QM	Course:	Quality Management, DPT404		
	Teacher:	Conny Johansson	Department:	IDE, University Of Karlskrona/Ronneby

The V-Model

Prepared for

Conny Johansson
Conny.Johansson@ide.hk-r.se
IDE, University Of Karlskrona/Ronneby

Prepared by

Christian Bucanac
c.bucanac@computer.org
Software Engineering Student,
University Of Karlskrona/Ronneby

1999-01-04

Contents

Introduction

This report is a part of the examination in the Quality Management course, DPT404 given at the University of Karlskrona/Ronneby.

The report contains a description of the German Federal Armed Forces software lifecycle process model, called The V-Model. It also contains a discussion of the model.

This report together with the other student's reports is used in the quality model seminar later during the course.

Description of The V-Model

In this section I describe The V-Model. In the following subsection "Brief Description" I describe the general structure of The V-Model. I have chosen to describe the Allocation of Methods and Functional Tool Requirements levels briefly in this subsection. The Lifecycle Process Model is described in more detail in the subsection called "The Lifecycle Process Model".
I have decided to do this since the Lifecycle Process Model is the heart of The V-Model. It describes the activities and products in The V-Model. The other levels complement the Lifecycle Process Model with methods and tools. What methods and tools are used to perform the activities and produce the products is less important. They differ from one project to another, but the Lifecycle Process Model stays the same.

Brief description

The general structure of The V-Model

The V-Model, encompasses these three levels:

- The Lifecycle Process Model (Procedure in Figure 1) – Answers the question "What has to be done?". The procedures establish what activities are to be performed, which results these activities should produce and what contents these results must have.
- The Allocation of Methods (Methods in Figure 1) – Answers the question "How is it done?". In this level it is determined what methods are to be used to perform the activities in the procedure level.
- The Functional Tool Requirements (Tool Requirements in Figure 1) – Answers the question "What is used to do it?". In this level it is established what functional characteristics the tools must have that are used to perform the activities.

At all levels the standards are structured according to the areas of functionality. These areas of functionality are called submodels. There are four submodels:

- Project management (PM) – This submodel plans, monitors, controls the project. It also passes information to the other submodels.
- System Development (SD) – This submodel develops the system or software.
- Quality Assurance (QA) – This submodel specifies the quality requirements and informs the other submodels of it. It specifies for example test cases and criteria to assure that the products and processes comply with the standards.
- Configuration Management (CM) – This submodel administrates the generated products.

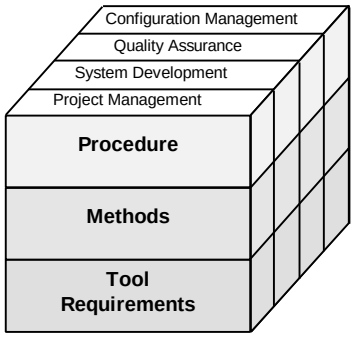

Figure 1 - The architecture of The V-Model

The Allocation of Methods and Functional Tool Requirements

The Allocation of Methods level regulates the selection and application of methods for all submodels. It complements the Lifecycle Process Model. It specifies how something is done, while the Lifecycle Process Model specifies what has to be done.

There are two kinds of methods, basic and complex. The basic methods refer to the procedures that describe a special limited aspect of the system (for example functional or data oriented aspect) or certain section of the system development (for example analysis or design). The basic methods are categorized. A category compromises those basic methods that offer different solutions approaches for a certain tasks of problems. Complex methods refer to the procedures that compromise various methodical components and integrate them into a total method.

For each activity, a method is selected. This selection or activity-method mapping is documented in the project manual. To make a selection easier the Allocation of Methods level has described a limited set of methods. Each method is described according to this structure:

- Identification and definition of the method.
- Brief characteristics of the method.
- Limits of the method.
- Specification of the method allocation – Specifies how the method is to be used in the activities.
- Interfaces – Interfaces to other methods that supplement each other are described. For example, the use case modeling has an interface to class object modeling, state transition modeling and interaction modeling.
- Further literature about the method.

As with the Allocation of Methods level, the Functional Tool Requirements level regulate the selection and application of tools for all submodels. It also complements the Lifecycle Process Model. In addition, it supports the Allocation of Methods level. A method can use one or several tools.

A tool is defined as " A tool is a software product supporting the development or maintenance/modification of IT systems". The tools are grouped into service units. A service unit defines the requirements for the tools in the specific service unit. A service unit can either or both be applied in an activity in the Life Cycle Process Model or a method in the Allocation of Methods level. The service units are further grouped into service complexes. A service complex can for example be one of the submodels. The service complexes build up the Reference Model of a Software Development Environment (SDE). It arranges all IT services offered by a SDE into a basic schema.

The selection of tools is done in four steps:

- In the first step you specify the required functionality of the tools on the basis of the functionality required for a project. This step results in a selection of some service units that fulfill the specified required functionality.

- In the second step there are two activities:
  - In the first activity you have the selected service units and the application conditions for the tools as input. The activity considers the special environmental allocation and possible prioritization of the requirements. The output of this activity is the functional requirements of the tools.
  - In the second activity you have the application conditions for the tools as input. The activity specifies the technical and organizational requirements. The output of this activity is the technical requirements.
- In the third step, the functional and technical requirements for the tools are summarized. The summarization results in an operational criteria catalogue. It represents the final requirements of the tools.
- In the fourth step you compare the different tools with help of the tool descriptions or profiles and the operational criteria catalogue. The comparison results into some applicable tools.

The Lifecycle Process Model

Basic elements

Activities and Products

The Lifecycle Process Model describes the processes in two basic concepts, activities and products. It also describes the states of the products and the interdependencies between the activities and products with the product flow schema.

Activities are worksteps in the development process. The execution and the results of an activity are exactly described. An activity may consist of subactivities. The activity at the highest level is called main activity. The result from an activity is a product. An activity generates, changes state or modifies a product. For each activity there is an activity description which is based on an activity pattern called activity schema.

A product can be a document, piece of software or hardware. A product is the result from an activity. As with activities, the products may be decomposed into subproducts. A product is described with a product schema. It contains a short synopsis of the product and a listing of the structural items of the product.

Activity schema

An activity schema pattern consists of:

- The name of the activity – Describes the name and the number of the activity. This may be a main activity or a subactivity. Subactivities are numbered with a dot notation.
- The product flow – Describes the flow of the product. It describes the input and output product of an activity. In the input, it is described from which activity the product comes from and it's state. When the product has be processed by the specific activity it is outputted. In the output, it is described to which activity the product is passed on to and it's state. See Figure 2.
- The handling – Describes how the activity must be handled during realization. In a main activity, the subactivities, their interdependencies and their results are graphically described in the handling section.
- The recommendations and explanations – Gives recommendations on how the activity can be handled. Explanations clarify the definitions of the activity and make it easier to understand.

from		Product	to	
Activity	State		Activity	State
External	—	Project Order	—	—
—	—	Project Manual. *Project Organization*	PM 1.3	b. proc.

Figure 2 - Example of product flow: PM 1.1: Setting up the project

Explanation to Figure 2**:**

- The Project Order comes from an external part and has no state. It does not go further to another state and does not change state.
- The Project Manual does not come from any activity and has no state. It is therefore newly created in this activity. It is passed on to the PM 1.3 activity (Generation of Project-Specific V-Model) with the state being processed. The Project Manual is one of the input products to the PM 1.3 activity.

Product states

A product undergoes different states during it's generation and processing. A change of state can only be triggered by an activity. In the activity schema it is described what state a product has when it enters the activity and when it leaves the activity.

A product may have these states:

- Planned – The product is being planned. The product does not exist yet. It is the initial state of all products.
- Being processed – The product is being processed. It is in control of a developer. It is either checked-out from the product library or checked-in to the product library.
- Submitted – The product is from the developer's point of view finished. The product is submitted to the QA for assessment. If the QA assessment rejects the product it is returned to the being processed state otherwise it is accepted.
- Accepted – The product has been accepted by the QA assessment and is therefore released. It can only be further modified if the version number is updated.

The states and transitions are modeled in Figure 3.

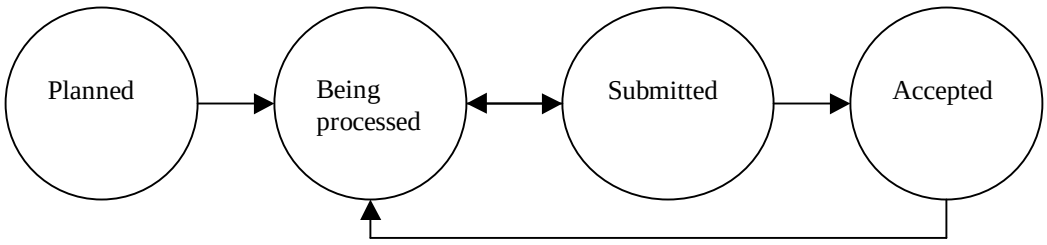

Figure 3 - The product states and transitions

Organization and Roles

The allocation of tasks to individual project members is done by roles. A set of roles are listed for each submodel. Each role has defined which activity it allocates, the responsibilities it has, skills needed, know-how needed and dependencies with other roles.

The organization is built up by selecting project members for each submodel and assigning them roles. A role may be given to one or several project members. One project member can also have several roles. This allocation of roles makes The V-Model impartial of a project's organization.

The Submodels

As mentioned earlier, there are four submodels in The V-Model. In this section there are described in more detail from the Lifecycle Process Models point of view.
All submodels consist of some activities and generate some products. They also cooperate together to achieve the project goal. This is shown in Figure 4.

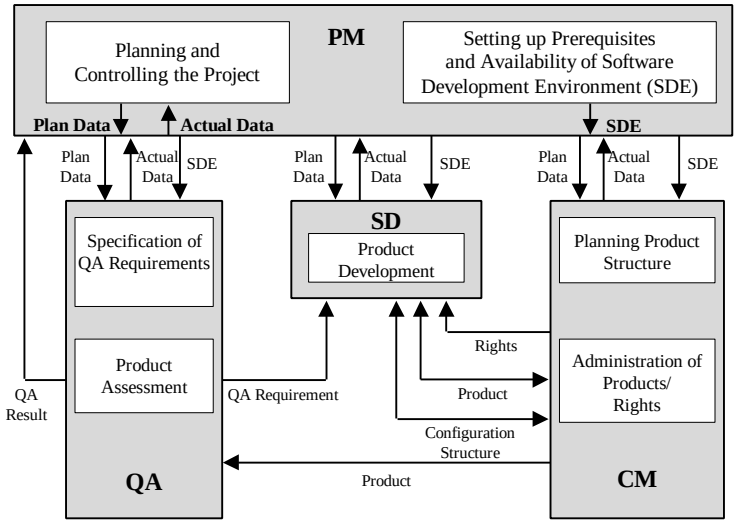

Figure 4 – The Cooperation of submodels

Project Management

The PM submodel regulates the activities of initiating, planning and monitoring the project. It compromises of these 14 main activities:

- Project Initialization – Defines the organizational framework in the project manual. A tailoring of The V-Model into a specific project V-Model is done according to the project criteria and conditions. Preliminary planning is done based on the project manual and documented in the project plan.
- Placement/Procurement – Involves sending out request of proposal to possible subcontractors, evaluating the responses and determining the most economic offer.
- Contractor Management – The goal of this activity is to supervise that the terms of the contract are fulfilled. This concerns both the contracts with the customer and subcontractors.
- Detailed Planning – The goal of this activity is to realize a detailed plan for the project. This is done with help of the existing preliminary planning and the specifications in the project manual.
- Cost/Benefit Analysis – The goal of this activity is to determine how profitable the planned solutions are. Each solution proposal is evaluated according to it's cost and benefits.
- Phase Review – After each project phase, a phase review is done. The review is done to check the actual project status according to the planned.
- Risk Management – The goal of this activity is to detect project risks as early as possible. The risks are managed by initiating preventive measures and supervising the efficiency of the initiated measures.
- Project Control – The goal of this activity is to control the progress of the project. If it deviates from the planned values, actions should be taken to correct the problem.
- Information Service/Reporting – The goal of this activity is to make information about the project available to the customer, subcontractors and project members.
- Training/Instruction – The goal of this activity is to train the project members, so that they can acquire the knowledge needed to fulfill their role/s.

- Supplying Resources – The goal of this activity is to supply the resources needed to realize the project's goal.
- Allocation of Work Orders – The goal of this activity is to initiate job sections allocated by the work orders.
- Staff Training – The goal of this activity is to introduce the project members to their job section. The project members task is explained and they are informed about the task.
- Project Completion – The goal of this activity is to complete the project. This involves writing a final project report with all project experiences.

System Development

The SD submodel regulates the activities of developing the system. It compromises these nine main activities:

- System Requirements Analysis – This activity involves setting up the system requirements from the user's point of view. The system is specified from the external point of view.
- System Design – This activity involves setting up the system architecture and an integration plan for the architecture.
- SW/HW Requirements Analysis – This activity involves updating the technical requirements and operational information with regard to the software and hardware requirements. This is done with help of the user requirements, system architecture and previously derived technical requirements.
- Preliminary SW Design – This activity involves designing the software architecture. This includes completing the interface descriptions and updating the software integration plan.
- Detailed SW Design – The software architecture and interface description are further detailed. The specification and details for each software module, component and database are made.
- SW Implementation – The software modules, components and databases are realized. The code is generated and compiled into executable form.
- SW Integration – The integration of the modules, components and databases is realized. This is possibly done in several steps.
- System Integration – The integration of the system is realized. Both software and hardware components are integrated according to the system architecture.
- Transition to Utilization – This activity compromises the tasks needed for putting the system in operation in the intended environment.

Quality Assurance

The QA submodel regulates the activities and products of the other submodels by assessing it before it is accepted. It compromises these five main activities:

- QA Initialization – This activity involves specifying the organizational and procedural scope of the QA activities. All specifications for the products and processes are documented in the QA plan.
- Assessment Preparation – This activity involves producing the assessment plan. It contains the definition of assessment methods, criteria, environment and test cases.
- Process Assessment of Activities – This activity involves doing assessment on the processes to determine if they are sufficient.
- Product Assessment – This activity involves assessing products. A product is either accepted and enters the accepted state or it is rejected and goes back to the being processed state.
- QA Reporting – This activity involves evaluating the assessment reports. This is done in QA reporting and is based on the number of problems, the severity of the problems, classification of problems and the cause of the problem.

Configuration Management

The CM submodel guarantees that a product can be uniquely identified at any time. The identification serves to control modifications and to guarantee integrity. It compromises these four main activities:

- CM Planning – This activity involves defining the organizational framework and to document it in the CM plan. Needed resources and tools are also documented in the CM plan.
- Product and Configuration Management – This activity involves managing the product library by cataloguing the products. A configuration of the whole system is also managed.

- Change Management – This activity involves managing a change through three steps. First a change is requested and registered by the CM. In the next step the change is evaluated and either accepted or rejected. In the third step, if accepted, the change is implemented and all affected by the change are informed.
- CM Services – CM Services are those activities that serve the project in some way. It includes cataloging the software products, coordinating the interfaces, backup, release management and recording the project history.

Discussion of The V-Model

Advantages

The V-Model advantages:

- 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.
- The V-Model supports tailoring. At each project start, The V-Model is tailored into a specific project V-Model. The tailoring makes it possible because The V-Model is organization and project independent.
- The V-Model provides concrete assistance on how to implement an activity. In each activity schema there are instructions, recommendation and detailed explanation of the activity. Practical examples are given to explain the activity.

Method advantages:

- The definition of a limited number of methods helps the project management to easier make a selection. You have a limited number of methods, you do not have to drown in searching for suitable methods.
- The overview and definition of methods simplify the selection and application of methods. With an overview of methods you can fast get an overview of suitable and not suitable methods.
- The V-Model supports a wide range of development methodologies. There are methods that both support structured and object-oriented system development.

Tool advantages:

- The selection of tools is facilitated by the definition of requirements on service units. The selection of tools is made on service units. This way no tool is favored. The tools are selected objectively according to which requirements it fulfills.
- Known and well-used tools guarantee a calculable productivity. The tools listed in The V-Model are proven to be useful in previous projects. This guarantees that they will be useful in future projects too.
- The productivity is increased by concentrating on the most important characteristics. By choosing tools through service units you concentrate on the tools characteristics, not it's reputation.

Disadvantages

The V-Model disadvantages:

- The V-Model addresses software development within a project rather than a whole organization. It is a lifecycle process model. It is only used once during a project. It does not address the whole organization.

- The V-Model is not complete as it argues that it is. It argues that the submodels cover all activity areas. They pretty much do that, but they do it on a too abstract level. It is hard to find out if peer reviews and inspections are done in The V-model. You may think that it is done in the self-assessment activity before a product is passed on to the QA for acceptance. This is not stated explicitly in The V-Model.
- There is no organizational institutionalization of the processes in The V-Model. The processes are temporarily institutionalized during the project, but when the project is finished they are abolished.

Method disadvantages:
- There are a limited number of methods that are defined. By limiting the number of methods you do not get an overview of all methods that could be used. You do not get the chance of consider other tools that might be better.
- The Methods Allocation is not complete as it argues it is. There are for example five activities in the SD submodel that have no methods allocated. There are no methods allocated for the CM submodel.

Tool disadvantage:
- The Tools are not complete as it argues it is. The SDE in The V-Model includes hardware. There are no hardware tools in The V-Model. A tool is defined as "A tool is a software product supporting the development or maintenance/modification of IT systems".

Reflections

The V-Model

I don't think it is possible to compare The V-Model with the CMM [CMM1.1] as [Zah98] does. The models do not have the same foundation. The V-Model is a lifecycle project process model while the CMM is a capability maturity model for an organization. The idea with the CMM is to improve an organization's processes. The goal of the V-Model is to successfully reach the goal of a project. The V-Model is a one-use process model during a project while the CMM is a continuous use process model throughout the whole organization.

The V-Model and the CMM are not comparable in the large scope, but I do think that they are comparable in the small scope. The different key process areas in CMM can be compared with the activities in The V-Model.
The V-Model is very simplified compare to the CMM. The V-model is too abstract in it's activity descriptions. You do not really know what is included in one activity or not. It is hard to see what The V-Model sees as important. In the CMM the activities are divided into several smaller key process areas. This makes it easy to see what is important and what is not. Another big difference is that The V-Model often just mentions activities. It does not go as deep and detailed as the CMM.
Why is for example configuration management more important than requirements management? There is a whole submodel for configuration management while requirements management is only done in two main activities in the SD submodel. By devoting a whole key process area for requirements management the CMM provides a strong focus on controlling the requirements. I believe that requirements management is more important than configuration management. Many projects fail because they fail to satisfy the customer's requirements.
Managing subcontractors is another important activity area. The V-Model has an activity in the PM submodel that handles both the contractor (customer) and subcontractor/s. These two areas have been very simplified into one activity. The CMM on the other hand has a key process area for both contractor (requirements management key process area) and subcontractor management.

It is interesting to note that The V-Model is not just for developing software. The V-Model is designed for developing systems that includes both software and hardware. This is very important because the software will run on hardware. Handling requirements for the hardware is a natural way of working in The V-Model. If there is not a suitable hardware, then it is developed.

For each new started project The V-Models is tailored into a project specific V-Model. When a project is finished the final project report is put in the organizations experience database. The question is how useful this experience database is for other projects. Each project has a specially tailored V-Model. The experiences form previous projects are somewhat unique because of this. I therefore don't think the data in the experience database are directly comparable

As I mentioned earlier The V-Model is very simplified and too abstract. This makes it very easy to understand. There are four submodels and three levels that can be modeled by a cube. There are main activities and subactivities. An activity has an input product and an output product. Each product has some states. With these simple building bricks you can build up a complex model. If you connect the dots between each input and output products in an activity, you get very complex model that is hard to get an overview of.

I only found two things that an organization benefits from when using The V-Model. It is the recording of project history in the CM submodel and the general data dictionary of the organization.
The project history is the experience database mentioned earlier. It contains information on the project development, difficulties and problems. It also contains information on how these difficulties and problems were handled. The general data dictionary contains terms and definitions that are used across the whole organization. It is specific for the business domain. Both the project and organization benefits from it. It is used and updated during the project.

I only found one reason for why The V-Model is named The V-Model. I think it has to do with the activity and verification flow of the SD submodel. It can be modeled as a V in Figure 5:

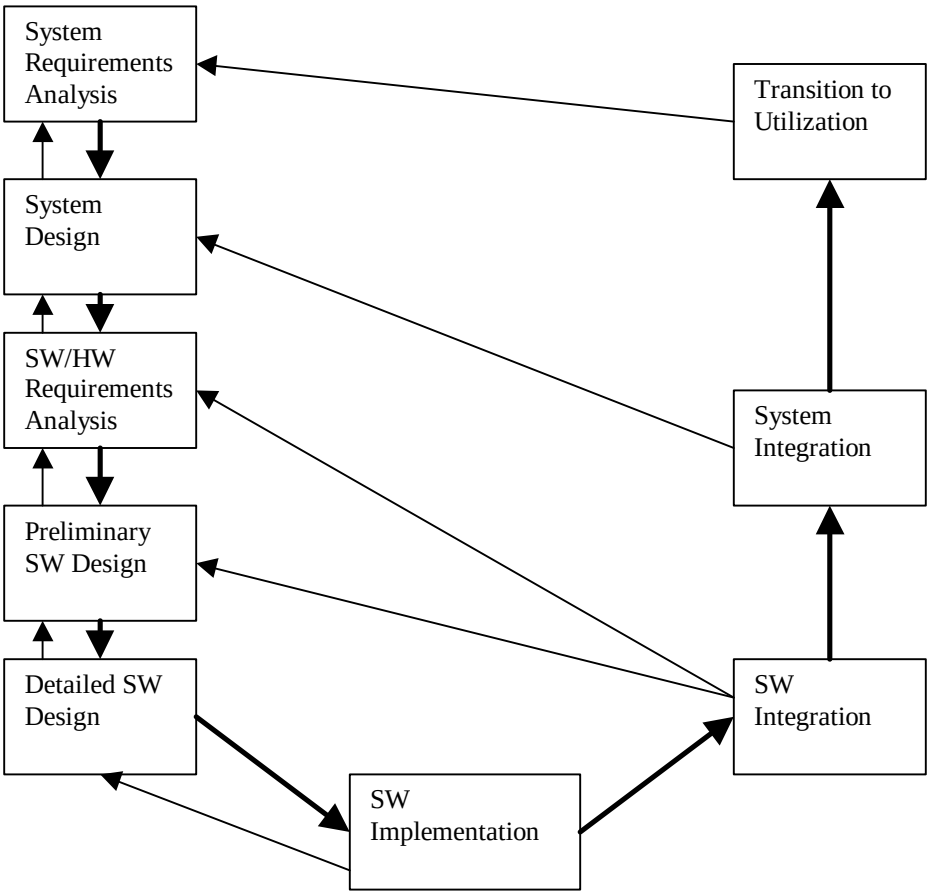

Figure 5 - The activity and verification flow in the SD submodel

The bold arrows are the activity flow. The other arrows are the verification of the products. For example, the implementation is verified against the detailed design.

Methods and Tools

I believe that the best thing with methods and tools is that they can be dynamically added and removed in The V-Model. The V-Model is very flexible in this sense. At each change control board meeting the methods and tools can be reviewed. New emerging methods and tools can easily be added to the model. The same goes with old and obsolete methods and tools. They can be removed easily.

Lifecycle Process Model

It is interesting to note how products are handled in the Lifecycle Process Model. All products can be seen as living products. They are refined, updated and detailed thought out the activities. Even when they are in the accepted state they can be considered to be living products. They can be modified and worked further on only if the version number is updated.

There are many advantages listed for the Lifecycle Process Model, but there are no disadvantages listed. The V-Model is not self critic and does not point out it's weaknesses and limits. This is a big disadvantage. The advantages are just listed and not explained further how and why.

Project Management Submodel

The PM submodel states that the staff members should participate in the planning of activities. They should participate in estimating and deciding the time schedule. I believe this is very good. Participatory planning commits the project members to the work, since they have participated in the planning rather than just given a plan without been asked about it.

The PM submodel has a job selection activity where project members are matched with roles. This is very important and problematic activity according to [Hum97]. I believe so too. It is hard to match roles/work orders with project members. You want to utilize every project member's skills to its full potential. It is important to assign tasks to project members so that they can enjoy the work. This is very important for if you are going to get the motivation from project members. If there is no motivation then the work will not be performed as good as with motivation present.

System Development Submodel

It is interesting to note that The V-Model has such a big emphasis on using off the shelf components. In the second system development activity, which is system design, it says that off the shelf components must be given special attention. Feasibility studies should be performed on the basis of the requirements and the off the shelf components should be evaluated.

Quality Assurance Submodel

The whole QA submodel is about assessing products and processes. There are no long-term plans on continuous process improvements. I am not surprised. The V-Model is a project process model, not an organization process model. There is no point in launching a substantial process improvement program.

The QA submodel assesses the products and processes. It compares the product and processes with the plans, standards and definitions. It either accepts or rejects the product or process. The submodel is on a too high abstract level so it is hard to find out if it suggests improvements if a product is rejected. I assume it is documented in the assessment report.

There is no software testing activity in the SD submodel. The testing in the SD submodel is done with self-assessment of the software product before it is submitted to the QA for acceptance. The QA tests the software product a second time before it is accepted.

The product acceptance by the QA is done in the same way as in the Cleanroom Process Model [Lin94]. In both cases the product is submitted to the QA. The QA verifies/assesses the product and either accepts or rejects it. If it is rejected it is returned to the authors. The difference is that the verifications are done more often, after each increment, in the Cleanroom Process Model.

When an assessment is finished the results are documented in an assessment report. The number of problems, severity of the problems, classification of problems and the cause of the problems is documented. It is interesting to compare this with inspections [GilGra93]. These things are also documented during an inspection. Finding out the cause of the problem is called root cause analysis in [GilGra93].

Conclusion

The V-Model is very easy to understand and use. It is built up with some basic building bricks. With these building bricks you can build up a very complex model that is hard to grasp. It is easy to use because it is practically oriented. It answers the three basic questions:

- What has to be done?
- How is it done?
- What is used to do it?

Two big advantages of The V-Model are:

- It is very flexible. It supports project tailoring and dynamically addition and removal of methods and tools. At each project start The V-Model is tailored into a specific project V-Model so that it suits the project. It is easy to add new emerging methods and tools and remove old and obsolete methods and tools.
- The V-Model is publicly developed and maintained. The users of The V-Model participate each year in the change control board. It processes all received change requests.

Three big disadvantages of The V-Model are:

- The V-Model is project oriented. It is a lifecycle project process model and is only used once during a project. It does not address the whole organization. I think that an organization, that uses The V-Model, would need a complementary process model on the organizational level.
- The V-Model is not self-critic. It does not point out the weaknesses and limitations of The V-Model. Instead it lists many advantages without further explaining them.
- Some activities in The V-Model are described in a too abstract level. You do not know what is included and what is excluded. It is too flexible in this sense. It leaves too much for the user of the model to decide.

References

[CMM1.1]
The Capability Maturity Model
Software Engineering Institute, Carnegie Mellon University
ISBN 0-201-54664-7

[GilGra93]
Software Inspection
Tom Gilb, Dorothy Graham
ISBN 0-201-63181-4

[Hum97]
Managing Technical People
Watts S. Humphrey
ISBN 0-201-54597-7

[IABG97]
The V-Model
Development Standard for IT-Systems of the Federal Republic of Germany, IABG
http://www.v-modell.iabg.de

[Lin94]
Cleanroom Process Model
Richard C. Linger
IEEE Software March 1994

[Zah98]
Software Process Improvement
Sami Zahran
ISBN 0-201-17782-X