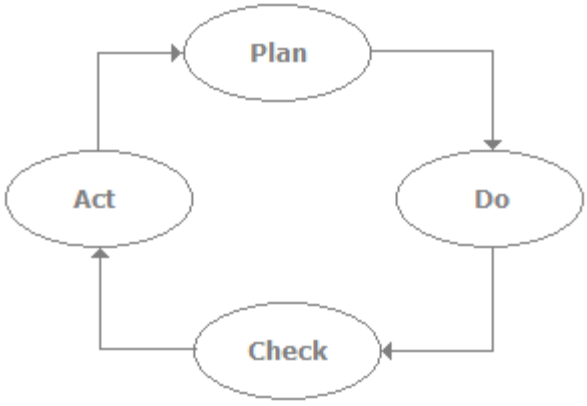
1. Can you explain the PDCA cycle and where testing fits in?

Software testing is an important part of the software development process. In normal software development there are four important steps, also referred to, in short, as the PDCA (Plan, Do, Check, Act) cycle.



Let's review the four steps in detail.

1. **Plan:** Define the goal and the plan for achieving that goal.
2. **Do/Execute:** Depending on the plan strategy decided during the plan stage we do execution accordingly in this phase.
3. **Check:** Check/Test to ensure that we are moving according to plan and are getting the desired results.
4. **Act:** During the check cycle, if any issues are there, then we take appropriate action accordingly and revise our plan again.

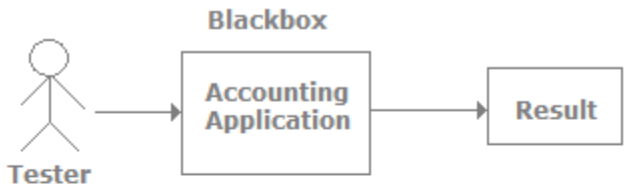
So developers and other stakeholders of the project do the "planning and building," while testers do the check part of the cycle. Therefore, software testing is done in check part of the PDCA cyle.

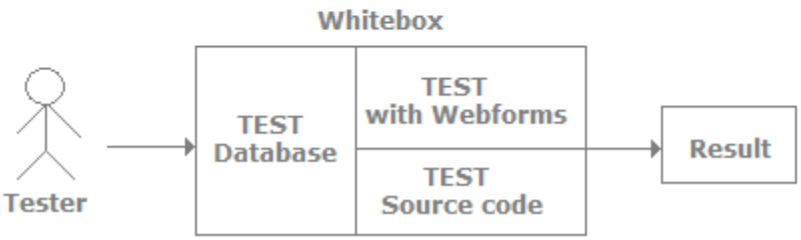
2. What is the difference between white box, black box, and gray box testing?

Black box testing is a testing strategy based solely on requirements and specifications. Black box testing requires no knowledge of internal paths, structures, or implementation of the software being tested.

White box testing is a testing strategy based on internal paths, code structures, and implementation of the software being tested. White box testing generally requires detailed programming skills.

There is one more type of testing called **gray box** testing. In this we look into the "box" being tested just long enough to understand how it has been implemented. Then we close up the box and use our knowledge to choose more effective black box tests.

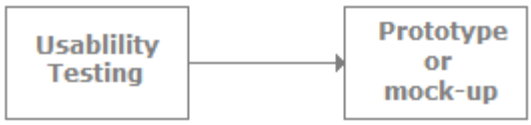




The above figure shows how both types of testers view an accounting application during testing. Black box testers view the basic accounting application. While during white box testing the tester knows the internal structure of the application. In most scenarios white box testing is done by developers as they know the internals of the application. In black box testing we check the overall functionality of the application while in white box testing we do code reviews, view the architecture, remove bad code practices, and do component level testing.

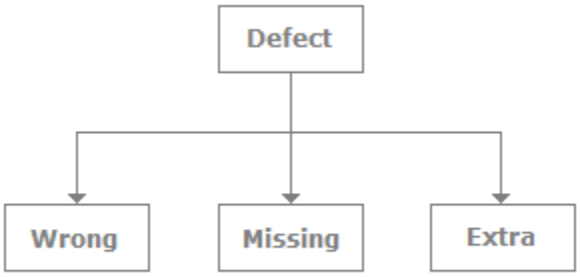
3. Can you explain usability testing?

Usability testing is a testing methodology where the end customer is asked to use the software to see if the product is easy to use, to see the customer's perception and task time. The best way to finalize the customer point of view for usability is by using prototype or mock-up software during the initial stages. By giving the customer the prototype before the development start-up we confirm that we are not missing anything from the user point of view.



4. What are the categories of defects?

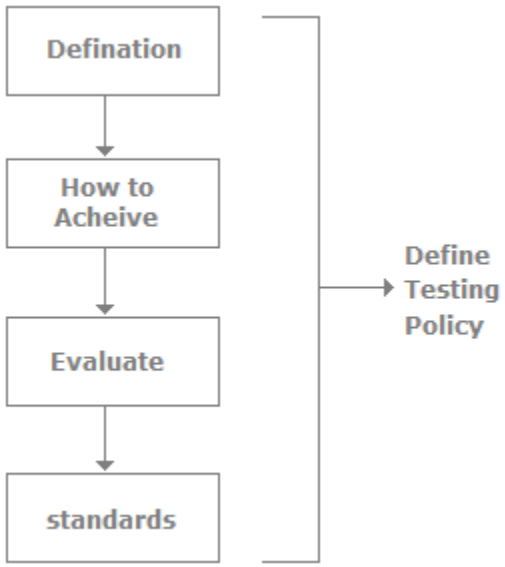
There are three main categories of defects:



1. **Wrong:** The requirements have been implemented incorrectly. This defect is a variance from the given specification.
2. **Missing:** There was a requirement given by the customer and it was not done. This is a variance from the specifications, an indication that a specification was not implemented, or a requirement of the customer was not noted properly.
3. **Extra:** A requirement incorporated into the product that was not given by the end customer. This is always a variance from the specification, but may be an attribute desired by the user of the product. However, it is considered a defect because it's a variance from the existing requirements.

5. How do you define a testing policy?

The following are the important steps used to define a testing policy in general. But it can change according to your organization. Let's discuss in detail the steps of implementing a testing policy in an organization.



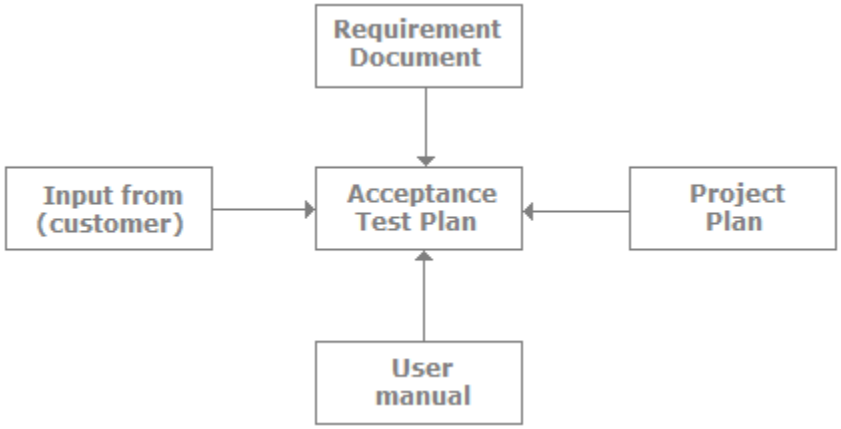
- **Definition:** The first step any organization needs to do is define one unique definition for testing within the organization so that everyone is of the same mindset.
- **How to achieve:** How are we going to achieve our objective? Is there going to be a testing committee, will there be compulsory test plans which need to be executed, etc?.
- **Evaluate:** After testing is implemented in a project how do we evaluate it? Are we going to derive metrics of defects per phase, per programmer, etc. Finally, it's important to let everyone know how testing has added value to the project?.
- **Standards:** Finally, what are the standards we want to achieve by testing? For instance, we can say that more than 20 defects per KLOC will be considered below standard and code review should be done for it.

6. On what basis is the acceptance plan prepared?

In any project the acceptance document is normally prepared using the following inputs. This can vary from company to company and from project to project.

1. **Requirement document:** This document specifies what exactly is needed in the project from the customers perspective.
2. **Input from customer:** This can be discussions, informal talks, emails, etc.
3. **Project plan:** The project plan prepared by the project manager also serves as good input to finalize your acceptance test.

The following diagram shows the most common inputs used to prepare acceptance test plans.



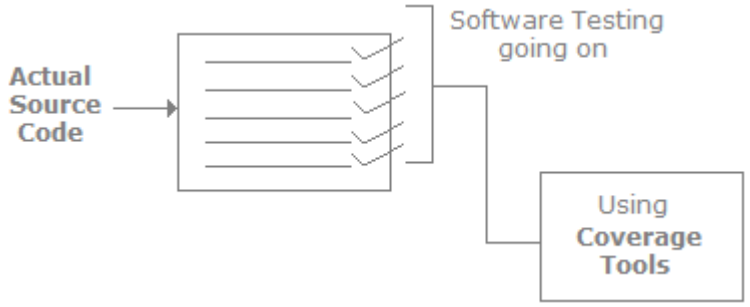
7. What is configuration management?

Configuration management is the detailed recording and updating of information for hardware and software components. When we say components we not only mean source code. It can be tracking of changes for software documents such as requirement, design, test cases, etc.

When changes are done in adhoc and in an uncontrolled manner chaotic situations can arise and more defects injected. So whenever changes are done it should be done in a controlled fashion and with proper versioning. At any moment of time we should be able to revert back to the old version. The main intention of configuration management is to track our changes if we have issues with the current system. Configuration management is done using baselines.

8. How does a coverage tool work?

While doing testing on the actual product, the code coverage testing tool is run simultaneously. While the testing is going on, the code coverage tool monitors the executed statements of the source code. When the final testing is completed we get a complete report of the pending statements and also get the coverage percentage.

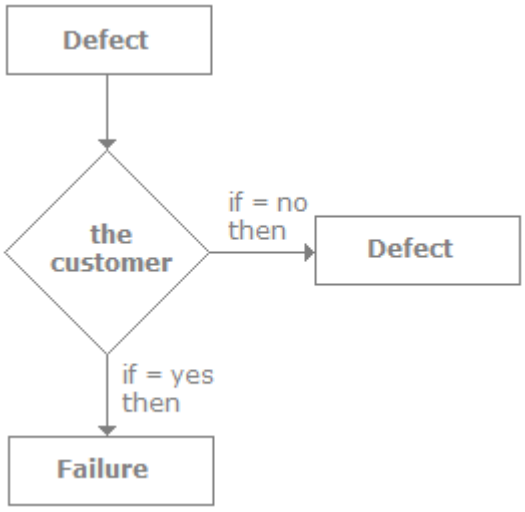


9. Which is the best testing model?

In real projects, tailored models are proven to be the best, because they share features from The Waterfall, Iterative, Evolutionary models, etc., and can fit into real life time projects. Tailored models are most productive and beneficial for many organizations. If it's a pure testing project, then the V model is the best.

10. What is the difference between a defect and a failure?

When a defect reaches the end customer it is called a failure and if the defect is detected internally and resolved it's called a defect.



11. Should testing be done only after the build and execution phases are complete?

In traditional testing methodology testing is always done after the build and execution phases.

But that's a wrong way of thinking because the earlier we catch a defect, the more cost effective it is. For instance, fixing a defect in

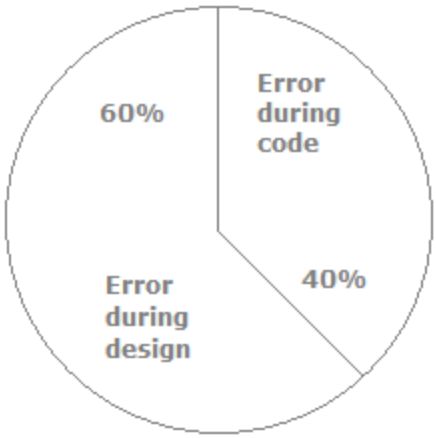
maintenance is ten times more costly than fixing it during execution.

In the requirement phase we can verify if the requirements are met according to the customer needs. During design we can check whether the design document covers all the requirements. In this stage we can also generate rough functional data. We can also review the design document from the architecture and the correctness perspectives. In the build and execution phase we can execute unit test cases and generate structural and functional data. And finally comes the testing phase done in the traditional way. i.e., run the system test cases and see if the system works according to the requirements. During installation we need to see if the system is compatible with the software. Finally, during the maintenance phase when any fixes are made we can retest the fixes and follow the regression testing.

Therefore, Testing should occur in conjunction with each phase of the software development.

12. Are there more defects in the design phase or in the coding phase?

The design phase is more error prone than the execution phase. One of the most frequent defects which occur during design is that the product does not cover the complete requirements of the customer. Second is wrong or bad architecture and technical decisions make the next phase, execution, more prone to defects. Because the design phase drives the execution phase it's the most critical phase to test. The testing of the design phase can be done by good review. On average, 60% of defects occur during design and 40% during the execution phase.



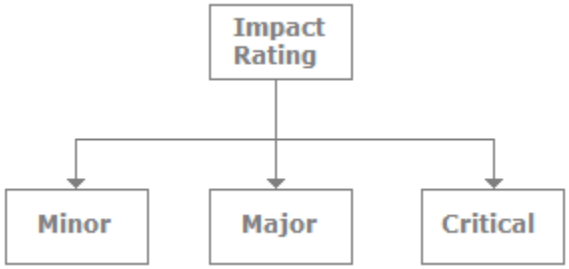
13. What group of teams can do software testing?

When it comes to testing everyone in the world can be involved right from the developer to the project manager to the customer. But below are different types of team groups which can be present in a project.

- Isolated test team
- Outsource - we can hire external testing resources and do testing for our project.
- Inside test team
- Developers as testers
- QA/QC team.

14. What impact ratings have you used in your projects?

Normally, the impact ratings for defects are classified into three types:



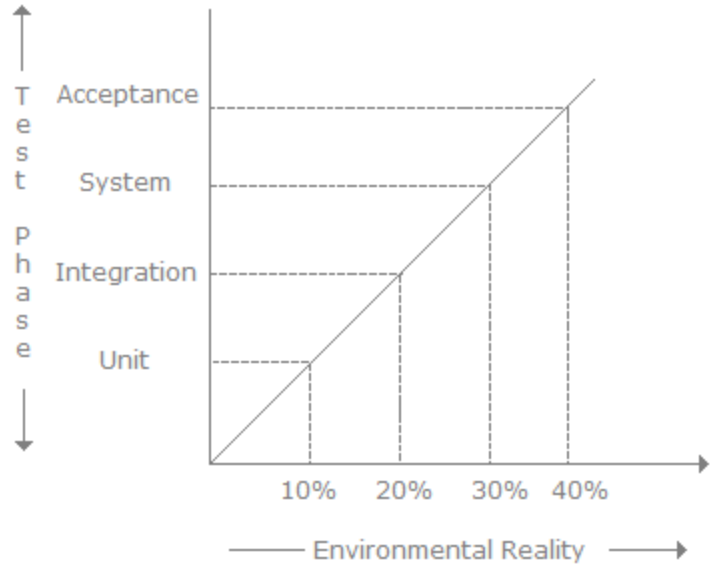
- **Minor:** Very low impact but does not affect operations on a large scale.
- **Major:** Affects operations on a very large scale.
- **Critical:** Brings the system to a halt and stops the show.

15. Does an increase in testing always improve the project?

No an increase in testing does not always mean improvement of the product, company, or project. In real test scenarios only 20% of test plans are critical from a business angle. Running those critical test plans will assure that the testing is properly done. The following graph explains the impact of under testing and over testing. If you under test a system the number of defects will increase, but if you over test a system your cost of testing will increase. Even if your defects come down your cost of testing has gone up.

16. What's the relationship between environment reality and test phases?

Environment reality becomes more important as test phases start moving ahead. For instance, during unit testing you need the environment to be partly real, but at the acceptance phase you should have a 100% real environment, or we can say it should be the actual real environment. The following graph shows how with every phase the environment reality should also increase and finally during acceptance it should be 100% real.



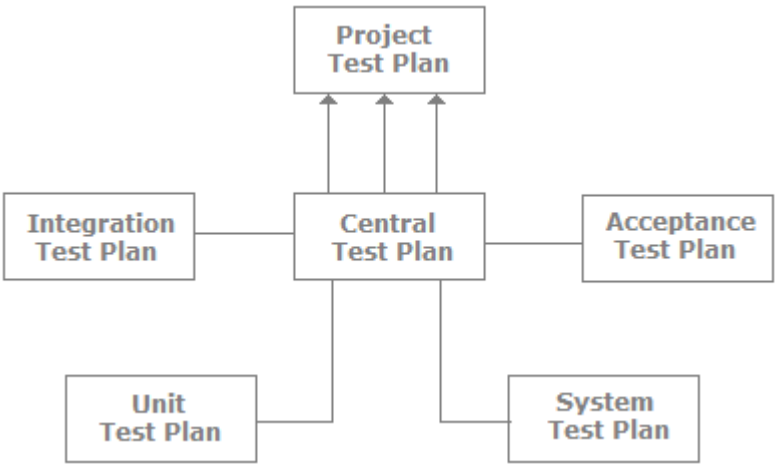
17. What are different types of verifications?

Verification is static type of s/w testing. It means code is not executed. The product is evaluated by going through the code. Types of verification are:

1. **Walkthrough:** Walkthroughs are informal, initiated by the author of the s/w product to a colleague for assistance in locating defects or suggestions for improvements. They are usually unplanned. Author explains the product; colleague comes out with observations and author notes down relevant points and takes corrective actions.
2. **Inspection:** Inspection is a thorough word-by-word checking of a software product with the intention of Locating defects, Confirming traceability of relevant requirements etc.

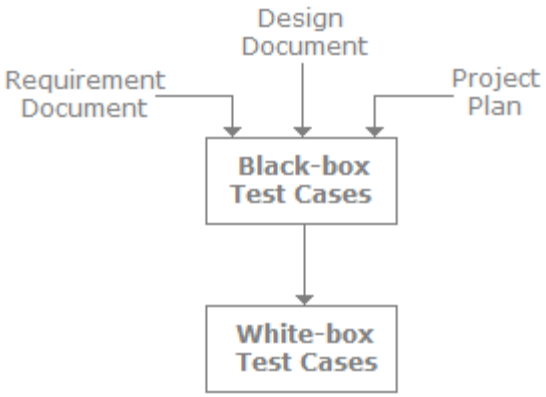
18. How do test documents in a project span across the software development lifecycle?

The following figure shows pictorially how test documents span across the software development lifecycle. The following discusses the specific testing documents in the lifecycle:



- **Central/Project test plan:** This is the main test plan which outlines the complete test strategy of the software project. This document should be prepared before the start of the project and is used until the end of the software development lifecycle.
- **Acceptance test plan:** This test plan is normally prepared with the end customer. This document commences during the requirement phase and is completed at final delivery.
- **System test plan:** This test plan starts during the design phase and proceeds until the end of the project.
- **Integration and unit test plan:** Both of these test plans start during the execution phase and continue until the final delivery.
- **19. Which test cases are written first: white boxes or black boxes?**
- Normally black box test cases are written first and white box test cases later. In order to write black box test cases we need the requirement document and, design or project plan. All these documents are easily available at the initial start of the project. White box test cases cannot be started in the initial phase of the project because they need more architecture clarity which is not available at the start of the project. So normally white box test cases are written after black box test cases are written.

  Black box test cases do not require system understanding but white box testing needs more structural understanding. And structural understanding is clearer i00n the later part of project, i.e., while executing or designing. For black box testing you need to only analyze from the functional perspective which is easily available from a simple requirement document.



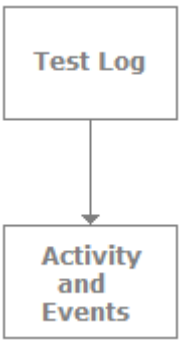
- **20. Explain Unit Testing, Integration Tests, System Testing and Acceptance Testing?**
- Unit testing - Testing performed on a single, stand-alone module or unit of code.

  Integration Tests - Testing performed on groups of modules to ensure that data and control are passed properly between modules.

  System testing - Testing a predetermined combination of tests that, when executed successfully meets requirements.

  Acceptance testing - Testing to ensure that the system meets the needs of the organization and the end user or customer (i.e., validates that the right system was built).
- **21. What is a test log?**
- The IEEE Std. 829-1998 defines a test log as a chronological record of relevant details about the execution of test cases. It's a detailed view of activity and events given in chronological manner.

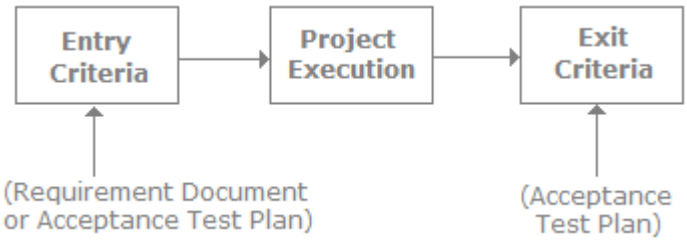
The following figure shows a test log and is followed by a sample test log.



- **22. Can you explain requirement traceability and its importance?**
- In most organizations testing only starts after the execution/coding phase of the project. But if the organization wants to really benefit from testing, then testers should get involved right from the requirement phase.

  If the tester gets involved right from the requirement phase then requirement traceability is one of the important reports that can detail what kind of test coverage the test cases have.
- **23. What does entry and exit criteria mean in a project?**
- Entry and exit criteria are a must for the success of any project. If you do not know where to start and where to finish then your goals are not clear. By defining exit and entry criteria you define your boundaries.

  For instance, you can define entry criteria that the customer should provide the requirement document or acceptance plan. If this entry criteria is not met then you will not start the project. On the other end, you can also define exit criteria for your project. For instance, one of the common exit criteria in projects is that the customer has successfully executed the acceptance test plan.



- **24. What is the difference between verification and validation?**
- Verification is a review without actually executing the process while validation is checking the product with actual execution. For instance, code review and syntax check is verification while actually running the product and checking the results is validation.

25. What is the difference between latent and masked defects?

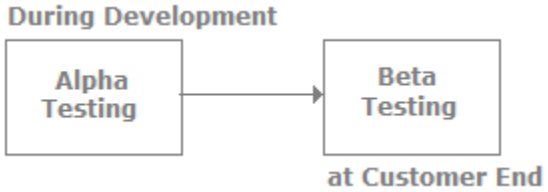
A latent defect is an existing defect that has not yet caused a failure because the sets of conditions were never met.

A masked defect is an existing defect that hasn't yet caused a failure just because another defect has prevented that part of the code from being executed.

26. Can you explain calibration?

It includes tracing the accuracy of the devices used in the production, development and testing. Devices used must be maintained and calibrated to ensure that it is working in good order.

27. What's the difference between alpha and beta testing?

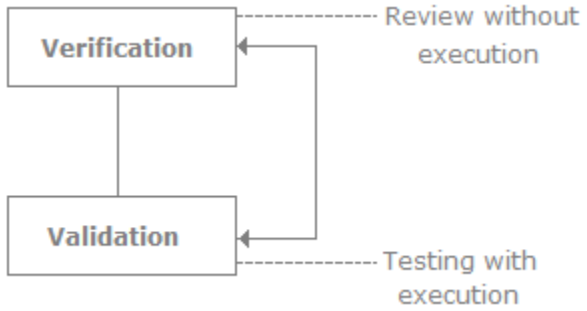


Alpha and beta testing has different meanings to different people. Alpha testing is the acceptance testing done at the development site. Some organizations have a different visualization of alpha testing. They consider alpha testing as testing which is conducted on early, unstable versions of software. On the contrary beta testing is acceptance testing conducted at the customer end.

In short, the difference between beta testing and alpha testing is the location where the tests are done.

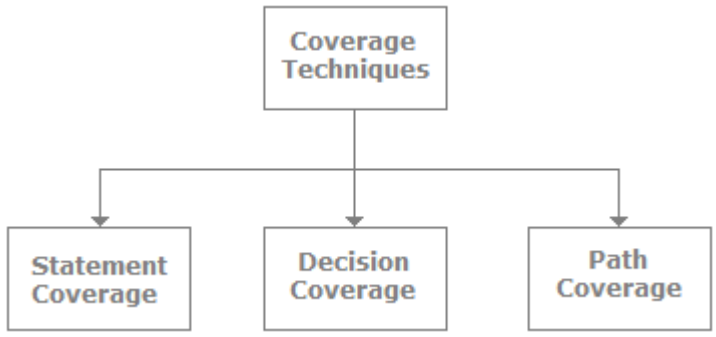
28. How does testing affect risk?

A risk is a condition that can result in a loss. Risk can only be controlled in different scenarios but not eliminated completely. A defect normally converts to a risk.



29. What is coverage and what are the different types of coverage techniques?

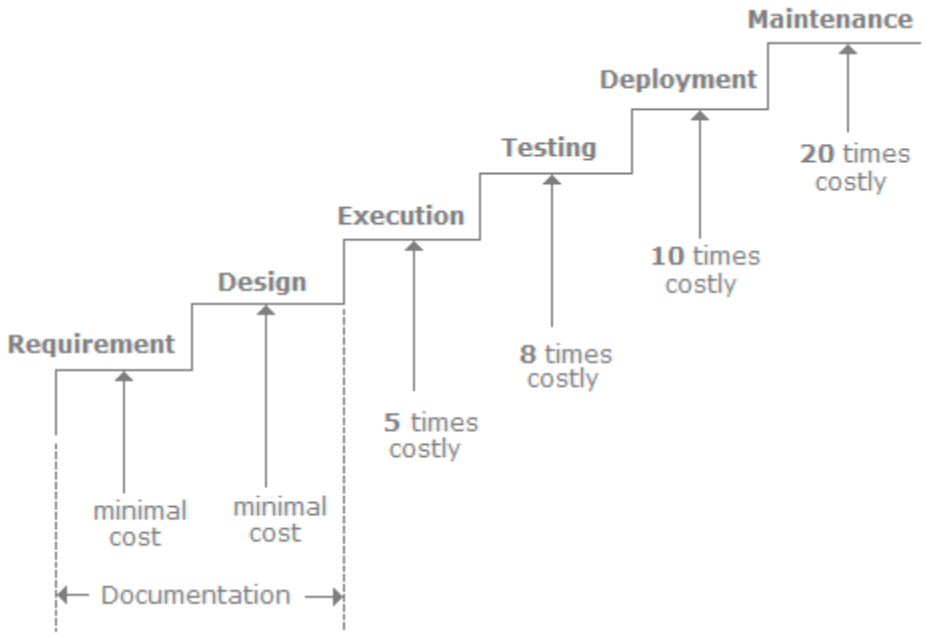
Coverage is a measurement used in software testing to describe the degree to which the source code is tested. There are three basic types of coverage techniques as shown in the following figure:



- **Statement coverage:** This coverage ensures that each line of source code has been executed and tested.
- **Decision coverage:** This coverage ensures that every decision (true/false) in the source code has been executed and tested.
- **Path coverage:** In this coverage we ensure that every possible route through a given part of code is executed and tested.

30. A defect which could have been removed during the initial stage is removed in a later stage. How does this affect cost?

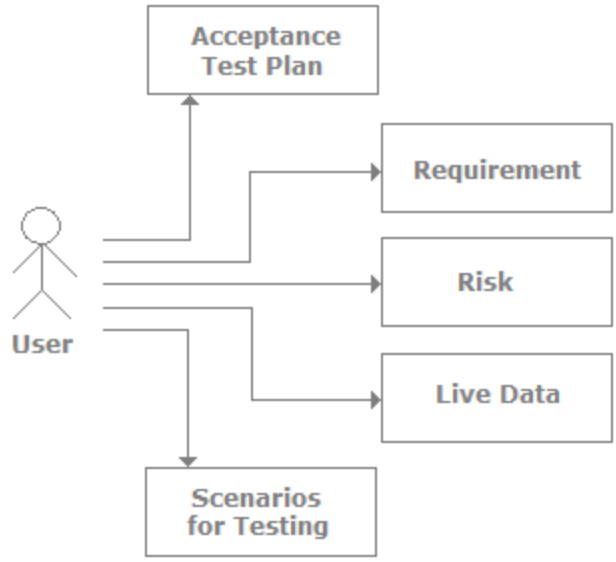
If a defect is known at the initial stage then it should be removed during that stage/phase itself rather than at some later stage. It's a recorded fact that if a defect is delayed for later phases it proves more costly. The following figure shows how a defect is costly as the phases move forward. A defect if identified and removed during the requirement and design phase is the most cost effective, while a defect removed during maintenance is 20 times costlier than during the requirement and design phases.



For instance, if a defect is identified during requirement and design we only need to change the documentation, but if identified during the maintenance phase we not only need to fix the defect, but also change our test plans, do regression testing, and change all documentation. This is why a defect should be identified/removed in earlier phases and the testing department should be involved right from the requirement phase and not after the execution phase.

31. What kind of input do we need from the end user to begin proper testing?

The product has to be used by the user. He is the most important person as he has more interest than anyone else in the project.



From the user we need the following data:

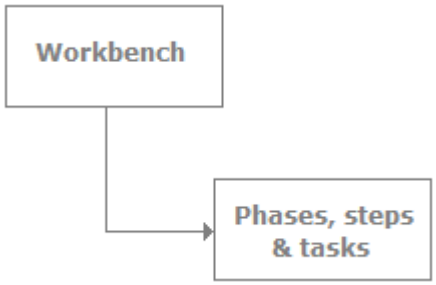
- The first thing we need is the acceptance test plan from the end user. The acceptance test defines the entire test which the product has to pass so that it can go into production.
- We also need the requirement document from the customer. In normal scenarios the customer never writes a formal document until he is really sure of his requirements. But at some point the customer should sign saying yes this is what he wants.
- The customer should also define the risky sections of the project. For instance, in a normal accounting project if a voucher entry screen does not work that will stop the accounting functionality completely. But if reports are not derived the accounting

department can use it for some time. The customer is the right person to say which section will affect him the most. With this feedback the testers can prepare a proper test plan for those areas and test it thoroughly.

- The customer should also provide proper data for testing. Feeding proper data during testing is very important. In many scenarios testers key in wrong data and expect results which are of no interest to the customer.

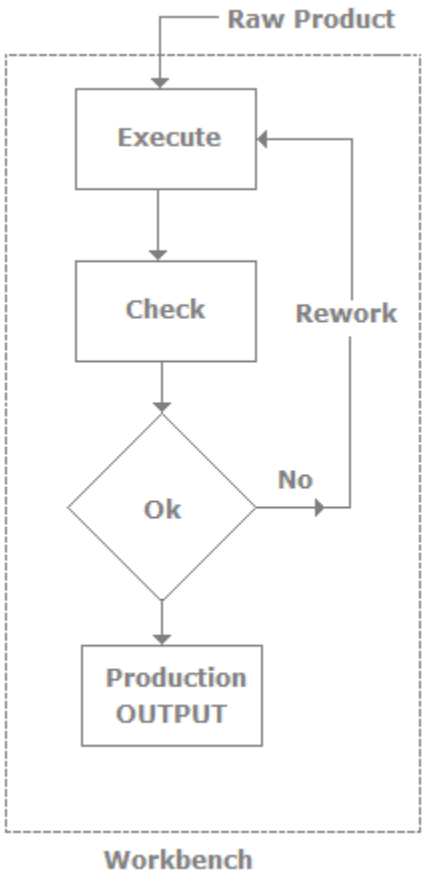
32. Can you explain the workbench concept?

In order to understand testing methodology we need to understand the workbench concept. A Workbench is a way of documenting how a specific activity has to be performed. A workbench is referred to as phases, steps, and tasks as shown in the following figure.



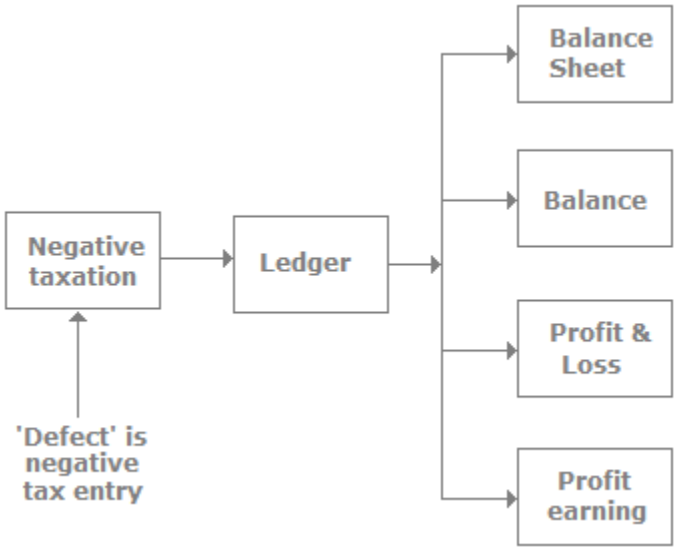
There are five tasks for every workbench:

- **Input:** Every task needs some defined input and entrance criteria. So for every workbench we need defined inputs. Input forms the first steps of the workbench.
- **Execute:** This is the main task of the workbench which will transform the input into the expected output.
- **Check:** Check steps assure that the output after execution meets the desired result.
- **Production output:** If the check is right the production output forms the exit criteria of the workbench.
- **Rework:** During the check step if the output is not as desired then we need to again start from the execute step.



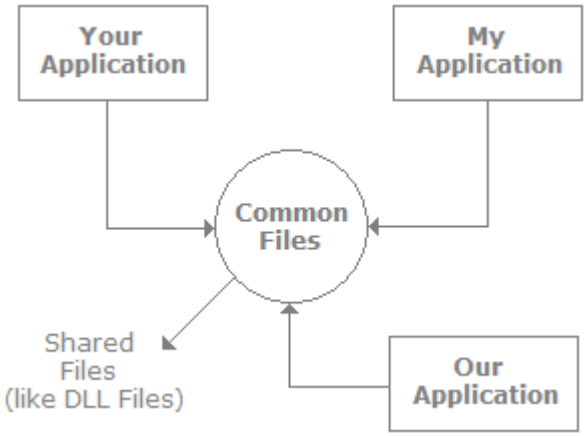
33. Can you explain the concept of defect cascading?

Defect cascading is a defect which is caused by another defect. One defect triggers the other defect. For instance, in the accounting application shown here there is a defect which leads to negative taxation. So the negative taxation defect affects the ledger which in turn affects four other modules.



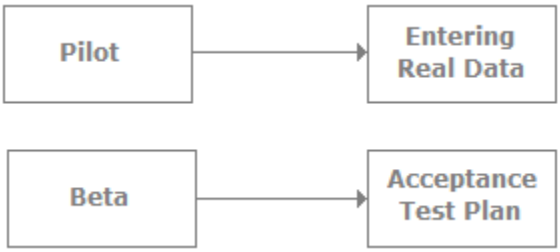
34. Can you explain cohabiting software?

When we install the application at the end client it is very possible that on the same PC other applications also exist. It is also very possible that those applications share common DLLs, resources etc., with your application. There is a huge chance in such situations that your changes can affect the cohabiting software. So the best practice is after you install your application or after any changes, tell other application owners to run a test cycle on their application.



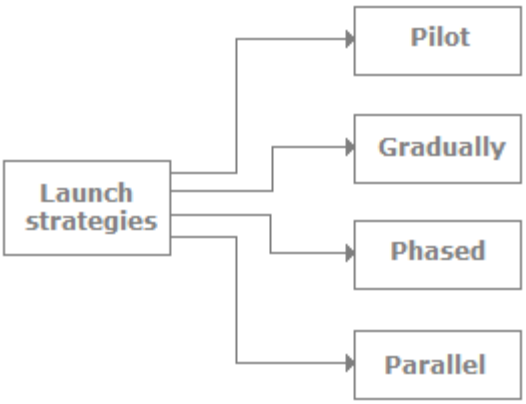
35. What is the difference between pilot and beta testing?

The difference between pilot and beta testing is that pilot testing is nothing but actually using the product (limited to some users) and in beta testing we do not input real data, but it's installed at the end customer to validate if the product can be used in production.



36. What are the different strategies for rollout to end users?

There are four major ways of rolling out any project:



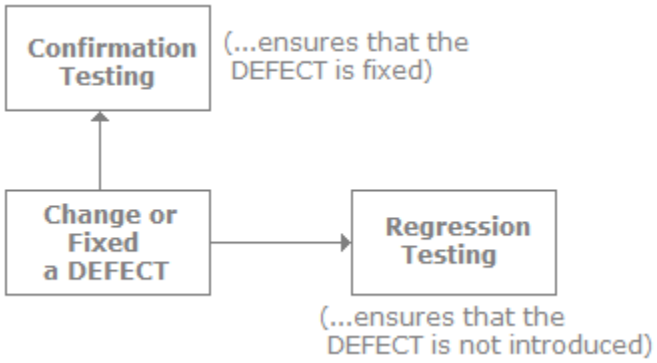
- **Pilot:** The actual production system is installed at a single or limited number of users. Pilot basically means that the product is actually rolled out to limited users for real work.
- **Gradual Implementation:** In this implementation we ship the entire product to the limited users or all users at the customer end. Here, the developers get instant feedback from the recipients which allow them to make changes before the product is available. But the downside is that developers and testers maintain more than one version at one time.

- **Phased Implementation:** In this implementation the product is rolled out to all users in incrementally. That means each successive rollout has some added functionality. So as new functionality comes in, new installations occur and the customer tests them progressively. The benefit of this kind of rollout is that customers can start using the functionality and provide valuable feedback progressively. The only issue here is that with each rollout and added functionality the integration becomes more complicated.
- **Parallel Implementation:** In these types of rollouts the existing application is run side by side with the new application. If there are any issues with the new application we again move back to the old application. One of the biggest problems with parallel implementation is we need extra hardware, software, and resources.
- **37. What's the difference between System testing and Acceptance testing?**
- *Acceptance testing* checks the system against the "Requirements." It is similar to System testing in that the whole system is checked but the important difference is the change in focus:

  System testing checks that the system that was specified has been delivered. *Acceptance testing* checks that the system will deliver what was requested. The customer should always do Acceptance testing and not the developer.

  The customer knows what is required from the system to achieve value in the business and is the only person qualified to make that judgement. This testing is more about ensuring that the software is delivered as defined by the customer. It's like getting a green light from the customer that the software meets expectations and is ready to be used.
- **38. Can you explain regression testing and confirmation testing?**
- Regression testing is used for regression defects. Regression defects are defects occur when the functionality which was once working normally has stopped working. This is probably because of changes made in the program or the environment. To uncover such kind of defect regression testing is conducted.

  The following figure shows the difference between regression and confirmation testing.



If we fix a defect in an existing application we use confirmation testing to test if the defect is removed. It's very possible because of this defect or changes to the application that other sections of the application are affected. So to ensure that no other section is affected we can use regression testing to confirm this.