1 ANS:

Verification is the process of checking that a software achieves its goal without any bugs. It is the process to ensure whether the product that is developed is right or not. It verifies whether the developed product fulfills the requirements that we have. Verification is static testing.
Verification means **Are we building the product right?**

Validation is the process of checking whether the software product is up to the mark or in other words product has high level requirements. It is the process of checking the validation of product i.e. it checks what we are developing is the right product. it is validation of actual and expected product. Validation is the dynamic testing.
Validation means **Are we building the right product?**

The difference between Verification and Validation is as follow:

Verification	Validation
It includes checking documents, design, codes and programs.	It includes testing and validating the actual product.
Verification is the static testing.	Validation is the dynamic testing.
It does *not* include the execution of the code.	It includes the execution of the code.
Methods used in verification are reviews, walkthroughs, inspections and desk-checking.	Methods used in validation are Black Box Testing, White Box Testing and non-functional testing.
It checks whether the software conforms to specifications or not.	It checks whether the software meets the requirements and expectations of a customer or not.
It can find the bugs in the early stage of the development.	It can only find the bugs that could not be found by the verification process.
The goal of verification is application and software architecture and specification.	The goal of validation is an actual product.
Quality assurance team does verification.	Validation is executed on software code with the help of testing team.
It comes before validation.	It comes after verification.
It consists of checking of documents/files and is performed by human.	It consists of execution of program and is performed by computer.

OR

Monitoring and Controlling are processes needed to track, review, and regulate the progress and performance of the project. It also identifies any areas where changes to the project management method are required and initiates the required changes.

Monitor and control project work: The generic step under which all other monitoring and controlling activities fall under:

Perform integrated change control: The functions involved in making changes to the project plan. When changes to the schedule, cost, or any other area of the project management plan are necessary, the program is changed and re-approved by the project sponsor.

Validate scope: The activities involved with gaining approval of the project's deliverables.

Control scope: Ensuring that the scope of the project does not change and that unauthorized activitiesare not performed as part of the plan (scope creep).

Control schedule: The functions involved with ensuring the project work is performed according to the schedule, and that project deadlines are met.

Control costs: The tasks involved with ensuring the project costs stay within the approved budget.

Control quality: Ensuring that the quality of the project?s deliverables is to the standard defined in the project management plan.

Control communications: Providing for the communication needs of each project stakeholder.

Control Risks: Safeguarding the project from unexpected events that negatively impact the project's budget, schedule, stakeholder needs, or any other project success criteria.

Control procurements: Ensuring the project's subcontractors and vendors meet the project goals.

Control stakeholder engagement: The tasks involved with ensuring that all of the project's stakeholdersare left satisfied with the project work.

2 ANS : There are four types of software maintenance:

Corrective Software

Adaptive Software

Perfective Software Preventive

Software

Corrective software maintenance is what one would typically associate with the maintenance of any kind. Correct software maintenance addresses the errors and faults within software applications that could impact various parts of your software, including the design, logic, and code. These corrections usually come from bug reports that were created by users or customers – but corrective software maintenance can help to spot them before your customers do, which can help your brand's reputation.

Adaptive Software Maintenance

Adaptive software maintenance becomes important when the environment of your software changes.This can be brought on by changes to the operating system, hardware, software dependencies, Cloud storage, or even changes within the operating system. Sometimes, adaptive software maintenance reflects organizational policies or rules as well. Updating services, making modifications to vendors, orchanging payment processors can all necessitate adaptive software maintenance.

Perfective Software Maintenance

Perfective software maintenance focuses on the evolution of requirements and features that existing inyour system. As users interact with your applications, they may notice things that you did not or suggestnew features that they would like as part of the software, which could become future projects or enhancements. Perfective software maintenance takes over some of the work, both adding features that can enhance user experience and removing features that are not effective and functional. This can include features that are not used or those that do not help you to meet your end goals.

Preventive Software Maintenance

Preventative Software Maintenance helps to make changes and adaptations to your software so that itcan work for a longer period of time. The focus of the type of maintenance is to prevent the deterioration of your software as it continues to adapt and change. These services can include optimizing code and updating documentation as needed.

Preventative software maintenance helps to reduce the risk associated with operating software for a long time, helping it to become more stable, understandable, and maintainable.

Updating software environments, reducing deterioration, and enhancing what is already there to help satisfy the needs of all users are also included in the software maintenance examples.

OR

PRODUCT IMPLEMENTATION:

The product that has been developed and thoroughly tested now needs to be implemented at a customer site.

• You need to prepare all master data and test transaction data for testing the implemented product.

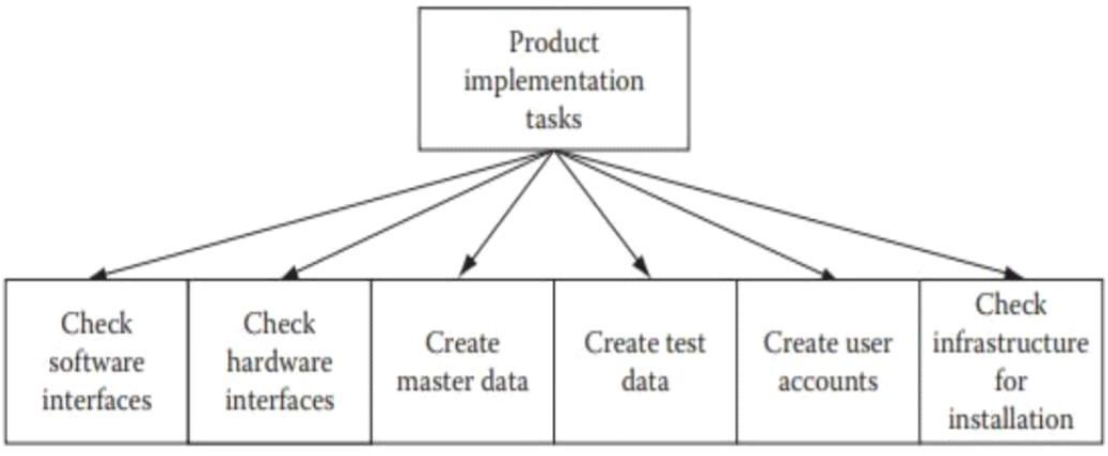
• You need to get all required hardware and software that need to be there for installing your software product.

• You need to make sure that you have developed and tested all the hardware and software interfaces for integrating your product, with existing legacy systems and infrastructure.

*****A product software implementation method is a blueprint to get users and/or organizations running with a specific software product.

• The method is a set of rules and views to cope with the most common issues that occur when implementing a software product.

*****It is stated that the implementation of (product) software consumes up to 1/3 of the budget of a software purchase (more than hardware and software requirements together).



3. (i) Test Automation

(ii) Defect Tracking

ANS: (i) Most testing tasks are done manually, as they are still difficult to automate. Wherever automation is possible, it can be evaluated. Care should also be taken not to do automation blindly. This is because the initial effort for automation is more than manual testing.Testing tasks include requirements and design document review, test case scenario creation, test case creation, test case execution, test case management, and defect tracking. Out of these tasks, test case execution and test case management are the only tasks for which good automation tools are available.

(ii) Defect tracking is one of the most important activities in a test project. During defect tracking it is ensured that defects are logged and get fixed.Defect count per hour per day is a common way of measuring performance of a test team. If the testing is done for an in-house software product, traditionally, it used to not be a performance evaluation measurement. What really counted was the number of defects found in production when the software product was deployed and used by end users. But it is too late a performance measurement.

A good defect tracking application should be deployed on a central server that is accessible to all test and development teams. Each defect should be logged in such a way that it could be understood by both development and testing teams.

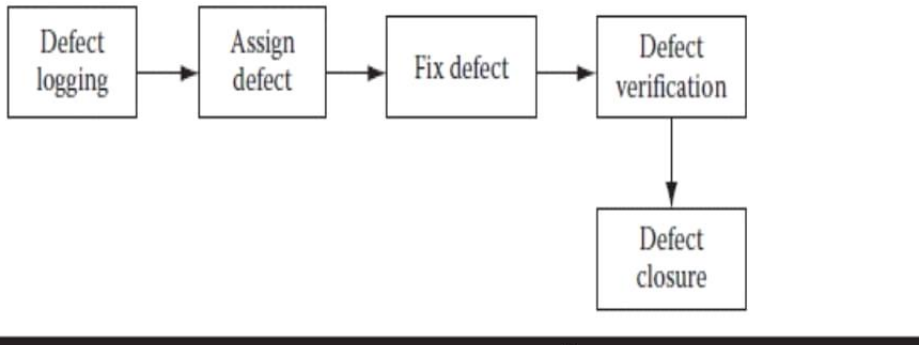


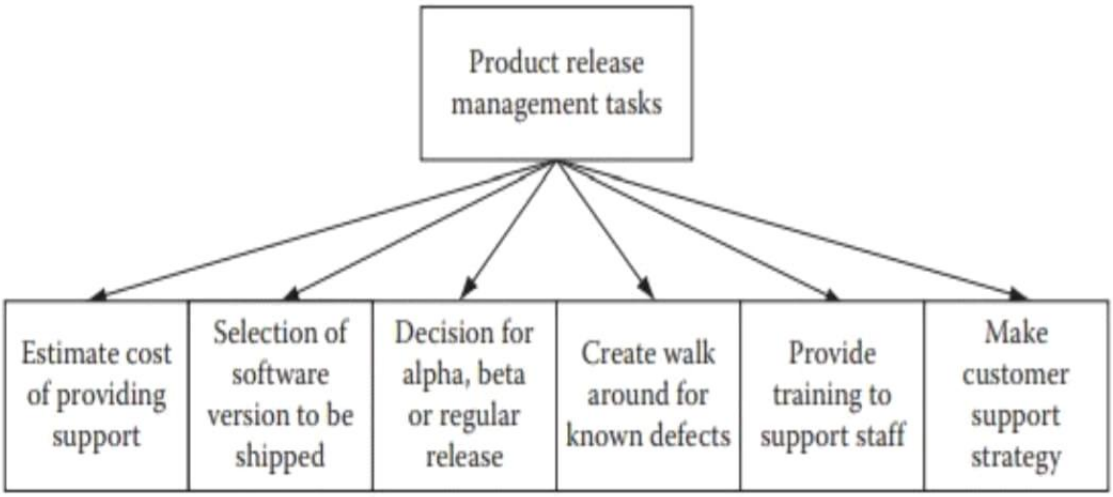
Figure 13.6 Defect life cycle.

OR

Q. Software product release with example.

Project teams working for software product vendors struggle to keep pace with release of the software product.

- There is pressure from the market to launch new versions by certain date.
- New features are to be added, porting the product to new platforms, old features are to be enhanced, existing bugs are to be removed.
- It is a constant struggle that calls for good product release strategies.
- Bargaining also has to be done for other requirements of bug fixes, feature enhancements, etc
- Product release management is such a dynamic environment that if proper planning is not done at a minute level and constant vigilance is not applied over project activities, then a huge mess can be created and there will be no time to clear it.
- The cost of support, depending on the number of estimated users, walk around, and remaining bugs should be figured out.
- These measures will ensure that the product is transitioned into market without facing major difficulties.



Release Management cycle