

## The Death of the V-Model

Ed Liversidge, Director, Harmonic Software Systems Ltd

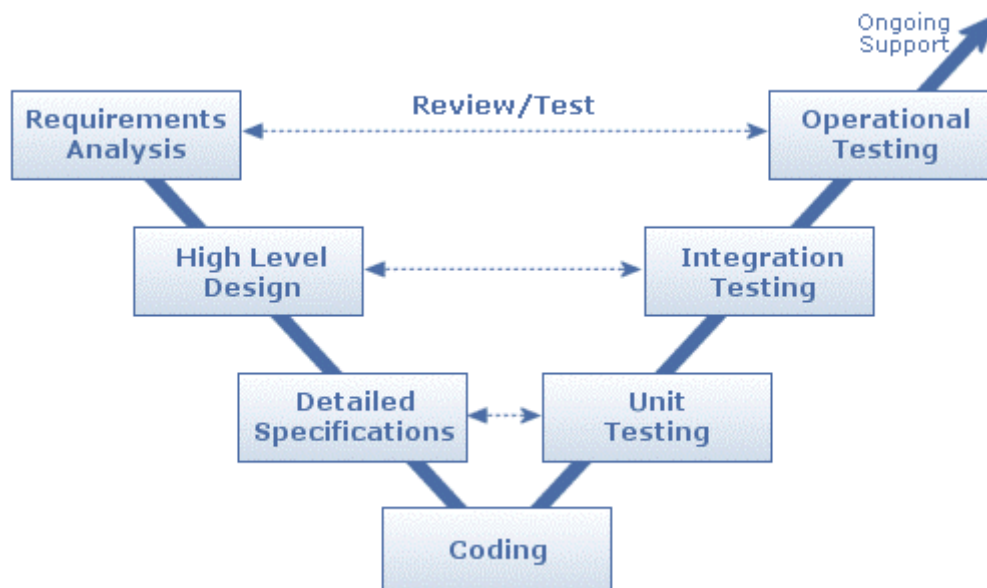
07 December 2005

### **Introduction**

The V-Model of software development is widely in use today, especially in the defence industry. It's a pity then, that it is fundamentally flawed, and that it is responsible for misleading project managers into thinking that the project they are about to undertake is well understood. The reality is that the more the V-Model is used as a tool to manage the software development process, the more likely that project is to fail.

### **The V-Model of Software Development**

Figure 1 shows a typical diagram of the V-Model. The development life cycle follows a fixed path from Requirements Analysis to Operational Testing and subsequent delivery to the customer. Testing tasks are generated from knowledge gained from development tasks, for instance the High Level Design will generate the Integration Tests. A project following this model will move through these tasks one at a time, moving on when the current task is completed.



**Figure 1 – The V-Model**

This model does have a number of good points, such as:

- It defines tangible phases of the process, and proposes a logical sequence in which these phases should be approached. It also defines a logical relationships between the phases.
- It demands that testing documentation is written as soon as possible, for example, the integration tests are written when the high level design is finished, the unit tests are written when the detailed specifications are finished.
- It gives equal weight to development and testing.
- It provides a simple and easy to follow map of the software development process.

However, this is as far as it goes. This simple model does not account for the complexities involved, and thus can prove to mislead a project manager if relied upon.

### **The Dying V-Model**

Let us take a look at the software development process for a moment. As stated by P.G. Armour, in "The Laws of the Software Process"<sup>1</sup>, software development can be viewed as a quest for knowledge. Like a climber planning a route over glaciers and up a mountain face, the destination is clear, but getting a safe route up the mountain and back down again takes a mixture of careful planning and an adaptive approach to events. The climbers may find impassable chasms, dangerous overhangs or unpredictable changes in weather. The plan will probably change.

In the software development world, you can bet your last dollar that the plan WILL change. That means that the software development model HAS to model change. The V-Model does nothing to accommodate change, and this is the primary reason why it fails as a model.

A second reason why the V-Model fails, is in the testing phases, and has been illustrated by Brian Marick<sup>2</sup>. He explains that implementing unit tests and integration tests as single, separate phases results in a thoughtless approach to testing. For example, a single unit test will require a custom test harness. Each unit may require a different test harness. For a large projects with lots of units, this could prove to be costly and problematic. It could be a better idea to test a unit when connected to the actual system, using the system to deliver test messages. The point is that no thought is being applied to the trade-offs involved in testing early, testing late, or testing as-you-go.

---

<sup>1</sup> "The Laws of the Software Process", Philip G. Armour, Auerback Publications, 2004

<sup>2</sup> "New Models for Test Development", Brian Marick, <http://www.testing.com>

## **The V-Model is Already Dead**

In reality, a development project that is working is not following the V-model, even if management believe that this is so. This is because good engineers are flexible and adapt to problems, they are able to expect and respond to change. They test code when it has been written, to get a feel for the system. Extremely good engineers even update the documentation when the design changes!

## **The Final Nail in the Coffin**

The most damaging aspect of the V-Model is not in the model itself. Any model is an approximation, and this model does at least provide some value. The biggest problem arises from the manager's steadfast reliance on the model, making the assumption that the model is a 'tool' to be 'used'. When the V-model is accepted as a good model for software development, it blinds the project manager to the realities of the software development process, and distorts this process to the point whereby it is counter productive. It is like a one dimensional view of a four dimensional universe – fundamentally flawed and very far removed from reality. As it is this bad, don't even bother to mention it except in the history books as a dead model.

## **Conclusions**

The V-Model is an inadequate model for software development for the following reasons:

- It is too simple to accurately reflect the software development process, and can lead managers into a false sense of security.
- It is inflexible; it has no ability to respond to change.
- It produces inefficient testing methodologies.

It is time to lay the V-Model to rest. RIP V-Model !!!

