**Formal inspection Process**

22.6 Inspection-pro(19.6).eps                                  00118328Macintosh HD                   B8AA5F2E:

A formal inspection consists of several activities:

1. **Planning** - The moderator selects the inspection team, obtains materials to be inspected from the author, and distributes them and any other relevant documents to the inspection team in advance. Materials should be distributed at least two or three days prior to the inspection. We leave the responsibility for requesting an inspection to the author, but you should establish some entry criteria for assessing readiness of a product for inspection. For code, you might require that it compiles cleanly and that the listing provided to inspectors includes line numbers.
2. **Overview meeting**- This meeting gives the author an opportunity to describe the important features of the product to the inspection team. It can be omitted if this information is already known to the other participants.
3. **Preparation** - Each participant is responsible for examining the work artifacts prior to the actual inspection meeting, noting any defects found or issues to be raised. Perhaps 75% of the errors found during inspections are identified during the preparation step. The product should be compared against any predecessor (specification) documents to assess completeness and correctness.  
     
   Checklists of defects commonly found in this type of work product should be used during preparation to hunt for anticipated types of errors. A sample checklist for code inspections is found in Figure 1. Checklists for inspecting nearly a dozen types of software work products can be found (along with tons of other useful information) in *Software Inspection Process* by Robert G. Ebenau and Susan H. Strauss (McGraw-Hill, 1994). If any standards are being used that pertain to this class of work product, each preparer should look for deviations from the standard.
4. **Inspection meeting** - During this session, the team convenes and is led through the work product by the moderator and reader. If the moderator determines at the beginning of the meeting that insufficient time has been devoted to preparation by the participants, the meeting should be rescheduled. During the discussion, all inspectors can report defects or raise other issues, which are documented on a form by the recorder.  
     
   The meeting should last no more than two hours. At its conclusion, the group agrees on an assessment of the product: accepted as is (I have never seen this happen); accepted with minor revisions; major revisions needed and a second inspection required; or rebuild the product.

**Causal analysis**- An important long-term benefit of an inspection program is the insight it can provide into the kinds of defects being created and process changes you can make to prevent them. This "causal analysis" step provides that understanding. While not essential to the current inspection, it can lead to improved quality on future work by helping to avoid making the same mistakes in the future.

1. **Rework** - The author is responsible for resolving all issues raised during the inspection. This does not necessarily mean making every change that was suggested, but an explicit decision must be made about how each issue or defect will be dealt with.
2. **Follow-up** - To verify that the necessary rework has been performed properly, the moderator is responsible for following up with the author. If a significant fraction (say, 10 percent) of the work product was modified, an additional inspection may be required. This is the final gate through which the product must pass in order for the inspection to be completed. You may wish to define explicit exit criteria for completing an inspection. These criteria might require that all defects are corrected and issues resolved, or that uncorrected defects are properly documented in a defect tracking system.

**Formal Technical Reviews**

* Formal Technical review is a software quality assurance activity performed by software engineer.

**Objectives of FTR**

1. FTR is useful to uncover error in logic, function and implementation for any representation of the software.
2. The purpose of FTR is to ensure that software meets specified requirements.
3. It is also ensure that software is represented according to predefined standards.
4. It helps to review the uniformity in software development process.
5. It makes the project more manageable.

* Besides the above mentioned objectives, the purpose of FTR is to enable junior engineer to observer the analysis, design, coding and testing approach more closely.
* Each FTR is conducted as meeting and is considered successfully only if it is properly planned, controlled and attended.

**Steps in FTR**

**1. The review meeting**

* Every review meeting should be conducted by considering the following constraints-

1. Involvement of people

Between 3 and 5 people should be involve in the review.

1. Advance preparation Advance preparation should occur but it should be very short that is at the most 2 hours of work for each person can be spent in this preparation
2. Short duration The short duration of the review meeting should be less than two hour.

* Rather than attempting to review the entire design walkthrough are conducted for modules or for small group of modules.
* The focus of the FTR is on work product (a software component to be reviewed). The review meeting is attended by the review leader, all reviewers and the producer.
* The review leader is responsible for evaluating for product for its deadlines. The copies of product material is then distributed to reviewers. -The producer organises “walkthrough” the product, explaining the material, while the reviewers raise the issues based on theirs advance preparation.
* One of the reviewers become recorder who records all the important issues raised during the review. When error are discovered, the recorder notes each.
* At the end of the review, the attendees decide whether to accept the product or not, with or without modification.

**2. Review reporting and record keeping**

* During the FTR, the reviewer actively records all the issues that have been raised.
* At the end of meeting these all raised issues are consolidated and review issue list is prepared.
* Finally, formal technical review summary report is produced.

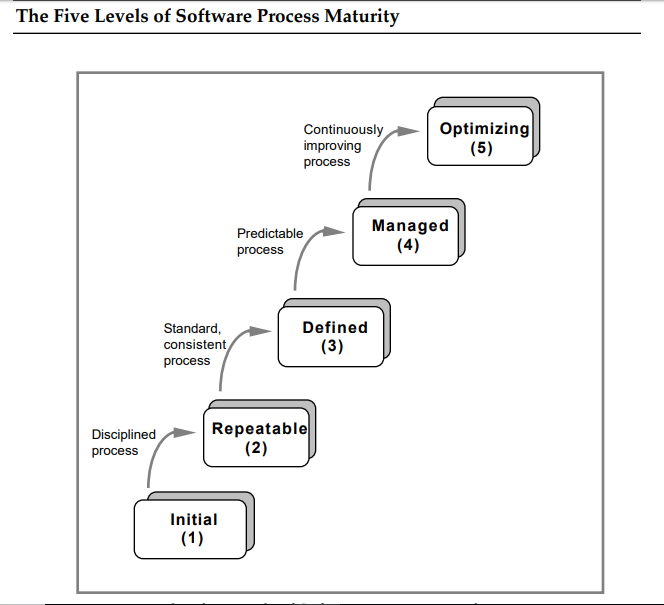
**3. Review guidelines**

* Guidelines for the conducting of formal technical review must be established in advance. These guidelines must be distributed to all reviewers, agreed upon, and then followed.
* For example,

Guideline for review may include following things

1. Concentrate on work product only. That means review the product not the producers.
2. Set an agenda of a review and maintain it.
3. When certain issues are raised then debate or arguments should be limited. Reviews should not ultimately results in some hard feelings.
4. Find out problem areas, but don’t attempt to solve every problem noted.
5. Take written notes (it is for record purpose)
6. Limit the number of participants and insists upon advance preparation.
7. Develop a checklist for each product that is likely to be reviewed.
8. Allocate resources and time schedule for FTRs in order to maintain time schedule.
9. Conduct meaningful trainings for all reviewers in order to make reviews effective.
10. Reviews earlier reviews which serve as the base for the current review being conducted.

**Capability Maturity Model(CMM)**



The following characterizations of the five maturity levels highlight the primary process changes made at each level:

1) Initial The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort.

2) Repeatable Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.

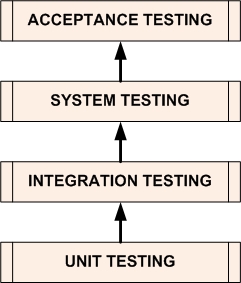
3) Defined The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software.

4) Managed Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.

5) Optimizing Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies

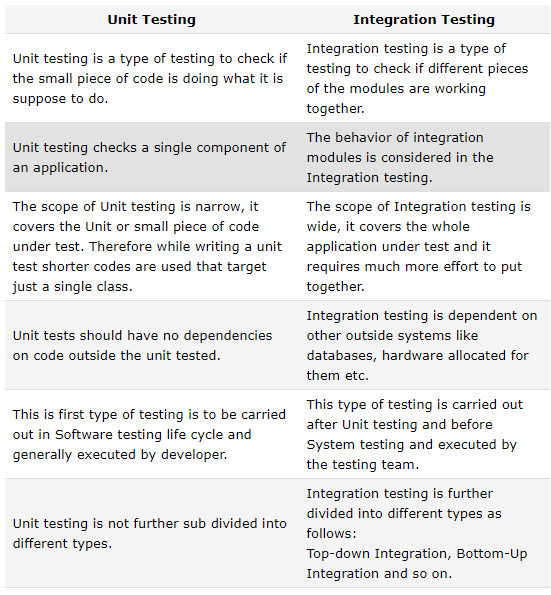
# Software Testing Levels

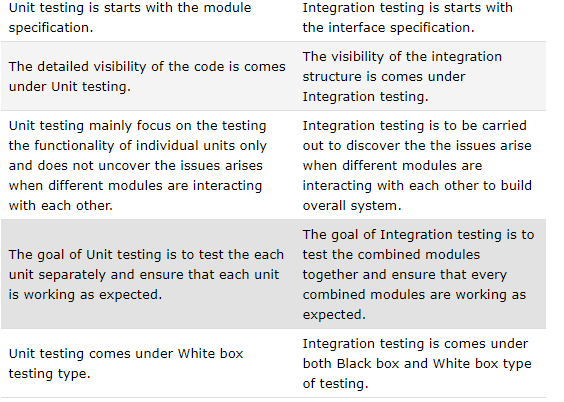
**SOFTWARE TESTING LEVELS** are the different stages in the software development life cycle where testing is conducted. There are four levels of software testing: Unit >> Integration >> System >> Acceptance.



## ****Levels****

|  |  |
| --- | --- |
| **Level** | **Summary** |
| [Unit Testing](http://softwaretestingfundamentals.com/unit-testing/) | A level of the software testing process where individual units/components of a software/system are tested. The purpose is to validate that each unit of the software performs as designed. |
| [Integration Testing](http://softwaretestingfundamentals.com/integration-testing/) | A level of the software testing process where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. |
| [System Testing](http://softwaretestingfundamentals.com/system-testing/) | A level of the software testing process where a complete, integrated system/software is tested. The purpose of this test is to evaluate the system’s compliance with the specified requirements. |
| [Acceptance Testing](http://softwaretestingfundamentals.com/acceptance-testing/) | A level of the software testing process where a system is tested for acceptability. The purpose of this test is to evaluate the system’s compliance with the business requirements and assess whether it is acceptable for delivery. |





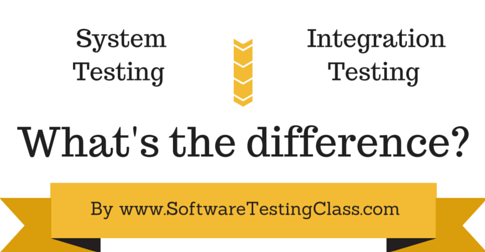
**What is System Testing?**

*System Testing* is testing of the software application as a whole to check if the system is complaint with the user requirements. It is an end to end user perspective testing intended to find defects in the software system.

System Testing is a type of black box testing technique thus the knowledge of internal code in not required. It is a high level testing always performed after integration testing. Regression and Re Testing is performed many times in system testing. The user can perform different type of tests under System Testing .It would depend on the user or the organisation to choose which type of system testing should be performed on the application. System testing can be broadly classified in two types:

1. Functional Testing
2. Non Functional Testing

**Objective of System Testing**:

[](https://i1.wp.com/www.softwaretestingclass.com/wp-content/uploads/2015/09/Difference-between-System-Testing-vs-Integration-Testing.png)

**System testing** is performed to check the following points:

* To check whether the software system is made according to the customer needs written in Software Requirements Specifications, it meets both functional and non-functional design requirements of the system.
* When all the modules are combined as a whole, many errors and facts may arise which may not give the expected results? So system testing is performed to find the defects or bugs in all the interfaces as well the whole system.
* To execute the real –life scenarios on the software. It is done on the staging server which is very much similar to the production server where the software would actually be deployed. The system test cases are made according to the end- to – end use perspective.

**What is Integration Testing?**

Integration testing tests the interface between modules of the software application.  The different modules are first testing individually and then combined to make a system. Testing the interface between the small units or modules is integration testing. It is usually conducted by software integration tester and in continuation to the development.  There are different techniques available for integration testing:

1. Big Bang Integration Testing: In type of integration testing all the modules are combined first and then tested together.
2. Top Bottom Integration Testing: This type of testing take place from top to bottom uses Stubs which are substitutes of components. The top module is tested first.
3. Bottom to Top Integration Testing: This type of testing take from bottom to top and uses Drivers which are substitutes of components. The bottom module is tested first.

**Objective of Integration Testing**:

Integration testing is performed to check the following points:

* To check whether the modules developed by individual developers when combined are according to standards and gives the expected results.
* When modules are combined, sometimes the data travelling between modules has many errors which may not give the expected results. So integration testing is performed to find the defects or bugs in all the interfaces.
* To check the integration between any third party tools used

**Difference between System Testing and Integration Testing:**

|  |  |
| --- | --- |
| **System Testing** | **Integration Testing** |
| 1. Testing the completed product to check if it meets the specification requirements. | 1. Testing the collection and interface modules to check whether they give the expected result |
| 2. Both functional and non-functional testing are covered like sanity, usability, performance, stress an load . | 2.Only  Functional testing is performed to check whether the two modules when combined give correct outcome. |
| 3. It is a high level testing performed after integration testing | 3. It is  a low level testing performed after unit testing |
| 4.  It is a black box testing technique so no knowledge of internal structure or code is required | 4. It is both black box and white box testing approach so it requires the knowledge of the two modules and the interface |
| 5. It is performed by test engineers only | 5. Integration testing is performed by developers as well test engineers |
| 6. Here the testing is performed on the system as a whole including all the external interfaces, so any defect found in it is regarded as defect of whole system | 6. Here the testing is performed on interface between individual module thus any defect found is only for individual modules and not the entire system |
| 7. In System Testing the test cases are developed to simulate real life scenarios | 7. Here the test cases are developed to simulate the interaction between the two module |
| 8. The System testing covers many different testing types like sanity, usability, maintenance, regression, retesting and performance | 8. Integration testing techniques includes big bang approach, top bottom , bottom to top and sandwich approach. |
|  |  |

**Example to differentiate between System testing and Integration Testing:**

**Project**: Online Shopping

**Modules:**The Integration Testing would include checking the interaction between different modules like :

1. Searching a product using search bar, checking if the search bar accepts space, characters.
2. Checking the search results after hitting enter button, if it is according to the searched words. Changing the search item and checking the results again
3. Clicking on the product we select
4. Check the product detail page
5. Adding the product to the cart by clicking on “Add to Cart”, check by adding few more items
6. Checking the shopping cart page, if all the data coming from the product detail page is proper.

**System:**The System Testing would include checking system as a whole:

1. Add items to the Product page from back-end, check if it stored correctly in database table.
2. Add few items to shopping cart and check if these items are correctly stored in temporary tables in database.
3. Delete few items from shopping cart, it should be deleted from DB as well.
4. Perform a check out transaction; see in DB if the payment option and other details are updated in DB as real time.
5. Checking the payment page and the transaction id generated. Checking the same in database

**Difference between Black Box Testing and White Box Testing.**

|  |  |  |
| --- | --- | --- |
| **#** | **Black Box Testing** | **White Box Testing** |
| 1 | Black box testing is the [Software testing method](http://www.softwaretestingclass.com/what-is-software-testing/) which is used to test the software without knowing the internal structure of code or program. | White box testing is the software testing method in which internal structure is being known to tester who is going to test the software. |
| 2 | This type of testing is carried out by testers. | Generally, this type of testing is carried out by software developers. |
| 3 | Implementation Knowledge is not required to carry out Black Box Testing. | Implementation Knowledge is required to carry out White Box Testing. |
| 4 | Programming Knowledge is not required to carry out Black Box Testing. | Programming Knowledge is required to carry out White Box Testing. |
| 5 | Testing is applicable on higher levels of testing like System Testing, Acceptance testing. | Testing is applicable on lower level of testing like Unit Testing, Integration testing. |
| 6 | Black box testing means functional test or external testing. | White box testing means structural test or interior testing. |
| 7 | In Black Box testing is primarily concentrate on the functionality of the system under test. | In White Box testing is primarily concentrate on the testing of program code of the system under test like code structure, branches, conditions, loops etc. |
| 8 | The main aim of this testing to check on what functionality is performing by the system under test. | The main aim of White Box testing to check on how System is performing. |
| 9 | Black Box testing can be started based on Requirement Specifications documents. | White Box testing can be started based on Detail Design documents. |
| 10 | The Functional testing, Behavior testing, Close box testing is carried out under Black Box testing, so there is no required of the programming knowledge. | The Structural testing, Logic testing, Path testing, Loop testing, Code coverage testing, Open box testing is carried out under White Box testing, so there is compulsory to know about programming knowledge. |

USER ACCEPTANCE TESTING(Alpha and beta)

***Top 13 Differences between Alpha and Beta Testing (Field Testing)***

|  |  |
| --- | --- |
| **Alpha Testing** | **Beta Testing (Field Testing)** |
| 1. It is always performed by the developers at the software development site. | 1. It is always performed by the customers at their own site. |
| 2. Sometimes it is also performed by Independent Testing Team. | 2. It is not performed by Independent Testing Team. |
| 3. Alpha Testing is not open to the market and public | 3. Beta Testing is always open to the market and public. |
| 4. It is conducted for the software application and project. | 4. It is usually conducted for software product. |
| 5. It is always performed in **Virtual Environment**. | 5. It is performed in **Real Time Environment**. |
| 6. It is always performed within the organization. | 6. It is always performed outside the organization. |
| 7. It is the form of Acceptance Testing. | 7. It is also the form of Acceptance Testing. |
| 8. Alpha Testing is definitely performed and carried out at the developing organizations location with the involvement of developers. | 8. Beta Testing (field testing) is performed and carried out by users or you can say people at their own locations and site using customer data. |
| 9. It comes under the category of both [White Box Testing and Black Box Testing](http://testingbasicinterviewquestions.blogspot.in/2014/09/top-10-differences-between-black-box.html). | 9. It is only a kind of Black Box Testing. |
| 10. Alpha Testing is always performed at the time of Acceptance Testing when developers test the product and project to check whether it meets the user requirements or not. | 10. Beta Testing is always performed at the time when software product and project are marketed. |
| 11. It is always performed at the developer’s premises in the absence of the users. | 11. It is always performed at the user’s premises in the absence of the development team. |
| 12. Alpha Testing is not known by any other different name. | 12 Beta Testing is also known by the name **Field Testing** means it is also known as field testing. |
| 13. It is considered as the User Acceptance Testing (UAT) which is done at developer’s area. | 13. It is also considered as the User Acceptance Testing (UAT) which is done at customers or users area. |

# ****What is User Acceptance Testing?****

In a Software Testing Lifecycle, the UAT, alpha and beta testing are the acceptance testing. These are performed at the end of the lifecycle when all the functional, non- functional and regression tests are completed.  User acceptance testing is the final stage where the end users can check the software for its compliance with the business requirements. The UAT is carried out by the end users who are aware of the requirements and understand the purpose of building the software. This testing is the last testing performed before the software goes alive.

# ****Objective of User Acceptance Testing?****

The software is coded by developers after interpreting the requirements given in the documents. The testers and developers test the software based upon their understanding of the requirements. Thus the software is developed according to the functional requirements of the client or the organisation, but there are some business needs which can only be understood by the end users of the software. These business requirements and processes can sometimes be miscommunicated or missed and thus lacking the software built. Thus User Acceptance testing has an important role in which the end users validate the software whether it meets those business needs before getting live.

During UAT, the end users use real life scenarios and build UAT Test cases for real data; hence it becomes an important part in software release cycle.  Thus any defects found during UAT are fixed at much lower costs than finding and fixing defects after release.

# ****Who carries the UAT:****

Many tester has question in there mind is that “Who is going to perform the User Acceptance Testing (UAT)?”. The answer to this question is simple, the UAT is carried out by the end users who are going to use the software. The UAT team is generally composed of Beta Testers. Also some organisations form a small group or team from its employees called UAT Team who test the software so that the software is validated from every aspect and every user role.

# ****Challenges faced during UAT:****

UAT is a very important and crucial part of software release cycle. Many organisations suffer losses due to flaws in the software release and improper UAT. Organisations face a lot of challenges during release of a software and UAT. Issues like poor participation in UAT, users reluctance to perfrom UAT, improper test planning are some of problems in UAT. Thus it needs to overcome these problems for successful software.

We will see some challenges faced by organisation during **UAT**.

**1. Improper test planning**: UAT is carried out in the last stage of software testing life cycle and most critical part. So delay in any of the pervious testing stages leads to pressure and short time for UAT. What happens is that in cases the test planning for system testing and UAT is overlapped. The software is deployed in UAT environment without even completing the functionality testing leading to inaccuracy in the software. There should be a proper UAT plan which should contain test cases and use cases based on real user scenarios. The test plan should also contain test objectives to validate the software better according to business requirements.

**2.  In correct UAT users**: The UAT Testers are not properly trained and don’t have complete knowledge of the new business needs of the software. The organisation put the team of unskilled workers to perform the UAT thus causing improper UAT. Since the UAT team is built new whenever there is a software release. So the team is not aware of the  effective UAT. Also sometimes the non-technical people are hired to perform UAT which face difficulties for technical issues.

**3. The UAT environment and deployment:**The UAT should be carried out on the different environment than the functional and system testing. If we perform the UAT on the same environment, it will lead to missing real data test cases. When there is a different environment for UAT, we need to take care that the latest version of the software is deployed. It will be a total waste of time if the software which is being tested is not the latest version.

**4. Handling the new business requirements and defects:**During UAT many issues are found due to unclear or ambiguity in the requirements and the testers raise the defects for the same. These defects are expected to be fixed by the customers ignoring the fact that it may take ample time to fix the change requests, hence at this point the project manager needs to take a decision fast otherwise it may lead to delay in the software release.

**5. Communication Gap between the different teams :** Generally there is always a communication problem between the technical team, UAT team and testing team if they are present at different locations. Email communication between different teams like offshore and onsite leads to a lot of delays which can take a full day for a small ambiguity in the software requirements. Thus there should be a proper planning and timely communication for an effective UAT. There should be tools where all the teams can collaborate and put down notes, log defects.

**6. Customers offload their responsibility to functional testing teams:** Due to busy schedule of the customers and unavailability of the users, the customers try to offload their work and ask the functional testing teams to perform the UAT. This leads to ignorance of the real user scenarios and inefficient UAT. When the software gets alive the end users find issues. Thus UAT should be assigned to correct users and skilled users having proper business knowledge.

**7. Not Accepting the software:** Sometimes the customers trying to point out mistakes to reject the software just to show their superiority. The business team tries to put down the development and testing team. It is very rare and happens only if there are some politics going on the organisation. It can be avoided by building positive relation with the teams.

# ****How to Overcome the Challenges face in UAT:****

**1. Planning the UAT well in advance:**We should plan the User acceptance well in advance. Performing random and informal UAT will not be effective in find the deep embedded defects which are the main troublesome for the software. If we do improper planning without any documentation, we will never be sure of the completion of the UAT. The planning should be in phases like at strategic level, logical level and later detailed level. The user should determine the standards for the UAT in documentation, change control process and time frames.

**2. Involve actual users in UAT :** Generally the companies hire a team of surrogate user who perform the UAT but are not the people who are the actual users. The actual users when work on the software find issues which cannot be seen by the surrogate users. In this case, when the real users are not ready to perform UAT, the company should hold few review meetings with the actual users.

**3. Determine the test intensity relative to the risk and skill of workers:**Some projects do not require exhaustive testing and projects might require extensive testing. We should perform a risk assessment to determine the area which are severe, critical and can be impacted, so that the team will focus more on those areas to detect defects and avoid negative consequences. The risk assessment should be formal and documented, quantified. Informal risk assessment cannot determine a critical failure.

**4. Baseline the real time conditions and not the user requirements:** It is the most important thing in UAT; UAT is incomplete if we do not consider the real life scenarios in testing. In UAT we need to perform both Verification and Validation. Verification is performed on specifications and requirements while validation is performed on real cases. UAT Test cases are built to test the software on real world conditions.

**5. Understand the phase of UAT:**Mostly UAT happens in the end of the project when the software is completed and installed. Before this all the testing is performed on the software parts but not as a system as a whole. End of the project is the worst time to find and fix defects because defects found during this time cost 10 times higher than usual to be fixed. We should involve users from he very beginning so that they can define the acceptance criteria when providing the inputs to the system.

**6. Review the test planning:**The Test Plans can have common mistakes or errors. It should be reviewed for proper standard by the QA Team or the facilitator or project manager

## [Software Quality Assurance Activities](http://www.helpingtesters.com/software-quality-assurance-activities/)

Software Quality Assurance (SQA) is the process of making sure that the software is free from defects or mistakes and performs all the functionalities without complaints just before the delivery.The SQA process talks about the evaluation of the software on the basis of certain activities.

The Software Quality Assurance is measured based on the internal and external quality features of the software. The external quality is measured based on the real-time activities in operational mode and how the software is useful for the end users.

The internal quality is measured based on the style and quality of the code written. Mostly the client will bother about the external quality only. But, in effect for a perfect performance of the software, the internal quality is an important aspect to be considered and maintained.

# ****How to Determine the Software Quality Assurance?****

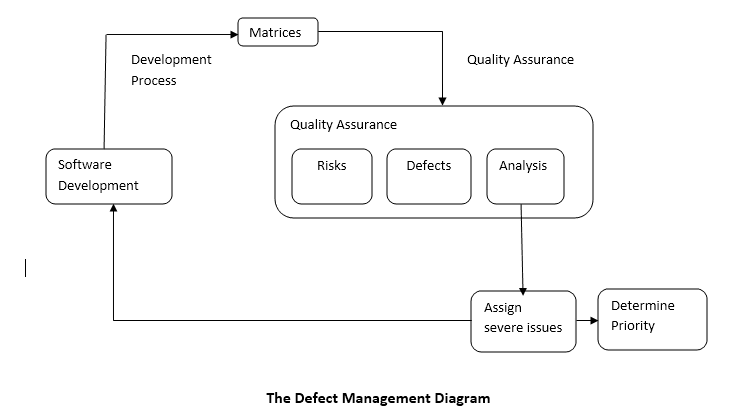
The 2 approaches to determine the Software Quality Assurance are:

## ****The Defect Management Approach****

The defects are categorized on the basis of the severity. The counts of the defects are taken and the actions are decided by analyzing the occurrence of defects. The defects come from very minute issues and extend to the coding defects, the non-completion of the requirements and of course if the application does not look good for the customers. Defect management process is based on some principles:

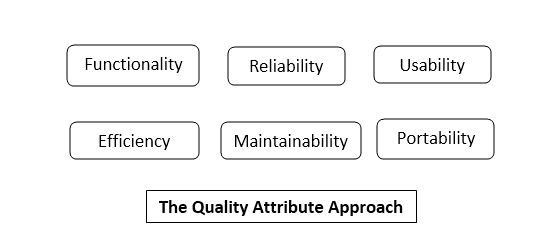
* Preventing defect is the primary goal of defect management approach. But preventing defects completely is not possible and so the purpose is to find out the defects as early as possible and to minimize the impact of the defects.
* To prevent the defects some process should be altered.
* The defect measurement processes should be integrated into the software development process, and thereby the process can be improved.
* Defect information always helps to improve the processes and hence the defect information is very useful for perfect completion of the software developed.

The diagram below explains the various stages of the defect management approach.

[](http://www.helpingtesters.com/wp-content/uploads/2016/09/Defect-Management-Diagram.png)

## ****The Software Quality Assurance Attribute Approach****

There is a  list of attributes which describes the step by step approach to obtain Software Quality Assurance. The attributes are given as in the diagram below:

[](http://www.helpingtesters.com/wp-content/uploads/2016/09/Quality-Attribute-Approach.png)

**Functionality:**The attributes considers the set of all the functions used in the software.

* Suitability: Ensures the functions of the software are appropriate.
* Accuracy: Ensures the accurate usage of the functions.
* Interoperability: Ensure the effective interaction of the software with other components.
* Security: Ensure the software is capable of handling any security issues

**Reliability:**The purpose of the attribute is  to check the capability of the system to perform without delay during any conditions

* Maturity: Less possibility of failure of the software in any activities.
* Recoverability: The rate of recovery ability once a failure occurs.

**Usability:**The purpose is to ensure the use of a function

* Understandability: How much effort a user needs to understand the functions.

**Efficiency:**The attribute depends on the architecture used and the coding practices.

**Maintainability:**The way to analyze and fix a fault/issue in the software

* Analyzability: Finding out the cause of failure.
* Changeability: How the system response to necessary changes.
* Stability: How stable the system is when the changes made.
* Testability: Testing efforts

**Adaptability:**Ability of the system to adopt the changes in its environment.

## ****SQA Activities to Assure the Software Quality****

The Software Quality Assurance of the software is analyzed and ensured by performing a series of activities. The activities are performed as step by step process and the result analysis is reported for the final evaluation process. The activities are performed as step by step process and the result analysis is reported for the final evaluation process.

### ****A Quality Management Plan****

A Quality Management Plan is designed and developed for the Software Quality Assurance Process. The plan includes the proper technical methods to manage the Software Quality Assurance activities. The plan requires a  tracking as a live plan based on the  SDLC.

### ****Applying Software Engineering Techniques****

The software engineering techniques are selected to achieve software quality. The techniques to be used for Software Quality Assurance are determined by analyzing the requirements collected. The requirement evaluation can be done by using some techniques eg: Facilitated  Application Specification Technique[FAST].

Also, a project estimate is prepared with the help of techniques such as Work Break Down[WBS] and Source Line of Code[SLOC] Estimation.

### ****Technical Reviews****

The Formal Technical Reviews[FTR] are conducted to assess the quality and design of the quality management plan. FTR is performed in the presence of the technical people and so will be helpful to find the defects in the early stages. FTR helps to avoid the need for reworking as the reviews in each phase are done with discussing the technical experts.

### ****Applying the Testing Strategy****

The testing strategy is designed and applied. The various levels of testing are designed and scheduled. The testing strategies are designed based on the policies of the company, the stages for each test phase execution are designed and scheduled for the concerned persons. Alpha testing and Beta testing with selected clients are also conducted to test the product before delivered.

### ****Ensuring Process Adherence****

The process adherence is the combination of 2 tasks product evaluation and process monitoring.Product evaluation is the process of ensuring all the requirements identified in the product development result to the completion of the functionalities.

Also, the evaluation process is conducted to monitor the standards and procedures.  
Process Monitoring is the process of comparing actual steps for the procedures with the expected steps designed in the documented procedures.

The process ensures the procedures and control processes described in the procedure documentation are carried out in the correct way and completed without any critical issues. These process evaluations are checked in the Audit Process.

### ****The Change Control Process****

The Change Control is the process which formalizes the request for changes, evaluates the quality/nature of changes, controls the impact of changes. The  Change Control Mechanism is designed and implemented during the design and development stages.

### ****Software Quality Assurance Audits****

Software Quality Assurance Audits inspects the Software Development Process by comparing to the established processes. Software Quality Assurance Auditor is the responsible person who reviews and checks the activities are executed to the highest possible standards. The quality of the project handling can be analyzed only through the results of the review submitted by the Software Quality Assurance Auditor.

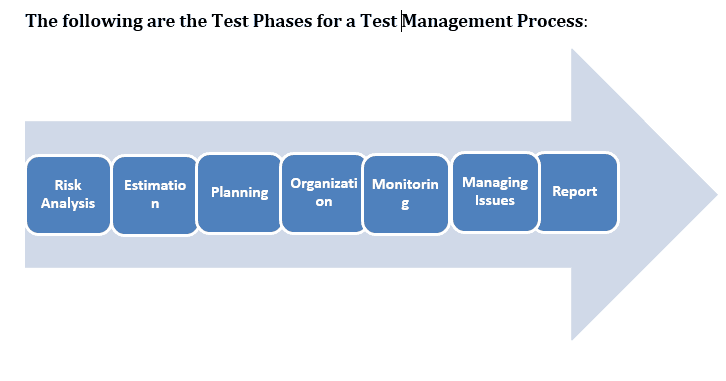
### ****Generate Reports****

Appropriate records for all the activities should be generated for future references. These activities evaluate the quality of a  project and also tests the way of handling project management processes. This will result in a review of the performance of the Test Engineer who is in charge of the Test Management  Processes.

## ****Software Quality Assurance Activities****

The Benefits of Software Quality Assurance are:

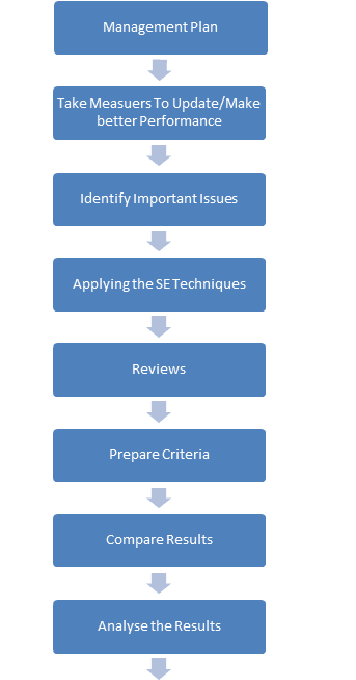
* Monitoring and Improving the Project Management Process.
* Ensuring the Standards are followed for handling procedures.
* Preventing the severe Software Quality Assurance Issues.

[](http://www.helpingtesters.com/wp-content/uploads/2016/09/Test-Phases-for-a-Test-Management-Process.png)

The Software Quality AssurancevActivities are designed and performed based on the test phases scheduled. The Software Quality Assurance diagram below explains each and every step of the Activities designed. In each SQA phase, the Software Quality Assurance team provides consultation and review of the project plan, work products, and procedures with regards to the organizational policies.

After the completion of the Software Assurance Implementation, an output for the test management review and audit is generated, and this report will be the evidence for the test engineer’s management quality and performance.

SOFTWARE QUALITY ASSURANCE ACTIVITIES

[](http://www.helpingtesters.com/wp-content/uploads/2016/09/Software-Quality-Assurance-Activities.png)

Once these activities are completed, the next step is to check for:

* Any defects/weaknesses in any activity/process
* Improve the system performance by correcting those weaknesses on a priority basis.

The software Quality Assurance [SQA] is accomplished by following some standards such as ISO 9000, CMMI or Six Sigma.

## ****The Project Delivery Life Cycle****

The project delivery life cycle incorporates QA activities and the deliverables.The Life Cycle has divided into 5 different phases and the activities and deliverables are associated with each phase.

**PHASE 1: ASSESSMENT**

This is the phase at which an assessment of the requirements are done and developed for realizing certain business objectives and project design. QA Deliverable: Software Quality Assurance Analyst submits a revised document on the requirement analysis and Quality Assurance Plan.

**PHASE 2: PLANNING**

This is the phase at which the strategic plan for the project around the information architecture is developed. The functions for various processes are also designed and double checked.

QA Activities:

* Deciding the standards and procedures.
* Develop Test Matrix: Design the test matrices.Decide the scope for testing and connect the test objectives to the requirement specifications.
* Auditing: The standards and procedures are audited and quality standards are ensured.

  QA Deliverables

* Test Matrix
* Revised Documents on the testing plan.

**PHASE 3: DESIGN**

This is the phase in which all the necessary system components are identified based on phase 1 and 2. Then detailed design specifications are created for each component.

QA Activities:

* Auditing Procedures and Standards
* Design QA PLAN, QA Test  Plan

QA Deliverables

* QA Plan for testing.
* A revision of the test matrix.

**PHASE 4: DEVELOPMENT**

This is the phase at which the developers constructs the project based on the design phase.

QA Activities:

* Planning Test cases: The test cases for the STLC are designed.
* Prepare the Quality Assurance Test cases.
* Prepare the test environment.

QA Deliverables

* Submits a set of Test cases.
* Set up the QA Environment

**PHASE 5: IMPLEMENTATION**

This is the phase in which the team concentrates on the testing and review of all aspects of the system. The team develop proper documentation for system training, market test plans etc…for making the system ready to launch

QA Activities

* Executes all test cases in the QA Test Plan

QA Deliverables

* Test Results and Reports.
* Bug Report.

### ****Conclusion****

The Software Quality Assurance is also known as the Management Review. The importance of the process is that it ensures the test manager performs the right things in the right way without failure in the specified activities. The set of Software Quality Assurance Activities is designed for the project manager to follow the predefined standard processes. Thus Software Companies are in need of an Software Quality Assurance or Management Review team in Test Management Process.