

Software Engineering

## ESOF – 3MIEIC05

T2 – Software Processes

[Duarte Nuno Esteves André Lima de Carvalho](mailto:Duarte%20Nuno%20Esteves%20André%20Lima%20de%20Carvalho)– [up201503661@fe.up.pt](mailto:up201503661@fe.up.pt)

João Álvaro Cardoso Soares Ferreira – [up201605592@fe.up.pt](mailto:up201605592@fe.up.pt)

[João Augusto dos Santos Lima](https://sigarra.up.pt/feup/pt/fest_geral.cursos_list?pv_num_unico=201605314) – [up201605314@fe.up.pt](mailto:up201605314@fe.up.pt)

# Índice

Conteúdo

[V-Model 3](#_Toc527250182)

# V-Model

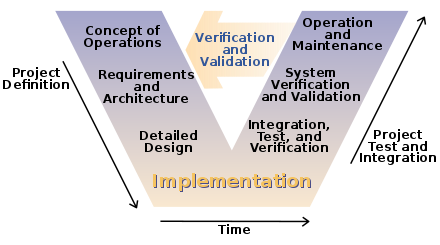


Figure - A basic representation of the V-Model software development cycle

A Brief Summary

The V-Model is a strict, linear and unique model for a software development cycle that originated in Germany, being the official project management methodology used by the government in the early 1990s.

Despite its origin as an equivalent to the PRINCE2 method for project management, it’s shown itself to be very relevant to software development along with systems engineering in the military and medical industries, and is still used by governments such as the USA’s or the UK’s. Its main asset is its reliability and guarantee of product quality, though at a loss for flexibility.

The system itself can be summed up in its shape: a V, with its two sides representing the following:

* the left side entails the verification stages, namely the requirements analysis, system design, architecture design and module design;
* the right side entails the validation stages, namely the unit, integration, system and acceptance testing.

At the bottom of the V we have the implementation, where the code is developed. The stages themselves aren’t set in stone as there isn’t one universal V-Model, but these are the four stages that, in one way or another (even maybe just named differently), a V-Model for software development goes through. It is also valuable to note is that while the original V-Model used verification via analysis, demonstration, investigation and testing, the model for software development uses tests.

It’s important to note that the tests that are developed for the right side of the V are defined simultaneously to the concepts that they are intended to test, having a corresponding stage on the opposite side – respectively, the requirement analysis and acceptance testing, the system design and system testing, the architectural design and integration testing and the module design and unit testing. These details are crucial, as it is what differentiates the V-Model from other models for software development. While others might also be similarly strict and develop the testing simultaneously to the conceptualization, the V-Model is the only one that accompanies the project’s development by cycling through the stages in this manner.

The Stages

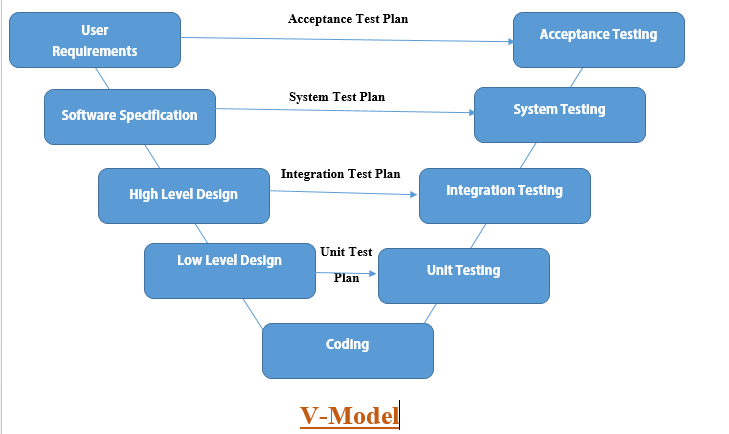


Figure - Extended Software Development V-Model

## ****User Requirements****

This is the first stage of development after communication with the client has been established. Here, the user’s needs are determined, as are the solutions that will be required to meet them.

## ****System Design****

In this stage, the goal is to create a document to determine the techniques, frameworks, materials and methods that will be optimal to develop the solutions outlined in the previous stage. Communication with the client is continued and their feedback is used for this stage as to what might or might not be feasible to develop.

## ****Architecture Design****

Also known as high-level design, in this stage the various concepts designed in the last two stages are connected in a functional manner, to determine how the application will link them all together. It’s in this stage that class diagrams are often drawn up, for example.

## ****Module Design****

Commonly represented as “low level design”, this final stage before the implementation is supposed to define all of the functions and break down the concepts in the previous stages so that the coders, in the following stage, can just start implementing the functions directly. A good tool to do this would be Doxygen documentation, for example.

## ****Implementation****

Usually the most time-consuming stage, the implementation is where all of the concepts and specifications done in the previous stages is written into code and made from theoretical to practical. All of it must be developed before the testing can begin.

## ****Unit Testing****

Developed during the module design by the development team, this phase should be able to rid the program of all bugs at a code level, being also the lengthiest of the testing stages.

## ****Integration Testing****

Developed during the architecture design, to check if all components are working together as intended.

## ****System Testing****

Often developed in contact with the consumer, these tests verify if the entirety of the application is performing as expected. This is developed during the system design.

## ****Acceptance Testing****

The last overall stage, acceptance testing is done by monitoring the contact the user has with the application, verifying that all needs are met. As mentioned before, these tests are developed during the user requirements stage.

Positives

* Due to a highly disciplined strict approach, there’s an overwhelming amount of testing, documentation and stability, leading to an overall more secure product;
* Simple to use and understand, especially for small projects;
* Prevents extensive debugging, as all errors are discovered at an appropriate stage;

Negatives

* It’s not an Agile model, lacking flexibility and a good response to changes during or after the development process;
* It promotes strictness and streamlining, sometimes overlooking creative solutions;
* Issues with time constraints;
* Ill-suited for overwhelming support after launch such as constant patches;
* Easy to apply and understand at first, but also prone to lead unexperienced developers to stick to its guidelines too much.

When to Use V-Model

It is ideal for projects where reliability and stability are a must-have, where costs and time-constraints aren’t an issue (as both might be unexpectedly high) and where patching and updates aren’t expected. It is thus very appropriate for matters of developing software for science, military and healthcare, as was mentioned beforehand.

Example of Success

As was just mentioned, the medical industry has adopted the V-Model for its software needs, as an there are quite a few difficulties with adopting software developed with an Agile method ([source](https://link.springer.com/chapter/10.1007%2F978-3-642-30439-2_13?LI=true)) and the V-Model’s reliability has shown it to be ideal for their needs. In this case, the V-Model’s success has shown itself in stability in its adoption and in the lack of any major failures in the medical industry that can be attributed to it.

Sources

[A Software Process Development, Assessment and Improvement Framework,for the Medical Device Industry - Dundalk Institute of Technology](http://eprints.dkit.ie/144/)

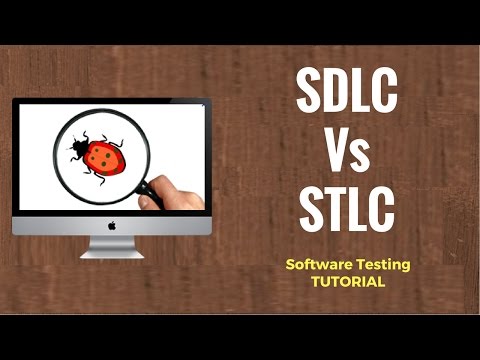
[What is V-Model? - airbrake.io](https://airbrake.io/blog/sdlc/v-model)

[V-Model vs Scrum: Who Wins? - reqtest.com](https://reqtest.com/agile-blog/v-model-versus-scrum-who-wins/)

[Good Design Practice for Medical Device and Equipment - Cambridge University](https://www-edc.eng.cam.ac.uk/downloads/gooddesignpractice1.pdf)

[Using V-Model for Testing - Carnegie Mellon University](https://insights.sei.cmu.edu/sei_blog/2013/11/using-v-models-for-testing.html)

introducing the V-Model with a Case Study

[](https://www.youtube.com/watch?v=An7HC1LolDM)

Course Design and Orienting Students with the "V Model" - MIT

[](https://www.youtube.com/watch?v=Wc0PmAIEUhM)