V-Model Views

Dieter Scheithauer
INCOSE ESEP
CASSIDIAN
Rechliner Str.
85077 Manching - Germany
dieter.scheithauer@cassidian.com

Kevin Forsberg
INCOSE Pioneer and ESEP
OGR Systems
1390 Grizzly Road
Portola, California – USA
kforsberg@ogrsystems.com

Abstract. The V-model is the most popular iconic illustration within systems engineering. Originally derived from the waterfall model to give a notion of a system's architecture, the V-model today is used widely in systems engineering training, and for communicating systems engineering ideas and concepts in the systems engineering community and beyond.
The contemporary understanding of the V-model is still based on definitions established more than two decades ago. Since then, systems engineering has advanced in directions beyond the systems engineering scenarios of the past. Systems-of-systems, systems engineering value stream thinking, and systems engineering outreach into commercial sectors challenge the current common interpretation of the V-model. It is the intention of this paper to establish a set of consistent V-model views integrating central systems engineering concepts and maintaining the central role of the V-model for communication within the systems engineering community and beyond.

Introduction

Icons are a powerful means of communication. They ease orientation in public facilities like airports, hotels, or train stations. As traffic signs, they help us to drive safely even when we take unknown directions. Their shape itself is part of the message. When naming the shapes of Figure 1 with the otherwise meaningless terms "omulvo" and "takete" most people will make the same choice without hesitation (Köhler 1947).

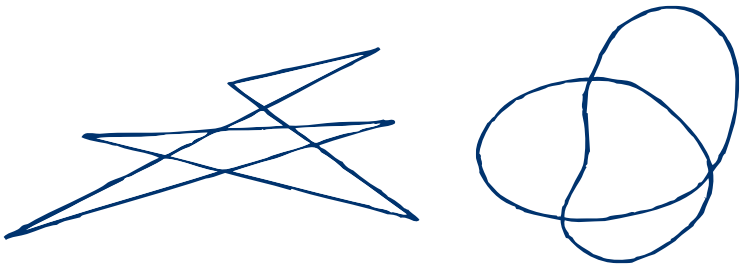

Figure 1. Naming Shapes – "Omulvo" and "Takete" (Köhler 1947).

The most concise icon in systems engineering is the V. Not necessarily the more artistic representation shown in the publications of Hal Mooz and Kevin Forsberg (Mooz and Forsberg 2001; Forsberg, Mooz and Cotterman 2005), but just two lines drawn on a piece of paper or on a board as a sketch for expressing systems engineering concepts. But unlike the initial example where just a phoneme is correlated with a graphical shape, the V is enriched with all the

systems engineering knowledge and experience of the observer. Due to its success, the V is used to explain many ideas and concepts in systems engineering. On the downside, inconsistencies are diluting more and more the communicative power of the V in systems engineering, since many people are creating their own V icons to address their specific situation.

It is the objective of this paper to propose a set of consistent V-model views in order to maintain the V as the dominant icon to express systems engineering concepts. This includes re-visiting the original definition of the V-model (Forsberg and Mooz 1991; Forsberg, Mooz and Cotterman 2005) with respect to the advances in systems engineering achieved within the last decade. Most notably, the following new concepts introduced to systems engineering have an impact on a further evolution of the V-model: (1) the notion of systems-of-systems (Haskins 2010); (2) value stream thinking in systems engineering (Ohno 1987; Womack and Jones 1996 and 2003; McManus 2005; Scheithauer 2012); (3) the outreach of systems engineering into commercial sectors.

Four consistent V-Model views are introduced by this paper:

- The Basic V expressing V-model fundamentals (see Figure 2)
- The Development V (see Figure 3)
- The Assurance V (see Figure 4)
- The Dynamic V (see Figure 5)

The first three views depict the overall systems engineering value stream, while the fourth view indicates iterations that may be performed over the whole or over parts of the V-Model.

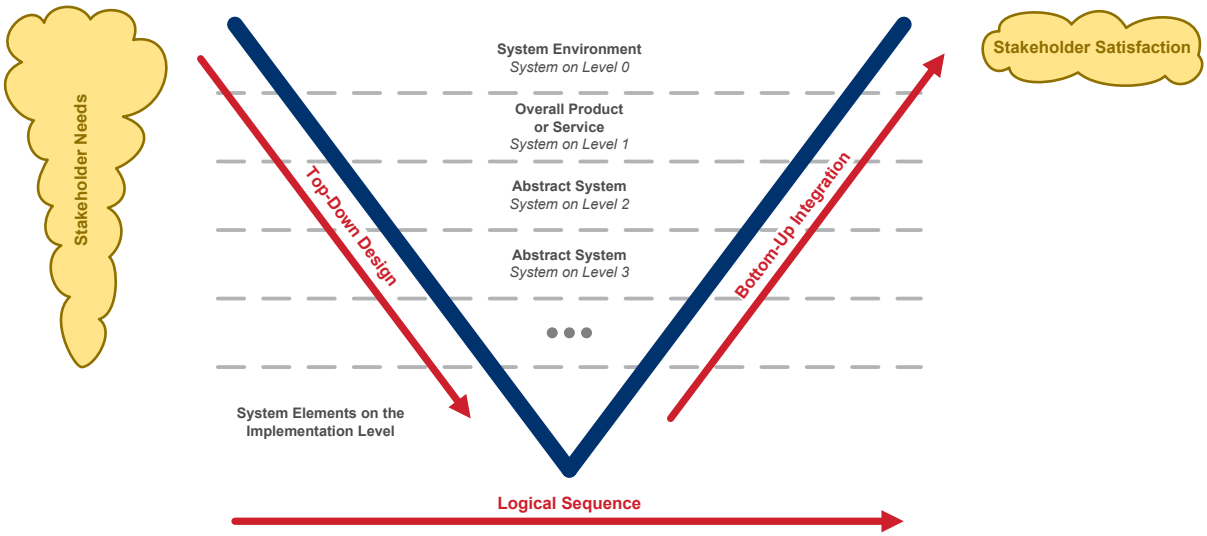

Figure 2. V-Model Fundamentals: The Basic V.

V-Model Legacy

According to Hal Mooz and Kevin Forsberg, the V-model has been invented twice in the late 1980's by NASA and in parallel by Kevin Forsberg and Hal Mooz (Forsberg and Mooz 1991). The main characteristics of the V-model are:

- Horizontal layers in the V-Model provide a notion of the hierarchical levels of a system architecture with the overall product or service under consideration at the top and with the system elements on the implementation level at the bottom.

- The direction from left to right expresses the time line. Usually, sequential milestone events structure the time continuum. Thus, the V-model expresses a sequential development approach. Other development philosophies are shown by breaking up the V in parts to provide examples for incremental and evolutionary development philosophies.

- A vertical line at the "time now" (not shown in the figures) will illustrate the critical iterative character of the V model. These iterations are part of the systems engineering process models that are not scaled temporally, but rather by events. While important to understand that the on-core and off-core activities are part of the heart of the V model, they are confusing when incorporated into a simple iconic model. For more detail see the earlier papers (Breidenthal and Forsberg 2008; Forsberg, Mooz, and Cotterman 2005).

- Going down on the left leg stands for top-down design. Progress is shown as a continuum of generated baselines.

- Proceeding up on the right leg means bottom-up system integration.

V-Model Weaknesses. From a today's point of advanced systems engineering knowledge, of accumulated experience and of applying systems engineering beyond its original context, the main characteristics of the V-Model may be challenged for the following weaknesses:

- Inventing new products and services deserves engineering activities regarding the operational context first. In the details of the V depiction, as well as the various architectural frame works (such as DoDAF, MoDAF, NAF) the system environment is described as a separate architectural layer above the overall product or service finally delivered to customers. Issues such as user needs and system affordability drive the evolution of the system solution. However the V icon does not show this.

- The meaning of the lowest architectural level at the peak of the V-Model is not well defined. General criteria for stopping the architectural system decomposition are as vague as the definition of the term implementation. Considering an industrial culture based on work-share and specialization in terms of commercial-off-the-shelf and non-development items only falls somewhat short about the complexity of industries.

- Interpreting the left-to-right direction as time line supports pure sequential systems engineering concepts, but introduces some twists for expressing re-work and iterative approaches. First, design iterations over the whole or parts of the system architecture can rarely be achieved at a single point in time. Re-working the design and creating updated configuration baselines for affected systems and system elements interfere considerably with development progress. As noted above, the activities off the core of the V are a vital part of the real work and value imbedded in the V model (Breidenthal and Forsberg 2008). The vertical iterations support iterative and agile development concepts, but the resulting model is no longer a simple iconic Vee, and so it is not elaborated upon here. Second, the need for re-work of the design due to system integration results is hidden despite the tremendous challenges for managing corrective design and implementation activities in

order to maintain consistent configuration baselines of all affected systems and system elements. Third, the linear mapping of the time line to life cycle phases disregards the repetition of some engineering activities in several life cycle phases. And at forth, mapping incremental or evolutionary development approaches to the V-model results in the distortion of the iconic V without providing a generalized view representing all real-world variations of pre-planned and event-driven iterations.

System Architecture Considerations

The system architecture of an overall system, e.g. products and services delivered to customers, is defined for engineering purposes. The system architecture is developed to partition the engineering effort according to a wide range of considerations including mastering system complexity, involved application domains, and organizational capabilities and boundaries. In Figure 2, the basic V-model is shown with the architectural system levels shown in horizontal layers. The different architectural layers are described below.

Note that the terms system and system element are used below in a recursive pattern. A system is decomposed into system elements on the next level that again can also be designated as systems. In contradiction, ISO/IEC 15288-2008 favors a tree metaphor where the system elements are the ultimate leaves of the tree (ISO and IEC 2008).

The System Environment. The upmost entry point of a system architecture is the system environment in which the overall system is operated. The system environment is the part of the surrounding world considered when engineering the overall system. In the end, customer and user satisfaction depends on operating the overall system in its surrounding world. A number of engineering activities are performed on the system environment. It has to be decided what is taken as part of the system environment, and what is neglected leading to the definition of the outer boundaries of the system environment. The system environment has to be understood in terms of functions and architecture. Innovative and competitive ideas for capability sustainment and enhancement beneficial to current and future customers have to be identified to generate promising future business cases. The operational requirements allocated to the overall system have to be developed. And, mission effectiveness and the system efficiency including affordability of the overall system's operation in the surrounding world have to be validated.

The Overall System. The overall system constitutes the product or service to be delivered finally to the customers. It is designed to satisfy the operational requirements. The design is expressed in terms of consistent and complementary system requirement, functional and architectural views. Continuously performed validation activities provide the confidence that stakeholder satisfaction is most likely. And, the verification of the overall integrated system provides the evidence that system requirements are met and that customer contracts are fulfilled.

Abstract Systems. System elements resulting from the overall system's decomposition are abstractions of portions of the overall system. Therefore, they are designated as abstract systems. Abstract system elements may be scoped according to functional considerations, or following other segregation principles. The engineering of each abstract system element demands a specific cooperation of multiple engineering disciplines with higher levels of

specialization on lower architectural system levels. Thus, the decomposition of the system architecture leads to team structures and responsibility assignments within the engineering organization. Engineering teams responsible for abstract systems actively contribute to satisfy the stakeholder needs on the overall system. They are involved in the elicitation of stakeholder requirements as they may understand some stakeholder needs inherited from higher level systems in more detail and occasionally their design decisions may lead to additional stakeholder requirements. Sometimes abstract system elements may have a corresponding representation within the product structure as a particular physical entity.

For each abstract system element, the allocated requirements are interpreted in course of the design. An abstract system element is defined by consistent and complementary requirement, functional and architectural views. The methods and practices applied for the development of a particular system element are selected according to the specific application domain. The hand-over to the system elements on the next lower architectural level is accomplished again by allocation of requirements that are feasible to be engineered and implemented on lower architectural levels. All abstract system elements are engineered having the stakeholder satisfaction with the overall system in its operational environment in mind, thus contributing to the quality management goal of customer satisfaction (ISO 2008).

The number of abstract system levels in each branch of the system architecture may vary. Each abstract system element has to add value to the overall system design in an effective and efficient manner. In some cases, the overall system may be directly broken down into system elements on the implementation level.

The Implementation Level. In contrast to the engineering of the abstract system elements, the engineering of the system elements on the implementation level sets the focus on the system requirements allocated to them without challenging the definition of the overall product or service and its abstract system elements. They are not concerned anymore with the stakeholder needs on the overall system. In many cases, system elements on the implementation level are procured as standard or customized parts from other industrial organizations. For example, a servo-hydraulic actuator is a system element on the implementation level for an aircraft manufacturer. For the actuator supplier, the actuator is the overall product. The same is true for the company providing a shut-off valve to the actuator manufacturer. Thus, the V-Model becomes re-entrant recursively. In other cases, system elements engineered in-house may be deemed to be on the implementation level like for example some electronic circuitry or basic software code. But be aware about the reliance on supplies from other industries for procuring for example electronic components or adequate compilers. In a sophisticated industrial culture it may hard or nearly impossible to identify the true peak of the V.

The V-Model as Value Stream

The idea of anticipating the V-model as a value stream is obvious. However, interpreting the left-to-right direction as timeline undermines this interpretation. The lean principle for flow (Oppenheim 2011) includes the absence of backflow. Showing vertical iterations on the left leg of the V to be performed at a single point in time, and hiding possible rework and redesign on the right leg as background details (Forsberg, Mooz, and Cotterman 2005) demands some twists in the reasoning to claim compatibility with all lean principles. This issue does not exist when the left to right direction is interpreted as logical sequence for a single iteration on the overall systems engineering value stream that may be performed on the whole or parts of the V

as later explained in the context of the Dynamic V (see Figure 5). This overall value stream then may put the major emphasis on the hand-over of information between systems and their system elements as shown in the Development V (see Figure 3)

Stakeholder Needs and Stakeholder Satisfaction. Stakeholder needs and stakeholder satisfaction themselves are shown off-side the V as they are not direct products of the systems engineering activities. The entry point for systems engineering is the elicitation of stakeholder requirements. Traditionally, the terms stakeholder needs and stakeholder requirements are used interchangeably in systems engineering (Haskins 2010). This position is currently changing. ISO 29148 (ISO 2011) is aware of the difference between stakeholder needs and stakeholder requirements, but still does not provide a proper definition of the term stakeholder needs.

It is the goal in systems engineering to satisfy all stakeholders concerned with the overall system over its whole system life cycle. However, the whole system is invented for those stakeholders that are concerned with the overall system during the in-service phase. For this reason, stakeholder satisfaction is indicated on the right of the V on the architectural level of the system environment in order to concentrate on essentials.

Logical Sequence. As stated above, interpreting the left-to-right direction as timeline prevents the V from being interpreted as value stream according to lean principles (Womack and Jones 1996; Oppenheim 2011). This deficiency is cured when the left-to-right direction is interpreted as logical sequence. Any re-work may then be expressed by moving back to the left. Thus, it becomes visible at which stage of the engineering process the re-work has to start and which system elements may be impacted by the changes.

Following this interpretation, the V expresses the overall systems engineering value stream. Of course, the engineering of each system element follows an engineering value stream on its own when widely managed independently. It is better to draw those value streams not in the shape of a V as they do not express an architectural decomposition. In order to avoid overlaps between the overall systems engineering value stream represented by the V and the engineering value streams for developing particular system elements, the following V-model views concentrate on global characteristics of the overall value stream.

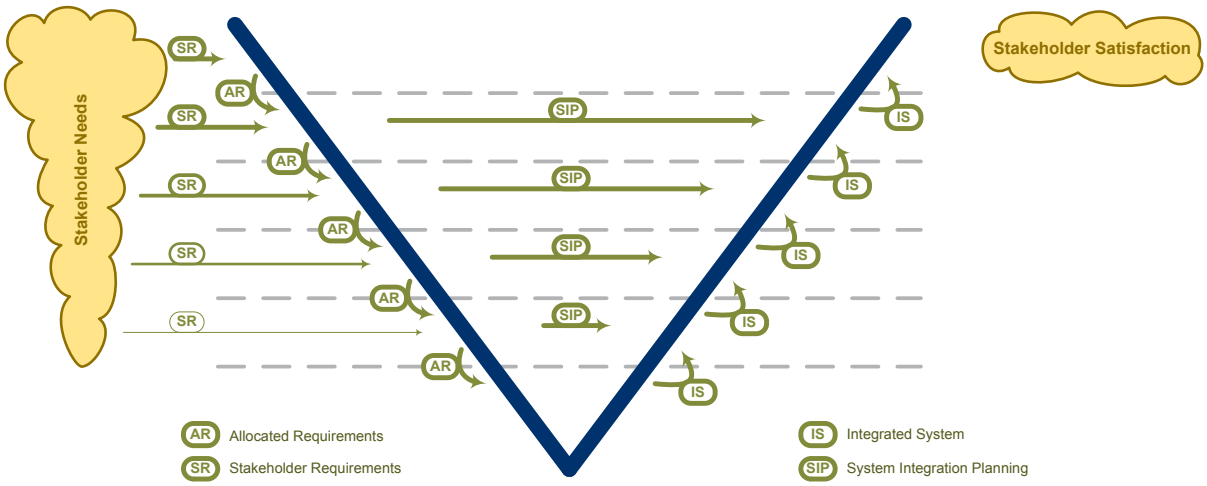

Figure 3. The Development V.

Development Activities in the V-Model: The Development V

From Stakeholder Needs to Stakeholder Requirements. Stakeholders have needs, but they are rarely conscious about all their needs. Stakeholders have expectations taken for granted. Consequently, they will miss to talk about some important basic needs. Instead they will emphasize felt deficiencies of current solutions and their wishes delighting them, if they would become true. A systems engineer without knowledge of the stakeholder's domain of expertise may be limited in the capability to transform all relevant stakeholder needs into stakeholder requirements. In general, it may be better to accept that stakeholder needs may not always be identified than postulating the identity of stakeholder needs and stakeholder requirements.

At the outset, stakeholder requirements on the system environment level have to provide a comprehensive view on the capabilities from which the operational requirements for the overall product or service are deduced. However, further stakeholder requirements may be derived for the overall product and service as well as for all the abstract system elements. Design decisions may lead to the identification of new stakeholders, additional needs from known stakeholders, and already known needs may be better understood by engineers from particular domains in charge for engineering specific logical system elements. Of course, it may be assumed that the need and the effort for stakeholder requirement elicitation will fade out with increasing architectural decomposition of the overall system into abstract system elements.

The Requirement Cascade. On the left leg of the V, the information hand-over between a system and its system elements on the next lower architectural level is facilitated by the allocation of requirements defined by the engineering team in charge of developing these requirements for the next lower architectural level. The engineering teams in charge of developing the system elements will interpret the allocated requirements. They will then generate consistent and complementary system requirement, functional and architectural views. On one hand, they have to ensure that they satisfy the expectations communicated by the requirements allocated to them. On the other hand, they also have to take care that their design solution is feasible and implementable on all lower architectural levels. Finally, requirements will be allocated to the system elements on the next lower architectural level. Note that the requirements allocated from the system environment to the overall system are commonly designated as operational requirements.

The hand-over of allocated requirements needs to be tightly controlled. First, the design of the next upper level system needs to have achieved a known level of maturity. This does not necessarily mean that the design has to be complete.

Second, a consistent configuration baseline of the next upper level system has to exist. Here, a consistent configuration baseline is characterized by a set of referenced work products describing the system requirement, functional and architectural views, and the allocated requirements assigned to all the system elements at the next lower architectural level. All content needs to be traceable and free of conflicts beyond the stated omissions and issues as knowledgeable at the particular point in the systems engineering process.

And third, appropriate agreements of the involved engineering teams are made, for example in the course of management reviews. At least three mandatory parties are involved in the agreement process. The originators of the allocated requirements taken as input for the system design have to assess, if their expectations are fulfilled by the system design. The engineering

design team of the particular system has to justify and defend their design decisions. The engineering teams responsible for the system elements on the next lower architectural level have to evaluate, if they have understood the requirements allocated to them and if the task is feasible for them.

System Integration Planning. When designing the overall product or service, or any abstract system element, the scope for system integration becomes clear as well. An overall system integration concept considering the whole system architecture of the overall system or service is generated. The demands for system integration of particular system elements are elaborated. This includes the generation of assurance objectives as an engineering activity subsumed to be represented by the left leg of the V. In order to provide a sound basis for preparing system integration, the hand-over between system design and system integration needs to be controlled by consistent configuration baselines again.

System integration planning may not only be performed on a per system element basis. In many cases, it is worthwhile to define an overall system integration concept defining rules on which level of the system architecture evidence for the fulfillment of certain categories of system requirements is generated most efficiently. Although important, this consideration is not addressed explicitly in the development V-model view.

Considering the lead time needed to provide all the system integration environments system integration should start early. All the preparatory work on system integration is represented by the right leg of the V as it does not interfere with the requirement cascade.

The System Integration Cascade. The actual system integration is dependent on the recursive bottom-up hand-over of integrated system elements up to the overall product or service. This hand-over is shown on the right leg of the V. In cases in which the overall product or service is one of a kind, the integration of the overall product or service in the system environment may be performed as a systems engineering activity. In all other cases, it is unlikely that it is a systems engineering responsibility to perform the transition into the in-service phase.

As a minimum, the following controls are necessary for evolving the system integration cascade: First, the integration activities on the respective system element need to have achieved a certain level of known maturity before they are actually integrated into the next upper level system. Second, the integrated system element has to be identified by a consistent configuration baseline with all omissions and deficiencies clearly stated in order to avoid damages and nuisance work when the integration of the next upper level system is commenced. And third, appropriate agreements of the involved engineering teams have been made, for example in the course of management reviews.

Assurance Activities in the V-Model: The Assurance V

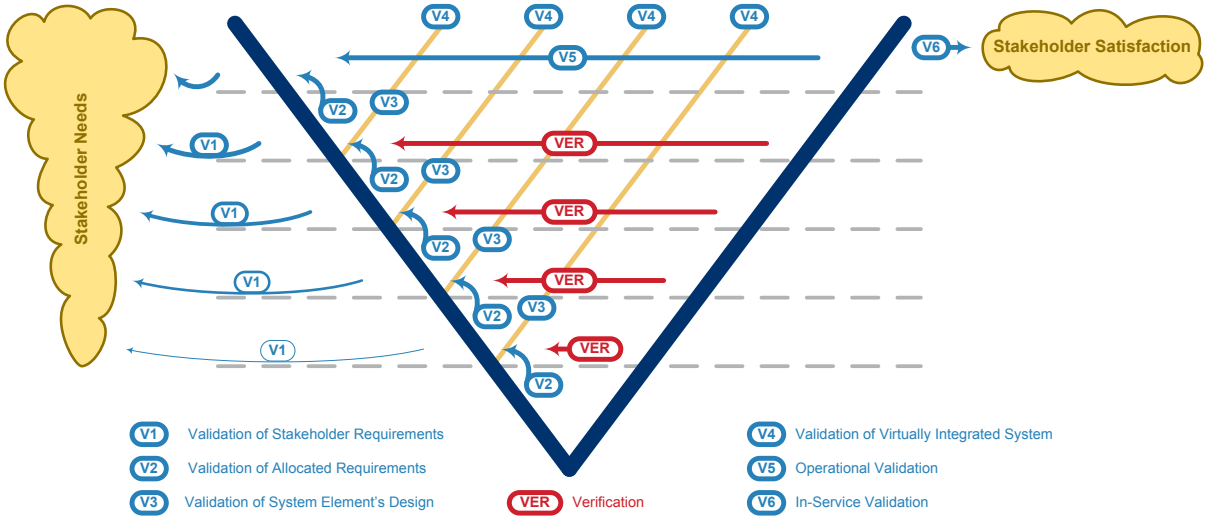

Figure 4. The Assurance V.

Assurance as Integral and Continuous Process. The movement for quality in the last decades is motivated by the fact that quality cannot be tested in. Assurance activities are continuously interwoven with development activities. Taiichi Ohno has coined the term autonomation to express not to allow forwarding deficient results from one process step further downstream to other process steps by using tools that have quality checks built in (Ohno 1988). Equivalently, any systems engineering work product should have passed quality checks before it is released to serve as a point of reference for downstream engineering activities. Thus, assurance activities occur everywhere in the V. Some distinct assurance activities are nevertheless worthwhile to be shown in the V.

Assurance activities have either the focus on the substantiation that system requirements represent stakeholder needs adequately, or on the demonstration that the system requirements are implemented correctly and completely, or on the proof that processes are defined and performed according to the demanded process quality. Here, the first two topics are addressed: validation and verification. Process assurance cannot be expressed in the V meaningfully.

In some product and service domains, further more specialized assurance processes may be defined like safety assessment or security assessment. Those processes are not specifically considered here. They apply some specific methods to satisfy validation and verification objectives in a certain engineering domain. Consequently, they are implicitly included in the considerations on validation and verification.

Dedicated Validation Activities. Because stakeholder needs may not be identified completely, validation will never lead to an exhaustive proof that stakeholder needs are fully identified and understood completely and correctly. On the other hand, stakeholder satisfaction results from the fulfillment of stakeholder needs. For these reasons, it is important to integrate distinct validation steps at multiple points in the overall systems engineering process. Furthermore, a defensive approach is taken not only to show that stakeholder needs are satisfied, but also to prove the absence of unintended functions that may have an adverse impact on humans and the environment.

As shown in Figure 4, Validation Step V1 is concerned with stakeholder requirements. The requirements text, the corresponding rationale, and existing examples are examined to check, if the stakeholder requirement is convincing to express a real stakeholder need well.

A similar communication situation occurs when allocated requirements are handed over from one engineering team to another. The engineering team receiving the allocated requirements is in a similar situation as when eliciting stakeholder requirements. This Validation Step V2 is accomplished by applying the same techniques as described above.

In Validation Step V3 the engineering team evaluates, if their design solution is appropriate for satisfying the stakeholder needs. All three essential defining views – system requirements, functions, and architectural decomposition – are examined to assess the level of achievable stakeholder satisfaction. This is a prerequisite before handing over allocated requirements to the system elements on the next architectural level.

The next validation opportunities arise when design results from lower level system elements become available and are progressively integrated into a virtual product or service in Validation Step V4.

Validation Steps V1 toV4 may all be performed before implementation is commenced. They are very valuable to mitigate the main risk of not achieving stakeholder satisfaction. Model Based Systems Engineering (MBSE) has the potential to boost the quality of validation results at this stage significantly and to increase thereby overall systems engineering efficiency.

However, not all assumptions on system requirements may be resolved yet. In this case, additional validation oriented assurance objectives may be imposed for resolving assumptions during system integration. As assurance objectives are mainly established for verification, and as validation and verification purposes are overlapping in most cases, no specific validation step is shown in Figure 4 for this in order not to capture a single activity twice.

The final validation step (Validation Step V5) during the value stream execution expressed by the V-model is performed when the overall product or service is actually operated in the real system environment. Frequently, this is called customer validation. Validation Step V5 is helpful to overcome initial in-service transition issues, to identify product or service improvements, and to record stakeholder requirements to be considered in the development of successor products and services.

Further validation activities may be performed in later system life cycle phases, in Figure 4 indicated as Validation Step V6. They provide hints for further product or service enhancements and extend the knowledge base on stakeholder needs for future products and services.

Note that validation with respect to the stakeholder needs on the overall product or service is not applicable to the implementation level in accordance with the definition of the implementation level given above.

Dedicated Verification Activities. Verification demands the availability of the integrated system element, or system, depending on the architectural level the verification is taking place. This is visualized in Figure 4 by the arrows pointing from right to left. Verification evidence is

generated during system integration by demonstrating compliance with the system requirements on the basis of the defined assurance objectives. This is the sole representation of verification in the V. However, to de-risk development continuous verification checks precede the final verification in the sense of the autonomation principle introduced above.

Verification is applicable to all levels of the system architecture except the system environment. Of course, verifying the system requirements on the overall system level demands a representation of the operational environment. This is depicted by the verification arrow on the overall system level. On the system environment there are no system requirements to be verified as the system environment goes simply beyond the delivery obligations. This does not mean that the verification of system requirements on the overall product or service would need no representation of the system environment, but this is all self-contained on the overall product or system level.

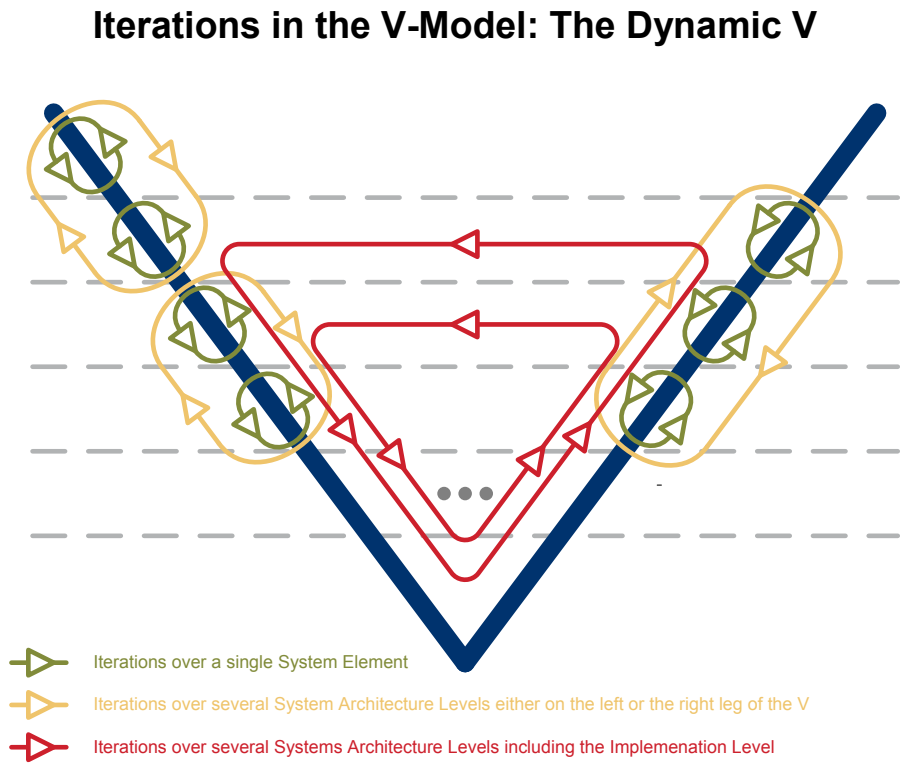

Figure 5. The Dynamic V.

The Necessity to Consider Iterations. Considering iterations in systems engineering is a valuable response in order to cope with the complexity of product and service development. Iterations may be pre-planned or event-driven. Mastering both kinds of iterations in an integrated manner provides agility to the execution of the systems engineering process (Scheithauer 2012).

Pre-planned iterations may occur due to the life cycle phase definitions of certain life cycle phase models, or due to incremental and evolutionary development philosophies. Also pre-planned repair cycles fall into this category when otherwise a waterfall model development philosophy is followed.

Event-driven iterations occur due to lessons learned during project execution. Lessons learned may be annoying, if they contain something already known within the organization. Or, they may reveal something new challenging the design and providing directions for further opportunities. In principle, pre-planned iterations are also a good means to cope with lessons learned during development. But similar to high volume mass production in batch mode this would mean to ignore the efficiency boost from lean thinking in manufacturing and agile approaches in software engineering that could be adopted to systems engineering. Event-driven iterations have the potential to enable faster learning, and by that to reduce cumulative uncertainty and risks (Scheithauer 2012).

When interpreting the V as the overall value stream in product and service development iterations may be generally indicated as feedback loops in the V as shown in Figure 5 schematically. A single iteration means to jump back to the left until the highest level system or system element is reached from which on the incorporation of the change needs to start. Thus, the V indicates the potential impact of each change when progressing to the right according to the logical sequence indicated by the V. Three distinct categories of iterations can be identified: (1) Iterations on single system elements, (2) iterations on the left or the right leg of the V, and (3) Iterations on multiple elements including the implementation level.

Iterations on Single System Elements. One essential prerequisite for effective and efficient execution of the overall systems engineering value stream represented by the V is the high quality of work performed on individual system elements in due time. A capability to provide high quality configuration baselines with a reasonable frequency lays the foundation for high reactivity within the overall systems engineering value stream. In the end, the system element with the weakest performance dominates the maximum dynamics achievable for the overall systems engineering value stream.

For the engineering of individual system elements, this means equivalently that advanced capabilities to cope with pre-planned and event-driven iterations are of paramount importance to recover any flaws internal to that system element rapidly without propagating any adverse impact to other system elements in the system architecture. In turn, it can be concluded that iterations of system elements are highly welcomed, if they eliminate self-contained deficiencies without adversely impacting the overall systems engineering value stream. Then, the classical project management over the whole system architecture facilitated by sound configuration baselines for handing over information between systems and system elements is not overloaded with nuisance issues.

For this reason, the feedback loops affecting just one system element either on the left or the right side of the V are drawn in green in Figure 4. Iterations on the left leg represent iterations on the system design, especially of system requirements, functions and the architectural decomposition. Iterations on the right leg may represent changes of any work products not impacting the system definition handed over to system elements on the next lower level on the left leg of the V.

Iterations on Left or Right Leg of the V. Iterations affecting several system elements demand principally a higher effort for performing the feedback loop than changes self-contained by one system element. They are the classical domain of project management with agreement processes facilitated by management reviews based on sound configuration baselines. Especially on the left leg of the V, systems engineering management is the prevailing discipline for coordinating all the engineering activities that are concurrently

performed on the overall product or service and all the abstract systems below. A capability to execute feedback loops over the overall product and service and all abstract system elements rapidly is beneficial for the generation of high quality proposals with no or only a minimum amount of TBDs left. Due to the management effort, Figure 5 shows this kind of feedback loops in amber.

Iterations on Multiple System Elements Including the Implementation Level. Whenever feedback loops cross the implementation level the effort will be high. Consequently, the color red is chosen in Figure 5. The only exceptions are systems for which the implementation effort is low like in some software applications. In all other cases, feedback loops of this kind should be avoided or minimized in number. Incremental development philosophies are usually applied to control the number of iterations passing the implementation level.

Conclusions

A set of four V-model views is proposed as a means for improved communication in the systems engineering community and beyond. The basic view defines a standardized system architecture decomposition emphasizing recursive decomposition principles and the V shape as representation of the over-arching systems engineering value stream compatible with lean principles. The development view concentrates on the hand-over of information within the system architecture. The assurance view shows the corresponding validation and verification activities. And, the dynamic view depicts the impact of pre-planned and event-driven iterations on the overall systems engineering value stream.

The standardized system architecture definition shows the system environment as a distinct layer above the overall product or service. Therefore, system-of-systems engineering is better expressible in the V explicitly.

The extended definition of the implementation level allows the V to be interpreted recursively in order to map supply chains typical for sophisticated industrial cultures based on work share, specialization, and standardization.

It is hoped by the authors that these V-model views become a popular part in the systems engineering assets.

References

Breidenthal, J. and K. Forsberg 2008. “Core and Off-Core Processes in Systems Engineering,” February 22. NPO-45745, NASA's Jet Propulsion Laboratory, Pasadena, California; Received JPL Outstanding Publication award.

Forsberg, K. and H. Mooz 1991. “The Relationship of System Engineering to the Project Cycle,” Proceedings of the National Council on Systems Engineering (NCOSE) Conference, Chattanooga, Tennessee, pp. 57-65, October. (Note: This is the first public presentation of the Vee as developed by Forsberg and Mooz in 1989.

Forsberg, K., H. Mooz, and H. Cotterman 2005. Visualizing Project Management: Models and Frameworks for Mastering Complex Systems. 3rd Edition. Hoboken, NJ (US): John Wiley and Sons.

Haskins, C., ed. 2010. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities.* Version 3.2. Revised by M. Krueger, D. Walden, and R. D. Hamelin. San Diego, CA (US): INCOSE.

ISO (International Organisation for Standardisation). 2008. EN ISO 9001-2008. *Quality Management Systems – Requirements*.

ISO and IEC (International Organisation for Standardisation and International Electrotechnical Commission). 2008. ISO/IEC 15288-2008. *Systems and Software Engineering – System Life Cycle Processes*.

ISO, IEC and IEEE (International Organisation for Standardisation, International Electrotechnical Commission, and Institute of Electrical and Electronics Engineers, Inc.). 2011. ISO/IEC/IEEE 29148-2011. *Systems and Software Engineering – Life Cycle Processes – Requirements Engineering*.

Köhler, W. 1947. *Gestalt Psychology: The Definitive Statement of the Gestalt Theory*. New York,NY (US): Liveright Publishing Corporation.

McManus, H. L. 2005. “Product Development Value Stream Mapping.” Release 1.0, Massachusetts Institute of Technology, Lean Advancement Initiative (Cambridge, MA, US).

Mooz, H., and K. Forsberg 2001. “A Visual Explanation of Development Methods and Strategies including the Waterfall, Spiral, Vee, Vee+, and V++ Models.” Paper presented at the 11th INCOSE International Symposium, Melbourne (AU): 1-5 July.

Ohno, T. 1988: *Toyota Production System: Beyond Large Scale Production*. Boca Raton, FL (US): CRC Press.

Oppenheim, B. W. 2011. *Lean Systems Engineering - With Lean Enablers for Systems Engineering*. Hoboken, NJ (US): John Wiley and Sons.

Scheithauer, D. 2012. "Managing Concurrency in Systems Engineering." Paper presented at the 22nd INCOSE International Symposium, Rome (IT): 9-12 July.

Womack, J. P., and D. T. Jones 1996 and 2003. *Lean Thinking: Banish Waste and Create Wealth in Your Corporation*. New York, NY (US): Free Press.

Biography

Dieter Scheithauer

Dr. Scheithauer, an INCOSE ESEP, studied electrical engineering with special emphasis on automatic control at the Universität der Bundeswehr München resulting in the degree of a Diplom-Ingenieur univ. in 1980 and a doctor’s degree (Dr.-Ing.) in 1987. His service as Technical Officer in the German Air Force ended in 1988. From 1988 to 1999 he was employed by Industrieanlagenbetriebsgesellschaft GmbH (IABG). He worked in a branch

mainly delivering technical expertise to the German Ministry of Defence and other government organizations. Throughout his professional career he contributed in various roles to the flight control system development for major European military aircraft and helicopter programs. Furthermore, he acted as project manager for unconventional airborne and ground-based systems. In 1999 he joined the European Aeronautic Defence and Space Company. Since then he has worked mainly in the field of process engineering. He now holds a position as Senior Expert Systems Engineering Processes within CASSIDIAN. Convinced of the importance of systems engineering for the company, he contributed to achieve its high level of recognition by CASSIDIAN and EADS today.
He is a former president and an honorable member of GfSE – The German Chapter of INCOSE. He represents CASSIDIAN on the INCOSE Corporate Advisory Board.

Kevin Forsberg
Dr. Forsberg, an INCOSE ESEP, draws on 27 years of industrial experience in System Engineering, Project, and Proposal Management and 28 years of successful consulting to both government and industry. He has developed leading industry training programs and has trained over 9,000 high technology project managers. Training courses co-developed by Dr. Forsberg have been taught in 39 countries. These courses have also been taught in a number of universities, including Santa Clara University, Stanford University, University of California, and American University.
He co-founded the Center for Systems Management (CSM) in 1989. He is currently president of OGR Systems, Inc., presenting training at many companies in the US and Europe.

Other accomplishments:
- Early member of the Lockheed Corona Project, America's first successful satellite
- Coauthor of Visualizing Project Management, J. Wiley & Sons (third edition, 2005)
- Coauthor of Communicating Project Management, J. Wiley & Sons, 2002
- Lockheed Program Manager of the Space Station Program (1981-83).
- Awarded the NASA Public Service Medal (1981) "in recognition of his outstanding technical and managerial contributions to the Space Shuttle Program." (He was Program Manager of the Space Shuttle tile project, from research through full-scale production.)
- Awarded the CIA Agency Seal Medallion for excellence in Project Management training, and in recognition of his pioneering efforts in the field of Project Management (1998).
- Awarded the Pioneer Award for INCOSE (International Council of System Engineering), jointly with Hal Mooz (2001).
- INCOSE Certified Systems Engineering Professional (CSEP) in 2004; ESEP in 2010
- Life-time Fellow, American Society of Mechanical Engineers (ASME) (1972)
- Associate Fellow, American Institute of Aeronautics and Astronautics (AIAA) (1968)
- Fellow, International Council of Systems Engineering (INCOSE) (2009)
- Member & co-chair, INCOSE Certification Advisory Group (CAG), 2004 to 2010.
- Co-chair of INCOSE Working Group on INCOSE SE Handbook, versions 2a, 3.0, 3.1,
- Author on BKCASE team, and member of six-person steering group (INCOSE/IEEE/Stevens Institute)
- Over sixty published articles in referred journals and proceedings

Education
- Ph.D., Engineering Mechanics, Stanford University (1961)
- M.S., Engineering Mechanics, Stanford University (1958)
- B.S., Civil Engineering, Massachusetts Institute of Technology (1956)