**V-Model**

Source: <https://www.tutorialspoint.com/sdlc/sdlc_v_model.htm>

<https://airbrake.io/blog/sdlc/v-model>

Barry Baum

**Application:**

Requirements are well defined, clearly documented and fixed.

Product definition is stable.

Technology is understood and not dynamic.

No ambiguous requirements.

Project is short.

Suited for restricted projects: stringency is key. Heavily used in medical device industry where the project length and scope are well-defined, technology stable, requirements are very clear.

Good for projects with a strict deadline and meet key milestone dates throughout the process. (Keeps a tight schedule)

**Advantages:**

Highly disciplined model providing high cohesion in the team.

Works well for smaller projects where requirements are very well understood.

Simple, easy to use.

Easy to manage due to rigidity of the model. Each phase has specific deliverables and a review process.

**Disadvantages:**

Not a good model for long/OO/ongoing/complex projects.

Not suitable where requirements are at a moderate/high risk of changing.

Difficult to go back and change functionality once the application hits testing.

No working software produced until late during the lifecycle.

Encourages ‘Design-by-Committee’ Development: While V-Model is certainly not the only development model to fall under this criticism, it cannot be denied that the strict and methodical nature of the V-Model and its various linear stages tend to emphasize a development cycle befitting managers and users, rather than developers and designers. With a method like V-Model, it can be all too easy for project managers or others to overlook the vast complexities of software development in favour of trying to meet deadlines, or to simply feel overly confident in the process or current progress, based solely on what stage in the life cycle is actively being developed.

