CHAPTER

6

SOFTWARE VERIFICATION AND VALIDATION

OBJECTIVES

This chapter aims to give an understanding of verification and validation and the different stages of software development under these two domains.

6.1 INTRODUCTION

We have seen two views of quality, viz. 'Conformance to requirements' which is defined as a developers view of quality and 'Fitness for use' which is defined as a customers view of quality. The first part may be termed 'verification' of the process used for developing a product while the second part may be called 'validation' of a product which is created by testers and given to customers for acceptance testing/using it in production.

Quality planning includes planning for the processes used to build and test software to create a right product. It may include software certification which may be termed 'validation', as well as finding conformance to the requirements defined by the processes and standards called 'verification'. Quality is not an accident but an outcome of planned activities carried out in various phases of Software Development Life Cycle (SDLC).

Quality planning in an organisation includes the following stages.

- Definition of processes, procedures, standards, and guidelines for developing and testing software. These may be refined and redefined as defects are found in the application, so that processes become more and more matured.
- Performing quality operations such as verification and validation as per process definition at different phases of SDLC.
- Auditing work products to ensure that processes are followed correctly and work products match exit criteria for the phase under consideration.
- Collecting metrics and defining actions for lacunae found in processes and opportunities for improvements on the basis of quantitative data.

E-next
https://E-next.in
Scanned by CamScanner

Verification and validation activities are two branches of software testing. They are complementary to each other, and not substitutes of each other. Verification and validation are completely dependent on each other. They are not even mutually exclusive. In addition to auditing, which works on sampling basis, verification and validation ensure that processes are followed correctly, the process definition is also appropriate and process capability is ensured by optimising processes through feedback loop. Moreover, the work product also matches with exit criteria as defined by various stages of development. Many organisational metrics related to processes as well as product are defined by collecting data through verification and validation. Proper verification and validation together can ensure quality of work product as well as quality of processes used for developing and testing software which will finally help in achieving business objectives. Only proper verification without proper validation, and vice versa cannot ensure a good software product. Rather, it needs a combination of both.

6.2 VERIFICATION

Verification is a disciplined approach to evaluate whether a software product fulfills the requirements or conditions imposed on them by the standards or processes. It is done to ensure that the processes and procedures defined by the customer and/or organisation for development and testing are followed correctly.

Verification is also called 'static technique' as it does not involve execution of any code, program or work product. It is done by systematically reading and assessing the contents of work product with an intention of detecting defect with respect to standards or processes. It helps in identification of defect and its location which then assists in correction. Some people follow a methodology of finding and fixing verification defects then and there only. One must do a root cause analysis and take corrective as well as preventive actions on the defects found.

6.2.1 ADVANTAGES OF VERIFICATION

- Verification can confirm that the work product has followed the processes correctly as defined by an organisation or customer (as the case may be). There is higher probability of success due to following processes religiously.
- It can find defects—in terms of deviations from standards—easily and also, the location of the defects. This helps in fixing the defects easily as well as conducting root cause analysis so that the same defects are not repeated.
- It can reduce the cost of finding and fixing defects as each work product is reviewed and corrected faster. Sometimes, defects are fixed as and when they are found.
- It can locate the defect easily as the work product under review is yet to be integrated with other work products. Cost of fixing defect is less as there is no/minimum impact on other parts of software.
- It can be used effectively for training people about processes and standards. Typically peer review, superior reviews, and walkthrough can be excellent tools for training.

6.2.2 DISADVANTAGES OF VERIFICATION

- It cannot show whether the developed software is correct or not. Rather, it shows whether the processes have been followed or not. If processes are not capable, then the outcome may not be good.

https://E-next.in
Scanned by CamScanner

- Actual working software may not be assessed by verification as it does not cover any kind of execution of a work product. 'Fit for use' cannot be assessed in verification.

6.2.3 PREREQUISITES FOR VERIFICATION

Depending upon the method followed and the work product under verification, there may be different inputs required for verification. Figure 6.1 is an indicative workbench definition for verification. It may differ significantly from one work product to another, for different customers and organisations.

For any process to happen, there must be some inputs coming to the work bench from the previous work bench, and some outputs going out from the work bench to the next work bench. Inputs for the verification process may include work product, quality plan or verification plan, and output from the verification work bench may include verification report, and defects. Some tools, checklists, guidelines, process definitions of verification, and standards are required for successful verification activity.

Prerequisites for verification may include the following

- Training required by people for conducting verification
- Standards, guidelines, and tools to be used during verification
- Verification Do and Check process definition

6.3 VERIFICATION WORK BENCH

A verification work bench is where verification activities are conducted, and may be a physical or virtual entity. For every work bench, the following basic things are required.

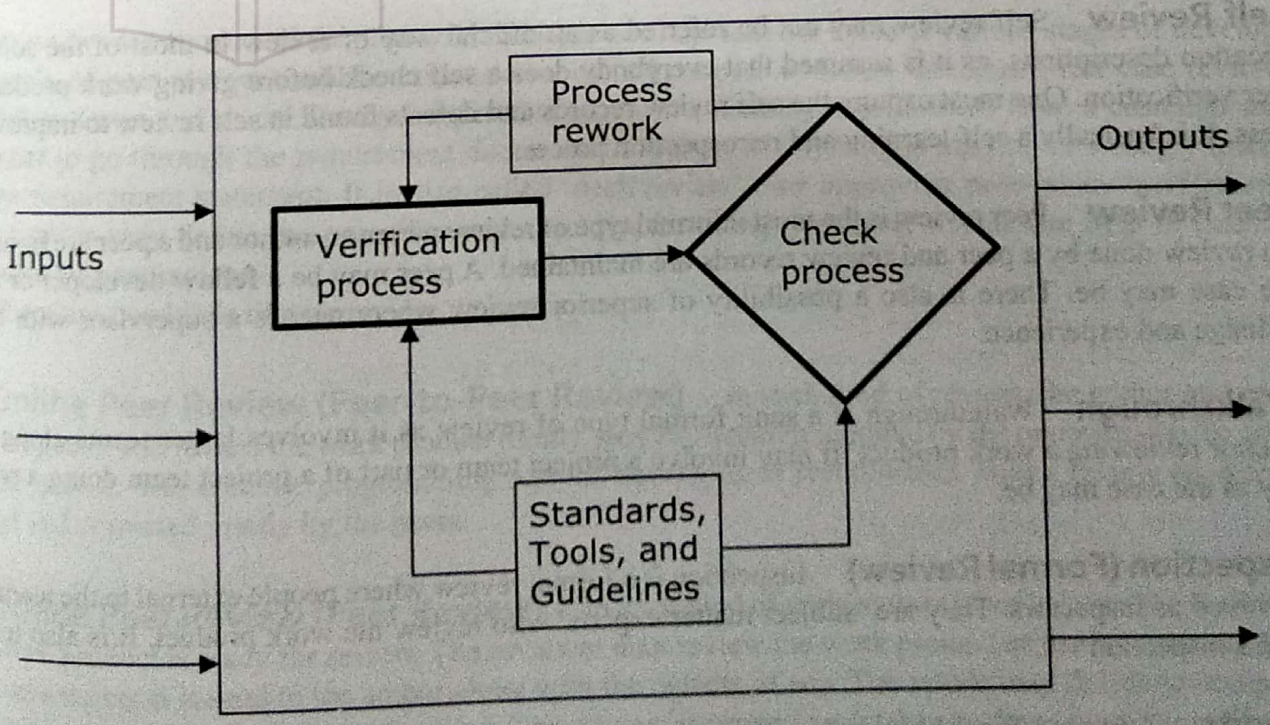

Fig. 6.1 Verification workbench

https://E-next.in
Scanned by CamScanner

Inputs There must be some entry criteria definition when inputs are entering the work bench. This definition should match with the output criteria of the earlier work bench. For example, if we are verifying a code file, then we must have a written and compilable code as an input to the work bench along with verification plan, verification artifacts, etc.

Outputs Similarly to the entry criteria, there must be some exit criteria from work bench which should match with input criteria for the next work bench. Outputs may include review comments and the work product with defects if any.

Verification Process Verification process must describe step-by-step activities to be conducted in a work bench. It must also describe the activities done while verifying the work product under review.

Check Process Check process must describe how the verification process has been checked. It is not a verification of the work product but a verification of the process used for verification. Quality plan must define the objectives to be achieved, and check processes must verify that the objectives have been really achieved.

Standards, Tools, and Guidelines These may be termed 'the tools' available for conducting verification activities. There may be coding guidelines or testing standards available for verifications. Sometimes checklists are used for doing verification.

6.4 METHODS OF VERIFICATION

There are many methods available for verification of software work products. Some of them are listed below.

- **Self Review** Self review may not be referred as an official way of review in most of the software verification descriptions, as it is assumed that everybody does a self check before giving work product for further verification. One must capture the self review records and defects found in self review to improve the process. It is basically a self-learning and retrospection process.

- **Peer Review** Peer review is the most informal type of review where an author and a peer are involved. It is a review done by a peer and review records are maintained. A peer may be a fellow developer or tester as the case may be. There is also a possibility of superior review where peer is a supervisor with better knowledge and experience.

- **Walkthrough** Walkthrough is a semi formal type of review as it involves larger teams along with the author reviewing a work product. It may involve a project team or part of a project team doing a review jointly as the case may be.

- **Inspection (Formal Review)** Inspection is a formal review where people external to the team may be involved as inspectors. They are 'subject matter experts' who review the work product. It is also termed 'Fagan's inspection'.

- **Audits** Audit is a formal review based on samples. Audits are conducted by auditors who may or may not be experts in the given work product.

E-next THE NEXT LEVEL OF EDUCATION

https://E-next.in Scanned by CamScanner

6.4.1 SELF REVIEW

Self review is not considered as an official review in software development as it is expected to be done by the author or creator of an artifact by default. It is a primary expectation that no one must say that his work is done before doing a self review and being satisfied that there are no problems in the output. Self review is excellent in defect prevention through self learning or retrospection. Defects found in self review can help in self education and self improvement.

Advantages of Self Review

- Self reviews are highly flexible with respect to time and defect finding, as one need not take an appointment or allocate specific time while doing it. It may be an online activity.
- Self review is an excellent tool for self learning, as learning from mistakes is a good way of improvement.
- There is no ego involved in self review as the findings need not be shared with anybody.

Disadvantages of Self Review

- Approach- or understanding-related defects may not be found in self review as it is difficult to think in some other way.
- People involved in self review may not conduct a review in reality due to time and focus issues. For example, developers may concentrate more on creating code rather than reviewing the code developed by them.
- Something which has been implemented correctly in initial development may get changed due to personal issues. Some people with less self confidence may change the correct implementation.

6.4.2 PEER REVIEW

Peer reviews are conducted frequently in software development life cycle at various stages of development. Example of peer review could be code review done by a peer or fellow developer, test case review done by fellow tester and so on. It may happen that a person collecting requirements from a customer may ask his peer to go through the requirement document to find out if anything is missing or wrongly interpreted in the requirement statement. It is also called 'desk review'. For improving peer-review performance, an organisation may define the processes and checklist for doing peer review. This can help in tracking the outcome of peer reviews.

There are two kinds of peer review

- **Online Peer Review (Peer-to-Peer Review)** In such kind of review, the author and reviewer meet together and review the work product jointly. Any explanation required by the reviewer may be provided by the author. The extreme programming way of development recommends online review as defects are found and corrected jointly by the peers.

- **Offline Peer Review (Peer Review)** In such kind of review, the author informs the reviewer that the work product is ready for review. The reviewer may review the work product as per his time availability. The review report is send to the author along with the defects, if any. The author may decide to accept/reject the review comments or defects identified. This review is prone to mistakes as there is no joint review, and the author and the reviewer do not discuss the defects. It is more on an opinion basis.

E-next THE NEXT LEVEL OF EDUCATION

https://E-next.in Scanned by CamScanner

Advantages of Peer Review

- Peer reviews are highly informal and unplanned in nature. It can happen at any time during development/testing life cycle. Sometimes, peer review is done jointly by the author and the peer, while at other times, the peer may complete the review of the artifact on his own and give feedback to the author.
- There are no (or less) egos or pride attached between the peers, and defects are accepted easily. Peer-to-peer relationships are important in such reviews.
- Mostly, defects are discussed and decision is reached very informally. This helps in finding and fixing the defects fast.
- Peer review is an excellent tool for educating peers about the artifact. While reviewing the artifacts, a reviewer must have some basic knowledge about the contents and background information. Thus, more number of people can be aware about the contents and approach followed in each artifact in addition to an author.

Disadvantages of Peer Review

- The other person doing the review may or may not be an expert on the artifacts under review, and his/her suggestions may not be valid always. Sometimes, a peer may change the work product without informing the author, or without noting down the defects in review reports. This may happen in offline reviews where a peer may make changes without telling the author.
- Peers may fix the defects without recording them, as they may not like to show defects of their partners/friends. Root causes of the defects may not be analysed as defect fixes may not be known.
- Sometimes, the correct implementation may get replaced by the wrong one as reviewer may update the artifact without recording a defect. There is no discussion, and hence, updation may go unnoticed.
- Approach-related defects may not be fixed, if both the author and the reviewer are at the same status and understand things in a similar way.

E-next THE NEXT LEVEL OF EDUCATION

6.4.3 SUPERIOR REVIEW

Sometimes, superior review is also considered as peer review, as a superior may be considered as peer of the author with some more experience and visibility. Superior reviews are conducted to avoid some limitations of peer review related to approach, visibility, etc. A superior such as team leader or module leader may have to review the artifacts made by his subordinates to find its suitability with reference to expected outcome.

Advantages of Superior Review

- Approach-related defects can be found easily as superiors are expected to have a better view of the entire project. A superior may have better understanding of overall requirements and designs than the author.
- Superiors can share better ways of doing things—learned from their experience—with their subordinates. Thus, superior review is also used as a tool for training.

Disadvantages of Superior Review

- Superior review can become a bottleneck if everything developed by a team needs to be reviewed by the superior. For example, if there are 10 people in a team of a module leader, then reviewing their work product may hamper his own work.
- It does not work efficiently when the superior is also new to the domain/approach, or if the project is in research and development kind of environment where exact solution or even approach may not be known to anybody.

https://E-next.in Scanned by CamScanner

6.4.4 WALKTHROUGH

Walkthrough is more formal than peer review but less formal than inspection. It is termed 'semi formal review' as only related people are involved in walkthrough. In a typical walkthrough, some members of a project team are involved in examining an artifact under review. Author of an artifact presents it, and the entire team discusses various aspects of the artifact. The comments/minutes of the discussion are documented and actions are tracked till closure. Walkthroughs can be effectively used for communication between the team.

Advantages of Walkthrough

- Walkthrough is an excellent tool for collaboration where joint decisions can be made by the team. Each and every member present must be involved in making decisions.
- Walkthrough is also useful for training the entire team at one time. Test scenarios and designs are generally subjected to walkthrough.
- People understand what they can expect from the document/approach under consideration.
- Problems can get recorded and suggestions can be received from the team for improving the work product further.

Disadvantages of Walkthrough

- Availability of people can be an issue when teams are large. Multilocations and distributed teams are major challenges in walkthrough. It is more formal than peer and procuring people can be a challenge.
- Team members may not be experts in giving comments and may need some training and basic knowledge about the project, artifact under walkthrough and so on.
- Time can be a constraint where schedules are very tight.

6.4.5 INSPECTION

Inspection is a very formal way of reviewing the work product. Agenda of an inspection is decided in advance. Moreover, people attending the inspection are given inputs and background documents beforehand. Generally, the author of the work product is not allowed/expected to be present during inspection. A presenter presents a work product, and a recorder makes notes of comments given by the inspectors. The actions identified during inspection are tracked till closure. The people present in inspection are experts in the domain under consideration. Sometimes, inspectors may be external to the organisation. These people are experts in their fields but may not understand customer requirements exactly. There can be issues like non-disclosure agreement which may affect inspection by outsiders.

Advantages of Inspection

- Experts opinion is available as they are present during review. Subject-matter experts can clarify many issues.
- Inspection must be time effective as availability of experts is limited. There is always a time pressure for inspection.
- Defects are recorded but solutions are not given by subject-matter experts. This helps the organisation to initiate own action plan for fixing the defects.

Disadvantages of Inspection

- Experts may not be ready with review comments before inspection and time may not be utilised properly. It is a common scenario that people start reading documents at the time of meeting.

E-next THE NEXT LEVEL OF EDUCATION

https://E-next.in Scanned by CamScanner

- Experts opinion may vary from realities as it may be derived from their judgments and experiences. Experts may not understand exactly what the needs of the customer are.
- Facilitator's job becomes critical in case there is a difference of opinion between two inspectors, as the final aim of the organisation/project is to get inputs from inspectors.

Guidelines for Successful Inspection Inspection is a very formal method of verifying software work products with respect to expectation, standards, and guidelines. Sometimes, the customer may participate in inspection process. A defined methodology of inspection must be followed for performing inspection. The processes for inspection must be defined in organisational process database. Some generic guidelines for inspection are given below.

- Organisation/project must plan for inspection well in advance so that all the stakeholders are aware of their roles, responsibilities, time schedule available, etc. There must be a process definition of how to do inspection. It becomes more important when inspectors are subject-matter experts from outside the organisation including the customer. Effectiveness of the inspection process must be compared with expected level of expectations, and deviations must be taken for process improvement.
- Allocation of trained resources for facilitating inspection process, recording the opinions of experts, and presenting artifacts are essential to maintain the agenda and inspection flow. Inspectors, typically outsiders, are expected to be subject-matter experts in their field. They carry knowledge—and also, prejudice and ego with them. If two inspectors have a conflict of opinion, their opinions need to be handled carefully. Wastage of time can prove to be very costly.
- Expertise in terms of logistics arrangement is also needed, if inspectors are external to the organisation. There must be venue bookings and other arrangements for the inspection.
- Moderator/leader must circulate the work product to be inspected to the inspection team in advance, so that inspectors come prepared and maximum can be achieved in the given time. Inspector cost is very high, and output has to be measured against these costs.
- Product must be clear, compiled, checked for grammar, formatting, etc. Inspectors are expected to give inputs about approach of doing things and contents, and not on language or grammar.
- Inspection team may be of 3–5 people, as it is difficult to handle a larger group. Less number of inspectors would mean bias in the observations.
- Maximum time allocated for the inspection must not exceed 1–1.5 hours, as people may not be able to concentrate for a very long time. It works as per the law of diminishing returns.
- It is the moderator's responsibility to maintain the agenda of inspection. It is expected that inspection must concentrate on the work product and not a person/author of the work product.
- Manager or superior of the author may not be involved in the inspection as poor remarks about a work product, if any, can create a bad impression of the author on the superior. It may affect the morale of the author in a negative manner. People may not like to receive criticism of their work products in front of their superiors.
- The presenter presents the work product while inspectors give their comments. Many organisations do not allow the author to be present during inspection as they may become defensive while facing criticism about their work product.
- During inspection, one must concentrate on locating defects and not on fixing them. How to fix these defects is a responsibility of an organisation/project/author. One can note down what is wrong but need not ask any explanation to the inspector about how to fix it.
- Classify and record defects in different categories. This can help when root causes of the defect are analysed and taken for further action. It can show the weak parts of the processes so that efforts can be concentrated in the defined areas.

E-next THE NEXT LEVEL OF EDUCATION

https://E-next.in Scanned by CamScanner

- Review inspection process for continuous improvement. Stronger parts may be noted, and weaker parts may be taken for improvements to make a process successful.

Phases of Inspection A typical inspection has the following phases

- *Planning for Inspection* Planning for inspection involves selecting people for inspection, allocating roles to other people (such as facilitator, recorder, and presenter); defining the entry and exit criteria for inspection, and selecting which parts of artifacts are to be looked at. The artifact must be delivered to concerned people well in advance so that they are ready with their comments.

- *Kick-off Inspection* Kick-off inspection may start by distributing artifacts, explaining the objectives of inspection, process to be followed for inspection, and checking entry criteria for the artifact as well as inspection process. Entire process must be known to people taking part in actual inspection and to the stakeholders of inspection.

- *Individual Preparation* It is expected that participants must come prepared for the inspection. Work must be reviewed and comments (including potential defects and questions) by each of the participants must be ready before the inspection meeting. Participants must have read the work product and have clarity about the problems and issues.

- *Inspection Meeting* Proceeds of the meeting such as discussions done or logging of comments, and preparation of minutes of meeting must be completed in defined time frame. The participants of the meeting may simply note defects, make recommendations for handling the defects or make decisions about the defects. Recorder shall note these comments.

- *Decision on Comments* An organisation has to decide the fate of comments once the inspection is over. It is not necessary that all comments will be accepted. Comments may be rejected, deferred or may undergo another iteration of inspection. For accepted comments, the fixing of defects is done by the author.

- *Follow Up* Completing the actions as identified in minutes of meeting like checking that the defects have been addressed and whether exit criteria has been met or not is required. The findings can be used to gather statistics about work product, project or process.

Roles and Responsibilities A typical inspection may include the following roles.

- *Manager* Manager is the person responsible for getting the work product inspected. For a project, he may be the project manager. Manager decides on the execution of inspection, defines the schedule, allocates time, defines the objectives to be met, and determines if the inspection objectives have been met or not at the end of the inspection process.

- *Moderator* Moderator is the person who leads the inspection of the artifacts, including planning the inspection, running the meeting and follow-up after the meeting. If necessary, moderator may mediate between the various points of view of different inspectors, and is often the person upon whom the success of the inspection rests. Moderator is also called 'facilitator' as he facilitates the entire process.

- *Author* Author is the writer or person with chief responsibility of the artifact to be inspected. He is the person who has created the artifact, and will be taking action based on the outcome of the inspection.

- *Reviewers (Checkers/Inspectors)* Individuals with specific technical or business background who, after necessary preparation, identify and describe findings in the work product under inspection in the form of comments. Reviewers must be chosen to represent different perspectives and roles in the inspection process and must take part in meetings.

https://E-next.in
Scanned by CamScanner

- *Recorder* Recorder is the person who documents all the issues, problems and open points that are identified during the meeting. The presence of a recorder is a must because if reviewers or inspectors are discussing something, they have to decide collectively about the comment. Recorder is also expected to write minutes of meeting.

Success Factors for Inspection

- Each inspection must have a clear, predefined objective that has to be achieved. An inspection can be called 'successful' only if these objectives are met.
- The right people must be involved in the inspection process. For success in an inspection, people with adequate knowledge, expertise, etc. must be involved.
- Defects found should be welcomed and expressed objectively. Inspection must tell clearly what is wrong and why it is wrong.
- People issues and psychological aspects are not to be dealt with in an inspection process. One must stick to the artifact only (e.g. making it a positive experience for the author).
- Inspection techniques applied should be suitable to the type and level of software work products and inspectors.
- Checklist or role playing is to be used if appropriate to increase effectiveness of defect identification.
- Training in inspection techniques is to be given in case inspectors are internal to the organisation.
- Management support is essential for a good inspection process (e.g. by incorporating adequate time for inspection activities in project schedules).
- There must be an emphasis on learning lessons and process improvement.

6.4.6 AUDIT

Audit is defined as an independent assessment of the process or product under progress. Historically, audit process was applied to financial transactions to know and understand the financial health of an organisation. Audit may be categorised into quality assurance activity or quality control activity depending upon its nature, breadth and depth. 100% auditing may be considered as quality control activity while sampling may be considered as quality assurance activity. Some people define auditing as a separate branch than quality control or quality assurance, as it involves combination of quality assurance activities and quality control activities at a time.

Software development and testing process audit is an examination of product to ensure that the products as well as processes used to build them meet predefined criteria. Audits are conducted by auditors—people with expertise in auditing processes and also some knowledge about work products (CMMi organisations may refer to such people as Software Quality Analysts (SQA)). Auditors may not be experts in subject matter but may have a fair amount of experience in the field as well as in auditing techniques. The outcome of audit is an audit report comprising the following.

- **Major and Minor Non-Conformances** Non-conformances are deviations from the established criteria used for developing a work product. They indicate a gap between what is required and what actually exists. Minor non-conformances indicate that there is some deviation at some places, but in other places, the processes have been followed correctly. Major non-conformances indicate a big issue where something required is missing completely.

- **Observations** Observations are findings which may get converted into future non-conformances, if proper care is not taken. They may be termed as conformances just on the verge of breakage.

https://E-next.in Scanned by CamScanner

- **Achievement or Good Practices** These are good achievements by areas under audit which can be used by others. There is no possibility of existence of any project where there is not a single positive aspect which can be shared across the platform.

In software development and testing phases, various audits are conducted, as mentioned below,

- **Kick-Off Audit** Kick-off audit is conducted at the start of the project. This audit covers the areas that ensure whether all the processes required at the start of the project are covered or not. It may start from proposal, contract, risk analysis, scope definition, team size and skill requirements, authorities and responsibilities of various roles in the team. Kick-off audits cover checking documentation and compliances required at the start of the project.

- **Periodic Software Quality Assurance Audit** Software quality assurance audit is famous by the name 'SQA audit' or 'SQA review'. It is conducted for a product under development and processes used as defined by quality assurance process definition for these work products. Auditor is expected to check whether different work products meet the defined exit criteria or not, and the processes used for building these work products are correctly implemented or not.

- **Phase-End Audit** Phase-end audit checks whether the phase defined in the development life cycle achieves its expected outcome or not. It is also used as a gate to decide whether the next phase can be started or not. These are also termed 'gate reviews'.

- **Predelivery Audit** Predelivery audit checks whether all the requisite processes of delivery are followed or not, and whether the work product meets the expected delivery criteria or not. Many organisations follow a delivery checklist approach for conducting predelivery audits. Only those work products which are successful in predelivery audits can be given to a customer.

- **Product Audit** Product audits can be covered in SQA audit. It is done by executing sample test cases to find whether the product meets its defined exit criteria or not. Sometimes, this is also considered as 'smoke testing'.

6.5 TYPES OF REVIEW ON THE BASIS OF STAGE/PHASE

The reviews may be categorised on the basis of their happening at particular phase or timeline during development life cycle, as given below.

6.5.1 IN-PROCESS REVIEW

Reviews conducted while different phases of software development life cycle are going on are defined as 'in-process review'. They are intended to check whether inputs to the phase are correct or not, and whether all the applicable processes/procedures during a phase are followed correctly or not. Process reviews help in correcting things before the phase ends, so that the objectives of the phase can be achieved.

Milestone Review Milestone reviews are conducted on periodic basis depending on the completion of a particular phase or a time frame defined or a milestone achieved during a software development life cycle. Examples of such reviews may be a requirement review at the end of requirement phase, design

https://E-next.in
Scanned by CamScanner

review at the end of design phase, weekly status review at the end of a week, or review conducted as per percentage completion (say 5% project completion review). These reviews confirm that the output from the phase matches with predefined quality goals and input requirements of the next phase, and development life cycle can go to the next phase. Three types of milestone reviews are explained below

- *Phase-End Review* Phase-end review is conducted at the end of a development phase under review such as requirements phase, design phase, coding, and testing. It can be applied when there are distinct phases of development as defined in waterfall development model such as requirements gathering, design creation (high level and low level designs), coding, testing at various levels, deployment, etc. and also, input and output criteria for the phases are clearly defined. Typical waterfall cycle is suitable for doing a phase-end review where distinction between different phases is clearly defined. It may be used during iterative development where one phase completion may be reviewed. As required by iterative development methodology, reviewed phase may undergo change when there are some changes coming in. Changes may be required due to some other phase of development.

Phase-end reviews are also termed 'gate reviews' by some organisations as the gate at phase end opens only when the output criteria of the phase are achieved by the work product. Phase-end reviews are used extensively in agile development as one phase completion is marked by review before the next phase is started. Figure 6.2 explains phase-end review in a typical waterfall development approach.

E-next THE NEXT LEVEL OF EDUCATION

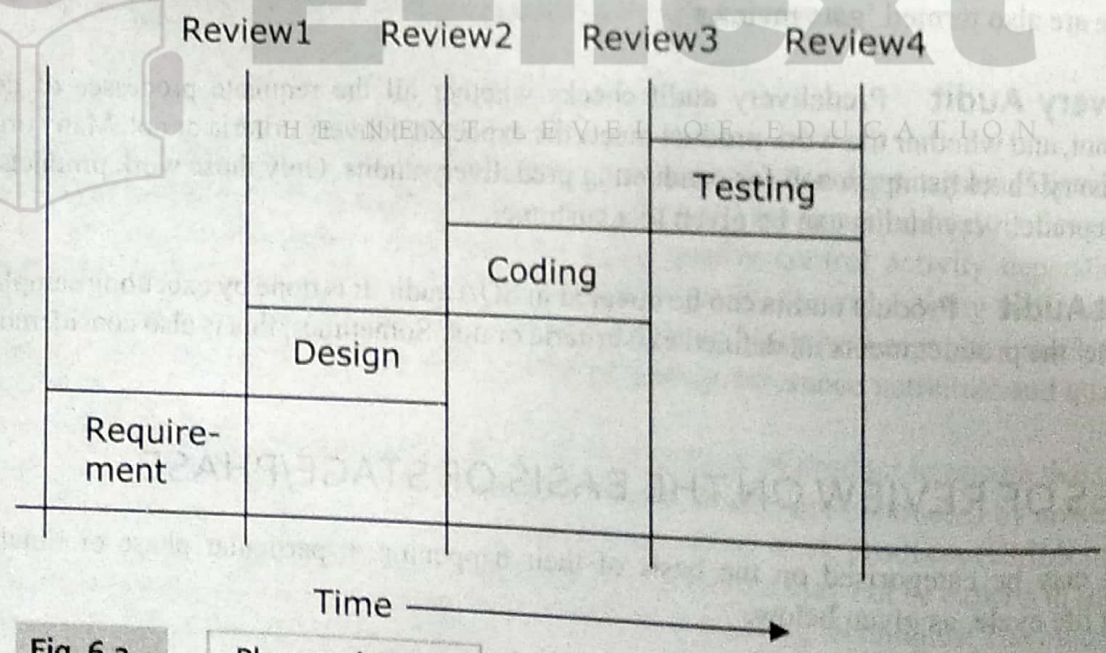

Fig. 6.2 Phase end review

- *Periodic Review* Waterfall is a classical model and applied very rarely in software development. Most of the times, we may not have a clear definition of phase end as there may be considerable overlap between different phases of development. Also, there is no clear definition of input and output criteria for each phase but they get defined as one progresses through software development life cycle. When one phase ends, in reality another phase may be half way through. In such cases, we may have review on some periodic basis

https://E-next.in
Scanned by CamScanner

such as weekly, monthly, quarterly, etc. Project plan must define various periods when review of project-related activities will be conducted. Stakeholders may be required to review the progress and take actions on any impediments. Many models such as CMMi mandate that various activities must be reviewed with stakeholders and senior management. Figure 6.3 shows a monthly review for a project where there are considerable overlaps between different phases.

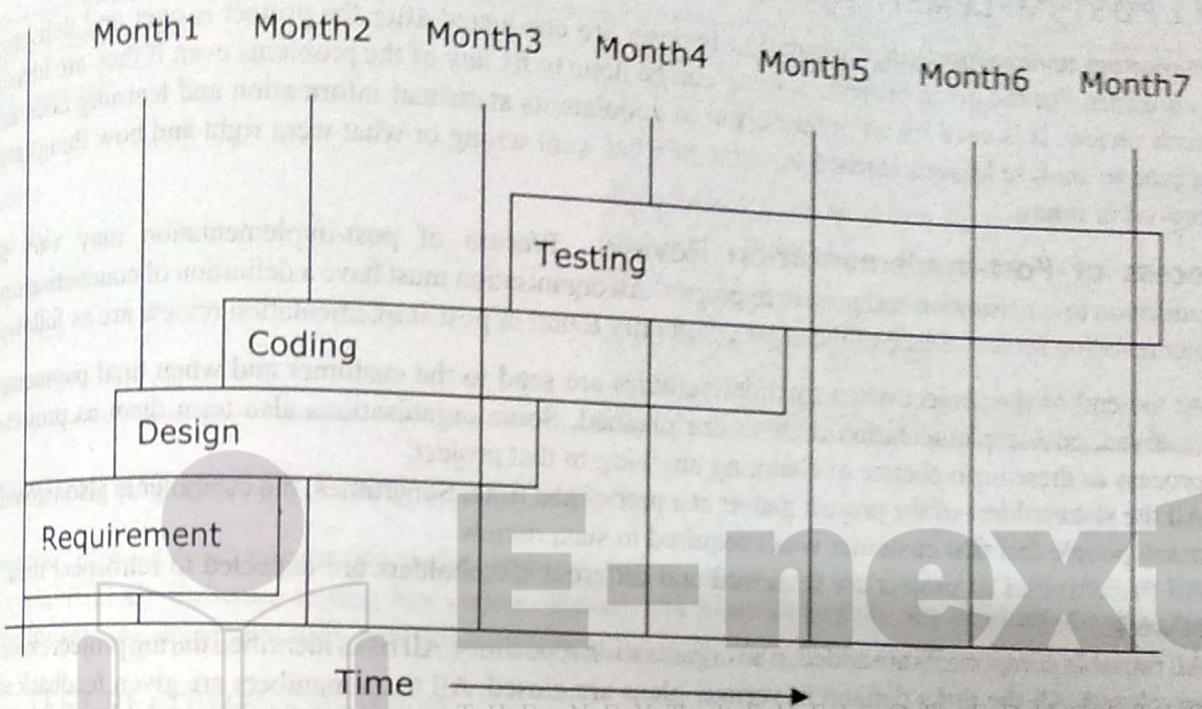

Fig. 6.3 Periodic review

- *Percent Completion* Percent completion review is a combination of periodic review and phase-end review where the project activities or product development activities are assessed on the basis of percent completion e.g. 5% completion or similar way. The review is conducted as the project progresses and the part of the project defined in percentage gets completed. At high activity level, month may be a very long duration for review as lot of things may change in a month. In such cases, percentage review is a better option. There may be some phases where development/testing activities may be very low and reviewing every month is of not much value as there is not much change. Here also, percentage is a better way of doing review.

Process of In-Process Review An organisation/project must have a definition of in-process review. It must have a list of stakeholders attending various reviews, and the work products under reviews. Sometimes, the customer is also invited for such reviews. Commonly found steps of in-process reviews are given below.

- Work product, and metrics undergoing review are created and submitted to the stakeholders well in advance.
- Inputs are received from stakeholders to initiate various actions.
- Risks identified by the project are reviewed and actions may be initiated as required. Risks which no longer exist may be closed and risks which cannot affect the project are retired.

E-next THE NEXT LEVEL OF EDUCATION

https://E-next.in Scanned by CamScanner

- Any issue related to any stakeholder is noted and follow-up actions are initiated. Issues may be like software availability, hardware availability, changing requirements, trainings required, etc.
- Review reports are generated at the end of review which acts as a foundation for the next review. In each review, actions and issues of earlier reviews are visited and any open items are taken for further actions.

6.5.2 POST-IMPLEMENTATION REVIEW

Post-mortem reviews/post-implementation reviews are conducted after the project is over and delivered to the customer. For the given project, nothing can be done to fix any of the problems even if they are identified in such review. It is used by an organisation to populate its statistical information and learning documents. It is used to analyse lessons learned in terms of what went wrong or what went right and how things can be improved in future.

Process of Post-Implementation Review Process of post-implementation may vary from organisation to organisation and project to project. An organisation must have a definition of conducting a post-implementation review. Some of the steps commonly found in post-implementation review are as follows.

- At the end of the project when final deliverables are send to the customer and when final payments are received, post-implementation reviews are planned. Some organisations also term them as project-end process as there is no chance of changing anything in that project.
- All the stakeholders of the project gather at a predefined time. Sometimes, the customer is also invited but many people feel that customer is not required in such review.
- All the activities of project are reviewed and different stakeholders are expected to retrospect their part of work.
- All reusable components are added in an organisation repository. All risks identified during project execution are closed. All the tasks defined in various plans are closed. All team members are given feedback about their performance in a project. Skill sets of all team members are updated in an organisation database.
- Each and every activity is listed to identify if something went exceptionally good or exceptionally bad when the project was executed. All project-related metrics are reviewed.

6.6 EXAMPLES OF ENTITIES INVOLVED IN VERIFICATION

The following Table 6.1 illustrates the artifacts prepared during development and testing life cycle and the entities reviewing them. These are indicative entities and may change as per specific circumstances, organisation and customer requirements.

Table 6.1 Entities performing verification

Verification	Entities performing verification
Requirement review	Business analysts, System analysts, Project team including architects, developer and customer/User
Design review	Project team, Customer/user
Code review	Development team, Customer/user
Project plan review	Project team, Customer/user, Suppliers
Test artifacts review	Test team, Development team, Customer/user

https://E-next.in
Scanned by CamScanner

All the verification techniques discussed above may not be used for all phases and work products. Verification techniques generally used in different phases of development lifecycle are shown in Table 6.2.

Table 6.2 SDLC phase and verification technique used

Phase	Verification technique used
Planning documents	Inspection, Walkthrough, Peer review
Requirements	Inspection, Peer review
Design	Walkthrough, Peer review
Coding	Peer review, Superior review
Test plan	Inspection, Walkthrough, Peer review
Test scenario	Walkthrough, Peer review
Test cases	Peer review, Superior review
Test results	Walkthrough, peer review

6.7 REVIEWS IN TESTING LIFECYCLE

Similar to software development life cycle, software testing has its own life cycle termed 'software testing life cycle' (STLC). Software testing has various phases which are defined in test planning domain. Let us define the reviews associated with execution of testing.

6.7.1 TEST READINESS REVIEW

Test readiness reviews are generally conducted by test managers, test leads or senior testers with project manager, module leaders or project leaders to ensure that product under development is ready for testing as per the phase of testing applicable. Test readiness review is done at each stage of development such as unit testing, integration testing, interface testing, system testing, and acceptance testing. Test readiness review for system testing may include the following stages,

- Review comments for different work products indicate the possible weak areas in software development, and possible problematic areas where defects can be found. All comments may not be defects but decision about comments (accept/reject/defer/change) must be finalised and actions must be taken according to decisions. Test manager must review all review comments of the artifacts such as peer review comments, walkthrough or inspection comments to check whether all comments are closed or not. If review comments are open, it is likely that defects would be found in the respective areas.
- Review of unit test report to check whether all unit testing defects are closed or not. If unit testing defects are open, the same defects can be found in system testing also. Sometimes, it may not be possible to close all unit testing defects but then, release note given to system testing must record such defects as known issues and their possible impact on the working of a product under testing.
- Most of the softwares need installation before they can be used. One needs to install software to check whether installation testing has been successful or not. Installation testing may include installation through CD or other media (like pen drives), installation from network, or remote installation as defined by requirement statement. If software cannot be installed, then it cannot be used by the users. Sometimes installation defects are called 'showstopper' as they can stop usage of software.

https://E-next.in
Scanned by CamScanner

Prerequisites Testing Software installation has some prerequisites such as operating system, database, and reporting services as the case may be. If requirement statement specifies that installation must check for the availability of such prerequisites, then it must be tested accordingly. Sometimes, if prerequisites are not available, installation may prompt, or prerequisites may be installed directly during installation. Also installation can be manual, autorun or semi-auto depending upon the requirements.

Updations Testing Sometimes, it is expected that while installing software, there must be a check whether the same or similar software already exists there or not. One may experience that while installing an operating system, it checks whether some operating system already exists and what is the version of the existing system. Generally, advance versions may not be overwritten by the older versions. Sometimes, it also prompts for a repair or new installation, if the versions available are older than the one being installed. The options must be guided by requirements.

Un-Installation Testing Un-installation testing is done when the developing organisation wishes to check that un-installation is clean. Any file existing before installation must not get replaced while installing the application. Similarly, when an application is un-installed, all the files installed must be removed from the disk. Applications running before installation must be running after installation and also once un-installation is complete.

- Smoke testing is done to check whether the application is alive or not. It refers to testing the application to understand if a user can work with it or not. Any living application must occupy some disc space, and must be shown in the programs available on a computer. Launching of an application and executing very high-level test cases must be possible. The purpose of smoke testing is to check whether the application can be worked with or not. It does not establish any functionality or correct behavior of the application. Some people refer to smoke testing as smell testing.
- Sanity testing, also called as Build Verification Testing (BVT) is done to check whether the major functionalities of an application are available to the user or not. In sanity testing, one works with software at macro level to understand its ability to produce consistent results and availability of major functionalities or features. Many people consider smoke and sanity testing at the same testing level. Build verification testing ensures that the results produced by executing test cases are repeatable and reproducible. If an application fails in sanity testing, there is no point in going ahead with system testing as the defects found may not be reproducible and correct/wrong behavior cannot be proved.

Test readiness review also includes review of all background arrangements including help files, release notes, and supporting documentation as defined in test plan document. Test readiness review may also include whether the test team has all information and hardware/software/licenses required for creation of test bed, availability of tools including regression testing tools, configuration management tools, defect tracking tools, training, and other resources for performing testing.

6.7.2 TEST COMPLETION REVIEW

Test completion review is conducted when a testing cycle is completed and test results are created. Outcome of such reviews includes analysis of testing process, number and type of defects found, and number of iterations completed. Test completion review must check the coverage of testing in terms of requirement coverage, functionality coverage, and feature coverage as the case may be (as intended in test plan), and the number of defects found as against the targeted number of defects. Final outcome of test completion is a recommendation

https://E-next.in
Scanned by CamScanner

about the status of an application. It must tell the development team whether the product can go to the next phase or needs any repairing, retesting and regression testing before it can be declared as successful.

6.8 COVERAGE IN VERIFICATION (TEST DESIGNING)

Different instances of verification offer different ways of measuring coverage achieved in verification. Definition of coverage must be in sync with the document or artifact under verification. Let us consider few coverage considerations in coding as an activity under verification (the same can be applied to other areas of development and testing).

- Statement coverage
- Path coverage
- Decision coverage
- Decision-to-decision path coverage

6.8.1 STATEMENT COVERAGE

Programs include numerous 'statements'. Definition of 'statement' must be done by an organisation before calculating any coverage. Some people consider ';' as a sign of statement while some people consider a code line with or without ';' as a statement. Definition must be used consistently atleast for the given project. Sometimes, non-executable lines are also considered as statement as they are very important from maintainability point of view, for example, comments.

When a verification activity is conducted, how many lines are verified against the total number of lines available in a given unit will represent the statement coverage. 100% may be targeted for a unit but for complete application, it may not be possible to verify each and every line in the application. One may concentrate on the important part of code which will be mostly used. In such cases, coverage may be less than 100%.

6.8.2 PATH COVERAGE

Path represents the sequence of control flow from entry to exit in a given code. Application may have several paths in which the flow may progress while executing the application. If there are no decision loops in the program, it will have a single path in which it can progress. As soon as decisions are introduced in the application, number of paths increase accordingly. When decisions are mutually exclusive, number of paths increase as per number of decisions. But if decisions are dependent on each other, increase in number of paths is exponential.

In verification activity, if each path is considered for verification, it may give 100% path coverage. Each path has different probability of happening and some paths with lesser probability of getting executed than others may not be verified, if time available for verification is a constraint. In such cases, path coverage may be less than 100%.

When the decision gives a range like 'for' or 'do while' loops, each outcome may represent a different path. One may apply techniques like 'equivalence partitioning', and 'boundary value analysis' to decide which path needs to be verified so that it would be equivalent to many other paths. In such cases, path coverage may be less than 100%.

Path Predicate Path prediction may be done depending upon the decision points available in the domain. Each decision gives several paths depending upon the type of decision included. Decision, depending upon the previous decisions, adds to number of paths exponentially while mutually independent decisions add the number of paths arithmetically.

https://E-next.in

Scanned by CamScanner

Generally, decisions like 'if and else' give two paths while decisions like '< or >' may give many more paths. Also, when outcome of one decision is an input to another decision, it multiplies the paths.

Achievable (Feasible) Path Achievable paths are the feasible paths when a user is actually using the application. If there is any path where the control would never go, it is termed 'infeasible path' or 'redundant path'. One must remove such path from the application being developed. Redundant path indicates a possible defect in verification.

Path Sensitising The basic strategy in path sensitising is to pick a defect or bug to be tested and sensitise paths; some of which go through that defect. A path is sensitised if the associated test checks for faults along the path; therefore the main objective is to find a set of tests which will determine if the defect has occurred or not. Path sensitising method uses less time and is more efficient in locating defects.

Path Profiling (Instrumentation) Path profiling is intended to find heavily-used paths from other paths present in the application where the control goes rarely. Heavily-used paths are more important for the application, and hence must be correct and deliver good performance to generate adequate confidence in users. Generally, verification must target to cover such paths for better product.

6.8.3 DECISION COVERAGE

When an application is getting executed at several places, it may have to make decisions depending upon the situation it faces. Each decision may have several paths. One may not be able to verify all paths and hence may concentrate on the decision part only. If one outcome is verified with an assumption that all other outcomes will be correct if the one selected is correct, it gives decision coverage less than 100%.

When the decision gives a range like 'for' or 'do while' loops, one may apply techniques like 'equivalence partitioning', and 'boundary value analysis' to decide which path needs to be verified. One may need to decide more probable paths for verification. In such cases, path coverage may be less than 100%.

6.8.4 DECISION–TO–DECISION PATH COVERAGE (DD PATH COVERAGE)

There may be a path between two decisions. If decisions are mutually exclusive, then there may be a single path between two decisions. If latter decisions are depending upon earlier decisions, then there may be several paths possible between different decisions.

Verification of path between two decisions will be used for statement coverage in addition to decision coverage. Decision coverage may be restricted to the decision part, while flow happening after the decision till next decision is captured by decision-to-decision path coverage.

6.8.5 NICE DOMAIN AND UGLY DOMAIN

During verification, it is considered that the domain under testing is a nice domain. The characteristics of nice domain are that the domain is linear, complete, systematic, orthogonal, consistently closed, simply connected and convex. This assumption makes an application or code review easier, as one set of transaction can be considered to represent all the sets available in that domain.

On the other hand, there can be some ugly domains which do not follow the assumptions given above. Such domains are nonlinear, less or not systematic, inconsistent and so on. If we consider an example of division of two numbers as 'a/b' then 'a' can assume any number and is a nice domain. But 'b' can have any value other than 'zero'.

https://E-next.in

Scanned by CamScanner

KV Charts A KV map (Karnaugh map or Veitch map) may contain number of boolean variables. Generally, more than 6 variables make the chart very complicated. Being a boolean in nature, each variable can contribute maximum two possibilities indicated as '0' and '1'. When we take multiple variables, one can plot all system states by considering different states of an individual variable. Once the variables are defined, the output possibilities are transcribed according to the grid location provided by the variables. Thus, for every possibility of a boolean input, the output possibility is defined.

When the KV map is completed, to derive a minimised function, the '1's or desired outputs are grouped into the largest possible rectangular group in which the number of grids boxes (output possibilities) in the groups must be equal to a power of 2. The boxes can be used more than once only if it generates the least number of groups. All '1's or desired output possibilities must be contained within a grouping.

The groups generated are then converted to boolean expression by locating and transcribing the variable possibility attributed to that box, and by the axiom laws of boolean algebra in which if the (initial) variable possibility and its inverse are contained within the same group, the variable term is removed. Each group provides a 'product' to create a 'sum-of-products' in the boolean expression. To determine the inverse of the KV map, the '0's are grouped instead of the '1's. The two expressions are non-complementary.

6.9 CONCERNS OF VERIFICATION

There are few concerns associated with verification activities. In general, verification is expected to find about 60% of the total number of defects. The defects found in verification must have some probability of impacting final users if they go undetected. Verification defects must not be only of academic nature but must also affect software negatively, if they are not removed. Effects may be in terms of not achieving some quality factors important from software usage perspective.

6.9.1 USE OF RIGHT VERIFICATION TECHNIQUE

Every verification technique has some advantages and disadvantages. One may have to use verification techniques judiciously. For test cases and code files, peer-to-peer review may be more advantageous from cost perspective while for requirement statement, inspection may be selected as a technique. Project manager/ organisation may decide the technique on the basis of various factors of evaluation such as cost-benefit analysis, importance from user's perspective and so on.

6.9.2 INTEGRATION OF VERIFICATION ACTIVITIES IN SDLC

As seen in V & V model, every phase of software development life cycle has associated phases of verification as well as validation. Selection of verification technique must be such that it must find the defect as early as possible. This can prevent stage contamination. The 'check' part must be very close to the 'do' part of the process so that defects in the 'do' process are found and removed. This improves development process capability.

6.9.3 RESOURCES AND SKILLS AVAILABLE FOR VERIFICATION

Very important inputs for verification are time and people with required skills. If reviewer/inspector is capable with sufficient knowledge and ability, verification can be very effective. Otherwise, it becomes ineffective as people may not find much value addition by conducting such a review. Defects found in verification must have some possibility of affecting final users, if they go undetected.

https://E-next.in
Scanned by CamScanner

6.10 VALIDATION

Validation is a disciplined approach to evaluate whether the final built software product fulfills its specific intended use. It is meant to validate the requirements as defined in requirement specification, ensure that the application as developed matches with the requirements defined, and is fit for the intended use. Real-life scenarios are achieved in test lab to perform validation of the application under testing. Validation steps may involve test bed preparation, test scenario definitions, test case definitions, test data definitions, test case execution, defect identification, retesting, and regression testing.

Validation is also called 'dynamic testing' as the application is executed during validation, with the intention to find defects. Validation must be done by independent users, functional experts, and black box testers to ensure independence of testing from development activities. It helps in analysing whether the software product meets the requirements as specified in requirement statement, and can give a user required level of confidence that the application will not fail in normal as well as abnormal circumstances.

6.10.1 ADVANTAGES OF VALIDATION

- Black box testing approach is used for system, integration and acceptance testing. It represents actual user interaction with the system without any consideration of internal structures or how the system is built. It is generally independent of the platform of development, database or any other technical aspect related to software development.
- Validation is the only way to show that software developed by following all processes of development is actually functioning as desired and is usable. It can answer the question—'Can we use the software?'

6.10.2 DISADVANTAGES OF VALIDATION

- No amount of testing can prove that software does not have defects. If defects are not found by conducting the defined test cases, we can only conclude that there is no defect in the set of transactions actually executed. There can be many more defects not captured by these test cases.
- For unit testing and module testing, stubs and drivers are needed to be designed and used during testing. Stubs and drivers need additional efforts of development and testing before they can be used.
- Sometimes, it may result into redundant testing as the tester is not aware of internal structure, and same part of the code gets executed again and again. This is a common scenario when we have some library functions which are called several times during execution, and they are tested each time since the tester does not know that the same function is being called again and again.

6.10.3 PREREQUISITES FOR VALIDATION

Depending upon the method followed for validation and work product under validation, there may be different inputs required for validation. Any process would need some inputs coming to the work bench and some outputs going out from the system. Inputs for the validation process may include work product, test plan, and test cases, whereas output from the validation work bench may include test report and defects.

Prerequisites for validation may include the following.

- Training required for conducting validation. Training may include domain knowledge and knowledge about testing and various test tools
- Standards, guidelines, and tools to be used during validation
- Validation 'do and check' process definition

https://E-next.in

Scanned by CamScanner

6.11 VALIDATION WORK BENCH

A validation work bench is a place where validation activities are conducted on the work products, and may be a physical or virtual entity. For every work bench, the following basic things are required. Figure 6.4 shows a validation workbench.

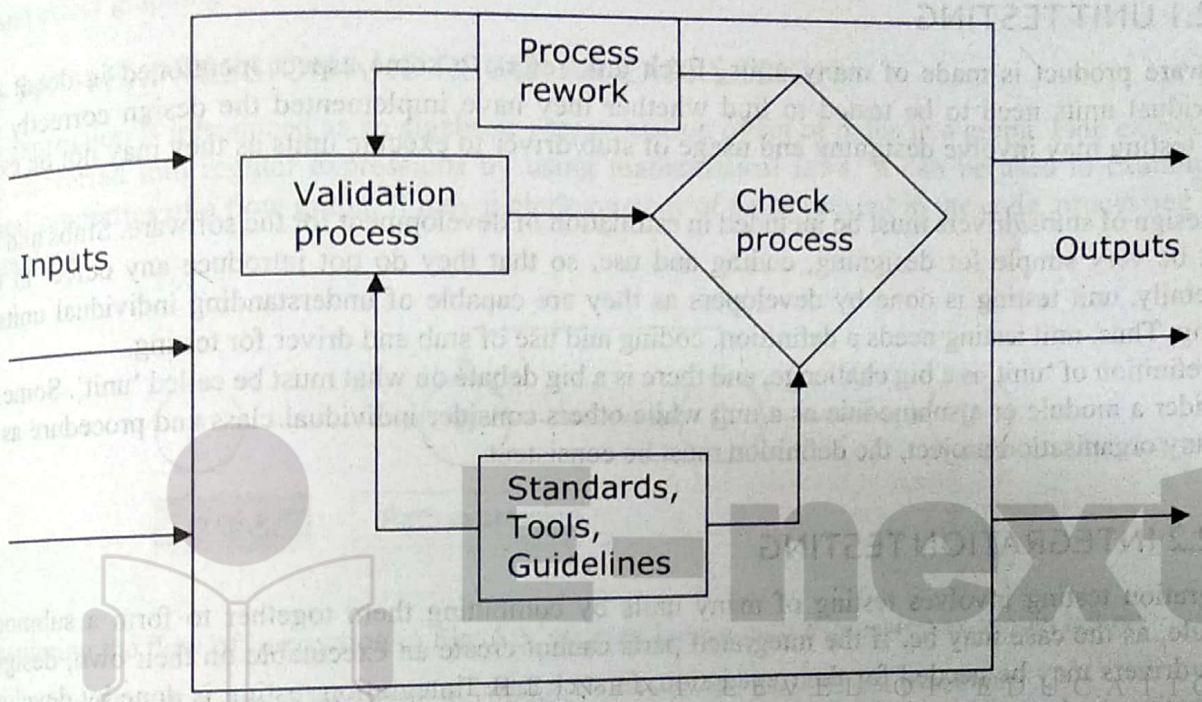

Fig. 6.4 Workbench for validation

Inputs There must be some entry criteria definition when inputs are entering the work bench. This definition should match with the output criteria of the earlier work bench. For example, if we are validating an application or part of it, then we must have a written and compilable code as an input to the work bench along with test plan and test cases/test data as per definition of input.

Outputs Similarly, there must be some exit criteria from work bench which should match with input criteria for the next work bench. Outputs may include validated work products, defects, and test logs.

Validation Process Validation process must describe step-by-step activities to be conducted in a work bench. It must also describe the activities done while validating the work product under testing.

Check Process Check process must describe how the validation process has been checked. It is not a validation of the work product but a process. Test plan must define the objectives to be achieved during validation, and check processes must verify that the objectives have been really achieved.

Standards, Tools, and Guidelines These may be termed 'the tools' available for validation. There may be testing guidelines or testing standards available. Sometimes checklists are used for doing validation.

https://E-next.in
Scanned by CamScanner

6.12 LEVELS OF VALIDATION

Similar to verification at different stages, validation can be applied at different stages of software development. Some of these stages are explained below

6.12.1 UNIT TESTING

Software product is made of many units. Each unit refers to some aspect mentioned in detail design. Individual units need to be tested to find whether they have implemented the design correctly or not. Unit testing may involve designing and usage of stub/driver to execute units as they may not be executed alone.

Design of stubs/drivers must be included in estimation of development for the software. Stubs and drivers must be very simple for designing, coding and use, so that they do not introduce any defect in testing. Generally, unit testing is done by developers as they are capable of understanding individual units under testing. Thus, unit testing needs a definition, coding and use of stub and driver for testing.

Definition of 'unit' is a big challenge, and there is a big debate on what must be called 'unit'. Some people consider a module or a submodule as a unit while others consider individual class and procedure as a unit. For any organisation/project, the definition must be consistent.

6.12.2 INTEGRATION TESTING

Integration testing involves testing of many units by combining them together to form a submodule or module, as the case may be. If the integrated parts cannot create an executable on their own, designing of stubs/drivers may be needed for their execution. In such case, integration testing is done by developers. If integration can be executed independently, it may be done by testers. Integration testing ensures that the units working correctly when tested individually are also working correctly when brought together. Integration testing helps in identifying if there are any issues of parameter passing as different units are interacting with each other. Test cases are defined by referring to low-level design. Integration testing mainly refers to detail design or low-level design.

6.12.3 INTERFACE TESTING

Interface testing involves testing of software with the environmental factors (such as database and operating system) where the application is supposed to work. If an application is supposed to work with some other applications, third-party components (such as communication software) combining all these applications and then conducting testing is termed 'interface testing'. Parameter passing is the most important criterion for interface testing. Test cases are defined by referring to high-level design or architectural design.

6.12.4 SYSTEM TESTING

System testing involves end-to-end testing of a system to find the behavior of a system with respect to expectations. It needs to be done by testers as if they are the users of the system. The testing process involves defining test scenario, test cases and test data for testing. System testing is actually a set of processes which may include functionality, user interface, performance, and security testing. Test cases are defined by referring to requirements and designs.

https://E-next.in

Scanned by CamScanner

6.12.5 CAUSE AND EFFECT GRAPHING

While using the system, there may be several inputs given to the system, and the system is expected to respond to those inputs. Equivalence partitioning is used extensively along with boundary value analysis for plotting the causes and their effects on the system. 'When somebody clicks a login button with valid login and password, the system allows that user to enter the system as a valid user'—this can be an example of cause and effect graphing.

6.12.6 PATH EXPRESSION AND REGULAR EXPRESSION

A path expression is introduced as an algebraic representation of set of paths in a graph. Path expressions can be converted into regular expressions by using mathematical laws. It can be used to examine the structural properties of a flow graph. It may include number of paths present in the code, processing time, and data flow.

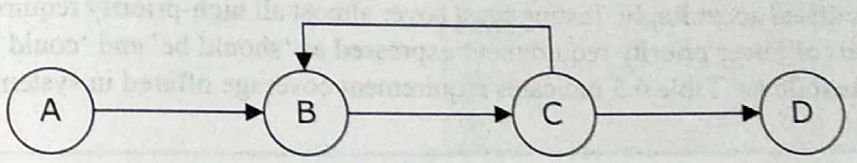

Fig. 6.5 Path expression

Considering the flow of instruction in Fig. 6.5, we may have a path expression as shown below.

AB + BC + CD + CB depending upon the number of times BC and CB get executed.

Path Sum Path sum denotes the number of paths in parallel as defined above.

Path Product When a larger path is made of two successive paths, the total number of paths would be a multiple of number of path in each instance. Thus, such paths are in a series to each other.

Path Loops Loops are infinite (or very large) number of parallel paths. It may be expressed in terms of definition of exponential values.

6.13 COVERAGE IN VALIDATION (PRIORITISATION/ SLICE BASED TESTING)

Different instances of validation may offer different coverages depending upon the test plan and definition of coverage. It has a relationship with requirement traceability matrix of an organisation and the customer. Validation mainly talks about black box testing methodology where we may not know how things are structured in an application. It mainly covers functionalities and requirements as defined in requirement statement.

When there is a time constraint and testing is expected to give results faster, 100% testing is not feasible. System may be released by testing the critical parts with respect to user requirements. This is also termed 'prioritisation technique' where primary usage parts of the system are tested fully so that the system will not

https://E-next.in
Scanned by CamScanner

have defects under normal circumstances. The slices are obtained on the basis of prioritisation as defined below.

Following are some of the famous coverages offered during validation.

- Requirement coverage
- Functionality coverage
- Feature coverage

Requirement Coverage Requirements are defined in 'requirement specifications' documents. Traceability matrix starts with requirements as defined in requirement statement and goes forward upto test results. Theoretically, all the requirements as defined in requirement statement must have corresponding test cases and test results. Passing or failure of such test cases can define whether the requirements have been achieved or not.

All requirements are not mandatory. They can be put in different classes such as 'must', 'should be', and 'could be', and prioritised accordingly. Testing must cover almost all high-priority requirements expressed as 'must' and some part of lower priority requirement expressed as 'should be' and 'could be'. Thus, coverage may be expressed as follows. Table 6.3 indicates requirement coverage offered in system testing.

Table 6.3 Requirement coverage definition

Priority of requirement	Coverage offered
P1 (Must)	100%
P2 (Should be)	50%
P3 (Could be)	25%

Functionality Coverage Sometimes, requirements are expressed in terms of functionality required for the application to work successfully. Functionalities are also defined at different priorities as per their importance to the users. All functionalities are not mandatory from the user's perspective. Test cases representing high-priority functionality must be tested to a larger extent than the functionalities with lower priorities. Thus, coverage may be expressed as follows. Table 6.4 indicates functionality coverage offered in system testing.

Table 6.4 Functionality coverage definition

Priority of functionality	Coverage offered
P1 (Must)	100%
P2 (Should be)	50%
P3 (Could be)	25%

Functionality coverage cannot help in identifying the coverage established by non-functional requirements such as performance and user interface.

https://E-next.in
Scanned by CamScanner

Feature Coverage An application may need some features as defined in requirements. Features are a group of functionalities doing same/similar things. The same feature may have different ways of doing things, and each of them is a functionality for the application. Accomplishing atleast one of the functionalities may help in achieving the user requirements as a feature. If we consider saving a file in Microsoft Word, then 'saving' represents the feature while 'different ways to save the file' may represent different functionalities. All may not be equally important and one can perform a probability analysis on the basis of user profile.

Feature coverage talks about covering a feature required by the user. It may mean covering atleast one of the functionalities which represents a feature provided, even if there are multiple ways of doing things. Thus coverage may be expressed as follows. Table 6.5 shows a feature coverage offered in system testing.

Table 6.5 Feature coverage definition

Priority of feature	Coverage offered
P_1 (Must)	100%
P_2 (Should be)	50%
P_3 (Could be)	25%

Specification-Based Testing Specification-based testing refers to the process of testing a program based on its specification, as defined in requirement specification, design specifications or user manual. Generally, it states how the applications behavior should be. In particular, we can develop test cases based on the specification, without seeing an implementation of the program. Furthermore, we can develop test cases before the programming on the basis of behavioral expectations.

Without writing a program to implement requirements based on the specification given and generating a set of test cases and test data that one thinks would be sufficient to test a program may be considered as specification testing. If one thinks that the specifications are incomplete in any way, then one has to state what assumptions one is making about how it should be completed or clarified.

Remember that a test case consists of the input data and the expected result for that input data. This testing generally concentrates on external functions, and does not test technical, system, and legal requirements completely.

6.14 ACCEPTANCE TESTING

Acceptance testing is generally done by the users and/or customers to understand whether the software satisfies their requirements or not, and to ascertain whether it is fit for use. There are some predefined methodologies, conditions and test cases in the contract which will be used for acceptance testing. Users/customers execute these test cases to show if the acceptance criterion for the application has been met or not. There are two (sometimes three) levels of acceptance testing.

- **Alpha Testing** Alpha testing represents the testing done by the customer in development environment in front of the development team. The testing is done at development site with dummy data, either created by developers or shared by customer. In reality, alpha testing may be done by testers in front of the customer to show that software is working. It can be used as a tool for training key users on the application. Any major

https://E-next.in
Scanned by CamScanner

problems in terms of requirement understanding, functioning of software, etc. can be cleared during alpha testing before software is formally delivered. Data used during testing may be shared by customer as a live business data or dummy data created by customer or tester. Test data created by testers must be reviewed by business experts/customer/users. There is no direct impact of such testing on customer business as testing is done off-site. This is also termed as 'dry run' by some people.

- **Beta Testing** Beta testing represents a business pilot where testing is actually conducted by customer in production/semi-production environment. It is done at selected customer locations by representatives from the customer side. Testers/developers may be appointed to help the customer in testing the application, and recording the problems, if any. Beta testing is extensively used as parallel testing to give confidence to the users that the software really works as per established existing system at customer's place.

- **Gamma Testing** Gamma testing is used for limited liability testing at selected places. Gamma testing is generally done by the product development organisation that wishes to have limited market demonstration and piloting of an application in the market. The application is given to few people for using it in production, environment, and feedback is obtained from them about the features incorporated as well as something missing or wrongly interpreted. This is used by product organisations to understand customer reaction to a new or enhanced product.

6.15 MANAGEMENT OF VERIFICATION AND VALIDATION (V & V)

Verification and validation techniques are complementary to each other, and not the replacement of each other. A project must estimate time and efforts required for preparing and executing verification and validation activities as per project/quality plan and test plan definitions. The steps involved in verification/validation are as follows.

THE NEXT LEVEL OF EDUCATION

6.15.1 DEFINING THE PROCESSES FOR VERIFICATION AND VALIDATION

An organisation must define the processes applied for verification and validation activities during the development life cycle phase of the project. The processes involved may be as follows.

- *Software Quality Assurance Process* These processes concentrate on developing the techniques/procedures for development and testing of the project. They may be more generic, indicating overall approach of handling verification and validation activities, or may be very specific for the project/customer. The processes may be inherited from the organisational process database, or may be tailored as required for a particular situation. Generally, quality assurance process forms a part of prevention cost, where care is taken so that minimum (zero) defects are introduced in the product.

- *Software Quality Control Process* These processes concentrate on developing the approach for verification and validation activities during software development and testing. They can be more specific for the project under development. The process definition must have the details of types of activities and stakeholders involved at each stage. Generally, quality control process forms a part of appraisal and failure cost. Quality control process must ensure that defects are found as near to its origin as possible, so that stage contamination can be reduced.

- *Software Development Process* These processes concentrate on the approach of developing software where minimum (zero) defects will be introduced. They may cover software quality control processes along with other SDLC processes. They define usage of development process, and templates, formats, guidelines

https://E-next.in

Scanned by CamScanner

and standards to be used during development and testing. These processes may be derived from organisational process database or may be acquired from external sources such as national or international standards. Sometimes, customer defined software processes are adopted by projects.

- *Software Life Cycle Definition* Software life cycle definition is essential to establish development and testing processes for the software. It depends on the life cycle definition of software. One may select a waterfall model or iterative development model as per specific requirements of customer.

6.15.2 PREPARE PLANS FOR EXECUTION OF PROCESS

When a project proposal is made, it must contain a definition of what is meant by a successful delivery, and how quality of deliverables will be achieved, ensured and tested during life cycle of the project. The contract must also cover the acceptance criteria and deliverable definitions. The plans involved are as follows.

- Software development plan and schedule which include responsibilities and time schedule for verification/validation activities must be defined at the start of the project. It must define entry and exit criteria for each activity. These plans must be refined as and when required.
- Software quality plan and software test plan must define the use of different methodologies for ensuring quality of deliverables, verification and validation activities associated with development. Software quality plan and software test plan must be refined if required.
- Software acceptance testing plan must be defined where acceptance criteria is finalised. Software acceptance test plan must cover the entry and exit criteria at each stage of development. Exit criteria at the earlier stage must match with entry criteria of the next stage.

6.15.3 INITIATE IMPLEMENTATION PLAN

The plans made at the time of proposal/contract must be implemented during development life cycle of a project. There must be formal records of requirements review, design review, code reviews, unit testing, integration testing, interface testing, system testing, and acceptance testing. The results must be recorded and corrective/preventive actions must be initiated from the results of verification and validation. Whenever required, verification and validation plans must be changed to accommodate the changes in development activities, scope, customer expectations and so on.

6.15.4 MONITOR EXECUTION PLAN

The verification and validation activities must be monitored during development life cycle execution. Project manager and test manager must oversee that the activities defined in various plans are being executed and the results are being logged in test log. The time required for various activities, number of test cases executed, and number of defects found must be monitored to find out their sufficiency and effectiveness for the purpose. Corrective/preventive actions must be planned when discrepancies are observed with respective planned arrangements, or defects are logged, or nonconformances are observed in the processes/work products.

6.15.5 ANALYSE PROBLEMS DISCOVERED DURING EXECUTION

Execution of verification and validation processes may bring out many problems in the product as well as processes associated with development and testing. Root cause analysis of problems and planning for improvement actions are essential parts of continuous improvement. An organisation is expected to perform a root cause analysis, initiate corrective actions to remove the chances of recurrence of the same problem, and take preventive actions along with correction.

https://E-next.in
Scanned by CamScanner

6.15.6 REPORT PROGRESS OF THE PROCESSES

During the life cycle execution of a project, there are milestone or period dependent reporting mechanisms established by plans and schedules. The outcome of verification and validation activities as planned must be formally reported to management, customer, and development team to make them aware of the project progress and problems faced by the product/processes during project life cycle.

6.15.7 ENSURE PRODUCT SATISFIES REQUIREMENTS

The ultimate aim of verification and validation activities undertaken during project execution is to achieve customer satisfaction. The requirements specified by the customer and defined by the organisation must be tested fully to ensure that these have been achieved in the product offered to the customer. Requirement traceability matrix helps project management in understanding the successful implementation of requirements in software on the basis of test execution log. Results of verification and validation must be logged in requirement traceability matrix. Defects found will show that which requirements have not been implemented successfully or are missing in the application under testing.

6.16 SOFTWARE DEVELOPMENT VERIFICATION AND VALIDATION ACTIVITIES

Verification and validation activities are spread over the life cycle of software development. The cycle of verification and validation may include the following.

Conceptualisation Conceptualisation is the first phase of developing a product or a project. For the customer, Conceptualisation means converting the thoughts or concepts into reality. It may be through 'proof of concept' approach or 'prototyping' approach. During proposal, the supplier may give some approach or solution, and the customer may have to evaluate the feasibility of such an approach or solution from product perspective. The supplier must understand what is expected by the customer, and whether it can be provided or not by an organisation. In Conceptualisation, the main stress of verification and validation activity is to determine feasibility of an approach for the project. Feasibility may be depending upon various factors such as technical feasibility, economic feasibility, and skill availability. Verification and validation ensure that an organisation is able to undertake a project and will be able to achieve its mission through the project. Verification and validation by the customer is intended to check the technical capabilities of a supplier to complete the project.

Requirement Analysis Requirement phase may start from conceptualisation. Requirements clarification happens through dialog with users/customer. It can be done by various approaches such as joint application development and customer survey. Requirement analysis verification and validation shows the feasibility of requirements and gives inputs to design approach. It can help in finding the gaps between proposal and requirements so that further clarifications can be achieved. Requirement reviews help in understanding different aspects of customer needs as well as various trade-offs done. Requirement validation can help in identifying any assumption as well as implied requirements in application development.

Design Requirements are implemented through design. Verification and validation of design include understanding that design is complete in all respects and matches with requirements. Requirement traceability ensures that all requirements are converted into design. Design validation involves dummy execution of a

https://E-next.in

Scanned by CamScanner

product design through dataflow diagrams or prototyping, to find whether the design reflects the requirements correctly along with its feasibility. When a flow is complete, design is assumed to be complete.

Coding Coding involves code review and testing of the units, as part of verification and validation, to make sure that requirements and design are correctly implemented. Units must be traceable to requirements through design.

Integration Integration validation and verification show that the individually tested units work correctly when brought together to form a module or submodule. It involves testing of modules/submodules to ensure proper working with respect to requirements and designs. Integration must satisfy low-level design definition such as parameter passing and communication between different units. Integration with other hardware/software is defined in architectural design. Interface testing must satisfy architectural design declared earlier.

Testing Test artifacts such as test plan, test scenario, test bed, test cases, and test data must be subjected to verification and validation activities. Test plan must be complete, covering all aspects of testing expected by the customer. Test bed must mimic real life scenario. Any simulators or any known issues related to test-environment creation must be noted in the risk-management section in test plan. Test scenario must be complete and feasible. Test cases must cover the application completely. Test cases for acceptance testing must be validated by customer/user/business analyst as the case may be.

Installation Application must be tested for installation, if installation is required by the customer. The documentation giving instructions for installation must be complete and sufficient to install the application. Verification and validation must ensure that there is adequate support and help available to common users for installation of an application. User must be guided through all prerequisites. Installation may even be interactive, if requirement specifies so.

Documentation There are many documents given along with a software product such as installation guide and user manual. They must be complete, detailed and informative, so that it can be referred by a common user. The list of documents must be mentioned in contract or statement of work so that compliance can be checked. The documents must undergo validation to ensure that documents and product are in sync with each other.

Tip — Tips for verification and validation activities

- Software testing includes verification and validation activities at each stage of development. It starts from proposal and ends only when the project crosses acceptance testing.
- Verification and validation activities cannot replace each other, rather they support each other.
- Maximum (about 60%) defects can be found in verification. Verification can reduce stage contamination as defect percolations are reduced.
- Verification has three different types, viz. reviews, walkthrough and inspection. Different artifacts undergo different review techniques. Test artifacts also need verification.
- Validation also occurs at different phases of development life cycle. About 25% of the defects can be found in unit testing. Over-reliance on system testing is not an economic solution of finding defects.

https://E-next.in
Scanned by CamScanner

Summary

This chapter provides a clear exposition of verification and validation activities. It covers various methods of verification such as reviews, walkthrough and inspection and various stages of verification such as requirement verification, design verification, and test artifacts verification. Advantages of different levels of reviews such as self review, peer review, walkthrough, and inspection have been dealt in detail. It also presents audits as an independent way than quality control and quality assurance. In-process reviews and post implementation reviews have also been elucidated.

The Second half of the chapter focusses on validation techniques such as unit testing, integration testing, interface testing and system testing. The chapter concludes with a clear overview of acceptance testing.

1) Discuss the advantages and disadvantages of verification.
2) Describe different types of verification on the basis of parties involved in verification.
3) Describe the advantages and disadvantages of self review.
4) Describe the advantages and disadvantages of peer review.
5) Describe the advantages and disadvantages of walkthrough review.
6) Describe the advantages and disadvantages of inspection.
7) Describe the process of inspection.
8) Describe the auditing process.
9) What are the different audits planned during development life cycle?
10) Explain various types of in-process review.
11) Explain the concept of post-mortem review. Why it is essential for a learning organisation?
12) Explain test readiness review.
13) Explain different ways of assessing coverage in verification.
14) What are the concerns of verification?
15) Discuss the advantages of validation.
16) How coverage is measured in case of verification?
17) How coverage is measured in case of validation?
18) Discuss the different levels of validation.
19) Explain different levels of acceptance testing.

https://E-next.in
Scanned by CamScanner