CHAPTER

7

V - TEST MODEL

OBJECTIVES

This chapter establishes 'V model' (validation model) and 'VV model' (verification and validation model). It also defines roles and responsibilities of three critical entities in software development.

7.1 INTRODUCTION

Testing is a lifecycle activity. It starts when the proposal of software development is made to a prospect, and ends only when application is finally delivered and accepted by the customer/end user. For product development, we may define each iteration of development as a separate project, and several projects may come together to make a complete product. In case of maintenance, the cycle may get repeated for every instance of change in present system. For a customer, it starts from a problem statement or conceptualisation of a new product, and ends with satisfactory product receipt, acceptance and usage. For every development activity, there is a testing activity associated with it, so that the phase achieves its milestone deliverable with minimum problem (theoretically, no problem). This is also termed 'certification approach of testing' or 'gate approach of testing', where the gate of a phase opens for delivery, only when the deliverable carries a certification that it meets exit criteria defined for that phase which is entry criteria for the next phase.

Each phase of software development activities must consider corresponding testing activity associated with it. Project plan and estimations in terms of time and effort for a project must consider all the activities of testing (verification and validation) along with development activities corresponding to different phases. The following diagram 7.1 shows the details of software lifecycle development activities, and software lifecycle testing activities associated with them.

7.2 V MODEL FOR SOFTWARE

Validation model describes the validation activities associated with different phases of software development.

- At the requirement phase, there is system testing and acceptance testing, where system testers and users confirm that requirements have been really met or not.

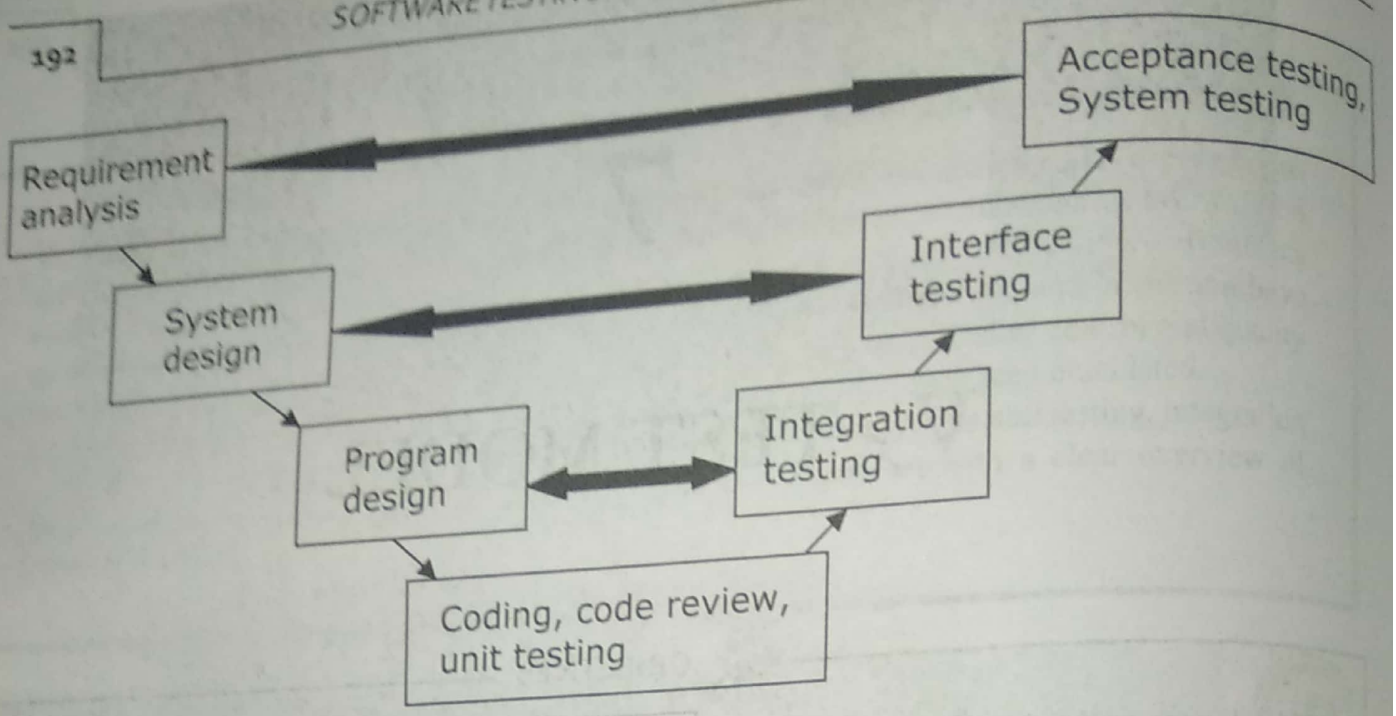

Fig. 7.1 V model for testing (Validation model)

- Design phase is associated with interface testing which covers design specification testing as well as structural testing.
- Program-level designs are associated with integration testing.
- At code level, unit testing is done to validate individual units.

7.2.1 STRUCTURED APPROACH TO TESTING

Testing activities for software development life cycle phases must be planned in advance, and conducted as per plan. Test policy and test strategy/test approach for performing verification and validation activities, responsibilities of stakeholders for supporting or doing these activities, inputs and outputs from each phase of the process, and milestone deliverables must be documented beforehand to avoid any problem in final deliverable to customer. People involved in these activities must have required knowledge, expertise, experience, and training for doing verification and validation activities. Generally, testing activities are referred in project plan but they are detailed in quality plan and verification plan. Some organisations also refer to this as 'verification and validation plan' (V & V plan).

7.2.2 ACTIVITIES DURING EACH PHASE OF SOFTWARE DEVELOPMENT LIFECYCLE

Activities of verification and validation are planned during different phases of software development life cycle, from proposal level till product is finally accepted by the customer. The software development process plan must define the development and testing activities to be conducted in each phase of development, and also, the people responsible for conducting and supporting those activities as stakeholders. Plan of software development must decide about 5 Ws (What, Where, When, Why, and Who) and H (How) with respect to people, processes, tools, training, etc. The information must be readily available, to the team doing these activities and the stakeholders, about their roles and responsibilities for those activities. Activities must be referred to during each phase of software development and testing.

7.2.3 ANALYSE STRUCTURES PRODUCED DURING DEVELOPMENT PHASES FOR ADEQUACY AND TESTABILITY

Documents and work products produced during a life cycle phase must be analysed beforehand to understand their coverage, relationships with different entities, structure in overall development, and traceability. An organisation must have a definition of processes, guidelines and standards which can be used for making such documents and artifacts. Documents and artifacts compliance must be measured in numerical terms with respect to adequacy for the purpose and testability (pertaining to customer needs or organisational standards).

7.2.4 GENERATE TEST SETS BASED ON STRUCTURES

Functional test scenario and test cases are generally developed based upon the functional requirements of the software. Functional requirements refer to the operational requirements of an application. Some requirements may be implemented through designs. In reality, developers implement designs and not requirements.

Structural test scenario and test cases are developed from the structures defined in the design specifications. They must correspond to the structures of the work product produced during development. Requirements/ designs are verified and validated by preparing use case diagrams, data flow diagrams, and prototypes with the question 'what happens when' to identify the completeness of design and requirements. For validation testing scenario, one must use techniques like boundary value analysis, equivalence partitioning, error guessing, and state transition for defining valid as well as invalid ways by which user may interact with the system.

7.2.5 ADDITIONAL ACTIVITIES DURING DESIGN AND CODING

Testing in low-level design and coding phases must confirm that first phase output of capturing the requirements and developing architecture matches with the inputs and outputs required by the low-level design phase. High-level design and low-level design must ensure that requirements are completely covered so that software developed covers all requirements. Verification and validation of design must ensure that the requirement verification and validation is proper and can be handled through the structures created for the purpose. Similarly, verification and validation of coding must ensure that all aspects of designs are covered by the code developed. Definition of next phase verification and validation activities can be initiated in previous phases with definition of output criteria.

7.2.6 DETERMINE THAT STRUCTURES ARE CONSISTENT WITH PREVIOUSLY GENERATED STRUCTURES

Software development is like a flow of events, starting from capturing the requirements till acceptance testing is completed successfully. The outputs of one phase must match with the input criteria of the next phase. There must be consistency between various life-cycle phases. If an architect defines one approach of implementation, then the designer must follow the same path while creating a low-level design. If a system designer decides some approach/reusability, then the developer must understand and implement the approach every time he writes a piece of code. Consistency between various phases can improve the maintainability of an application.

7.2.7 REFINE AND REDEFINE TEST SETS GENERATED EARLIER

Test artifacts such as test scenario, test cases, and test data may be generated in each phase of software development life cycle from requirements, design, and coding, as the case may be. The test artifacts so

generated must be reviewed and updated continuously as per review comments and changes in requirements, designs and related artifacts. One must challenge the validity of each artifact to ensure that it meets the desired results in terms of output criteria. If something is found to be missing, extra or wrongly interpreted in any of the artifacts, it must be corrected before testing execution begins.

7.3 TESTING DURING PROPOSAL STAGE

Requirement statement development starts from system proposal. A proposal is created when the customer asks for information, quotation, proposal, etc. At proposal stage, system description may not be very clear. It must talk about major problems to be solved, the possible solution for which the system is designed, and any major constraints in such definition. People use different approaches such as formal/informal proof of concept, modeling, prototyping, etc. The success of all these approaches during proposal stage lies in successfully defining the problem and proposed solution. It may also consider major constraints related to people, technology, etc.

Feasibility study may or may not be a part of the proposal. Sometimes, conducting feasibility study also needs approval as it may involve some cost and effort, and customer may not want people from other organisations to understand things completely. Feasibility study is done by the customer as well as the supplier in search of a possible solution/approach to solve the problem faced by the customer. It may include technical feasibility, economic feasibility, implementation feasibility, organisational fit, process fit, and people fit.

7.4 TESTING DURING REQUIREMENT STAGE

Requirement gathering stage must cover all the requirements for the system, defined in different groups such as technical, economic, legal, operational and system requirements. Verification of problem definition and requirements definition is the foundation on which the system requirement specification is developed further.

Characteristics of good requirements are mentioned below.

Adequate The requirements must explain the entire system covering end-to-end scenario from the user's side. All the assumptions and constraints/limitations in the system must be defined and documented with possible answers. Many times, scenario or use cases are applied to define end-to-end scenario to find if there have been any definition gaps while creating requirement statement. Requirements must not talk about any impossible thing happening.

Clear/Unambiguous Customer expectation must be reflected in requirements clearly. Requirements must be very clear to the architect, designer as well as developer. It must describe various functions, outputs in various forms, system performance requirements, and interfaces with other systems, in the system as applicable. There must not be any doubt about the outcome of system working to customer as well as developing organisation. Final-user scenario must be defined from requirements so generated. Definitions of actors and transactions must be very clear. Requirements must be prioritised and any trade-off done must be explained.

Verifiable/Testable The requirements must be verifiable and testable. The requirement statement is used for defining functional and structural test scenarios and test cases. It must be used for defining functional as well as structural test data and test events. Testers must be able to search the expected results from the requirement statement.

Measurable Tests defined from requirements must have the expected results, possibly in numerical terms or atleast measurable terms which can be verified during testing. When tester executes any test case,

there must not be any doubt about whether the customer expectations have been met or not. Requirement can be a definite number or a range with boundaries defined with terms like minimum, maximum, etc.

Feasible Requirements must be feasible. It must not refer to some thing which is impossible, or not implementable with given technology, approach, software or system configuration. Requirements must define any future state of organisation which can be achieved through proposed solution.

Not Conflicting with Each Other Two requirements must be in sync with each other. They must not specify anything which is contrary to each other. Requirements must be supplementing and supporting each other. In case of any conflicting requirements, one must ask the customer for a trade-off decision. Generally, requirements are prioritised to avoid any conflicting status. If there is any conflict, the requirement with greater priority would be implemented after getting consent from the customer.

Requirements must include constraints in terms of user profiles, system designs, hardware setups, performance criteria, and quality attributes of the proposed system. It must describe very clearly to the system architects about the number of users expected with average/maximum load, response time expected under different load conditions, minimum and maximum system resources available, types of users, authentications or security requirements.

7.5 TESTING DURING TEST-PLANNING PHASE

Test-planning phase includes defining a test strategy and test approach for the given application, writing test plan, test scenario, test cases, and test data. The senior management from testing/test manager is responsible for defining test strategy/approach, while the test lead/test manager develops a test plan to incorporate test strategies, define schedules, methods of testing, evaluation criteria to declare the outcome of testing, and define test-team approach with roles and responsibilities and structure for the system under testing. Testing artifacts must be reviewed for their consistency and accuracy. Verification of testing artifacts may include the following.

- **Generate Test Plan to Support Development Activities** Test plan must be consistent with the application development methodology, schedule, and deliverables. It must describe respective verification and validation activities to be conducted during each phase of software development life cycle. Test plan must refer to test approach or test strategy, define test objectives, scope of testing, assumptions and risks associated with testing. It must contain methods used for defining test data, test cases, test scenario, and execution of testing activities. It must include how many defects would be expected to be found in a work product during testing, the method to be followed when the defects found will be logged in defect management system, information and analysis required to be shared with stakeholders, and any recommendation at the end of testing.

- **Generate Test Cases Based on System Structure** Functional test cases must be based on functional requirements, and structural test cases must be defined on the basis of design and non-functional requirements of the system. It must be used to define test data using different techniques available such as boundary value analysis, equivalence partitioning, error guessing, and state transition. Test cases and test data must be derived from test scenario.

- **Analyse Requirement/Design Coverage** It is very difficult for any test team to create a test suite to cover 100% requirements and designs produced during software development life cycle. The test

manager decides the objective of requirement coverage and design coverage in test plan, and the customer approves it. Coverage less than 100% indicates a risk, and customer must be involved in making such decision. In case of any shortfall in coverage with respective objectives defined, the test team must analyse the situation and perform risk-benefit analysis of lesser coverage with the help of customer and take corrective measures if required. Theoretically, there is no possibility of less coverage as we cannot expect any requirement/design component given to the customer without testing. Practically, it may not be possible, or it may not be required to cover all requirements, but only P1 requirements may be covered to larger extent. One must conduct the analysis and decide the strategy for coverage.

7.6 TESTING DURING DESIGN PHASE

Design is the backbone of a software application. A successful design can convert the requirements into a good application. Design may be made by system architects or designers, and reviewed and approved by project manager and/or customer. Design must reflect the requirements correctly. Verification and validation of design may include the following.

- **Consistency with respect to Requirements** Designs must be consistent with requirements defined. Sometimes, customer expectations as defined in requirements may be contradicting with each other, and one must perform trade-off. If there is any trade-off between the requirements for implementation, it must be mentioned clearly in design, and customer approval should be taken. The requirement coverage of design must be analysed and measured. If there is no design for a given set of requirements, it will never get implemented. On the other hand, if there is a design component not referring to any requirements, it is considered as a defect. This is something extra given which is not required by the customer.

- **Analyse Design for Errors** Designs generated must be reviewed and tested for completeness and accuracy. A design is implemented by coding. Errors in the design will directly reflect the errors in coding and application so developed. Consistency between design elements must be maintained to avoid any mismatches. Reusability can create robust design and flexible code. Designs must be optimised. They must also be traceable to requirements.

- **Analyse Error Handling** A design must define the error-handling process during application use. It must consider all possible errors such as erroneous data input and invalid transactions, and how design must handle them. Error handling of all kinds and possibilities must be covered in designing aspects. An organisation must have standards for error handling, error messaging and user interactions so that designers are very clear about handling them during design. Error handling plays very crucial role in usability testing.

- **Developers Verify Information Flow and Logical Structure** The designs must be analysed for logical flow of information during transactions. Data flow diagrams or dummy execution of the system is done, and outputs derived are measured against expected outputs. Designs must be consistent with the requirements and user expectations. Logic behind different algorithms and handling of different conditions must be analysed to find out completeness as well as redundancy of designs.

- **Testers Inspect Design in Detail** Testers use design for defining structural test scenario, test cases, and structural test data. Test scenario must be end-to-end scenario considering data flow in the system, and must contain valid as well as invalid conditions, and error handling during user interactions with the system. The normal user must be protected by a good designer from wrong working of the system.

7.6.1 ASPECTS TO BE CHECKED

- **Missing Test Cases** Test cases not defined for a particular scenario (either requirement or design) must be caught in test-case review. Missing test cases must be added to close the review comments. Testers must try to write the test scenario using the requirement statement without referring to design. This helps in identifying the missing test cases. Similarly, integration and interface test cases must be written on the basis of design without checking coding.

- **Faulty Logic** If the logic or algorithm described in design is not correct as per requirement statement, then it must be found by testing. Defects so found must be corrected by designers in design, and then by developers in code. The designs must be traceable back to requirements. Analysis of design must be done.

- **Module Interface Mismatch** The data input/output from one module to another must be checked for consistency with design. Parameter passing is a major area of defects in software, where communication in two modules is affected. Analysis of interfaces between different systems must be done as defined by the requirement specifications, and also how it is handled in design. Interface test cases must test the scenario where one system is communicating with another system.

- **Data Structure Inconsistency** Review of mismatch between data structures and definitions between different modules and system must be done for verification of designs. Data formats from different areas must match during transactions to avoid any loss of data due to mismatch of formats.

- **Erroneous Input/Output** If the system needs to be protected from erroneous operations such as huge/invalid input/output, the design must describe how it will handle the situation. A user must be protected from any mishap through proper designs. Different controls such as protective or, detective or corrective controls can be applied to give sufficient protection to users.

- **User Interface Inadequacies** System may need users to input some information at some point when they are interacting with the system. Entry may be through different methods such as keyboard entry, electronic entry, mouse click, system-to-system data transfer and so on. Similarly, the system may ask users to perform different activities and output may be given to users in different ways such as screen displays, reports, etc. The user interfaces defined by the design must be adequate for the purpose of communication, so that users can work with the system comfortably.

- **Correctness of Decisions and Conditions along All Paths** A system may go through different paths and branches of algorithms based on situations faced by it. The design must be such that all the possible conditions must be defined completely. No single condition must have two different logics or outcomes possible. Logic must be checked for any redundant scenario.

- **Inconsistency with respect to Requirements and High-Level Design** Design and development must be consistent with requirements and high-level design. As the system architecture is defined in high-level design which must reflect system requirements, any deviation can lead to extra or missing functionalities, or inappropriately implemented system.

7.7 TESTING DURING CODING

Coding is the most crucial stage in software development where product is actually build. In most of the projects, the number of developers as well as code files is very large in comparison to qualified people

who can review them. Many organisations are completely dependent on self review and peer review for verification of code. Also, unit testing is done by the developers on their own program/code file to ensure that they achieve their objectives as defined by low-level designs. It is crucial but important for the organisation to establish/institutionalise verification/validation of a code so that the product matches with requirement specifications.

7.7.1 ASPECTS TO BE CHECKED

- **Coding Standards/Guidelines Implementation** Many organisations have coding standards/guidelines defined and used in coding phase of product development. When the code files are written, the developers must use these standards/guidelines. While reviewing code, the peer must make sure that standards and guidelines are implemented correctly. Coding standards and guidelines define the best way or suggested way of doing things. It helps in optimisation and better readability/maintainability of code in future.

- **Coding Optimisation** Coding standards must also talk about optimisation of code. It talks about how nesting must be done, how declaration of variables and functions must be done, how reusable components must be used and so on. Peer review must verify that the written code is properly adhering to optimisation guidelines.

- **Code Interpreting Design** Coding must interpret designs correctly. Coding files and what they are supposed to implement must be defined in low-level design. There must not be anything more or less than what has been defined in low-level design.

- **Unit Testing** Unit testing must be done by the developers to ensure that written code is working as expected. Sometimes, unit testing is done by peer of an author (of a code) to maintain independence of testing with respect to development. Unit test cases must cover valid as well as invalid conditions faced by users. Unit test case logs must be prepared and available for peer review, SQA review as well as customer audits. Defects in unit testing must be logged, and correction as well as corrective/preventive actions must be taken to close them.

7.8 VV MODEL

'V model' is also termed 'validation model' or 'test model' as it mainly considers only validation activities associated with software development which are popularly known as 'testing activities'. But, quality checking involves verification as well as validation activities. 'VV model' considers all the activities related to verification as well as validation. It is also termed 'Verification and Validation Model' or 'Quality model'.

'VV model' talks about verification and validation activities associated with software development during entire life cycle. Let us talk in brief about various activities associated with each phase of software development life cycle.

Requirements As requirements are obtained from customer using techniques described earlier (like oint application development, customer or market survey, and prototyping) there must be some arrangement or conducting requirement review. The intention would be to find if there is any gap existing between user equirements and requirement definition, and also to check whether all the requirements have been captured orrectly and completely or not. Requirement review may concentrate on the structure of requirement pecifications statement using a template and content of requirement specifications as per process definition.

Requirement Verification Generally requirement verification is done through inspection of requirement specification document using checklist, standards or guidelines. Experts in domain, customer representative, and other stakeholders may be involved in conducting such an inspection. Outcome of the inspection must be recorded and defects must be corrected before going further to next stages of development life cycle. Trade-offs, if any, must be documented. Many organisations have a checklist approach to fix the problems in requirement statement before it goes for inspection.

Requirement Validation Requirement validation takes place at two or more stages during software development. The first stage of validation involves writing complete use cases by referring to requirement statement. Any assumption made while writing use case may be a possible gap unless all stakeholders agree that they are implied requirements. Use cases must cover all the possible scenarios. The second stage of validation is through system testing. System test scenario and test cases are defined using requirement specifications. Requirement traceability through test scenario and test cases establish coverage of requirements.

Another requirement validation, though by review, happens when a customer reviews requirement specification and signs off as accepted. Anything signed by the customer as acceptance is considered 'validation'. Requirement validation also happens when customer conducts acceptance testing on the work product. Any defect (popularly called as anomaly) is recorded as a possible defect. One may have to refer to requirement statement to find whether reported defect is actually a defect or not. Sometimes, there may be a deliberate decision taken while creating requirement statement which is found during acceptance testing. This may not be considered as a defect.

Design Design may include high-level design or architectural design, and low-level design or detail design. Designs are created by architects (high-level designs)/designers (low-level designs) as the case may be.

Design Verification Verification of design may be a walkthrough of design document by design experts, team members and stakeholders of the project. Project team along with architect/designer may walkthrough the design to find the completeness and give comments, if any. Traceability of design with requirement must be established in requirement traceability matrix. Many organisations follow some specific tools or methodologies (like UML) to create designs. Use of any case tool must be validated so that defects introduced by tools are known beforehand.

Design Validation Validation of design can happen at two or more stages during software development life cycle. The first stage of validation happens when data flow diagrams can be created by referring to the design document. If the flow of data is complete, design is considered to be complete. Any interruption in flow of data indicates lacunae in design. This is also termed 'flow anomaly'. Sometimes, there is a circular reference in data flow which will lead to a loop which will never stop. Another option may be to create models or prototype to validate design. The second stage of validation happens at integration testing and interface testing. Integration testing is an activity to bring the units together and test them as a module. All units are created as per low-level design and tested in unit testing. Once it is confirmed that they work individually, they are integrated to form a module or system as the case may be. Interface testing is an activity of testing connectivity and communication of application with the outside world. Once the system is integrated as defined in earlier phase, one may have to test it for outside connections like database, browser, operating system, communication software, and user interface. Interface is defined by architectural designs.

Another design validation, though by review, happens when a customer reviews the design specification and signs off as accepted.

Coding Coding is an activity of writing individual units as defined in low level design. It is done by developers where low-level designs are implemented. At some places, database (including tables and stored procedures) is also handled by different teams.

Code Verification As coding is done, it undergoes a code review (generally peer review). Peer review helps in identification of errors with respect to coding standards, indenting standards, commenting standards, and variable declaration issues. Checklist approach is used in code review.

Code Validation Validation of coding happens through unit testing where individual units are tested separately. The developer may have to write special programs (such as stubs and drivers) so that individual units can be tested. The executable is created by combining stubs, drivers and the units under testing. This executable is tested with the test cases mainly derived from low-level design. Test results and defects are then logged.

Another code validation may happen, when customer reviews/performs unit testing of the code file and signs off as accepted.

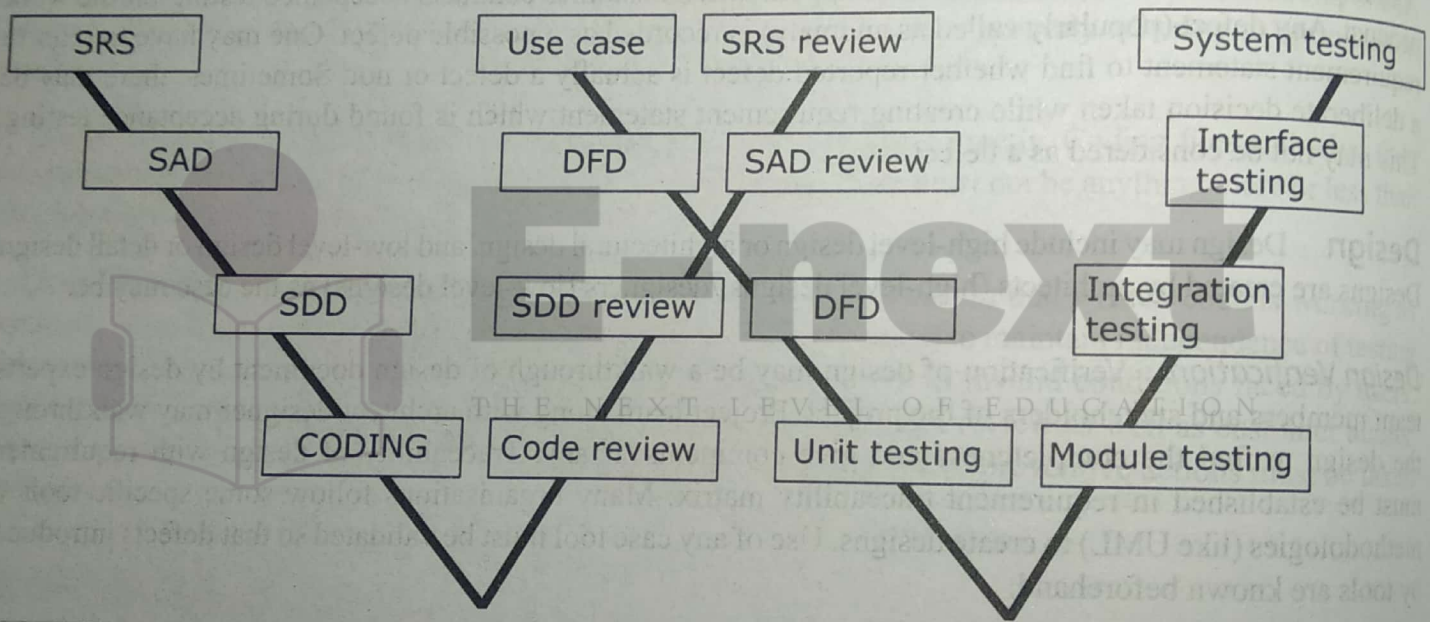

Fig. 7.2 VV model (Verification and Validation model

7.9 CRITICAL ROLES AND RESPONSIBILITIES

In software verification and validation, three critical roles can be identified. It does not refer to different persons, but activities done when different roles are taken by those entities. Depending upon organisation size, maturity, and type of the project, these roles may be performed by different entities at different stages. There is a possibility of overlap between these roles, or gap between these roles. Gap indicates lacunae in organisation structure which must be fixed immediately. Let us try to define these roles and their responsibilities.

Development Development team may be comprised of various roles under them. They may be performing various activities as per their roles and responsibilities at different stages. These subroles may include development manager, project leads, module leads, team leads, developers, and unit testers as per organisation structure and project requirements. Some of the activities related to development group may be as follows

- Project planning activities include requirement elicitation, estimation, project planning, scheduling, and definition of quality attributes required by the customer. Project planning is the foundation of the project's success. Generally, senior people in development (such as project manager) may have the responsibility to create a project plan.
- Resourcing may include identification and organisation of adequate number of people, machines, hardware, software, and tools as required by the project. It may also involve an assessment of skills required by project, skills already available with them and any training needs of team. This may result into different plans such as training plan, procurement plan, etc.
- Interacting with customer and other stakeholders as per project requirements. The interactions may be to gather requirements, get sign offs for each phase, get queries resolved, or send deliverables to the customer.
- Defining policies and procedures for creating development work, verification and validation activities to ensure that quality is built properly, and delivering it to test team/customer as the case may be.
- Supporting testing team by acknowledging defects, giving inputs about requirements, giving executable on time, solving the queries raised by testing team or sending it to customer, as the case may be.
- Doing development-related activities such as capturing requirements, creating designs, coding, and implementation. It may also include quality control related activities such as reviews, walkthrough, etc.

Testing Testing team may include test manager, test leads, and testers as per scope of testing, size of project, and type of customer. Generally, it is expected that a test team would have independence of working and they do not have to report to development team. Roles and responsibilities of test team may include the following.

- Test planning including test strategy definition, and test case writing. Test planning may include estimation of efforts and resources required for testing.
- Resourcing may include identification and organisation of adequate number of people, machines, hardware, software, and tools as required by the project. It may also involve an assessment of skills required by project, skills already available with them and any training needs.
- Interacting with customer and other stakeholders as per project requirements. It may include asking queries, responding to customer/development team requests and so on.
- Defining policies and procedures for creating and executing tests as per test strategy and test plan. Testers may have to take part in verification and validation activities related to test artifacts.
- Supporting development team by providing adequate information about the defects. If required, testers may have to reproduce defects in front of customer/development team.
- Doing acceptance testing related activities such as training and mentoring to users from customer side before/during acceptance testing activities.

Customer Customer may be the final user group, or people who are actually sponsoring the project. Customers can be internal to an organisation or external to the organisation. Roles and responsibilities of a customer may include the following

- Specifying requirements and signing off requirement statement and designs as per contract. It may also include solving any queries or issues raised by development/test team.
- Participating in acceptance testing as per roles and responsibilities defined in acceptance test plan. A customer may be responsible for alpha, beta and gamma testing, as the case may be.

- Review and approve various artifacts as defined by contract or statement of work. A customer may have to participate in various reviews, walkthroughs, and inspection activities as defined.

Tips for verification and validation

- Understand verification and validation activities during software development and testing. Testing includes both verification as well as validation.
- Understand the roles and responsibilities of each activity related to verification and validation. They are always interrelated to each other.
- Understand 'V model' of validation as well as 'VV model' of verification and validation.

Summary

This chapter offers an elaborate discussion of 'V model' of software validation and 'VV model' of software verification and validation. We have also seen activities related to verification and validation, and roles and responsibilities of three critical entities in entire development. Understanding of these roles and responsibilities is crucial to derive a good product.

1) What are the characteristics of good requirements?
2) Describe V & V activities during proposal.
3) Describe V & V activities during requirement generation.
4) Describe V & V activities for test artifacts.
5) Describe V & V activities during designs.
6) Describe V & V activities during coding.
7) What are the roles and responsibilities of development group?