages-and-when-to-use-it/), [**Spiral model**](http://tryqa.com/what-is-spiral-model-advantages-disadvantages-and-when-to-use-it/), [**Prototype model**](http://tryqa.com/what-is-prototype-model-advantages-disadvantages-and-when-to-use-it/) etc.). Each process model follows a particular life cycle in order to ensure success in process of software development.

Software life cycle models describe phases of the software cycle and the order in which those phases are executed. Each phase produces deliverables required by the next phase in the life cycle. Requirements are translated into design. Code is produced according to the design which is called development phase. After coding and development the testing verifies the deliverable of the implementation phase against requirements. The testing team follows [**Software Testing Life Cycle (STLC)**](http://tryqa.com/what-is-software-testing-life-cycle-stlc/) which is similar to the development cycle followed by the development team.

There are following six phases in every Software development life cycle model:

1. Requirement gathering and analysis
2. Design
3. Implementation or coding
4. Testing
5. Deployment
6. Maintenance

**1) Requirement gathering and analysis:**  Business requirements are gathered in this phase. This phase is the main focus of the project managers and stake holders. Meetings with managers, stake holders and users are held in order to determine the requirements like; Who is going to use the system? How will they use the system?  What data should be input into the system?  What data should be output by the system?  These are general questions that get answered during a requirements gathering phase. After requirement gathering these requirements are analyzed for their validity and the possibility of incorporating the requirements in the system to be development is also studied.

Finally, a Requirement Specification document is created which serves the purpose of guideline for the next phase of the model. The testing team follows the Software Testing Life Cycle and starts the [**Test Planning**](http://tryqa.com/what-is-the-purpose-and-importance-of-test-plans/) phase after the requirements analysis is completed.

**2)  Design:**  In this phase the system and software design is prepared from the requirement specifications which were studied in the first phase. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture. The system design specifications serve as input for the next phase of the model.

In this phase the testers comes up with the [**Test strategy**](http://tryqa.com/what-are-the-test-approaches-or-strategies-in-software-testing/), where they mention what to test, how to test.

**3)  Implementation / Coding:**  On receiving system design documents, the work is divided in modules/units and actual coding is started. Since, in this phase the code is produced so it is the main focus for the developer. This is the longest phase of the software development life cycle.

**4)**[**Testing**](http://tryqa.com/what-is-a-software-testing/)**:**  After the code is developed it is tested against the requirements to make sure that the product is actually solving the needs addressed and gathered during the requirements phase. During this phase all types of [**functional testing**](http://tryqa.com/what-is-functionality-testing-in-software/) like [**unit testing**](http://tryqa.com/what-is-unit-testing/), [**integration testing**](http://tryqa.com/what-is-integration-testing/), [**system testing**](http://tryqa.com/what-is-system-testing/), [**acceptance testing**](http://tryqa.com/what-is-acceptance-testing/) are done as well as [**non-functional testing**](http://tryqa.com/what-is-non-functional-testing-testing-of-software-product-characteristics/) are also done.

**5)  Deployment:** After successful testing the product is delivered / deployed to the customer for their use.

As soon as the product is given to the customers they will first do the [**beta testing**](http://tryqa.com/what-is-beta-testing/). If any changes are required or if any bugs are caught, then they will report it to the engineering team. Once those changes are made or the [**bugs**](http://tryqa.com/what-is-defect-or-bugs-or-faults-in-software-testing/) are fixed then the final deployment will happen.

**6) Maintenance:** Once when the customers starts using the developed system then the actual problems comes up and needs to be solved from time to time. This process where the care is taken for the developed product is known as maintenance.

# What are the Software Development Models?

The **software development models** are the various processes or methodologies that are being selected for the development of the project depending on the project’s aims and goals. There are many development life cycle models that have been developed in order to achieve different required objectives. The models specify the various stages of the process and the order in which they are carried out.

The selection of model has very high impact on the testing that is carried out. It will define the what, where and when of our planned testing, influence regression testing and largely determines which test techniques to use.

There are various Software development models or methodologies. They are as follows:

1. [**Waterfall model**](http://tryqa.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/)
2. [**V model**](http://tryqa.com/what-is-v-model-advantages-disadvantages-and-when-to-use-it/)
3. [**Incremental model**](http://tryqa.com/what-is-incremental-model-advantages-disadvantages-and-when-to-use-it/)
4. [**RAD model**](http://tryqa.com/what-is-rad-model-advantages-disadvantages-and-when-to-use-it/)
5. [**Agile model**](http://tryqa.com/what-is-agile-model-advantages-disadvantages-and-when-to-use-it/)
6. [**Iterative model**](http://tryqa.com/what-is-iterative-model-advantages-disadvantages-and-when-to-use-it/)
7. [**Spiral model**](http://tryqa.com/what-is-spiral-model-advantages-disadvantages-and-when-to-use-it/)
8. [**Prototype model**](http://tryqa.com/what-is-prototype-model-advantages-disadvantages-and-when-to-use-it/)

Choosing right model for developing of the software product or application is very important. Based on the model the development and testing processes are carried out.

Different companies based on the software application or product, they select the type of development model whichever suits to their application. But these days in market the ‘**Agile Methodology**‘ is the most used model. ‘**Waterfall Model**‘ is the very old model. In ‘Waterfall Model’ testing starts only after the development is completed. Because of which there are many defects and failures which are reported at the end. So,the cost of fixing these issues are high. Hence, these days people are preferring ‘Agile Model’. In ‘Agile Model’ after every sprint there is a demo-able feature to the customer. Hence customer can see the features whether they are satisfying their need or not.

‘**V-model**‘ is also used by many of the companies in their product. ‘V-model’ is nothing but ‘Verification’ and ‘Validation’ model. In ‘V-model’ the developer’s life cycle and tester’s life cycle are mapped to each other. In this model testing is done side by side of the development.

Likewise ‘Incremental model’, ‘RAD model’, ‘Iterative model’ and ‘Spiral model’ are also used based on the requirement of the customer and need of the product.

Start learning about the models with [**Waterfall model**](http://tryqa.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/) and its advantages and disadvantages.

The Waterfall Model was first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**.  It is very simple to understand and use.  In a waterfall model, each phase must be completed fully before the next phase can begin. This type of [**software development model**](http://tryqa.com/what-are-the-software-development-models/) is basically used for the for the project which is small and there are no uncertain requirements. At the end of each phase, a review takes place to determine if the project is on the right path and whether or not to continue or discard the project. In this model [**software testing**](http://tryqa.com/what-is-a-software-testing/) starts only after the development is complete. In **waterfall model phases** do not overlap.

**Diagram of Waterfall-model:**



**Advantages of waterfall model:**

* This model is simple and easy to understand and use.
* It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
* In this model phases are processed and completed one at a time. Phases do not overlap.
* Waterfall model works well for smaller projects where requirements are very well understood.

**Disadvantages of waterfall model:**

* Once an application is in the [**testing**](http://tryqa.com/what-is-a-software-testing/) stage, it is very difficult to go back and change something that was not well-thought out in the concept stage.
* No working software is produced until late during the life cycle.
* High amounts of risk and uncertainty.
* Not a good model for complex and object-oriented projects.
* Poor model for long and ongoing projects.
* Not suitable for the projects where requirements are at a moderate to high risk of changing.

**When to use the waterfall model:**

* This model is used only when the requirements are very well known, clear and fixed.
* Product definition is stable.
* Technology is understood.
* There are no ambiguous requirements
* Ample resources with required expertise are available freely
* The project is short.

Very less customer interaction is involved during the development of the product. Once the product is ready then only it can be demoed to the end users. Once the product is developed and if any failure occurs then the cost of fixing such issues are very high, because we need to update everywhere from document till the logic.

# What is V-model- advantages, disadvantages and when to use it?

V- model means Verification and Validation model. Just like the [**waterfall model**](http://tryqa.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/), the V-Shaped life cycle is a sequential path of execution of processes. Each phase must be completed before the next phase begins. **V-Model** is one of the [**many software development models**](http://tryqa.com/what-are-the-software-development-models/).Testing of the product is planned in parallel with a corresponding phase of development in **V-model**.

**Diagram of V-model:**



The various phases of the V-model are as follows:

**Requirements** like BRS and SRS begin the life cycle model just like the waterfall model. But, in this model before development is started, a [**system test**](http://tryqa.com/what-is-system-testing/) plan is created.  The [**test plan**](http://tryqa.com/what-is-the-purpose-and-importance-of-test-plans/) focuses on meeting the functionality specified in the requirements gathering.

**The high-level design (HLD)** phase focuses on system architecture and design. It provide overview of solution, platform, system, product and service/process. An [**integration test**](http://tryqa.com/what-is-integration-testing/) plan is created in this phase as well in order to test the pieces of the software systems ability to work together.

**The low-level design** **(LLD)** phase is where the actual software components are designed. It defines the actual logic for each and every component of the system. Class diagram with all the methods and relation between classes comes under LLD. [**Component tests**](http://tryqa.com/what-is-component-testing/) are created in this phase as well.

**The implementation** phase is, again, where all coding takes place. Once coding is complete, the path of execution continues up the right side of the V where the test plans developed earlier are now put to use.

**Coding:** This is at the bottom of the V-Shape model. Module design is converted into code by developers. [**Unit Testing**](http://tryqa.com/what-is-unit-testing/) is performed by the developers on the code written by them.

**Advantages of V-model:**

* Simple and easy to use.
* Testing activities like planning, [**test designing**](http://tryqa.com/what-is-test-design-or-how-to-specify-test-cases/) happens well before coding. This saves a lot of time. Hence higher chance of success over the waterfall model.
* Proactive defect tracking – that is defects are found at early stage.
* Avoids the downward flow of the defects.
* Works well for small projects where requirements are easily understood.

**Disadvantages of V-model:**

* Very rigid and least flexible.
* Software is developed during the implementation phase, so no early prototypes of the software are produced.
* If any changes happen in midway, then the test documents along with requirement documents has to be updated.

**When to use the V-model:**

* The V-shaped model should be used for small to medium sized projects where requirements are clearly defined and fixed.
* The V-Shaped model should be chosen when ample technical resources are available with needed technical expertise.

High confidence of customer is required for choosing the V-Shaped model approach. Since, no prototypes are produced, there is a very high risk involved in meeting customer expectations.

# What is Incremental model- advantages, disadvantages and when to use it?

In incremental model the whole requirement is divided into various builds. Multiple development cycles take place here, making the life cycle a [**“multi-waterfall” cycle**](http://tryqa.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/).  Cycles are divided up into smaller, more easily managed modules. Incremental model is a type of software development model like [**V-model**](http://tryqa.com/what-is-v-model-advantages-disadvantages-and-when-to-use-it/), [**Agile model**](http://tryqa.com/what-is-agile-model-advantages-disadvantages-and-when-to-use-it/) etc.

In this model, each module passes through the requirements, design, implementation and [**testing**](http://tryqa.com/what-is-a-software-testing/) phases. A working version of software is produced during the first module, so you have working software early on during the [**software life cycle**](http://tryqa.com/what-are-the-software-development-life-cycle-phases/). Each subsequent release of the module adds function to the previous release. The process continues till the complete system is achieved.

For example:

In the diagram above when we work **incrementally**we are adding piece by piece but expect that each piece is fully finished. Thus keep on adding the pieces until it’s complete. As in the image above a person has thought of the application. Then he started building it and in the first iteration the first module of the application or product is totally ready and can be demoed to the customers. Likewise in the second iteration the other module is ready and integrated with the first module. Similarly, in the third iteration the whole product is ready and integrated. Hence, the product got ready step by step.

**Diagram of Incremental model:**



**Advantages of Incremental model:**

* Generates working software quickly and early during the software life cycle.
* This model is more flexible – less costly to change scope and requirements.
* It is easier to test and debug during a smaller iteration.
* In this model customer can respond to each built.
* Lowers initial delivery cost.
* Easier to manage risk because risky pieces are identified and handled during it’d iteration.

**Disadvantages of Incremental model:**

* Needs good planning and design.
* Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.
* Total cost is higher than [**waterfall**](http://tryqa.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/).

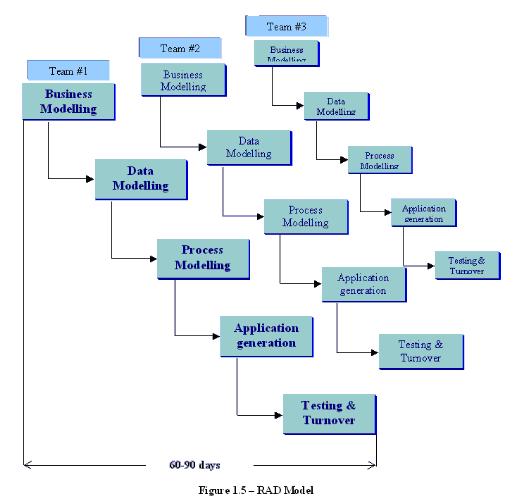
**When to use the Incremental model:**

* This model can be used when the requirements of the complete system are clearly defined and understood.
* Major requirements must be defined; however, some details can evolve with time.
* There is a need to get a product to the market early.
* A new technology is being used
* Resources with needed skill set are not available
* There are some high risk features and goals.

# What is RAD model- advantages, disadvantages and when to use it?

RAD model is Rapid Application Development model. It is a type of [**incremental model**](http://tryqa.com/what-is-incremental-model-advantages-disadvantages-and-when-to-use-it/). In RAD model the components or functions are developed in parallel as if they were mini projects. The developments are time boxed, delivered and then assembled into a working prototype.  This can quickly give the customer something to see and use and to provide feedback regarding the delivery and their requirements.

**Diagram of RAD-Model:**



The phases in the rapid application development (RAD) model are:

**Business modeling:** The information flow is identified between various business functions.  
**Data modeling:** Information gathered from business modeling is used to define data objects that are needed for the business.  
**Process modeling:** Data objects defined in data modeling are converted to achieve the business information flow to achieve some specific business objective. Description are identified and created for CRUD of data objects.  
**Application generation:** Automated tools are used to convert process models into code and the actual system.  
**Testing and turnover:** Test new components and all the interfaces.

**Advantages of the RAD model:**

* Reduced development time.
* Increases reusability of components
* Quick initial reviews occur
* Encourages customer feedback
* Integration from very beginning solves a lot of [**integration issues**](http://tryqa.com/what-is-system-integration-testing/).

**Disadvantages of RAD model:**

* Depends on strong team and individual performances for identifying business requirements.
* Only system that can be modularized can be built using RAD
* Requires highly skilled developers/designers.
* High dependency on modeling skills
* Inapplicable to cheaper projects as cost of modeling and automated code generation is very high.

**When to use RAD model:**

* RAD should be used when there is a need to create a system that can be modularized in 2-3 months of time.
* It should be used if there’s high availability of designers for modeling and the budget is high enough to afford their cost along with the cost of automated code generating tools.
* RAD [**SDLC model**](http://tryqa.com/what-are-the-software-development-models/) should be chosen only if resources with high business knowledge are available and there is a need to produce the system in a short span of time (2-3 months).

# What is Agile model – advantages, disadvantages and when to use it?

**Agile development model** is also a type of [**Incremental model**](http://tryqa.com/what-is-incremental-model-advantages-disadvantages-and-when-to-use-it/). Software is developed in incremental, rapid cycles. This results in small incremental releases with each release building on previous functionality. Each release is thoroughly [**tested**](http://tryqa.com/why-is-testing-necessary/) to ensure [**software quality**](http://tryqa.com/what-is-software-quality/) is maintained. It is used for time critical applications.  Extreme Programming (XP) is currently one of the most well known agile [**development life cycle model**](http://tryqa.com/what-are-the-software-development-models/).

**Diagram of Agile model:**



**Advantages of Agile model:**

* Customer satisfaction by rapid, continuous delivery of useful software.
* People and interactions are emphasized rather than process and tools. Customers, developers and testers constantly interact with each other.
* Working software is delivered frequently (weeks rather than months).
* Face-to-face conversation is the best form of communication.
* Close, daily cooperation between business people and developers.
* Continuous attention to technical excellence and good design.
* Regular adaptation to changing circumstances.
* Even late changes in requirements are welcomed

**Disadvantages of Agile model:**

* In case of some software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the software development life cycle.
* There is lack of emphasis on necessary designing and documentation.
* The project can easily get taken off track if the customer representative is not clear what final outcome that they want.
* Only senior programmers are capable of taking the kind of decisions required during the development process. Hence it has no place for newbie programmers, unless combined with experienced resources.

**When to use Agile model:**

* When new changes are needed to be implemented. The freedom agile gives to change is very important. New changes can be implemented at very little cost because of the frequency of new increments that are produced.
* To implement a new feature the developers need to lose only the work of a few days, or even only hours, to roll back and implement it.
* Unlike the [**waterfall model**](http://tryqa.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/) in agile model very limited [**planning**](http://tryqa.com/what-is-the-purpose-and-importance-of-test-plans/) is required to get started with the project. Agile assumes that the end users’ needs are ever changing in a dynamic business and IT world. Changes can be discussed and features can be newly effected or removed based on feedback. This effectively gives the customer the finished system they want or need.
* Both system developers and stakeholders alike, find they also get more freedom of time and options than if the software was developed in a more rigid sequential way. Having options gives them the ability to leave important decisions until more or better data or even entire hosting programs are available; meaning the project can continue to move forward without fear of reaching a sudden standstill.

You can refer to our introduction to [**Agile Methodology**](http://tryqa.com/what-is-agile-methodology-examples-when-to-use-it-advantages-and-disadvantages/) if you would like to understand Agile better however, the above information is sufficient for the ISTQB Foundation Level exam.

# What is Iterative model- advantages, disadvantages and when to use it?

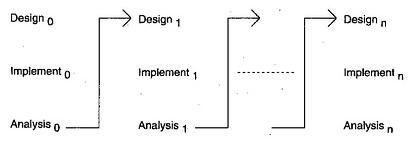
An iterative [**life cycle model**](http://tryqa.com/what-are-the-software-development-models/) does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which can then be reviewed in order to identify further requirements. This process is then repeated, producing a new version of the software for each cycle of the model.

For example:



In the diagram above when we work **iteratively**we create rough product or product piece in one iteration, then review it and improve it in next iteration and so on until it’s finished. As shown in the image above, in the first iteration the whole painting is sketched roughly, then in the second iteration colors are filled and in the third iteration finishing is done. Hence, in iterative model the whole product is developed step by step.

**Diagram of Iterative model:**



**Advantages of Iterative model:**

* In iterative model we can only create a high-level design of the application before we actually begin to build the product and define the design solution for the entire product. Later on we can design and built a skeleton version of that, and then evolved the design based on what had been built.
* In iterative model we are building and improving the product step by step. Hence we can track the defects at early stages. This avoids the downward flow of the defects.
* In iterative model we can get the reliable user feedback. When presenting sketches and blueprints of the product to users for their feedback, we are effectively asking them to imagine how the product will work.
* In iterative model less time is spent on documenting and more time is given for designing.

**Disadvantages of Iterative model:**

* Each phase of an iteration is rigid with no overlaps
* Costly system architecture or design issues may arise because not all requirements are gathered up front for the entire lifecycle

**When to use iterative model:**

* Requirements of the complete system are clearly defined and understood.
* When the project is big.
* Major requirements must be defined; however, some details can evolve with time.

# What is Spiral model- advantages, disadvantages and when to use it?

The spiral model is similar to the [**incremental model**](http://tryqa.com/what-is-incremental-model-advantages-disadvantages-and-when-to-use-it/), with more emphasis placed on risk analysis. The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation. A software project repeatedly passes through these phases in iterations (called Spirals in this model). The baseline spiral, starting in the planning phase, requirements are gathered and risk is assessed. Each subsequent spirals builds on the baseline spiral. Its one of the [**software development models**](http://tryqa.com/what-are-the-software-development-models/) like [**Waterfall**](http://tryqa.com/what-is-waterfall-model-advantages-disadvantages-and-when-to-use-it/), [**Agile**](http://tryqa.com/what-is-agile-model-advantages-disadvantages-and-when-to-use-it/), [**V-Model**](http://tryqa.com/what-is-v-model-advantages-disadvantages-and-when-to-use-it/).

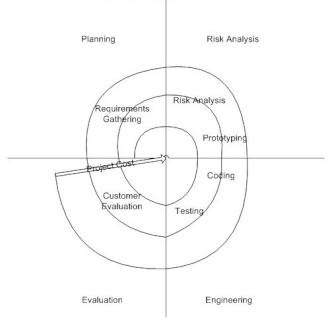
**Planning Phase:**Requirements are gathered during the planning phase. Requirements like ‘BRS’ that is ‘Bussiness Requirement Specifications’ and ‘SRS’ that is ‘System Requirement specifications’.

**Risk Analysis:** In the**risk analysis phase**, a process is undertaken to identify risk and alternate solutions.  A prototype is produced at the end of the risk analysis phase. If any risk is found during the risk analysis then alternate solutions are suggested and implemented.

**Engineering Phase:** In this phase software is **developed**, along with [**testing**](http://tryqa.com/what-is-a-software-testing/) at the end of the phase. Hence in this phase the development and testing is done.

E**valuation phase:**This phase allows the customer to evaluate the output of the project to date before the project continues to the next spiral.

**Diagram of Spiral model:**

**[](http://tryqa.com/wp-content/uploads/2012/01/Spiral-model.jpg)**

**Advantages of Spiral model:**

* High amount of risk analysis hence, avoidance of Risk is enhanced.
* Good for large and mission-critical projects.
* Strong approval and documentation control.
* Additional Functionality can be added at a later date.
* Software is produced early in the [**software life cycle**](http://tryqa.com/what-are-the-software-development-life-cycle-phases/).

**Disadvantages of Spiral model:**

* Can be a costly model to use.
* Risk analysis requires highly specific expertise.
* Project’s success is highly dependent on the risk analysis phase.
* Doesn’t work well for smaller projects.

**When to use Spiral model:**

* When costs and risk evaluation is important
* For medium to high-risk projects
* Long-term project commitment unwise because of potential changes to economic priorities
* Users are unsure of their needs
* Requirements are complex
* New product line
* Significant changes are expected (research and exploration)

# What is Prototype model- advantages, disadvantages and when to use it?

The basic idea in **Prototype model** is that instead of freezing the requirements before a design or coding can proceed, a throwaway prototype is built to understand the requirements. This prototype is developed based on the currently known requirements. Prototype model is a [**software development model**](http://tryqa.com/what-are-the-software-development-models/). By using this prototype, the client can get an “actual feel” of the system, since the interactions with prototype can enable the client to better understand the requirements of the desired system.  Prototyping is an attractive idea for complicated and large systems for which there is no manual process or existing system to help determining the requirements.

The prototype are usually not complete systems and many of the details are not built in the prototype. The goal is to provide a system with overall functionality.

**Diagram of Prototype model:**



**Advantages of Prototype model:**

* Users are actively involved in the development
* Since in this methodology a working model of the system is provided, the users get a better understanding of the system being developed.
* Errors can be detected much earlier.
* Quicker user feedback is available leading to better solutions.
* Missing functionality can be identified easily
* Confusing or difficult functions can be identified  
  Requirements validation, Quick implementation of, incomplete, but  
  functional, application.

**Disadvantages of Prototype model:**

* Leads to implementing and then repairing way of building systems.
* Practically, this methodology may increase the complexity of the system as scope of the system may expand beyond original plans.
* Incomplete application may cause application not to be used as the  
  full system was designed  
  Incomplete or inadequate problem analysis.

**When to use Prototype model:**

* Prototype model should be used when the desired system needs to have a lot of interaction with the end users.
* Typically, online systems, web interfaces have a very high amount of interaction with end users, are best suited for Prototype model. It might take a while for a system to be built that allows ease of use and needs minimal training for the end user.
* Prototyping ensures that the end users constantly work with the system and provide a feedback which is incorporated in the prototype to result in a useable system. They are excellent for designing good human computer interface systems.

# What are Software Testing Levels?

Testing levels are basically to identify missing areas and prevent overlap and repetition between the development life cycle phases. In software development life cycle models there are defined phases like requirement gathering and analysis, design, coding or implementation, testing and deployment.  Each phase goes through the testing. Hence there are various levels of testing. The various levels of testing are:

1. [**Unit testing:**](http://tryqa.com/what-is-unit-testing/) It is basically done by the developers to make sure that their code is working fine and meet the user specifications. They test their piece of code which they have written like classes, functions, interfaces and procedures.
2. [**Component testing:**](http://tryqa.com/what-is-component-testing/) It is also called as module testing. The basic difference between the unit testing and component testing is in unit testing the developers test their piece of code but in component testing the whole component is tested. For example, in a student record application there are two modules one which will save the records of the students and other module is to upload the results of the students. Both the modules are developed separately and when they are tested one by one then we call this as a component or module testing.
3. [**Integration testing:**](http://tryqa.com/what-is-integration-testing/) Integration testing is done when two modules are integrated, in order to test the behavior and functionality of both the modules after integration. Below are few types of integration testing:
   * [**Big bang integration testing**](http://tryqa.com/what-is-big-bang-integration-testing/)
   * Top down
   * Bottom up
   * Functional incremental (explained later under Integration Testing)

Sometimes there can be several levels of integration testing :

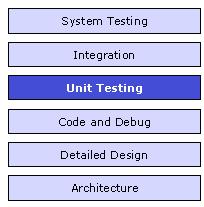
* + [**Component integration testing:**](http://tryqa.com/what-is-component-integration-testing/) In the example above when both the modules or components are integrated then the testing done is called as Component integration testing. This testing is basically done to ensure that the code should not break after integrating the two modules.
  + [**System integration testing:**](http://tryqa.com/what-is-system-integration-testing/) System integration testing (SIT) is a testing where testers basically test that in the same environment all the related systems should maintain data integrity and can operate in coordination with other systems.

1. [**System testing:**](http://tryqa.com/what-is-system-testing/) In system testing the testers basically test the compatibility of the application with the system. System integration testing may be performed after system testing or in parallel with system testing.
2. [**Acceptance testing:**](http://tryqa.com/what-is-acceptance-testing/) Acceptance testing are basically done to ensure that the requirements of the specification are met.
   * [**Alpha testing:**](http://tryqa.com/what-is-alpha-testing/) Alpha testing is done at the developers site. It is done at the end of the development process
   * [**Beta testing:**](http://tryqa.com/what-is-beta-testing/) Beta testing is done at the customers site. It is done just before the launch of the product.

# What is Unit testing?

A **unit test** is the smallest testable part of an application like functions, classes, procedures, interfaces. Unit testing is a method by which individual units of source code are tested to determine if they are fit for use.

* **Unit tests are basically written and executed by software developers** to make sure that code meets its design and requirements and behaves as expected.
* The goal of unit testing is to segregate each part of the program and test that the individual parts are working correctly.
* This means that for any function or procedure when a set of inputs are given then it should return the proper values. It should handle the failures gracefully during the course of execution when any invalid input is given.
* A unit test provides a written contract that the piece of code must assure. Hence it has several benefits.
* Unit testing is basically done before integration as shown in the image below.



**Method Used for unit testing:** White Box Testing method is used for executing the unit test.

**When Unit testing should be done?**

Unit testing should be done before Integration testing.

**By whom unit testing should be done?**

Unit testing should be done by the developers.

**Advantages of Unit testing:**

1. Issues are found at early stage. Since unit testing are carried out by developers where they test their individual code before the integration. Hence the issues can be found very early and can be resolved then and there without impacting the other piece of codes.

2. Unit testing helps in maintaining and changing the code. This is possible by making the codes less interdependent so that unit testing can be executed. Hence chances of impact of changes to any other code gets reduced.

3. Since the bugs are found early in unit testing hence it also helps in reducing the cost of bug fixes. Just imagine the cost of bug found during the later stages of development like during system testing or during acceptance testing.

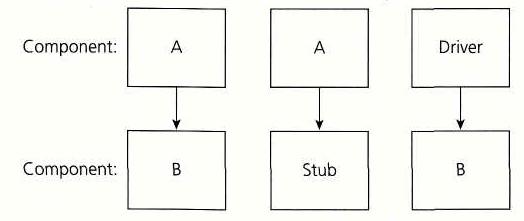
4. Unit testing helps in simplifying the debugging process. If suppose a test fails then only latest changes made in code needs to be debugged.

# What is Component testing?

**What is Component testing?**:Component testing is a method where testing of each component in an application is done separately.  Suppose, in an application there are 5 components. Testing of each 5 components separately and efficiently is called as component testing.

* Component testing is also known as module and program testing. It finds the defects in the module and verifies the functioning of software.
* Component testing is done by the tester.
* Component testing may be done in isolation from rest of the system depending on the development life cycle model chosen for that particular application. In such case the missing software is replaced by **Stubs** and **Drivers**and simulate the interface between the software components in a simple manner.
* Let’s take an example to understand it in a better way.Suppose there is an application consisting of three modules say, module A, module B and module C. The developer has developed the module B and now wanted to test it. But in order to test the module B completely few of it’s functionalities are dependent on module A and few on module C. But the module A and module C has not been developed yet.In that case to test the module B completely we can replace the module A and module C by stub and drivers as required.
* **Stub:**A stub is called from the software component to be tested. As shown in the diagram below ‘Stub’ is called by ‘component A’.
* **Driver:**A driver calls the component to be tested. As shown in the diagram below ‘component B’ is called by the ‘Driver’.

Below is the diagram of the component testing:



As discussed in the previous article of the ‘Unit testing’ it is done by the developers where they do the testing of the individual functionality or procedure. After unit testing is executed, component testing comes into the picture. Component testing is done by the testers.

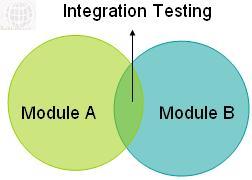
Component testing plays a very important role in finding the bugs. Before we start with the integration testing it’s always preferable to do the component testing in order to ensure that each component of an application is working effectively.

Integration testing is followed by the component testing.

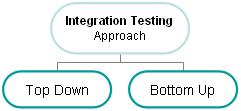
# What is Integration testing?

**Integration testing** tests integration or interfaces between components, interactions to different parts of the system such as an operating system, file system and hardware or interfaces between systems.

* Also after integrating two different [**components**](http://tryqa.com/what-is-component-testing/) together we do the integration [**testing**](http://tryqa.com/what-is-a-software-testing/). As displayed in the image below when two different modules ‘Module A’ and ‘Module B’ are integrated then the integration testing is done.

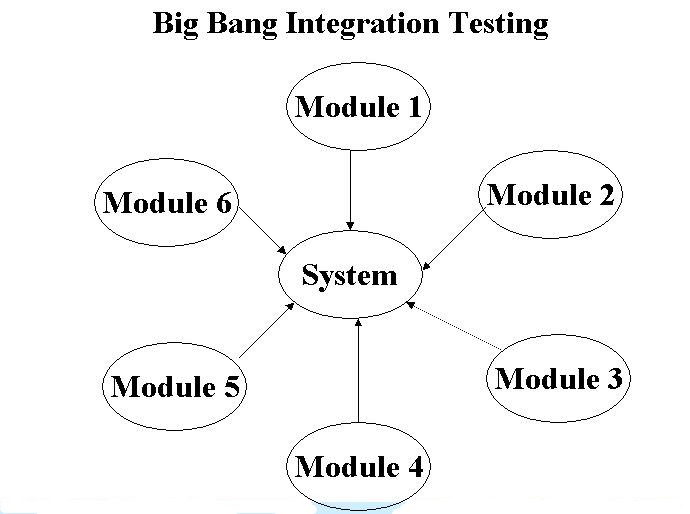


* Integration testing is done by a specific integration tester or test team.
* Integration testing follows two approach known as ‘Top Down’ approach and ‘Bottom Up’ approach as shown in the image below:

**[](http://tryqa.com/wp-content/uploads/2014/09/Integration_testing.jpg)**

Below are the integration testing techniques:

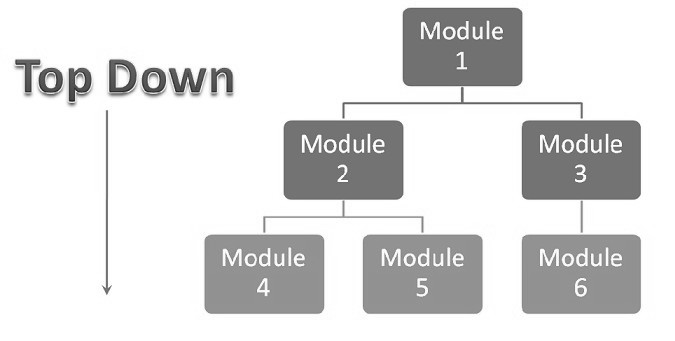
**1.**[**Big Bang integration testing**](http://tryqa.com/what-is-big-bang-integration-testing/)**:**

In Big Bang integration testing all components or modules are integrated simultaneously, after which everything is tested as a whole. As per the below image all the modules from ‘Module 1’ to ‘Module 6’ are integrated simultaneously then the testing is carried out.**[](http://tryqa.com/wp-content/uploads/2012/01/What-is-big-bang-integration-testing1.jpg)**

**Advantage:** Big Bang testing has the advantage that everything is finished before integration testing starts.

**Disadvantage:** The major disadvantage is that in general it is time consuming and difficult to trace the cause of failures because of this late integration.

2. **Top-down integration testing:** Testing takes place from top to bottom, following the control flow or architectural structure (e.g. starting from the GUI or main menu). Components or systems are substituted by stubs. Below is the diagram of  ‘Top down Approach’:



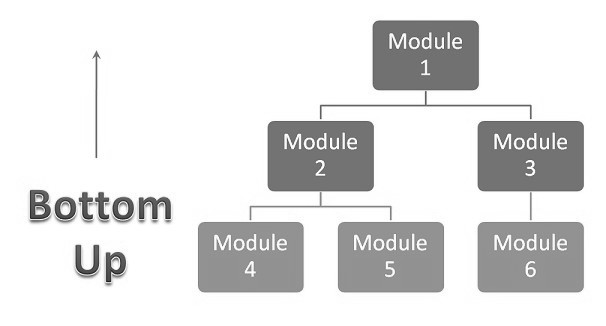
**Advantages of Top-Down approach:**

* The tested product is very consistent because the integration testing is basically performed in an environment that almost similar to that of reality
* Stubs can be written with lesser time because when compared to the drivers then Stubs are simpler to author.

**Disadvantages of Top-Down approach:**

* Basic functionality is tested at the end of cycle

3. **Bottom-up integration testing:**Testing takes place from the bottom of the control flow upwards. Components or systems are substituted by drivers. Below is the image of ‘Bottom up approach’:



**Advantage of Bottom-Up approach:**

* In this approach development and testing can be done together so that the product or application will be efficient and as per the customer specifications.

**Disadvantages** **of Bottom-Up approach:**

* We can catch the Key interface defects at the end of cycle
* It is required to create the test drivers for modules at all levels except the top control

[**Incremental testing**](http://tryqa.com/what-is-incremental-testing-in-software/)**:**

* Another extreme is that all programmers are integrated one by one, and a test is carried out after each step.
* The incremental approach has the advantage that the defects are found early in a smaller assembly when it is relatively easy to detect the cause.
* A disadvantage is that it can be time-consuming since stubs and drivers have to be developed and used in the test.
* Within incremental integration testing  a range of possibilities exist, partly depending on the system architecture.

**Functional incremental:**Integration and testing takes place on the basis of the functions and functionalities, as documented in the functional specification.

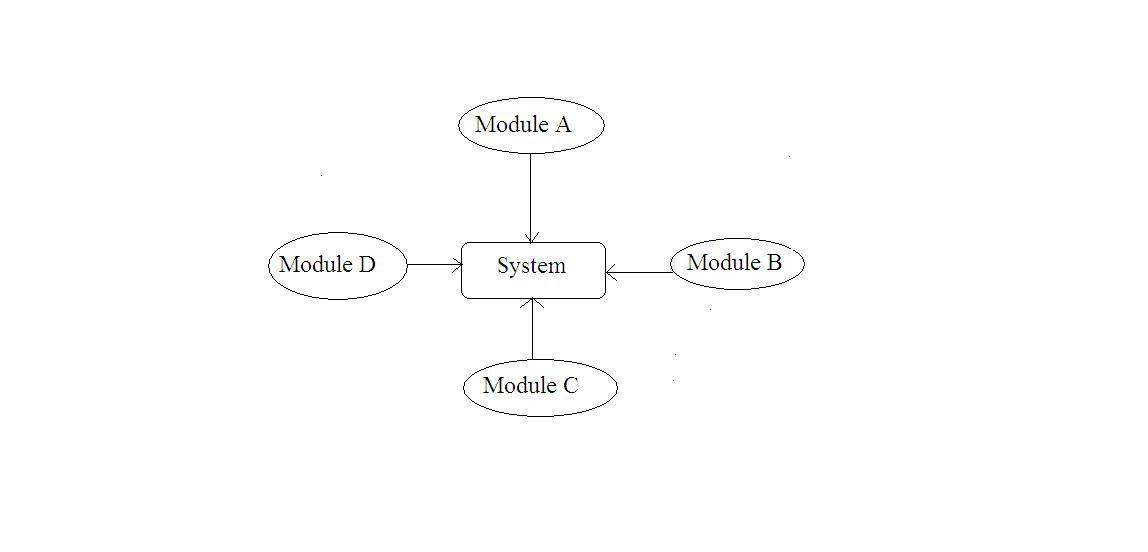
There are several types of integration testing like [**Big Bang integration testing**](http://tryqa.com/what-is-big-bang-integration-testing/), [**Component integration testing**](http://tryqa.com/what-is-component-integration-testing), [**System integration testing**](http://tryqa.com/what-is-system-integration-testing/) etc and these are covered in detail in subsequent topics.

# What is Big Bang integration testing?

In Big Bang integration testing all components or modules are integrated simultaneously, after which everything is tested as a whole.

* In this approach individual modules are not integrated until and unless all the modules are ready.
* In Big Bang integration testing all the modules are integrated without performing any [**integration testing**](http://tryqa.com/what-is-integration-testing/) and then it’s executed to know whether all the integrated modules are working fine or not.
* This approach is generally executed by those developers who follows the ‘Run it and see’ approach.
* Because of integrating everything at one time if any failures occurs then it become very difficult for the programmers to know the root cause of that failure.
* In case any bug arises then the developers has to detach the integrated modules in order to find the actual cause of the bug.

Below is the image of the big bang integration testing:



Suppose a system consists of four modules as displayed in the diagram above. In big bang integration all the four modules ‘Module A, Module B, Module C and Module D’ are integrated simultaneously and then the testing is performed. Hence in this approach no individual integration testing is performed because of which the chances of critical failures increases.

**Advantage of Big Bang Integration:**

* Big Bang testing has the **advantage** that everything is finished before integration testing starts.

**Disadvantages of Big Bang Integration:**

* The major **disadvantage** is that in general it is very time consuming
* It is very difficult to trace the cause of failures because of this late integration.
* The chances of having critical failures are more because of integrating all the components together at same time.
* If any bug is found then it is very difficult to detach all the modules in order to find out the root cause of it.
* There is high probability of occurrence of the critical bugs in the production environment

# What is Incremental testing in software?

The **incremental testing approach** has the advantage that the defects are found early in a smaller assembly when it is relatively easy to detect the cause.

* Another advantage is that all programs are integrated one by one and a test is carried out after each step.
* A disadvantage is that it can be time-consuming since stubs and drivers have to be developed and used in the test.
* Within incremental [**integration testing**](http://tryqa.com/what-is-integration-testing/) a range of possibilities exist, partly depending on the system architecture:
  + **Top down:**Testing takes place from top to bottom, following the control flow or architectural structure (e.g. starting from the GUI or main menu). Components or systems are substituted by stubs.
  + **Bottom up:**Testing takes place from the bottom of the control flow upwards. Components or systems are substituted by drivers.
  + **Functional incremental:**Integration and testing  takes place on the basis of the functions and functionalities, as documented in the functional specification.

# What is Component integration testing?

* It tests the interactions between software components and is done after component testing.
* The software components themselves may be specified at different times by different specification groups, yet the integration of all of the pieces must work together.
* It is important to cover negative cases as well because components might make assumption with respect to the data.

# What is System integration testing?

* System integration testing (SIT) tests the interactions between different systems and may be done after [**system testing**](http://tryqa.com/what-is-system-testing/).
* It verifies the proper execution of software components and proper interfacing between components within the solution.
* The objective of SIT Testing is to validate that all software module dependencies are [**functionally correct**](http://tryqa.com/what-is-functional-testing-testing-of-functions-in-software/) and that data integrity is maintained between separate modules for the entire solution.
* As testing for dependencies between different components is a primary function of SIT Testing, this area is often most subject to [**Regression Testing**](http://tryqa.com/what-is-regression-testing-in-software/).

# What is System testing?

* In **system testing** the behavior of whole system/product is tested as defined by the scope of the development project or product.
* It may include [**tests based on risks**](http://tryqa.com/what-is-risk-based-testing/) and/or [**requirement specifications**](http://tryqa.com/what-is-functional-testing-testing-of-functions-in-software/), business process, use cases, or other high level descriptions of system behavior, interactions with the operating systems, and system resources.
* System testing is most often the final test to verify that the system to be delivered meets the specification and its purpose.
* System testing is carried out by specialists testers or [**independent testers**](http://tryqa.com/what-is-independent-testing-its-benefits-and-risks/).
* System testing should investigate both [**functional**](http://tryqa.com/what-is-functionality-testing-in-software/) and [**non-functional requirements**](http://tryqa.com/what-is-non-functional-testing-testing-of-software-product-characteristics/) of the testing.

# What is Acceptance testing or User Acceptance Testing (UAT)?

After the system test has corrected all or most defects, the system will be delivered to the user or customer for **Acceptance Testing** or **User Acceptance Testing (UAT)**.

* Acceptance testing is basically done by the user or customer although other stakeholders may be involved as well.
* The goal of acceptance [**testing**](http://tryqa.com/what-is-a-software-testing/) is to establish confidence in the system.
* Acceptance testing is most often focused on a validation type testing.
* Acceptance testing may occur at more than just a single level, for example:
  + A **Commercial Off the shelf (COTS)**software product may be acceptance tested when it is installed or integrated.
  + **Acceptance testing of the**[**usability of the component**](http://tryqa.com/what-is-usability-testing-in-software-and-its-benifits-to-end-user/)may be done during component testing.
  + **Acceptance testing of a new functional enhancement**may come before [**system testing**](http://tryqa.com/what-is-system-testing/).
* The **types of acceptance testing**are:
  + The**User Acceptance test:**focuses mainly on the functionality thereby validating the fitness-for-use of the system by the business user. The user acceptance test is performed by the users and application managers.
  + The **Operational Acceptance test:**also known as Production acceptance test validates whether the system meets the requirements for operation. In most of the organization the operational acceptance test is performed by the system administration before the system is released. The operational acceptance test may include testing of backup/restore, disaster recovery, maintenance tasks and periodic check of security vulnerabilities.
  + **Contract Acceptance testing**: It is performed against the contract’s acceptance criteria for producing custom developed software. Acceptance should be formally defined when the contract is agreed.
  + **Compliance acceptance testing:**It is also known as regulation acceptance testing is performed against the regulations which must be adhered to, such as governmental, legal or safety regulations.

# What is Alpha testing?

Alpha testing is one of the most common [**software testing**](http://tryqa.com/what-is-a-software-testing/) **[strategy](http://tryqa.com/what-are-the-test-approaches-or-strategies-in-software-testing/" \o "Test Strategy)**used in software development. Its specially used by product development organizations.

* This **test takes place at the developer’s site**. Developers observe the users and note problems.
* Alpha testing is testing of an application when development is about to complete. Minor design changes can still be made as a result of alpha testing.
* Alpha testing is typically performed by a group that is independent of the design team, but still within the company, e.g. in-house software test engineers, or software QA engineers.
* Alpha testing is final testing before the software is released to the general public. It has two phases:
  + In the **first phase** of alpha testing, the software is tested by in-house developers. They use either debugger software, or hardware-assisted debuggers. The goal is to catch bugs quickly.
  + In the**second phase** of alpha testing, the software is handed over to the software QA staff, for additional testing in an environment that is similar to the intended use.
* Alpha testing is simulated or actual operational testing by potential users/customers or an independent test team at the developers’ site. Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing, before the software goes to beta testing.

# What is Beta testing?

**Beta Testing** is also known as field testing. It takes place at **customer’s site**. It sends the system/software to users who install it and use it under real-world working conditions.

* A beta test is the second phase of [**software testing**](http://tryqa.com/what-is-a-software-testing/) in which a sampling of the intended audience tries the product out. (Beta is the second letter of the Greek alphabet.) Originally, the term [**alpha testing**](http://tryqa.com/what-is-alpha-testing/) meant the first phase of testing in a software development process. The first phase includes [**unit testing**](http://tryqa.com/what-is-unit-testing/), [**component testing**](http://tryqa.com/what-is-component-testing/), and [**system testing**](http://tryqa.com/what-is-system-testing/). Beta testing can be considered “pre-release testing.
* The goal of beta testing is to place your application in the hands of real users outside of your own engineering team to discover any flaws or issues from the user’s perspective that you would not want to have in your final, released version of the application. Example: Microsoft and many other organizations release beta versions of their products to be tested by users.

**Open and closed beta:**  
Developers release either a **closed beta** or an **open beta**;

* Closed beta versions are released to a select group of individuals for a user test and are invitation only, while
* Open betas are from a larger group to the general public and anyone interested. The testers report any bugs that they find, and sometimes suggest additional features they think should be available in the final version.

### **Advantages of beta testing**

* You have the opportunity to get your application into the hands of users prior to releasing it to the general public.
* Users can install, test your application, and send feedback to you during this beta testing period.
* Your beta testers can discover issues with your application that you may have not noticed, such as confusing application flow, and even crashes.
* Using the feedback you get from these users, you can fix problems before it is released to the general public.
* The more issues you fix that solve real user problems, the higher the quality of your application when you release it to the general public.
* Having a higher-quality application when you release to the general public will increase customer satisfaction.
* These users, who are early adopters of your application, will generate excitement about your application.

# What are Software Test Types?

**Software Test types** are introduced as a means of clearly defining the objective of a certain level for a program or project.  A test type is focused on a particular test objective, which could be the testing of the function to be performed by the component or system.

The test objective could be to test [**non-functional quality**](http://tryqa.com/what-is-non-functional-testing-testing-of-software-product-characteristics/)characteristics, such as reliability or usability; the structure or architecture of the component or system; or related to changes, i.e confirming that defects have been fixed ([**confirmation testing or retesting**](http://tryqa.com/what-is-retesting/)) and looking for unintended changes ([**regression testing**](http://tryqa.com/what-is-regression-testing-in-software/)).

Depending on its objectives, testing will be organized differently. Hence there are four software test types:

1. [**Functional testing**](http://tryqa.com/what-is-functional-testing-testing-of-functions-in-software/)
2. [**Non-functional testing**](http://tryqa.com/what-is-non-functional-testing-testing-of-software-product-characteristics/)
3. [**Structural testing**](http://tryqa.com/what-is-structural-testing-testing-of-software-structurearchitecture/)
4. Change related testing

# What is Functional testing (Testing of functions) in software?

In **functional testing** basically the testing of the functions of **[component](http://tryqa.com/what-is-component-testing/)**or [**system**](http://tryqa.com/what-is-system-testing/) is done. It refers to activities that verify a specific action or function of the code. Functional test tends to answer the questions like “can the user do this” or “does this particular feature work”. This is typically described in a requirements specification or in a functional specification.

The techniques used for functional testing are often specification-based. Testing functionality can be done from two perspective:

* **Requirement-based testing:**In this type of testing the requirements are prioritized depending on the [**risk criteria**](http://tryqa.com/what-is-risk-based-testing/) and accordingly the tests are prioritized. This will ensure that the most important and most critical tests are included in the testing effort.
* **Business-process-based testing:**In this type of testing the scenarios involved in the day-to-day business use of the system are described. It uses the knowledge of the business processes. For example, a personal and payroll system may have the business process along the lines of: someone joins the company, employee is paid on the regular basis and employee finally leaves the company.

What is Non-functional testing (Testing of software product characteristics)?

FacebookLinkedInTwitterGoogle+Email

In **non-functional testing** the quality characteristics of the component or system is tested. Non-functional refers to aspects of the software that may not be related to a specific function or user action such as scalability or security. Eg. How many people can log in at once? Non-functional testing is also performed at all levels like [**functional testing**](http://tryqa.com/what-is-functional-testing-testing-of-functions-in-software/).

Non-functional testing includes:

* Reliability testing
* Usability testing
* Efficiency testing
* Maintainability testing
* Portability testing
* Baseline testing
* Compliance testing
* Documentation testing
* Endurance testing
* Load testing
* Performance testing
* Compatibility testing
* Security testing
* Scalability testing
* Volume testing
* Stress testing
* Recovery testing
* Internationalization testing and Localization testing
* [**Reliability testing**](http://tryqa.com/what-is-reliability-testing-in-software/)**:**Reliability Testing is about exercising an application so that failures are discovered and removed before the system is deployed. The purpose of reliability testing is to determine product reliability, and to determine whether the software meets the customer’s reliability requirements.
* [**Usability testing**](http://tryqa.com/what-is-usability-testing-in-software-and-its-benifits-to-end-user/)**:**In usability testing basically the testers tests the ease with which the user interfaces can be used. It tests that whether the application or the product built is user-friendly or not.

Usability testing includes the following five components:

* 1. **Learnability:**How easy is it for users to accomplish basic tasks the first time they encounter the design?
  2. **Efficiency:** How fast can experienced users accomplish tasks?
  3. **Memorability:** When users return to the design after a period of not using it, does the user remember enough to use it effectively the next time, or does the user have to start over again learning everything?
  4. **Errors:** How many errors do users make, how severe are these errors and how easily can they recover from the errors?
  5. **Satisfaction:** How much does the user like using the system?
* [**Efficiency testing**](http://tryqa.com/what-is-efficiency-testing-in-software/)**:**Efficiency testing test the amount of code and testing resources required by a program to perform a particular function. Software Test Efficiency is number of test cases executed divided by unit of time (generally per hour).
* [**Maintainability testing**](http://tryqa.com/what-is-maintainability-testing-in-software/)**:**It basically defines that how easy it is to maintain the system. This means that how easy it is to analyze, change and test the application or product.
* [**Portability testing**](http://tryqa.com/what-is-portability-testing-in-software/)**:**It refers to the process of testing the ease with which a computer software component or application can be moved from one environment to another, e.g. moving of any application from Windows 2000 to Windows XP. This is usually measured in terms of the maximum amount of effort permitted. Results are measured in terms of the time required to move the software and complete the and documentation updates.
* [**Baseline testing**](http://tryqa.com/what-is-baseline-testing-in-software/)**:**It refers to the validation of documents and specifications on which test cases would be designed. The requirement specification validation is baseline testing.
* [**Compliance testing**](http://tryqa.com/what-is-compliance-testing-in-software/)**:**It is related with the IT standards followed by the company and it is the testing done to find the deviations from the company prescribed standards.
* [**Documentation testing**](http://tryqa.com/what-is-documentation-testing/)**:**As per the IEEE Documentation describing plans for, or results of, the testing of a system or component, Types include test case specification, test incident report, test log, test plan, test procedure, test report. Hence the testing of all the above mentioned documents is known as documentation testing.
* [**Endurance testing**](http://tryqa.com/what-is-endurance-testing-in-software/)**:**Endurance testing involves testing a system with a significant load extended over a significant period of time, to discover how the system behaves under sustained use. For example, in software testing, a system may behave exactly as expected when tested for 1 hour but when the same system is tested for 3 hours, problems such as memory leaks cause the system to fail or behave randomly.
* [**Load testing**](http://tryqa.com/what-is-load-testing-in-software/)**:**A load test is usually conducted to understand the behavior of the application under a specific expected load. Load testing is performed to determine a system’s behavior under both normal and at peak conditions. It helps to identify the maximum operating capacity of an application as well as any bottlenecks and determine which element is causing degradation. E.g. If the number of users are in creased then how much CPU, memory will be consumed, what is the network and bandwidth response time
* [**Performance testing**](http://tryqa.com/what-is-performance-testing-in-software/)**:**Performance testing is testing that is performed, to determine how fast some aspect of a system performs under a particular workload. It can serve different purposes like it can demonstrate that the system meets performance criteria. It can compare two systems to find which performs better. Or it can measure what part of the system or workload causes the system to perform badly.
* [**Compatibility testing**](http://tryqa.com/what-is-compatibility-testing-in-software/)**:**Compatibility testing is basically the testing of the application or the product built with the computing environment. It tests whether the application or the software product built is compatible with the hardware, operating system, database or other system software or not.
* [**Security testing**](http://tryqa.com/what-is-security-testing-in-software/)**:**Security testing is basically to check that whether the application or the product is secured or not. Can anyone came tomorrow and hack the system or login the application without any authorization. It is a process to determine that an information system protects data and maintains functionality as intended.
* [**Scalability testing**](http://tryqa.com/what-is-scalability-testing-in-software/)**:**It is the testing of a software application for measuring its capability to scale up in terms of any of its non-functional capability like load supported, the number of transactions, the data volume etc.
* [**Volume testing**](http://tryqa.com/what-is-volume-testing-in-software/)**:**Volume testing refers to testing a software application or the product with a certain amount of data. E.g., if we want to volume test our application with a specific database size, we need to expand our database to that size and then test the application’s performance on it.
* [**Stress testing**](http://tryqa.com/what-is-stress-testing-in-software/)**:**It involves testing beyond normal operational capacity, often to a breaking point, in order to observe the results. It is a form of testing that is used to determine the stability of a given system. It put greater emphasis on robustness, availability, and error handling under a heavy load, rather than on what would be considered correct behavior under normal circumstances. The goals of such tests may be to ensure the software does not crash in conditions of insufficient computational resources (such as memory or disk space).
* [**Recovery testing**](http://tryqa.com/what-is-recovery-testing-in-software/)**:**Recovery testing is done in order to check how fast and better the application can recover after it has gone through any type of crash or hardware failure etc. Recovery testing is the forced failure of the software in a variety of ways to verify that recovery is properly performed. For example, when an application is receiving data from a network, unplug the connecting cable. After some time, plug the cable back in and analyze the application’s ability to continue receiving data from the point at which the network connection got disappeared. Restart the system while a browser has a definite number of sessions and check whether the browser is able to recover all of them or not.
* [**Internationalization testing and Localization testing**](http://tryqa.com/what-is-internationalization-testing-and-localization-testing-in-software/)**:**Internationalization is a process of designing a software application so that it can be adapted to various languages and regions without any changes. Whereas Localization is a process of adapting internationalized software for a specific region or language by adding local specific components and translating text.

# What is functionality testing in software?

**Functionality testing** is performed to verify that a software application performs and functions correctly according to design specifications. During functionality testing we check the core application functions, text input, menu functions and installation and setup on localized machines, etc.

 The following is needed to be checked during the functionality testing:

* Installation and setup on localized machines running localized operating systems and local code pages.
* Text input, including the use of extended characters or non-Latin scripts.
* Core application functions.
* String handling, text, and data, especially when interfacing with non-Unicode applications or modules.
* Regional settings defaults.
* Text handling (such as copying, pasting, and editing) of extended characters, special fonts, and non-Latin scripts.
* Accurate hot-key shortcuts without any duplication.

Functionality testing verifies that an application is still fully functional after [**localization**](http://tryqa.com/what-is-internationalization-testing-and-localization-testing-in-software/). Even applications which are professionally internationalized according to world-readiness guidelines require functionality testing.

What is reliability testing in software?

FacebookLinkedInTwitterGoogle+Email

**Reliability Testing** is about exercising an application so that [**failures**](http://tryqa.com/what-is-a-failure-in-software-testing/) are discovered and removed before the system is deployed. The purpose of reliability testing is to determine product reliability, and to determine whether the software meets the customer’s reliability requirements.

* According to ANSI, Software Reliability is defined as: the probability of failure-free software operation for a specified period of time in a specified environment. Software Reliability is not a direct function of time. Electronic and mechanical parts may become “old” and wear out with time and usage, but software will not rust or wear-out during its life cycle. Software will not change over time unless intentionally changed or upgraded.
* Reliability refers to the consistency of a measure. A test is considered reliable if we get the same result repeatedly. Software Reliability is the probability of failure-free software operation for a specified period of time in a specified environment. Software Reliability is also an important factor affecting system reliability.
* Reliability testing will tend to uncover earlier those failures that are most likely in actual operation, thus directing efforts at fixing the most important faults.
* Reliability testing may be performed at several levels. Complex systems may be tested at component, circuit board, unit, assembly, subsystem and system levels.

Software reliability is a key part in software quality. The study of software reliability can be categorized into three parts:

1.Modeling  
2.Measurement  
3. Improvement

1. **Modeling:** Software reliability modeling has matured to the point that meaningful results can be obtained by applying suitable models to the problem. There are many models exist, but no single model can capture a necessary amount of the software characteristics. Assumptions and abstractions must be made to simplify the problem. There is no single model that is universal to all the situations.

2. **Measurement:** Software reliability measurement is naive. Measurement is far from commonplace in software, as in other engineering field. “How good is the software, quantitatively?” As simple as the question is, there is still no good answer. Software reliability can not be directly measured, so other related factors are measured to estimate software reliability and compare it among products. Development process, faults and failures found are all factors related to software reliability.

3. **Improvement:** Software reliability improvement is hard. The difficulty of the problem stems from insufficient understanding of software reliability and in general, the characteristics of software. Until now there is no good way to conquer the complexity problem of software. Complete testing of a moderately complex software module is infeasible. Defect-free software product can not be assured. Realistic constraints of time and budget severely limits the effort put into software reliability improvement.

What is Usability testing in software and it’s benefits to end user?

FacebookLinkedInTwitterGoogle+Email

In **usability testing** basically the testers tests the ease with which the user interfaces can be used. It tests that whether the application or the product built is user-friendly or not.

* Usability Testing is a [**black box testing**](http://tryqa.com/what-is-black-box-specification-based-also-known-as-behavioral-testing-techniques/) technique.
* Usability testing also reveals whether users feel comfortable with your application or Web site according to different parameters – the flow, navigation and layout, speed and content – especially in comparison to prior or similar applications.
* Usability Testing tests the following features of the software.

— How easy it is to use the software.  
— How easy it is to learn the software.  
— How convenient is the software to end user.

 Usability testing includes the following five components:

1. **Learnability:**How easy is it for users to accomplish basic tasks the first time they encounter the design?
2. **Efficiency:** How fast can experienced users accomplish tasks?
3. **Memorability:** When users return to the design after a period of not using it, does the user remember enough to use it effectively the next time, or does the user have to start over again learning everything?
4. **Errors:** How many errors do users make, how severe are these errors and how easily can they recover from the errors?
5. **Satisfaction:** How much does the user like using the system?

Benefits of usability testing to the end user or the customer:

— Better quality software  
— Software is easier to use  
— Software is more readily accepted by users  
— Shortens the learning curve for new users

Advantages of usability testing:

* Usability test can be modified to cover many other types of testing such as [**functional testing**](http://tryqa.com/what-is-functional-testing-testing-of-functions-in-software/), [**system integration testing**](http://tryqa.com/what-is-system-integration-testing/), [**unit testing**](http://tryqa.com/what-is-unit-testing/), [**smoke testing**](http://tryqa.com/what-is-smoke-testing-when-to-use-it-advantages-and-disadvantages-2/) etc.
* Usability testing can be very economical if planned properly, yet highly effective and beneficial.
* If proper resources (experienced and creative testers) are used, usability test can help in fixing all the problems that user may face even before the system is finally released to the user. This may result in better performance and a standard system.
* Usability testing can help in discovering potential bugs and potholes in the system which generally are not visible to developers and even escape the other type of testing.

Usability testing is a very wide area of testing and it needs fairly high level of understanding of this field along with creative mind. People involved in the usability testing are required to possess skills like patience, ability to listen to the suggestions, openness to welcome any idea, and the most important of them all is that they should have good observation skills to spot and fix the issues or problems.

# What is Efficiency testing in software?

Efficiency testing test the amount of code and testing resources required by a program to perform a particular function. Software Test Efficiency is number of test cases executed divided by unit of time (generally per hour).

It is internal in the organization how much resources were consumed how much of these resources were utilized.

Here are some formulas to calculate **Software Test Efficiency** (for different factors):

* Test efficiency = (total number of defects found in unit+integration+system) / (total number of defects found in unit+integration+system+User acceptance testing)
* Testing Efficiency = (No. of defects Resolved / Total No. of Defects Submitted)\* 100

Software Test Effectiveness covers three aspects:

— How much the customer’s requirements are satisfied by the system.  
— How well the customer specifications are achieved by the system.  
— How much effort is put in developing the system.

# What is Maintainability testing in software?

FacebookLinkedInTwitterGoogle+Email

It basically defines that how easy it is to maintain the system. This means that how easy it is to analyze, change and test the application or product.

Maintainability testing shall use a model of the maintainability requirements of the software/system. The maintainability testing shall be specified in terms of the effort required to effect a change under each of the following four categories:

* **Corrective maintenance –** Correcting problems. The maintainability of a system can be measured in terms of the time taken to diagnose and fix problems identified within that system.
* **Perfective maintenance –** Enhancements. The maintainability of a system can also be measured in terms of the effort taken to make required enhancements to that system. This can be tested  by recording the time taken to achieve a new piece of identifiable functionality such as a change to the database, etc. A number of similar tests should be run and an average time calculated. The outcome will be that it is possible to give an average effort required to implement specified functionality. This can be compared against a target effort and an assessment made as to whether requirements are met.
* **Adaptive maintenance –** Adapting to changes in environment. The maintainability of a system can also be measured in terms on the effort required to make required adaptations to that system. This can be measured in the way described above for perfective maintainability testing.
* **Preventive maintenance –** Actions to reduce future maintenance costs. This refers to actions to reduce future maintenance costs.

# What is Portability testing in software?

FacebookLinkedInTwitterGoogle+Email

It refers to the process of testing the ease with which a computer software component or application can be moved from one environment to another, e.g. moving of any application from Windows 2000 to Windows 10. This is usually measured in terms of the maximum amount of effort permitted. Results are measured in terms of the time required to move the software and complete the and documentation updates.

Being able to move software from one machine platform to another either initially  or from an existing environment. It refers to system software or application software that can be recompiled for a different platform or to software that is available for two or more different platforms.

The iterative and incremental development cycle implies that portability testing is regularly performed in an iterative and incremental manner.

**Portability testing** must be automated if adequate [**regression testing**](http://tryqa.com/what-is-regression-testing-in-software/) is to occur.

The objectives of Portability testing are to:

* Partially validate the system (i.e., to determine if it fulfills its portability requirements):
  + Determine if the system can be ported to each of its required environments:
    - Hardware ram and disk space
    - Hardware processor and processor speed
    - Monitor resolution
    - Operating system make and version
    - Browser make and version
  + Determine if the look and feel of the webpages is similar and functional in the various browser types and their versions.
* Cause failures concerning the portability requirements that help identify [**defects**](http://tryqa.com/what-is-defect-or-bugs-or-faults-in-software-testing/) that are not efficiently found during unit and integration testing.
* Report these [**failures**](http://tryqa.com/what-is-a-failure-in-software-testing/) to the development teams so that the associated defects can be fixed.
* Help determine the extent to which the system is ready for launch.
* Help provide project status metrics (e.g., percentage of use case paths successfully tested).
* Provide input to the defect trend analysis effort.

Portability tests include tests for:

**Installability:** Installability testing is conducted on the software used to install other software on its target environment.

**Co-existence or compatibility:**Co-existence is the software product’s capability to co-exists with other independent software products in a common environments sharing common resources**.**

**Adaptability:**Adaptability is the capability of the software product to be adapted to different specified environments without applying actions or means other than those provided for this purpose for the system.

**Replaceability:**Replaceability is the capability of the product to be used in place of another specified product for the same purpose in the same environment.

Examples of portability testing of an application that is to be portable across multiple:

* Hardware platforms (including clients, servers, network connectivity devices, input devices, and output devices).
* Operating systems (including versions and service packs).
* Browsers (including both types and versions).

# What is Baseline testing in software?

FacebookLinkedInTwitterGoogle+Email

* It is one of the type of [**non-functional testing**](http://tryqa.com/what-is-non-functional-testing-testing-of-software-product-characteristics/).
* It refers to the validation of documents and specifications on which test cases would be designed. The requirement specification validation is baseline testing.
* Generally a baseline is defined as a line that forms the base for any construction or for measurement, comparisons or calculations.
* Baseline testing also helps a great deal in solving most of the problems that are discovered. A majority of the issues are solved through baseline testing.

# What is Compliance testing in software testing?

FacebookLinkedInTwitterGoogle+Email

* It is a type of non-functional [**software testing**](http://tryqa.com/what-is-a-software-testing/).
* It is related with the IT standards followed by the company and it is the testing done to find the deviations from the company prescribed standards.
* It determines,whether we are implementing and meeting the defined standards.
* We should take care while doing this testing,Is there any drawbacks in standards implementation in our project and need to do analysis to improve the standards.
* Its basically an audit of a system carried out against a known criterion.

# What is documentation testing in software testing?

FacebookLinkedInTwitterGoogle+Email

Documentation testing is a non-functional type of [**software testing**](http://tryqa.com/what-is-a-software-testing/).

* It is a type of [**non-functional testing**](http://tryqa.com/what-is-non-functional-testing-testing-of-software-product-characteristics/).
* Any written or pictorial information describing, defining, specifying, reporting, or certifying activities, requirements, procedures, or results’. Documentation is as important to a product’s success as the product itself. If the documentation is poor, non-existent, or wrong, it reflects on the quality of the product and the vendor.
* As per the IEEE Documentation describing plans for, or results of, the testing of a system or component, Types include test case specification, test incident report, test log, test plan, test procedure, test report. Hence the testing of all the above mentioned documents is known as documentation testing.
* This is one of the most cost effective approaches to testing. If the documentation is not right: there will be major and costly problems. The documentation can be tested in a number of different ways to many different degrees of complexity. These range from running the documents through a spelling and grammar checking device, to manually reviewing the documentation to remove any ambiguity or inconsistency.
* Documentation testing can start at the very beginning of the software process and hence save large amounts of money, since the earlier a [**defect**](http://tryqa.com/what-is-defect-or-bugs-or-faults-in-software-testing/) is found the less it will cost to be fixed.

# What is Endurance testing in software testing?

FacebookLinkedInTwitterGoogle+Email

Endurance testing is a non functional type of [**software testing**](http://tryqa.com/what-is-a-software-testing/).

* It is a type of non-functional testing.
* It is also known as Soak testing.
* Endurance testing involves testing a system with a significant load extended over a significant period of time, to discover how the system behaves under sustained use. For example, in software testing, a system may behave exactly as expected when tested for 1 hour but when the same system is tested for 3 hours, problems such as memory leaks cause the system to fail or behave randomly.
* The goal is to discover how the system behaves under sustained use. That is, to ensure that the throughput and/or response times after some long period of sustained activity are as good or better than at the beginning of the test.
* It is basically used to check the memory leaks.

# What is Load testing in software testing?

FacebookLinkedInTwitterGoogle+Email

Load testing is a type of [**non-functional testing**](http://tryqa.com/what-is-non-functional-testing-testing-of-software-product-characteristics/). A load test is type of [**software testing**](http://tryqa.com/what-is-a-software-testing/) which is conducted to understand the behavior of the application under a specific expected load. Load testing is performed to determine a system’s behavior under both normal and at peak conditions.

* It helps to identify the maximum operating capacity of an application as well as any bottlenecks and determine which element is causing degradation. E.g. If the number of users are increased then how much CPU, memory will be consumed, what is the network and bandwidth response time.
* Load testing can be done under controlled lab conditions to compare the capabilities of different systems or to accurately measure the capabilities of a single system.
* Load testing involves simulating real-life user load for the target application. It helps you determine how your application behaves when multiple users hits it simultaneously.
* Load testing differs from [**stress testing**](http://tryqa.com/what-is-stress-testing-in-software/), which evaluates the extent to which a system keeps working when subjected to extreme work loads or when some of its hardware or software has been compromised.
* The primary goal of load testing is to define the maximum amount of work a system can handle without significant performance degradation.
* Examples of load testing include:
  + Downloading a series of large files from the internet.
  + Running multiple applications on a computer or server simultaneously.
  + Assigning many jobs to a printer in a queue.
  + Subjecting a server to a large amount of traffic.
  + Writing and reading data to and from a hard disk continuously.

# What is Performance testing in software?

FacebookLinkedInTwitterGoogle+Email

* It is  a type of non-functional testing.
* Performance testing is testing that is performed, to determine how fast some aspect of a system performs under a particular workload.
* It can serve different purposes like it can demonstrate that the system meets performance criteria.
* It can compare two systems to find which performs better. Or it can measure what part of the system or workload causes the system to perform badly.
* This process can involve quantitative tests done in a lab, such as measuring the response time or the number of MIPS (millions of instructions per second) at which a system functions.
* Why to do [**performance testing**](http://tryqa.com/what-is-performance-testing-in-software/):
* Improve user experience on sites and web apps
* Increase revenue generated from websites
* Gather metrics useful for tuning the system
* Identify bottlenecks such as database configuration
* Determine if a new release is ready for production
* Provide reporting to business stakeholders regarding performance against expectations

# What is Compatibility testing in software testing?

FacebookLinkedInTwitterGoogle+Email

* It is a type of non-functional testing.
* Compatibility testing is a type of [**software testing**](http://tryqa.com/what-is-a-software-testing/) used to ensure compatibility of the system/application/website built with various other objects such as other web browsers, hardware platforms, users (in case if it’s very specific type of requirement, such as a user who speaks and can read only a particular language), operating systems etc. This type of testing helps find out how well a system performs in a particular environment that includes hardware, network, operating system and other software etc.
* It is basically the testing of the application or the product built with the computing environment.
* It tests whether the application or the software product built is compatible with the hardware, operating system, database or other system software or not.

# What is Security testing in software testing?

FacebookLinkedInTwitterGoogle+Email

* It is a type of non-functional testing.
* Security testing is basically a type of [**software testing**](http://tryqa.com/what-is-a-software-testing/) that’s done to check whether the application or the product is secured or not. It checks to see if the application is vulnerable to attacks, if anyone hack the system or login to the application without any authorization.
* It is a process to determine that an information system protects data and maintains functionality as intended.
* The security testing is performed to check whether there is any information leakage in the sense by encrypting the application or using wide range of software’s and hardware’s and firewall etc.
* Software security is about making software behave in the presence of a malicious attack.
* The six basic security concepts that need to be covered by security testing are: confidentiality, integrity, authentication, availability, authorization and non-repudiation.

# What is Scalability testing in software testing?

FacebookLinkedInTwitterGoogle+Email

* It is a type of non-functional testing.
* Testing the ability of a system, a network, or a process to continue to function well when it is changed in size or volume in order to meet a growing need.
* It is the testing of a software application for measuring its capability to scale up in terms of any of its non-functional capability like load supported, the number of transactions, the data volume etc.
* Example: An ecommerce site may be able to handle orders for up to 100 users at a time but scalability testing can be performed to check if it will be able to handle higher loads during peak shopping seasons.

# What is Volume testing in software testing?

FacebookLinkedInTwitterGoogle+Email

* It is a type of non-functional testing.
* Volume testing refers to testing a software application or the product with a certain amount of data. E.g., if we want to volume test our application with a specific database size, we need to expand our database to that size and then test the application’s performance on it.
* “Volume testing” is a term given and described in Glenford Myers’ The Art of[***Software Testing***](http://tryqa.com/what-is-a-software-testing/), 1979. Here’s his definition: **“Subjecting the program to heavy volumes of data. The purpose of volume testing is to show that the program cannot handle the volume of data specified in its objectives”** – p. 113.
* The purpose of **volume testing** is to determine system performance with increasing volumes of data in the database.

# What is Stress testing in software testing?

FacebookLinkedInTwitterGoogle+Email

* It is a type of [**non-functional testing**](http://tryqa.com/what-is-non-functional-testing-testing-of-software-product-characteristics/).
* It involves testing beyond normal operational capacity, often to a breaking point, in order to observe the results.
* It is a form of [**software testing**](http://tryqa.com/what-is-a-software-testing/) that is used to determine the stability of a given system.
* It  put  greater emphasis on robustness, availability, and error handling under a heavy load, rather than on what would be considered correct behavior under normal circumstances.
* The goals of such tests may be to ensure the software does not crash in conditions of insufficient computational resources (such as memory or disk space).

Difference between Volume, Load and stress testing in software

* FacebookLinkedInTwitterGoogle+Email
* Very simply we can put the difference between Volume, Load and stress testing as:
* [**Volume Testing**](http://tryqa.com/what-is-volume-testing-in-software/) = Large amounts of data  
  [**Load Testing**](http://tryqa.com/what-is-load-testing-in-software/) = Large amount of users  
  [**Stress Testing**](http://tryqa.com/what-is-stress-testing-in-software/) = Too many users, too much data, too little time and too little room

# What is Recovery testing in software?

FacebookLinkedInTwitterGoogle+Email

* It is a type of [**non-functional testing**](http://tryqa.com/what-is-non-functional-testing-testing-of-software-product-characteristics/).
* Recovery testing is done in order to check how fast and better the application can recover after it has gone through any type of crash or hardware failure etc.
* Recovery testing is the forced [**failure**](http://tryqa.com/what-is-a-failure-in-software-testing/) of the software in a variety of ways to verify that recovery is properly performed.
* For example: When an application is receiving data from a network, unplug the connecting cable. After some time, plug the cable back in and analyze the application’s ability to continue receiving data from the point at which the network connection was broken.
* Example: Restart the system while a browser has a definite number of sessions and check whether the browser is able to recover all of them or not.

# What is Internationalization testing and Localization testing in software?

FacebookLinkedInTwitterGoogle+Email

* It is a type of non-functional testing.
* Internationalization is a process of designing a software application so that it can be adapted to various languages and regions without any changes.
* Whereas Localization is a process of adapting internationalized software for a specific region or language by adding local specific components and translating text.

What is Confirmation testing in software?

* FacebookLinkedInTwitterGoogle+Email
* **Confirmation testing or**[**re-testing**](http://tryqa.com/what-is-retesting/)**:** When a test fails because of the defect then that defect is reported and a new version of the software is expected that has had the defect fixed. In this case we need to execute the test again to confirm that whether the defect got actually fixed or not. This is known as confirmation testing and also known as re-testing. It is important to ensure that the test is executed in exactly the same way it was the first time using the same inputs, data and environments.
* Hence, when the change is made to the defect in order to fix it then confirmation testing or re-testing is helpful.

# What is Regression testing in software?

When any modification or changes are done to the application or even when any small change is done to the code then it can bring unexpected issues. Along with the new changes it becomes very important to test whether the existing functionality is intact or not. This can be achieved by doing the regression testing.

* The purpose of the regression testing is to find the [**bugs**](http://tryqa.com/what-is-defect-or-bugs-or-faults-in-software-testing/) which may get introduced accidentally because of the new changes or modification.
* During [**confirmation testing**](http://tryqa.com/what-is-confirmation-testing-in-software/) the defect got fixed and that part of the application started working as intended. But there might be a possibility that the fix may have introduced or uncovered a different defect elsewhere in the software. The way to detect these ‘**unexpected side-effects**’ of fixes is to do regression testing.
* This also ensures that the bugs found earlier are NOT creatable.
* Usually the regression testing is done by automation tools because in order to fix the defect the same test is carried out again and again and it will be very tedious and time consuming to do it manually.
* During regression testing the [**test cases**](http://tryqa.com/test-case/) are prioritized depending upon the changes done to the feature or module in the application. The feature or module where the changes or modification is done that entire feature is taken into priority for testing.
* This testing becomes very important when there are continuous modifications or enhancements done in the application or product. These changes or enhancements should NOT introduce new issues in the existing tested code.
* This helps in maintaining the quality of the product along with the new changes in the application.

**Example:**

Let’s assume that there is an application which maintains the details of all the students in school. This application has four buttons Add, Save, Delete and Refresh. All the buttons functionalities are working as expected.

Recently a new button ‘Update’ is added in the application. This ‘Update’ button functionality is tested and confirmed that it’s working as expected. But at the same time it becomes very important to know that the introduction of this new button should not impact the other existing buttons functionality.

Along with the ‘Update’ button all the other buttons functionality are tested in order to find any new issues in the existing code. This process is known as regression testing.

**Types of Regression testing techniques:**We have four types of regression testing techniques. They are as follows:

**1) Corrective Regression Testing:** Corrective regression testing can be used when there is no change in the specifications and test cases can be reused.

**2) Progressive Regression Testing:** Progressive regression testing is used when the modifications are done in the specifications and new test cases are designed.

**3) Retest-All Strategy:** The [**retest**](http://tryqa.com/what-is-retesting/)-all strategy is very tedious and time consuming because here we reuse all tests which results in the execution of unnecessary test cases. When any small modification or change is done to the application then this strategy is not useful.

**4) Selective Strategy:** In selective strategy we use a subset of the existing test cases to cut down the retesting effort and cost. If any changes are done to the program entities, e.g. functions, variables etc., then a test unit must be rerun. Here the difficult part is to find out the dependencies between a test case and the program entities it covers.

**When to use it:**

Regression testing is used when:

* Any new feature is added
* Any enhancement is done
* Any bug is fixed
* Any performance related issue is fixed

**Advantages of Regression testing:**

* It helps us to make sure that any changes like bug fixes or any enhancements to the module or application have not impacted the existing tested code.
* It ensures that the bugs found earlier are NOT creatable.
* Regression testing can be done by using the automation tools
* It helps in improving the quality of the product.

**Disadvantages of Regression testing:**

* If regression testing is done without using automated tools then it can be very tedious and time consuming because here we execute the same set of test cases again and again.
* Regression test is required even when a very small change is done in the code because this small modification can bring unexpected issues in the existing functionality.

# What is Structural testing (Testing of software structure/architecture)?

FacebookLinkedInTwitterGoogle+Email

* The structural testing is the testing of the structure of the system or component.
* Structural testing is often referred to as ‘white box’ or ‘glass box’ or ‘clear-box testing’ because in structural testing we are interested in what is happening ‘inside the system/application’.
* In structural testing the testers are required to have the knowledge of the internal implementations of the code. Here the testers require knowledge of how the software is implemented, how it works.
* During structural testing the tester is concentrating on how the software does it. For example, a structural technique wants to know how loops in the software are working. Different test cases may be derived to exercise the loop once, twice, and many times. This may be done regardless of the functionality of the software.
* Structural testing can be used at all levels of testing. Developers use structural testing in component testing and component integration testing, especially where there is good tool support for code coverage. Structural testing is also used in system and acceptance testing, but the structures are different. For example, the coverage of menu options or major business transactions could be the structural element in system or acceptance testing.

# What is Maintenance Testing?

FacebookLinkedInTwitterGoogle+Email

Once a system is deployed it is in service for years and decades. During this time the system and its operational environment is often corrected, changed or extended. Testing that is provided during this phase is called maintenance testing.

Usually maintenance testing is consisting of two parts:

* **First** one is, testing the changes that has been made because of the correction in the system or if the system is extended or because of some additional features added to it.
* **Second**one is regression tests to prove that the rest of the system has not been affected by the maintenance work.

# What is Impact analysis in software testing?

Impact analysis is basically analyzing the impact of the changes in the deployed application or product.

It tells us about the parts of the system that may be unintentionally affected because of the change in the application and therefore need careful regression testing.  This decision is taken together with the stakeholders.