V model

The **V-model** represents a software development process (also applicable to hardware development) which may be considered an extension of the waterfall model. Instead of moving down in a linear way, the process steps are bent upwards after the coding phase, to form the typical V shape. The V-Model demonstrates the relationships between each phase of the development life cycle and its associated phase of testing. The horizontal and vertical axes represents time or project completeness (left-to-right) and level of abstraction (coarsest-grain abstraction uppermost), respectively.

The **V-Model** is a systems development model designed to simplify the understanding of the complexity associated with developing systems. In systems engineering it is used to define a uniform procedure for product or project development.

"V" Model is one of the SDLC Methodologies.

In this methodology Development and Testing takes place at the same time with the same kind of information in thei hands.

Typical "V" shows Development Phases on the Left hand side and Testing Phases on the Right hand side.

Development Team follow "Do-Procedure" to achive the goals of the company

and

Testing Team follow "check-Procedure" to verify them.

V-Model is a Testing Life Cycle Model which is entirely different from Software Development Life Cycle.
V-Model is used for tester's where as SDLC we have developers and testers.

SDLC models are

1. Waterfall model 2.Prototype Model 3. Spiral Model

TLC Models are

1. Fish Model and 2. V-Model

In V-Model the strategy goes like this

Gathering phase User Acceptance Testing.

Analysing phase System Testing

Design Phase (HLD and LLD) Integration Testing

Coding phase Unit Testing

build

Here unit testing is done related to Coding
Integration testing is based on Design phase
System Testing is done related to Requirements (Analysing phase)
User acceptance is done based on Gathering phase.

Overview

The V-model is a graphical representation of the systems development lifecycle. It summarizes the main steps to be taken in conjunction with the corresponding deliverables within computerized system validation framework.

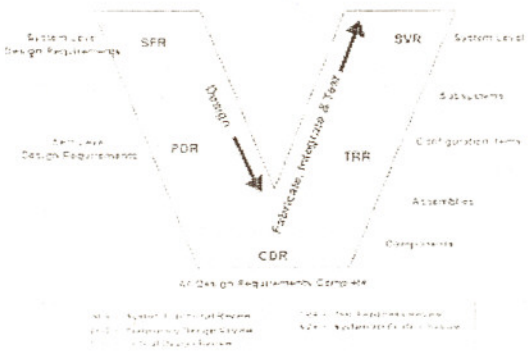
The *VEE* represents the sequence of steps in a project life cycle development. It describes the activities and results that have to be produced during product development. The left side of the "V" represents the decomposition of requirements, and creation of system specifications. The right side of the *VEE* represents integration of parts and their verification. V stands for "Verification and Validation".

Objectives

The V-Model provides guidance for the planning and realization of projects. The following objectives are intended to be achieved by a project execution:

- **Minimization of Project Risks**: The V-Model improves project transparency and project control by specifying standardized approaches and describing the corresponding results and responsible roles. It permits an early recognition of planning deviations and risks and improves process management, thus reducing the project risk.
- **Improvement and Guarantee of Quality**: As a standardized process model, the V-Model ensures that the results to be provided are complete and have the desired quality. Defined interim results can be checked at an early stage. Uniform product contents will improve readability, understandability and verifiability.
- **Reduction of Total Cost over the Entire Project and System Life Cycle**: The effort for the development, production, operation and maintenance of a system can be calculated, estimated and controlled in a transparent manner by applying a standardized process model. The results obtained are uniform and easily retraced. This reduces the acquirers dependency on the supplier and the effort for subsequent activities and projects.
- **Improvement of Communication between all Stakeholders**: The standardized and uniform description of all relevant elements and terms is the basis for the mutual understanding between all stakeholders. Thus, the frictional loss between user, acquirer, supplier and developer is reduced.

V Model topics



Advantages

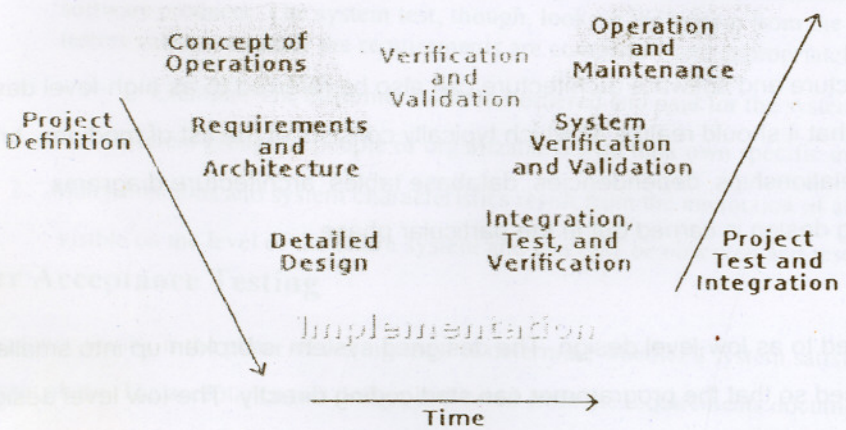
These are the advantages V-Model offers in front of other systems development models:

- The users of The V-Model participate in the development and maintenance of The V-Model. A change control board publicly maintains the V-Model. The change control board meets once a year and processes all received change requests on The V-Model.
- At each project start, the V-Model can be tailored into a specific project V-Model, this being possible because the V-Model is organization and project independent.
- The V-Model provides concrete assistance on how to implement an activity and its work steps, defining explicitly the events needed to complete a work step. each activity schema contains instructions, recommendations and detailed explanations of the activity.

Limits

The following aspects are not covered by the V-Model, they must be regulated in addition, or the V-Model must be adapted accordingly.

- The placing of contracts for services is not regulated.
- The organization and execution of operation, maintenance, repair and disposal of the system are not covered by the V-Model. However, planning and preparation of a concept for these tasks are regulated in the V-Model.
- The V-Model addresses software development within a project rather than a whole organization.



Verification Phases

Requirements analysis

In the Requirements analysis phase, the requirements of the proposed system are collected by analyzing the needs of the user(s). This phase is concerned about establishing what the ideal system has to perform. However it does not determine how the software will be designed or built. Usually, the users are interviewed and a document called the user requirements document is generated.

The user requirements document will typically describe the system's functional, interface, performance, data, security, etc requirements as expected by the user. It is used by business analysts to communicate their understanding of the system to the users. The users carefully review this document as this document would serve as the guideline for the system designers in the system design phase. The user acceptance tests are designed in this phase. There are different method

for gathering requirements of both soft and hard methodologies including; interviews, questionnaires, document analysis, observation, throw-away prototypes, use cases and status and dynamic views with users.

System Design

Systems design is the phase where system engineers analyze and understand the business of the proposed system by studying the user requirements document. They figure out possibilities and techniques by which the user requirements can be implemented. If any of the requirements are not feasible, the user is informed of the issue. A resolution is found and the user requirement document is edited accordingly

The software specification document which serves as a blueprint for the development phase is generated. This document contains the general system organization, menu structures, data structures etc. It may also hold example business scenarios, sample windows, reports for the better understanding. Other technical documentation like entity diagrams, data dictionary will also be produced in this phase. The documents for system testing are prepared in this phase.

Architecture Design

The phase of the design of computer architecture and software architecture can also be referred to as high-level design. The baseline in selecting the architecture is that it should realize all which typically consists of the list of modules, brief functionality of each module, theirinterface relationships, dependencies, database tables, architecture diagrams, technology details etc. The integration testing design is carried out in the particular phase.

Module Design

The module design phase can also be referred to as low-level design. The designed system is broken up into smaller units or modules and each of them is explained so that the programmer can start coding directly. The low level design document or program specifications will contain a detailed functional logic of the module, in pseudocode:

- database tables, with all elements, including their type and size
- all interface details with complete API references
- all dependency issues
- error message listings
- complete input and outputs for a module

The unit test design is developed in this stage.

Validation Phases

Unit Testing

In computer programming, unit testing is a method by which individual units of source code are tested to determine if they are fit for use. A unit is the smallest testable part of an application. In procedural programming a unit may be an individual function or procedure. Unit tests are created by programmers or occasionally by white box testers. The purpose is to verify the internal logic code by testing every possible branch within the function, also known as test coverage. Static

analysis tools are used to facilitate in this process, where variations of input data are passed to the function to test every possible case of execution.

Integration Testing

In integration testing the separate modules will be tested together to expose faults in the interfaces and in the interaction between integrated components. Testing is usually black box as the code is not directly checked for errors.

System Testing

System testing will compare the system specifications against the actual system. After the integration test is completed, the next test level is the system test. System testing checks if the integrated product meets the specified requirements. Why is this still necessary after the component and integration tests? The reasons for this are as follows:

Reasons for system test

1. In the lower test levels, the testing was done against technical specifications, i.e., from the technical perspective of the software producer. The system test, though, looks at the system from the perspective of the customer and the future user. The testers validate whether the requirements are completely and appropriately met.
    - Example: The customer (who has ordered and paid for the system) and the user (who uses the system) can be different groups of people or organizations with their own specific interests and requirements of the system.
2. Many functions and system characteristics result from the interaction of all system components, consequently, they are only visible on the level of the entire system and can only be observed and tested there.

User Acceptance Testing

Acceptance testing is the phase of testing used to determine whether a system satisfies the requirements specified in the requirements analysis phase. The acceptance test design is derived from the requirements document. The acceptance test phase is the phase used by the customer to determine whether to accept the system or not.

Acceptance testing helps

- to determine whether a system satisfies its acceptance criteria or not.
- to enable the customer to determine whether to accept the system or not.
- to test the software in the "real world" by the intended audience.

Purpose of acceptance testing: to verify the system or changes according to the original needs.

[edit]Procedures

1. Define the acceptance criteria:
    - Functionality requirements.
    - Performance requirements.
    - Interface quality requirements.
    - Overall software quality requirements.

2. Develop an acceptance plan:
    - Project description.
    - User responsibilities.
    - Acceptance description.
    - Execute the acceptance test plan.

Agile development

as a basis but advocates a lighter and more people-centric viewpoint than traditional approaches. Agile processes use feedback, rather than planning, as their primary control mechanism. The feedback is driven by regular tests and releases of the evolving software.

There are many variations of agile processes:

- In Extreme Programming (XP), the phases are carried out in extremely small (or "continuous") steps compared to the older, "batch" processes. The (intentionally incomplete) first pass through the steps might take a day or a week, rather than the months or years of each complete step in the Waterfall model. First, one writes automated tests, to provide concrete goals for development. Next is coding (by a pair of programmers), which is complete when all the tests pass, and the programmers can't think of any more tests that are needed. Design and architecture emerge out of refactoring, and come after coding. Design is done by the same people who do the coding. (Only the last feature — merging design and code — is common to *all* the other agile processes.) The incomplete but functional system is deployed or demonstrated for (some subset of) the users (at least one of which is on the development team). At this point, the practitioners start again on writing tests for the next most important part of the system.

- Scrum

Extreme Programming

Extreme Programming (XP) is a software development methodology which is intended to improve software quality and responsiveness to changing customer requirements. As a type ofagile software development, it advocates frequent "releases" in short development cycles (timeboxing), which is intended to improve productivity and introduce checkpoints where new customer requirements can be adopted.

Other elements of extreme programming include: programming in pairs or doing extensive code review, unit testing of all code, avoiding programming of features until they are actually needed, a flat management structure, simplicity and clarity in code, expecting changes in the customer's requirements as time passes and the problem is better understood, and frequent communication with the customer and among programmers. The methodology takes its name from the idea that the beneficial elements of traditional software engineering practices are taken to "extreme" levels, on the theory that if