Architecting of Systems for Participation in System-of-Systems

Dr. Dieter Scheithauer, INCOSE ESEP
Breitensteinstr. 26
83727 Schliersee
GERMANY

dieter.scheithauer@hitseng.eu

ABSTRACT

Systems architecting, requirements engineering and model-based systems engineering are the dominant schools of thought in systems engineering today. They evolved in response to each other, partly in agreement and partly in opposition. To some extent they invented their own terminology and semantics. Due to overlaps, systems engineering as a whole lost some coherence of terminology and semantics. This paper proposes a fusion of systems architecting, requirements engineering and model-based systems engineering. Three fundamental principles are taken for guidance: making the best use of human cognitive capabilities, applying value stream thinking, and considering the whole system life cycle.

On the basis of previous work, two essential solution elements are described: the strict interpretation of the V as the overall systems engineering value stream, and appropriate system life cycle phase models. In the overall systems engineering value stream the commonality of the engineering of systems-of-systems with the engineering of any other system or system element in the system architecture becomes visible. However, the integration of architectural frameworks and value stream thinking bears still major challenges. The system life cycle considerations emphasise the integration of systems engineering in an enterprise in support of portfolio strategy development and business planning.

1 INTRODUCTION

For bringing a new high-integrity technical system into being, contributions from many disciplines are required, each exploiting its comprehensive body of knowledge and experience. Without human imagination and human creativity all efforts would run dry and would lead to nothing. Technical systems are invented from humans for the benefits of humans, and hopefully also for the benefit of the planet. Processes, methods and procedures that exploit human capabilities in a holistic manner have the highest potential to improve effectiveness and efficiency.

Aunt colonies are not a good model for human societies, even though some similar behavioural patterns are shared. The cognitive capabilities are definitely different and lead to richer social behaviours among humans. But as Heinz von Förster has argued in an attack on behaviourism, the possibility of becoming manipulated and trivialised is one way of cognitive reaction on external threats [4]. Nevertheless, some authors believe that primarily the restless effort of large communities of engineers are the basis for success [28]. Views like that provoked Maier and Rechtin to classify engineering as a pure deductive process, e.g. saying that "engineering deals almost entirely with measurables using analytic tools derived from mathematics and the hard sciences" [20]. In opposition, they claim systems architecting as being the inductive part considering issues on a large scale and being in charge of the real decisions that matter. Could it just be that the inductive and deductive approach are best applied together? Deduction leads to differentiation creating new terms and

new semantics while induction creates meaning by setting terms and semantics into context. Are we not going too far when system architects talk about logical views what in systems engineering terminology is called requirements [19]?

In the history of systems engineering there were already too many new findings and viewpoints claimed as real paradigm shifts in the sense of Kuhn's definition [16]. For this reason, this paper starts with a simple narrative of the history of systems engineering to support the holistic claim of systems engineering also beyond the imagined schism between systems architecting and systems engineering. The holistic viewpoint is then defined using research results from cognitive psychology. On this basis, a distinction of the systems engineering value stream and the system life cycle is introduced integrating the inductive and deductive facets of systems engineering. The remaining two sections describe the Overall systems engineering value stream and the early life cycle phases summarising mainly material from a number of previous papers [24, 25, 26, 27].

2 SYSTEMS ARCHITECTING – THE BETTER SYSTEMS ENGINEERING?

2.1 A Short Historical Narrative about the Evolution of Systems Engineering

A narrative is a sequence of statements following a pattern "then … and then … and then …". The meaning of "and then" ranges from temporal relations to strong causal dependencies. The range in between is populated by any kind of reasonability [16]. Temporal relations are less well memorable than causal dependencies. Temporal memories fade over time, and become rather condensed. Reasonable and causal Relations expressing reasonability (including strong causality) are transferred to semantic memory, and will be used by further cognitive functions [9].

The following narrative tries to stick to the historical sequence as anticipated and experienced over the last 35 years by the author. Easily, the reader may imply reason between the statements, episodes and facts. This will be unavoidable, and is intended as well. However, the author does not claim causality according to scientific standards. More detailed social studies on the history of systems engineering and related sciences would be required to separate correlational from causal dependencies.

The recognition of the system life cycle as a system property initiated systems engineering as a distinct integrative engineering discipline. In order to achieve cost efficiency in large, government funded programmes, engineers started to explicitly consider all conditions a system is exposed to during its life cycle. Life cycle phase models were invented and detailed. Life cycle phase models defined the activities and expected outcomes of each phase in order to synchronise progress. Major decision points were introduced at the beginning and at the end of each phase. In most cases they were combined judging the success of the previous phase and adjusting the further route of project conduction. Over time, system life cycle phase models became more and more fine grained with respect to early system life cycle phases up to development. Unfortunately, not many information about this period can be found in the Internet, but the author remembers system life cycle phase model definitions with up to thirteen or fourteen separate phases. Today, system life cycle phase models are still valued as a heritage, but not much use is made of them actually. The ISO standards are rather brief and inconsistent in their descriptions [11, 12].

Software engineering entered the scene for discussing development phase models. Royce proposed the waterfall model in 1970. Due to practical experience the waterfall model was modified by allowing iterations. Incremental development philosophies were proposed not to develop all feature at once, but to add a set of new features with every iteration. With the spiral model, Boehm proposed around 1985 an evolutionary approach to improve functionality with each iteration. Craig Larman and Victor R. Basili describe this episode in more detail [18]. Under the assumption that software maintenance means re-entering development, post-development life cycle phases were not considered for being special. This omission has

left many issues regarding software-intense embedded systems open.

Meanwhile, remarkable budget overruns, time delays and quality deficiencies were still observed in practice despite the most sophisticated system life cycle models. In order to let take over industry more responsibility for their doing, fixed-price development contracts became introduced in the 1980s. Requirements engineering was promoted to ensure high-quality contract specifications. However, this brought no real relieve in case of high innovation challenges when both, customer and developer, had to learn what is feasible in practice, and what not.

The V-model became an important part of systems engineering and has been quite popular since its invention in 1991 [5]. Its interpretation was detailed and expanded over time [6]. Incremental and evolutionary approaches were blended into the Vee, but disturbing its shape. The double Vee was invented. And, in 2013 a strict interpretation of the V as the overall systems engineering value stream was proposed [25] leaving all reminiscences to system life cycle phase models behind.

With proposing systems architecting, Rechtin and Maier reacted to the excesses of requirements engineering that represented the mainstay of systems engineering in this period [20]. Although requirements engineering brought some improvements, it failed to eliminate budget overspending, time delays and quality deficiencies completely. Unfortunately, the two schools of thought achieved a weak compromise only. Since then, they have evolved nearly as two distinct disciplines. Systems architecting paved the way for the development of architecture frameworks and occupied systems-of-systems (SoS) as their sole playing field. Requirements engineering maintained its strong position for the development of technical products. The real victims of the struggle between requirements engineering and systems architecting were systems engineering as a whole and the functional definition of systems. Functional definition survived mainly as a part of requirements engineering called functional analysis [11], and as an integral part of architecture framework views [21]. Systems engineering as a whole suffered from losing some coherence of its terminology and semantics.

With these deficiencies it was no real surprise when Model Based Systems Engineering (MBSE) was introduced around 2004 claiming to become a real paradigm shift in systems engineering [7]. MBSE has a high potential of course. In fact, it uses the progress in modelling and simulation technologies, and information technology infrastructure. The conduct of systems architecting on the basis of architectural frameworks profited a bit more from MBSE than requirements engineering. Overall, functional definition climbed up to an adequate prominent role in the current understanding of systems engineering.

So far our narrative may close with an all's well that ends well. It should have become clear that systems architecting is not the better systems engineering, but an essential contribution to systems engineering.

2.2 A Systems Engineering Synthesis

A systems engineering synthesis of the available systems engineering assets should be oriented towards the goals that need to be achieved. That systems engineering has an improvement potential is obvious as budget overruns, time delays and quality deficiencies are still observable. It is more surprising when complex military procurement programmes run smoothly meeting all expectations than winning public attention due to significant deviations from plans. Thus, the primary goal is easily to state: meeting all quality characteristics according to the planned time schedule and within the allocated budget.

To achieve this goal, we have to consider three essential factors. First, the planning has to be realistic at project start. The quality characteristics imposed on the system, the allocated budget and the envisaged time schedule must fit to each other. Second, the system should satisfy its quality characteristics at project end. Third, the process transforming the system from the initial state to the final state has to be effective. If the process is also efficient, actual project duration and actual consumption are optimised.

One important effect of the complexity of a system is that the final knowledge about the system at the end of the system life cycle exceeds the initial knowledge about the system at the beginning of the system life cycle. The knowledge gain is shown schematically in Figure 1 as the learning curve. Assuming that everything is known about the system at the end of the life cycle, the area between the 100 percent line and the learning curve is finite. The size of the area may be interpreted as an indicator of uncertainty and risk: as smaller the area as better. Minimising the area may be achieved by increased initial knowledge and by faster learning. The initial knowledge grows with experience and learning to master the technologies used for the system. However, it is wise never to claim that the initial knowledge equals the final knowledge. This is a characteristic of trivial or complicated systems, but not for complex systems. Faster learning is a process issue comprising the art of making action driving decisions that the right design decisions are made by the right people at the right point in time. The design decisions themselves close the knowledge gap. In order to keep uncertainties and risks on a manageable level, the decomposition of the system life cycle into system life cycle phases and managing portions of the system life cycle as separate projects are good practices.

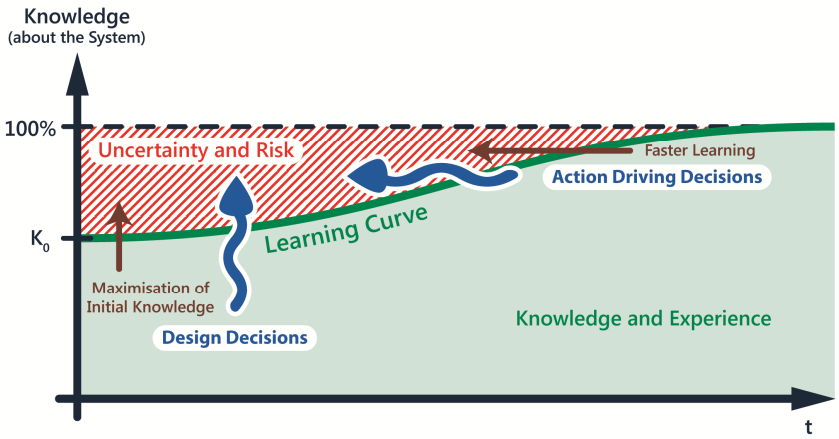

Figure 1: The Systems Engineering Learning Curve.

Focussing now at the final state of the engineered system at the end of all development activities, we have to consider the quality of the configuration baseline describing and identifying the system. It is obvious that high system quality is equivalent to a consistent and high-quality configuration baseline. As it may take years to reach the end of development, good systems engineering practices will focus continuously on the quality of the evolving configuration baseline. In other words, every day it should be possible to establish a consistent configuration baseline saying what is achieved, what is omitted, and which inconsistencies exist.

Common management practices are often lazy, more waiting for than controlling the evolution of configuration baselines. Generating Gantt charts directly from a work breakdown structure, setting milestone dates for the completion of all artefacts, e.g. documents and models, and then monitoring their actual completion dates, are unsuitable to actively control the constant evolution of consistent and high-quality configuration baselines. Controlling the system development flow following a sound value stream definition are the two inevitable prerequisites to improve the quality of systems engineering management. Queuing theory provides the scientific evidence for the benefits of flow management [23]. The same result is obtained by comparing document-centred, process-oriented and value stream approaches regarding their capability to maintain consistent configuration baselines continuously throughout development [27].

The whole systems engineering value stream may be described as overall systems engineering value stream expressing the flow of configuration baselines over the system architecture [25] and work product generation sequences for defining the value stream for generating the configuration baseline of a particular system or

system element [26]. The benefits have been proven over the last twenty years under various conditions [24].

System life cycle phases models and the systems engineering value stream are interwoven. The systems engineering value stream or portions of it will be executed in particular life cycle phases, but not exclusively in just one specific system life cycle phase. However, the expected results may vary due to the specific goals of the particular life cycle phase. We dive deeper into this matter below in the section about the system life cycle.

At the end of this section we return to the current state of systems engineering as described in the narrative above. What are the opportunities for advancing the art of systems engineering further? The outlook is promising indeed. Results from cognitive psychology provide the evidence that a system definition in terms of system requirement, functions and architecture has the potential to maximise human understanding of the particular system, see Figure 2. These are the essential views exploiting our human brain power best.

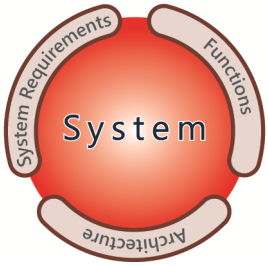

Figure 2: The three Essential Views on a System.

Our brain features two operating channels [8]. One channel deals with dynamic state changes in our environment. This operating channel makes us real-time capable. D. Kahnemann calls it fast thinking [13]. Fast thinking is highly associative. All what we have learned theoretically or by doing plus our lifetime experience is merged to act and react in our individual environment appropriately. Conscious thinking seems only to be involved when we sense and perceive deviations from well-known environmental patterns. For illustration, driving a car may serve as an example. A beginner needs a lot of conscious processing to watch the surrounding traffic and to coordinate the operation of throttle and brake, setting direction lights and turning the steering wheel. With increasing practice all this coordination is achieved unconsciously. Similarly, driving a new route for the first time seems to take longer than after having used it every day for several years because most stimuli are known after a while and subjected to unconscious processing.

D. Kahnemann designates the second operating channel as slow thinking [13]. This operational channel is important to get an understanding of the world learning to make distinctions and identifying the properties of things. Observing children learning to speak provides a lot of insight. A young boy may call everything that passes by on wheels a car. A few years later, this boy distinguishes all types of motorised and non-motorised vehicles with ease. He will also identify differences that definitely nobody has explained to him explicitly. A lot of conscious thinking is involved here, but sometimes insight comes at night, or when busy doing other things [14].

The real-time thinking channel maps to the functional description. All types of functional models are powerful communication means. However, the downside is the highly associative nature of this operating channel. As everyone has built up a unique memory it is not easy to safeguard that what the originator intends to communicate is really understood accordingly by the recipients.

Architectural descriptions are linked to slow thinking. We identify entities, categories, things, systems - or however we want to call it - as wholes and associate them with features and behaviours. As we do not know

the richness of associated features and behaviours in the mind of other people, similar communication problems may be encountered like with functional models.

To safeguard communication success, the good old natural languages provide the highest versatility and flexibility to adjust to a wide variety of communication scenarios limited only by the skill level of mastering the used language by the people participating in the communication. In critical communication situations, most people will opt for talking together directly. This underpins the importance of system requirements. Systems requirements express the commitments the engineering team in charge for engineering the particular system is taking responsibility. The demand of a requirements based demonstration of compliance in many industrial domains underpins the importance of system requirements.

In summary, system requirements, functional descriptions and architectural descriptions together are the best means to describe a system. The functional and architectural views exploit the full human cognitive capabilities. System requirements safeguard successful communication in multidisciplinary engineering teams and beyond. It does not mean that all system features are fully visible in all views. Some past exaggerations in requirements engineering have demonstrated that this would be a nonsense indeed. High quality is achieved when the three views are complementary and consistent to each other and when they together provide a comprehensive, if not complete definition of the system.

Requirements engineering, systems architecting and model based systems engineering have achieved levels of maturity that a fusion of their particular strengths may be achieved in due time. For the harmonisation, human cognitive capabilities, systems engineering value stream thinking and system life cycle efficiency provide the most relevant guiding principles.

3 THE OVERALL SYSTEMS ENGINEERING VALUE STREAM

3.1 The Basic V

The purpose of performing the systems engineering value stream is to transform stakeholder needs into products or services that satisfy primarily the stakeholders who are utilising the system for their purposes. For this reason, stakeholder needs and stakeholder satisfaction are shown off-side the V, see Figure 3.

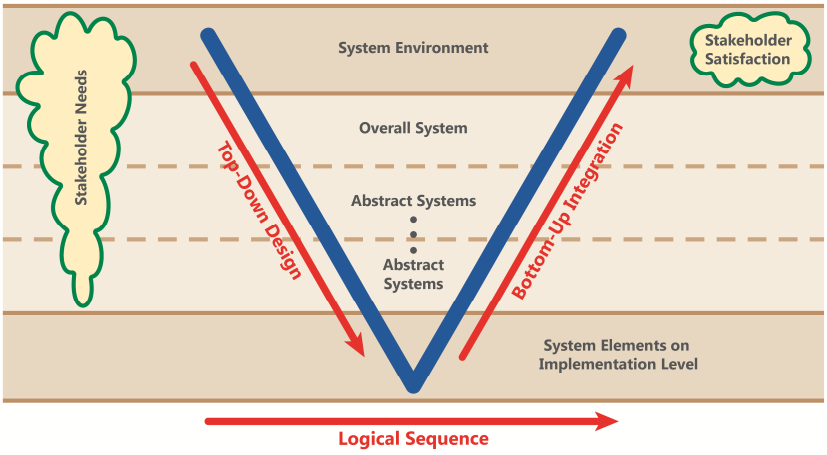

Figure 3: The Basic V.

The vertical dimension shows the system architecture. Top-down design and bottom-up generation generate the V shape. The V provides a notion of the complete system architecture, but with an organisational focus

on a single enterprise. The V is not well suited to map complete supply chains in market economies. Even in case of a pure green-field approach, there is no real answer to the question what is at the peak at the bottom of the V. Is it the mining of the ore to get the metal the product is built of? And even if, what is with the installations in the mine that are made from the same metal? The linearity of the V cannot depict the economic networks in sophisticated technical cultures by default.

To consider organisational boundaries in the systems architecture, bears the advantage that areas of entrepreneurial responsibility become visible. The layer of the overall system stands for the products delivered by the enterprise. Product liability and product safety are imposed on the overall system. In the European Union, two directives ensures common legislation in all EU countries [2, 3]. For specific application domains, regulatory requirements may be established interpreting and amending the laws, especially regarding safety concerns.

Above the overall system, all higher levels of the system architecture and also neighbouring systems are subsumed in the system environment. The system environment is the part of the surrounding world considered when engineering the overall system. 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. Eventually, customer and user satisfaction depends on operating the overall system in its surrounding world.

System elements resulting from the overall system's decomposition performed within the enterprise 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. Their characteristics are identified by configuration baselines. 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 have to contribute to satisfy the stakeholder needs on the overall system [10]. 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. Thus, the system architecture is distinct from the spatial containment hierarchy.

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 system elements on the implementation level designate all system elements procured from other parties. The supplier organisations do not share responsibility for the overall systems of their customers in a legal sense. Their starting point are the requirements allocated to their overall system that is a system element on the implementation level in the eyes of their customers. System elements on the implementation level may be standardised items, existing systems that are re-usable for the new purpose, customisations of existing systems, or new designs to be developed on demand according to the allocated requirements.

Figure 4 shows an illustrative example how the interaction of two enterprises can be expressed by the interaction of the overall systems engineering value streams of two enterprises. Enterprise A is contracted to

develop a fighter aircraft. They start to analyse the system environment in order to understand all relevant characteristic of the system environment. Their overall system is the fighter aircraft. The architectural decomposition defines a flight control system that is designed in-house. An electro-hydraulic primary actuator customised for the particular aircraft needs to be procured from a supplier specialised in these matters. In order not to jeopardise the own overall system, Enterprise A has to ensure that the allocated requirements imposed on the primary actuator are achievable under the particular constraints. Enterprise B will analyse the allocated requirements taking all their knowledge and experience into account. Their overall system is the primary actuator itself. The electrical actuator input is transformed to mechanical units by a Direct Drive Valve. For the construction of this valve, position transducers are needed. These position transducers may be standard items.

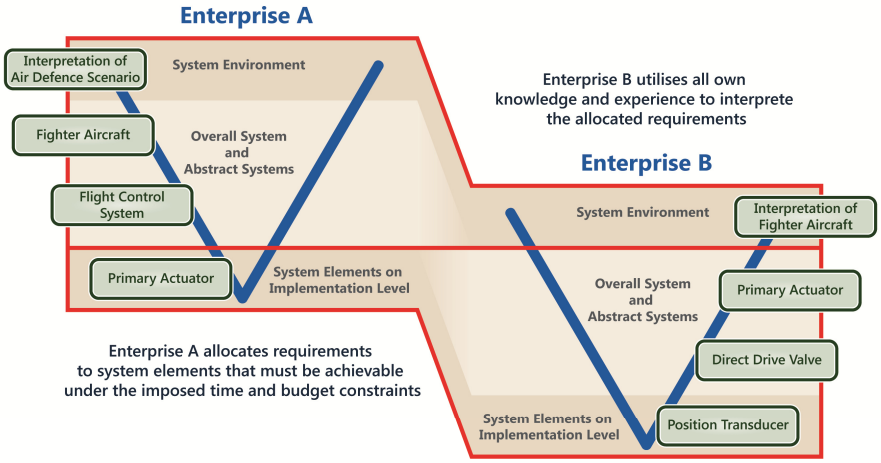

Figure 4: The Interaction between Customers and Suppliers – Example.

The horizontal dimension of the Basic V expresses the logical sequence of the systems engineering and the associated information flow. In earlier interpretations of the V, the left-to-right direction has been the time line [6]. This change has more than marginal impacts. It removed the last reminiscence of expressing sequential progress in absolute time inherited from system life cycle phase models. For a single iteration, the V is mapped to the time line once. This may be supported by a template modelling the value stream. Multiple iterations and concurrency can be expressed in time schedules, and the planning support of the project planning tools may be exploited for minimising waiting and balancing resource utilisation. Strict sequential performance of each iteration allows the continuous control of the evolution of all configuration baselines and ensures that temporary configuration baselines are consistent over the whole system architecture.

3.2 The Development V

The Development V comprises four sub-processes as shown in Figure 5 by the red areas. These four processes are sufficient to describe the development flow. A mapping to other development process breakdowns like the one in ISO 15288 in most parts is quite easy with the exception of the System Integration Planning Process.

The Stakeholder Requirement Definition Process is concerned with identifying stakeholders and their role with respect to the system. The needs of these stakeholders have to be transformed into stakeholder requirements that are able to communicate the stakeholder needs. However, this may be a cumbersome

activity. 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. Sometimes they may opt for specific solutions because they were involved in corrective actions previously, and want to avoid the same troubles again. 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. Missing some stakeholder needs or not understanding some right are permanent concerns throughout the engineering activities.

Most stakeholder requirements are elicited on the levels of the system environment and the overall system. But further stakeholder requirements may be defined for the abstract systems for two reasons. Design decisions may have a mutual impact on the stakeholder needs, for example in the design of human-machine interfaces. And, some stakeholder needs may be better understood by the engineers of the particular design teams of the abstract elements due their domain specific knowledge that may not be available by the design teams of the higher level systems.

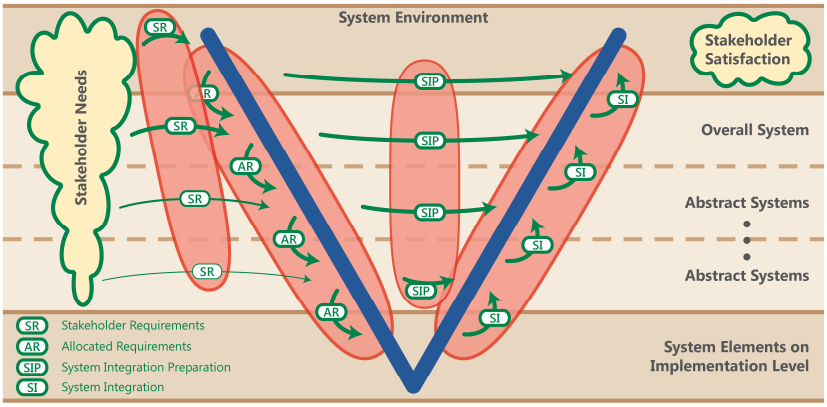

Figure 5: The Development V.

The System Requirement Definition and System Design Process is concerned with the generation of the three complementary and consistent views of system requirements, functions and architectural decomposition that together have to describe the system comprehensively. Thus, it is the process defining the system. The top-down information flow may provide the impression that system definition would be performed best as a deductive process managed in a command and control fashion. System definition needs to be a value-generating exercise. If there are no specific characteristics emergent for this system, there would be no need for this system as a distinct node in the system architecture. The inductive approach of systems architecting needs to be applied for every system and system element of the system architecture.

Furthermore, due to the specialisation principle it has to be assumed that the engineering team in charge of defining the system is the most qualified group of people within the organisation for the job. Taking into account that modern quality management systems rely on the quality responsibility shared by all parts of the organisation [10], it is a must that the system definition of each system adds value to the overall system.

The considerations on the cognitive capabilities of humans above lead to the recommendation not to segregate the definition of system requirements from the definition of the functions and the further architectural decomposition as all three views should evolve together to keep consistency of the system definition. The installation of integrated multidisciplinary teams is recommended in order to consider all relevant aspects together. Architectural frameworks like the NATO Architecture Framework (NAF) [21] help to address all issues comprehensively, but too many views developed in isolation bear the risk of

inconsistency, gaps and redundant information. Due to the associative richness of their minds, experienced engineers may see more in a view than intended by the engineer who has generated it and as allowed according to the method definition. Even in the old days with just two views, the segregation of data flow from control flow prevented nobody to imagine a reasonable control flow when analysing a data flow diagram. Usually, this became evident in case of problem resolutions demanding corrective actions and re-work.

Although system requirements definition and system design are mutually dependent, it makes sense to maintain a basic activity sequence, at least as far as the release sequence of work products is concerned. Figure 6 defines a sequence of four activities.

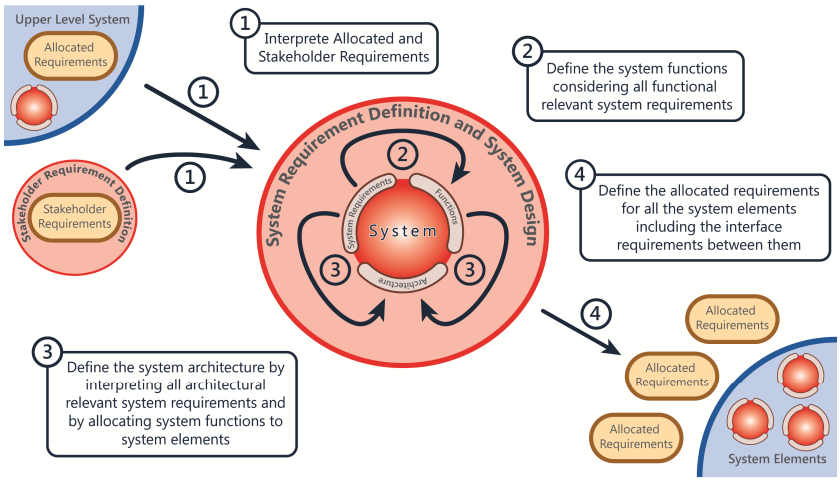

Figure 6: System Requirement Definition and System Design.

In the first step, the allocated requirements and further stakeholder requirements are analysed. For achieving a good understanding of the underlying stakeholder needs, allocated requirements should be followed upwards the system architecture to the initial stakeholder requirements. This includes interface requirements up to the architectural level introducing the corresponding interfaces. For interfaces external to the overall system, this means to go upwards through all architectural levels to the overall system. In many cases, system interface requirements may find their origin in internal interfaces established somewhere upwards in the system architecture. For all requirements found by this traceability exercise, the recorded rationale needs to be analysed in order to understand the implications on the own system or system element. Furthermore, from all requirements found, the downward traceability in other branches of the system architecture should be followed. Thus, the engineering team gets full awareness which other systems may have a stake in their external interfaces, and which refinements of the interface requirement have occurred in other branches. The first step results in a first set of system requirements. Compared to the allocated requirements imposed on the system and further stakeholder requirements elicited, the system requirements may be refined in order to be precise in the language of the engineering disciplines represented in the engineering team in charge of the development of the particular system.

The second step starts with identifying functions and sub-functions based on the system requirements. It continues with further elaboration of the functions and sub-functions resulting in functional models of the system. This may lead to further system requirements to make additional commitments visible and to ensure that appropriate assurance activities are performed.

The freedom in designing functions is particularly constrained by the external interfaces imposed on the system. To some extent, the functional interfaces may be refined for own purposes. As far as external

interfaces are concerned, compatibility with the corresponding interface design in other branches needs to be maintained. For this purpose, harmonization of interface requirements on a peer-to-peer basis may be preferred, if possible. Tracing detailed interface characteristics always up to the level of the introduction of the interfaces has two detrimental impacts on process efficiency and system quality. First, many detailed requirements have to be traced over a number of levels in the system architecture without adding value to most intermediate levels as they are of no relevance for the emergent characteristics of the particular systems. This just increases the administrative effort for requirements management. Second, the reactiveness within the overall systems engineering value stream suffers due to the time needed for generating updated configuration baselines of all the affected systems. Most likely engineering work will progress on the basis of a high number of assumptions. In consequence, the ability to continuously take consistent high-quality configuration baselines of the overall system will be lost.

Another concern is the appropriate level of depth in all modelling activities. To get a comprehensive understanding of the implications of alternative solutions sometimes rather detailed models may be generated and analysed. However, not any model detail should become a mandatory requirement imposed on lower level system elements. Otherwise, the permissible solution space for lower level systems may be too restricted for optimising the design of these lower level system.

The architectural definition defines the system elements by allocating functions to the system elements and considering all architectural relevant system requirements. Internal interfaces between the system elements on the next lower architectural level are established. Furthermore, inherited external interfaces have to be allocated to the system elements designated for satisfying those external interfaces. As far as further system interface requirements are established or existing system interface requirements are refined, harmonisation with the other systems or system elements involved in those system interfaces has to be achieved.

In the last step, the requirement sets containing the allocated requirements for the system elements need to be established. The hand-over of allocated requirements needs to be tightly controlled. The quality of the allocated requirements should be maximised in order to avoid nuisance iterations over several levels of the system architecture. The system design should have reached an acceptable level of maturity. This does not necessarily mean that the design has to be complete for each iteration. A configuration baseline should ensure that the allocated requirement sets for all the system elements fit to each other. Please note, it is recommended here to contain the allocated requirements imposed on a single system element in a work product on its own to support the development flow without an adverse impact on configuration management.

Not uncommonly, interface requirements have to be refined to capture the viewpoint of the particular system element better. When the system element is designated as a system element on the implementation level, e.g. is procured from a supplier, it is important at this step to ensure that system interfaces have been defined completely. For this purpose, interface control documents should be compiled. The information content should be a summary of all interface requirements including all refinements. Completeness of an interface control document may be evaluated by checking that all information requested by the interface control document template is available. If this is not the case, the system design is not complete. Due to the implications of immature interface control documents, it is then recommended to update the system requirements and the system design first.

System Integration Preparation comprises the four activities that link the left leg and the right leg of the V directly for each particular system. The corresponding information flow by-passes the V. First, the assurance requirements have to be established. Assurance requirements define the specific means and objectives for demonstrating compliance with the system requirements. Second, a system integration concept should be generated considering all assurance requirements. Preferably, there should be one system integration concept capturing the integration of the overall system and all abstract systems together. A common testing strategy with delegation principles how different kinds of tests are satisfied by the compliance demonstration on a

particular architectural level. Third, appropriate system integration environments need to be established according to the demands of the system integration concept. Fourth, the system integration should be planned in detail.

System Integration comprises the virtual system integration that utilises executable models and the actual system integration. Virtual integration can be commenced when more detailed models from lower level systems become available. However, a policy and an infrastructure is needed to allow virtual integration with low effort. 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. For enabling iterations and concurrent system integration activities, the system integration concept needs to consider the impact of system elements not providing their complete functionality deliberately. Correct, consistent and high-quality configuration baselines are a mandatory prerequisite to protect people, the system integration environments, and the equipment to be integrated from being injured, killed or damaged.

3.3 The Assurance V

Assurance activities comprise validation, verification and process assurance. Assurance activities should be performed as continuously as possible. Before the release of each version of any work product, the quality of the content needs to be examined. The results should be recorded. When configuration baselines are established to control the hand-over of information in the overall systems engineering value stream, the quality of all the referenced work products in conjunction needs to be assessed. This dedicated assurance activities regarding validation and verification are depicted in the Assurance V, see Figure 7. Process Assurance is not shown at all.

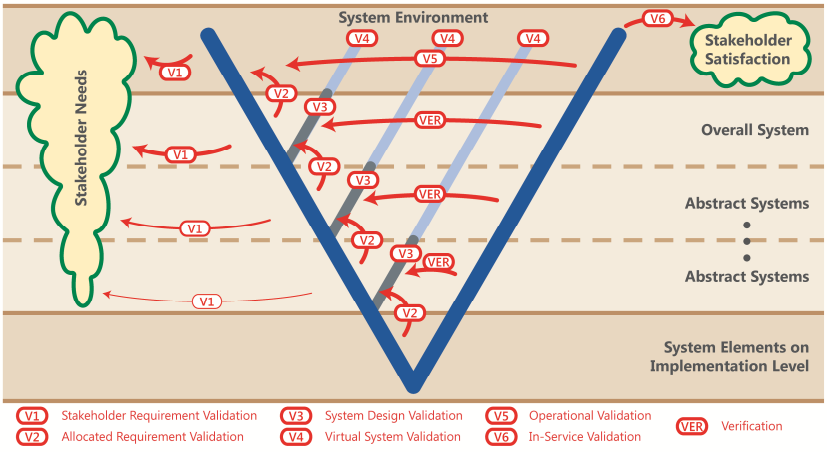

Figure 7: The Assurance V.

Verification is a rather straight forward process. Verification generates the evidence that the integrated system satisfies the system requirements. Verification is performed in the context of system integration after the system is successfully integrated. The virtual integrated system may be utilised for some verification purposes when the fidelity of the model has been proven to be appropriate. For assurance requirements demanding other means than tests, the appropriate manual inspections, automatic analyses, reviews and analyses of previous service experience have to be performed. Finally, a verification coverage analysis summarises the verification results including observed deviations from expected results.

Validation is concerned with the question, if the system as defined fulfils all relevant stakeholder needs. Performing validation takes a quite different approach as verification. In verification, the inputs on which the facts have been generated are retrospectively challenged. This includes that validation wants to detect

missing stakeholder requirements. On the one hand, validation searches for expected features missing. On the other hand, validation is looking for unintended functions that could harm people or the environment due to missing stakeholder requirements demanding protection. From the perspective of the available knowledge about the system, validation is looking for intangible features of the system. There is no straight forward way to do this. For this reason, validation should follow a defensive approach by evaluating all newly generated information about the system with respect to stakeholder needs. The Assurance V defines six validation steps in total.

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 position as when eliciting stakeholder requirements. The Validation Step V2 is accomplished by applying the same techniques as described above for Validation Step V1.

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 lower 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 to V4 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.

3.4 The Dynamic V

Iterations over the V are unavoidable when the initial knowledge is significantly below the 100 percent of the

final knowledge as discussed above. Iterations over the systems engineering value stream are caused due to three different categories of reasons. First, the same systems engineering activities are repeatedly performed in different system life cycle phases. The objectives regarding the expected outcome may vary depending on the overall purpose of the particular system life cycle phase. Second, incremental or evolutionary development philosophies may be applied. Third, lessons learned during the execution of the systems engineering value stream may demand incorporation of appropriate changes. Whatever the cause might be, all iterations need to be properly managed to ensure always consisted, high quality configuration baselines.

The Dynamic V in Figure 8 shows three different types of iterations. Iterations over a single system or system element are drawn in green. The colour green is chosen to express that iterations of this type should be highly welcomed. On the left leg of the V, they improve the maturity of the allocated requirements forwarded to the system elements. On the right leg of the V, they enhance the maturity of integrated system elements forwarded to commence integration of the upper level system. Each iteration is associated with some extra administration effort. However, the administration effort for iterations of this type is remarkable lower than for the other two types. When the quality of configuration baselines is higher up from the beginning for each system and system element, the capacities to manage the other two types of iterations are less absorbed by nuisance problems that could have been easily contained in the scope of a single system element. Thus, iterations affecting several levels of the system architecture may be performed with higher efficiency and in a more agile manner.

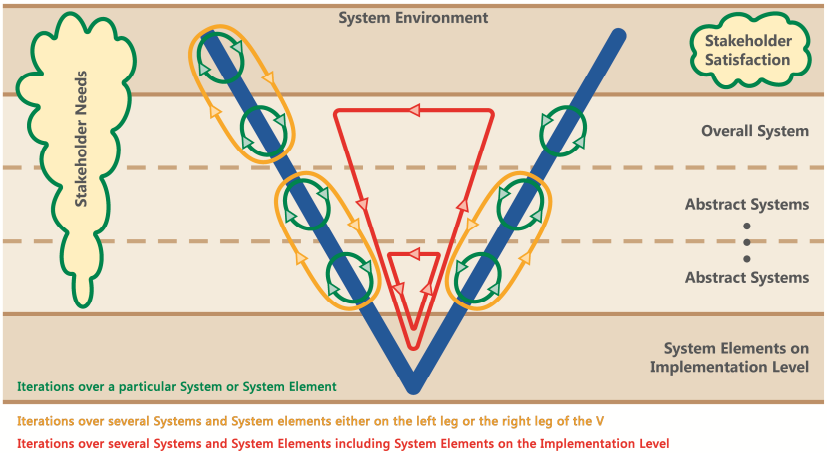

Figure 8: The Dynamic V.

The second type of iterations, drawn in amber, affect several systems either on the left leg or the right leg of the V that are under the control of a single enterprise. Applied on the left leg, they enhance the maturity of the system architecture before procurement contracts with the suppliers of the system elements are signed. With less demand for renegotiating contracts with the typical legal and economic impacts, a project will benefit in two ways. First, less project time will be lost due to waiting for updated contract agreements. Second, suppliers will find less opportunities for justifying claims for additional budget. On the right leg of the V, iterations of this type improve the reactivity to cope with anomalies, especially when system integration is performed on several levels of the system architecture concurrently.

The third type of iterations affect several levels of the system architecture including the implementation level. For good reasons, they are drawn in red. Usually, they are the most costly. Preferably, they should be pre-planned. Pure event-driven iterations of this type should be avoided with one exception. If the implementation effort is quite low, they may be regularly applied, for example in most agile software development methods.

4 THE SYSTEM LIFE CYCLE

4.1 System Life Cycle Efficiency and Sustainability

A system has a system life cycle as an inherent property. The system life cycle perspective is rather different from the systems engineering value stream perspective. The systems engineering value stream focusses on how engineering teams are working together within an enterprise to conceptualise and to develop a system. From a system life cycle perspective, the producing enterprise is only one of the enterprises that have some sort of interference with the system. But the producing enterprise has a rather specific role due to being responsibility in terms of product liability and product safety. For the European Union the legislation is defined by two directives that are implemented as national laws in the EU countries [2, 3]. Due to these obligations, the producing enterprise is placed on the system life cycle. Of course, all the other enterprises are also produce products or services they are liable for. All these enterprises interfere with the system life cycle of the system in focus via their particular products and services. They have to be considered by the producing enterprise in addition to their own interests regarding the topics system life cycle efficiency and system sustainability.

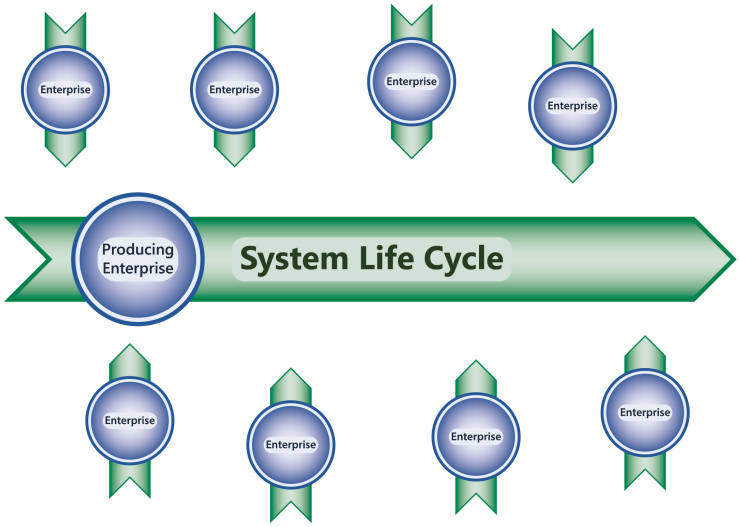

Figure 9: The System Life Cycle and the Enterprises Involved.

The term system life cycle efficiency is preferred here over the older term cost efficiency because it is closer to common language. Experts in optimisation have a wider understanding of cost and cost functions than people less attached to disciplines like operations research and automatic control. System life cycle efficiency is defined here as the relation of the benefits gained out of a system throughout its life cycle and the system's resource consumption over its life cycle. In calibrated units, a system life cycle efficiency greater than one indicates added value. If the system life cycle efficiency is less or equal one, it will not survive for long. Of course, the calibration is a tedious task by itself and the resulting figure may be challenged from many sides. However, for assessing market opportunities the estimation of system life cycle efficiency is an important tool.

A reasonable system life cycle efficiency is an important prerequisite for a sustainable system, but without the opportunity of all the enterprises (including individually acting people) to benefit from the system, links in the chain will break and may jeopardise the continuation of the system life cycle. Therefore, the system life cycle is as well an important input to the systems engineering value stream as it is also the outcome.

System life cycle phase models help to structure the system life cycle and to identify interaction with other systems in particular system life cycle phases. A one fits all solution for the whole life cycle is unlikely although the content of individual life cycle phases may have a potential for standardisation. In a further step, all system stakeholders can be identified. Thus, a carefully analysed system life cycle is supportive to the stakeholder requirement definition process and helps in the end to achieve system life cycle efficiency and system sustainability.

4.2 A Generic Business Development Cycle

Figure 11 shows a generic business development cycle to be applied by any kind of enterprise. It combines elements from holistic marketing approaches [15] and integrated business planning [1, 22] with systems engineering. A product portfolio strategy bolstered by sound product concepts, reliable marketing plans, an enterprise oriented technical roadmap, and the business plan itself are the main outcomes of this process.

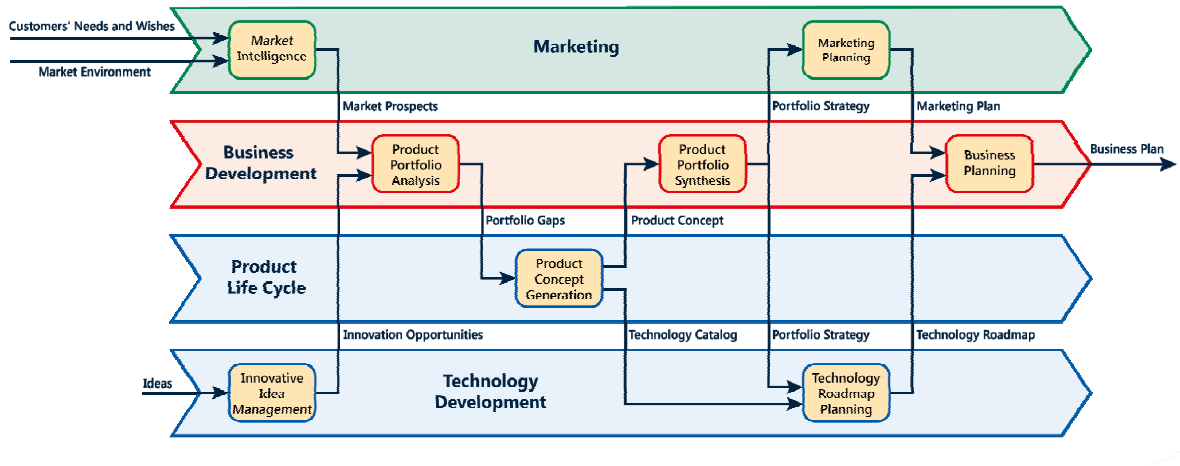

Figure 10. The Business Development Cycle.

Market intelligence is concerned with the identification of current and future needs of customers and further stakeholder groups who could be involved in product utilisation. In depth knowledge of the product application domains as well as of stakeholder views and habits are necessary pre-conditions for successful market intelligence. The stakeholder requirements definition process provides the methods and tools to record stakeholder requirements derived from the identified needs. The outcome of the market intelligence activities are the future market prospects.

The future oriented competencies and capabilities of an enterprise are not at least dependent on the innovative ideas generated or adopted by the members of the enterprise. The innovative ideas may be categorized as follows: (1) ideas for new products, (2) technological ideas enabling the invention of new products, and (3) innovative ideas increasing the efficiency of product solutions.

Based on the market prospects, an analysis of the product portfolio investigates whether (1) current products in the portfolio can satisfy the needs, (2) new product ideas may find their customers in future, (3) capability enhancements of existing products find their business case, (4) improved product efficiency would open the market to new customer groups, or (5) new products need to be invented according to the business strategy of the enterprise. Depending on the outcome, the demand for creating new product concepts, altering existing product concepts, or maturing existing product concepts further may arise.

In the scope of system concept generation, the context has to be investigated in which a product should provide its benefits to the customers, users and any other stakeholders who will be concerned with the product in the utilization phase. This will lead to an understanding of the system environment including its functions and internal structure. Operational requirements are derived for the intended product in order to provide customers and their associated stakeholders with new or improved beneficial capabilities. Alternative product solutions are investigated, and evaluated regarding how effectively they can serve the intended missions. At this stage, searching for all kinds of risks that may adversely impact successful system concept implementation is more efficient than recording detailed designs that just reflect what is known already by the enterprise.

Considering the product concepts and their maturity, the product portfolio strategy is updated. The maturity of each element in the product portfolio is rated. For planned product elements, the envisioned entry into service dates are defined. Priorities are allocated to all the elements of the product portfolio.

With the updated product portfolio strategy, marketing plans can be established or re-iterated. Quantitative forecasts for future sales are the outcome. Probability ratings are included for individual products and markets assisting in setting priorities.

In parallel to market planning, technology roadmaps are established. Inputs for technology roadmap planning are the product portfolio strategy and the technology catalogues associated to all the elements of the planned product portfolio. Technology development projects are defined and planned. Preferably, a particular technology development project should be of benefit for several elements of the planned product portfolio.

For each technology project, criteria are defined to be satisfied for fulfilling the technology readiness levels with respect to the designated projects. However, this may be a cumbersome process. With every iteration of performing the business planning process, product concepts may be updated and the corresponding technology catalogues may be altered. For this reason, it may be more reasonable to define the technology project milestone criteria somewhat independent from the technology readiness level criteria applicable to each product the technology is designated to. To ensure that technology projects do not deviate from what is required by the corresponding products, running technology development projects need to be reviewed regarding the fulfilment criteria for passing project milestones and for progressing through technology readiness levels.

With the information from the marketing plans and the technology roadmap planning, now decisions can be made for which purposes the enterprise's budgets shall be spent. This includes (1) financing marketing and sales campaigns, (2) investments into technology development projects, and (3) budgets for product development, manufacturing, and product delivery and support.

The business planning process should be performed iteratively. For markets with long product life cycles, one iteration per year may be sufficient. But for fast developing and changing markets a higher iteration rate with several iterations per year will be beneficial. That does not necessarily mean that all activities run with the same iteration rate. For production related planning activities on volatile markets, short term adjustments may be necessary while the development of the product portfolio strategy runs at a slower pace.

5 CONCLUSIONS

The architecting of systems comprises a part of systems engineering emphasising the value adding creative aspects that are especially important rather early in the system life cycle. Systems architecting is in opposition to other schools of thought that foster more the deductive approaches of systems engineering. This paper develops an integrative view. Research results from cognitive psychology allow the conclusion that human brain power is best exploited when a system is defined in terms of system requirements,

functions and architectural decomposition by a single integrated multi-disciplinary engineering team. Together these complementary views are best suited to provide a consistent and comprehensive system definition. These views have to be generated for all systems and system elements in the systems architecture. Thus, any claims for exclusive allocation of inductive and deductive approaches to specific levels in the system architecture or particular stages in the system life cycle are denied.

The systems engineering value stream provides an integrated process view how to perform systems engineering effectively and efficiently. System architecture frameworks and model-based systems engineering are valuable assets of the systems engineering tool box, but their integration into the systems engineering value stream is still faced with some serious challenges. A few are explicitly considered in this paper. (1) How it may be avoided that the many views defined in architecture frameworks are generated and maintained isolated from each other? (2) How binding are detailed models of a system for the definition of the system elements, e.g. segregating mandatory parts of models from parts that are allowed to be re-defined? (2) How may virtual product integration be enabled merging detailed models from lower levels of the system architecture into models describing higher level systems?

The systems engineering value stream is defined in order to enable the concurrent execution of iterations with strict and continuous control of the evolution of consistent high-quality configuration baselines. It is concerned with how all the engineering teams in charge of individual systems or system elements are working together. System life cycle phase models are concerned with all the enterprises interfering with a system over its systems life cycle. This is a rather different view from the system engineering value stream. A generic business planning cycle is proposed describing how systems architecting in the earliest stages of the system life cycle may be effectively integrated with the other enterprise processes.

REFERENCES

[1] Capgemini Consulting. 2011. Integrated Business Planning – Steering Towards Profit. White Paper. (http://www.capgemini-consulting.com/sites/default/files/resource/pdf/ Integrated_Business_Planning.pdf)

[2] EC (European Community). 2001. 2001/95/EC. *Directive 2001/95/EC of the European Parliament and of the Council of 3 December 2001 on General Product Safety*.

[3] EEC (European Economic Community). 1985. 85/374/EEC. *Council Directive of 25 July 1985 on the approximation of the laws, regulations and administrative provisions of the Member States concerning liability for defective products*.

[4] V. Förster, H., and Pörksen, B. 2011. *Wahrheit ist die Erfindung eines Lügners – Gespräche für Skeptiker*. Neunte Auflage. Heidelberg (DE): Carl-Auer-Systeme Verlag.

[5] 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.

[6] 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.

[7] Friedenthal, S. A., and C. Kobryn 2004. „Extending UML to Support a Systems Modeling Language." Paper presented at the 14th INCOSE International Symposium, Toulouse (FR): 20-24 June.

[8] Goldstein, E. B. 2010. *Sensation and Perception*. 8th International Edition. Belmont, CA (US): Wadsworth.

[9] Goldstein, E. B. 2014. *Cognitive Psychology – Connecting Mind, Research, and Everyday Experience*. 4th Edition. Stamford, CT (US): Cengage Learning.

[10] ISO (International Organisation for Standardisation). 1994. EN ISO 9001. *Quality Management Systems – Requirements*.

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

[12] ISO and IEC (International Organisation for Standardisation and International Electrotechnical Commission). 2010. ISO/IEC TR 24748-1. *Systems and Software Engineering – Life Cycle Management - Guide for Life Cycle Management*.

[13] Kahnemann, D. 2012. *Thinking, Fast and Slow*. New York NY, 2011: Farrar, Straus and Giroux.

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

[15] Kotler, P., and K. L. Keller 2012. *Marketing Management.* 14th Edition, Global Edition. Harlow (UK): Pearson Education Ltd.

[16] Koschorke, A. 2012. *Wahrheit und Erfindung – Grundzüge einer allgemeinen Erzähltheorie*. München (DE): S. Fischer Verlag.

[17] Kuhn, T. S. 2012. *The Structure of Scientific Revolutions*. 50th Anniversary Edition. Chicago IL (US): The University of Chicago Press.

[18] Larman, C., and V. R. Basili 2003. “Iterative and Incremental Development: A Brief History.” IEEE Computer (June 2003): 2-11.

[19] Luzeaux, D. 2010. “System of Systems: From Concept to Actual Development.” In *Systems of Systems*, edited by Luzeaux, D., and Ruault, J.-R. London (UK): ISTE Ltd.

[20] Maier, M. W. and E. Rechtin 2009. *The Art of Systems Architecting*. 3rd Edition. Boca Raton, FL (US): CRC Press.

[21] NATO (North Atlantic Treaty Organisation). 2010. *NATO Architecture Framework, Version 3.1*.

[22] PricewaterhouseCoopers. 2012: Integrated Business Planning. White Paper. (http://www.pwc.com.au/consulting/assets/publications/Integrated-Business-Planning-Oct12.pdf)

[23] Reinertsen, D. G. 2009. *The Principles of Product Development Flow: Second Generation Lean Product Development*. Redondo Beach, CA (US): Celeritas Publishing.

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

[25] Scheithauer, D., and K. Forsberg 2013. "V-Model Views." Paper presented at the 23nd INCOSE

International Symposium, Philadelphia PA (US): 24-27 June.

[26] Scheithauer, D. 2014. "The Role of Systems Engineering in Business Planning." Paper presented at the 24th INCOSE International Symposium, Las Vegas (NV): 30 June - 3 July.

[27] Scheithauer, D. 2014. "Systems Engineering Value Stream Modelling." Paper presented at the EMEA Systems Engineering Conference, Cape Town (ZA): 27-30 October.

[28] Vincenti, W. G. 1990. *What Engineers Know and How They Know It: Analytical Studies from Aeronautical History*. Baltimore, MD (US), London (UK): The John Hopkins University Press.