**V-Model (V-Modell)**

The V-Model also known as the verification and validation model is a software development process which is considered to be an extension of the traditional waterfall method with extensive focus on testing.

Instead of moving in a linear fashion, the V-Model first defines everything from the concepts, down to the low-level design, but at the same time it specifies all the criteria it must be verified against.

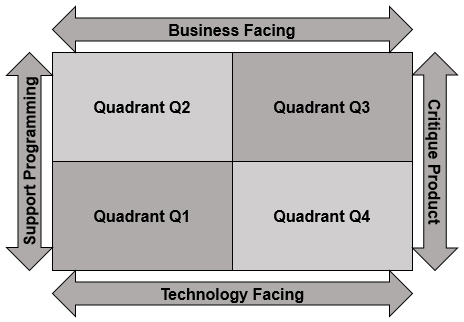
After the implementation, it verifies the application based on the previously written validation points. If the application passes the verification step it is ready to be shipped to the customer.   
  
**Pros**

* The final product will always be at a very high-quality standard as the product is only shipped once the application has passed all the tests
* It shows a clear roadmap which the developers can follow (step by step process)
* Since testing is so tightly incorporated less time will be wasted in later stages of development thus making the application less time-consuming in later stages of development

**Cons**

* As it is an extension of the waterfall method it is inherently less flexible when compared to agile methodologies such as The Quadrants model
* May not be correct for a more chaotic development team/environment
* It may trick developers into a false sense of security because of the high focus on tests
* Documentation is necessary at all time which is very time consuming
* Since no prototypes are created there is a very high risk involved with meeting customer expectations

**The Quadrants model**

****The quadrants merely act as a map to guide developers when they plan their tests and make sure all resources are available in order to accomplish them. The Quadrants consist of four sections (Q1,Q2,Q3 and Q4, note the numbering system does not imply any particular order as it’s an arbitrary numbering style to shorten the naming of each quadrant)

To decide if a test is business facing or technology facing:

A test is business facing if it answers business domain related questions. These questions are best understood by business experts and help answer how the application will function in a specific scenario

A test is technology facing if it answers a technology framed question. These questions are best understood by programmers/system architects who understand what needs to be implemented based on these questions.

**Pros**

* As the quadrants model is an agile testing methodology it is very flexible in the environment it exists in
* Customers can have first hand experience with the product and can give immediate feedback
* If set up correctly it can provide a comparably high-quality product to the V-Model

**Cons**

* The lack of documentation can be problematic in certain projects
* There are no specific rules about what goes into which quadrant
* Initial discussions need to be held for each quadrant so some time is spent at the start of each quadrant discussing testing

After carefully reviewing the methodologies we decided to choose the V-Model rather than the agile quadrants model as it better suits our current needs of focusing on testing in the GTL project. Although the Agile Quadrants does better suit our chaotic style of work, the project requirements put a heavy emphasis on testing, which is more appropriate to the waterfall methodology so we thought the V-Model would be a natural fit. Also, as the quadrants require time to negotiate as to what goes into which quadrant (and the fact that we are working on a tight schedule) it further pushed us towards the alternative. Although we can’t fully adopt the V-Model, we can integrate it into smaller sections of the project, which are mainly related to programming tasks. This is done so every time we finish a task we know its instantly ready to be integrated into the final software solution as the it already meets to the quality standards (due to the validation nature of the approach).  
In the actual implementation of the methodology we did modify the V-Model to fit our needs, for example: we omitted the sequentially as it limited our flexibility which we were not prepared to give up this major plus point of agile.