32nd Annual INCOSE international symposium
hybrid event
Detroit, MI, USA
June 25 - 30, 2022

A 4-Box Development Model for Complex Systems Engineering

Erik Herzog, Åsa Nordling Larsson, Olof Sundin, Anna Forsgren Goman
Saab Aeronautics
{erik.herzog, asa.nordling-larsson, olof.sundin, anna.goman}@saabgroup.com

Abstract. Over the years, much Systems Engineering effort has been focused on making development activities predictable. Yet, with increasing system complexity, methodology improvements have a hard time keeping up. In this paper, we argue that there is a need to ensure that development methodology is flexible and that development models must be crafted for ensuring that an organization has many options open if parts of the development activities becomes delayed.

This paper introduces a development model that provides the desired level of flexibility. The underlying case is that of fighter aircraft development. The dual-Vee model is used as a baseline, weaknesses in that model are identified and the models developed within Saab Aeronautics for fighter aircraft development is introduced and illustrated. The key element highlighted in the paper is that in order to ensure flexibility, development activities has to be asynchronous with activities integrating product configurations. Lessons learned are identified and the applicability of the proposed models are discussed.

Introduction

People involved in the development of a system or set of systems need common terms of reference for enabling communication with regard to development strategy and planning, over what activities have been performed and which are still outstanding. Such terms of reference can be labeled, e.g., Systems Engineering Management Plan, SEMP. Multiple representations have been proposed, e.g., waterfall (Royce, 1970), Vee (Forsberg and Mooz 1991), Spiral (Boehm 1986), Dual Vee (Mooz and Forsberg, 2006), Wedge (Halligan, 2007) and recently there have been a push for what may collectively be called agile approaches, e.g., SAFE (Leffingwell et al, 2017). ISO 15288 (ISO 15288, 2015) is especially interesting as it defines necessary activities, but, intentionally, provide little guidance on whether the activities are to be performed in sequence or in parallel.

When considering such models we have to keep George Box's (Box and Draper 1986) insight in mind "... all models are approximations. Essentially, all models are wrong, but some are useful. However, the approximate nature of the model must always be borne in mind....". Hence, a good model capturing the nature of how development is planned and executed must provide sufficient explanation on how development is performed within an enterprise and with sufficiently well defined semantics such that ambiguity is kept at a minimum. In addition, any user of a model must understand the consequences of the approximations made – and how to make adaptations such that the model can be applied to the real world problems at hand.

Today, incremental or continuous development is the norm in many industries. In the aerospace industry this is evident in the 'block' capability levels defined for US and European military aircraft. A block level defines a customer-oriented capability certified for operational use. Increments are also

evident in the internal development progress within an organization. For instance, in the development of the JAS-39 Gripen fighter aircraft at Saab Aeronautics many intermediate development editions are produced for each customer block level.

This paper outlines problem areas with the traditional Vee model as identified within Saab Aeronautics, and illustrates how the organization has developed an alternative set of models that address the identified weaknesses. The paper presents the models used within the organization along with examples on how they are applied and discusses strengths and weaknesses of the approach.

It shall be noted that the models presented in this paper are tailored for the development of safety critical systems where conformance against airworthiness requirements must be declared prior to flight and where flight-testing is an expensive activity. Hence the proposed solution may not be adequate for other product classes.

The rest of this paper is outlined as follows. First a review of the traditional Vee models is presented, followed by a list of perceived weaknesses for complex systems development, e.g., fighter aircraft development. Then the development models applied within Saab Aeronautics are introduced and exemplified. The paper is concluded with a discussion on the approach, its strengths and weaknesses as well as lessons learned.

Some views on development

When creating a model capturing the activities applied for the development of a complex system we have to keep in mind that all activities described are artificial constructs. Any model created will include abstractions and definitions that may or may not coincide with the actual activities performed by a development team.

The Vee model has a very strong heritage within INCOSE. It exists in multiple variants as well as extensions. It is beyond the scope of this paper to capture all variants of the Vee and provide a complete reference section – so it is appropriate to include an apology for all relevant references not included below.

Forsberg and Mooz have been very influential in introducing and extending the Vee model over multiple INCOSE Symposia. From the initial paper at the first NCOSE symposia in 1991 (Forsberg and Mooz 1991) they are very clear in that there are two distinct dimensions to the Vee:

- The product structure oriented view capturing the evolution of a system and its subsystems (the Architecture Vee). Typically, the Architecture Vee does not include an explicit temporal dimension.

- The internal system element development process view (the Entity Vee). When introducing technical reviews the Entity Vee include a distinct temporal dimension.

This separation and integration of the views described with elegance in the Dual Vee model (Mooz and Forsberg 2006).

Below, Figures 1-3 are taken from the Dual Vee paper to illustrate the different focuses of the Architecture and Entity views and their interaction.

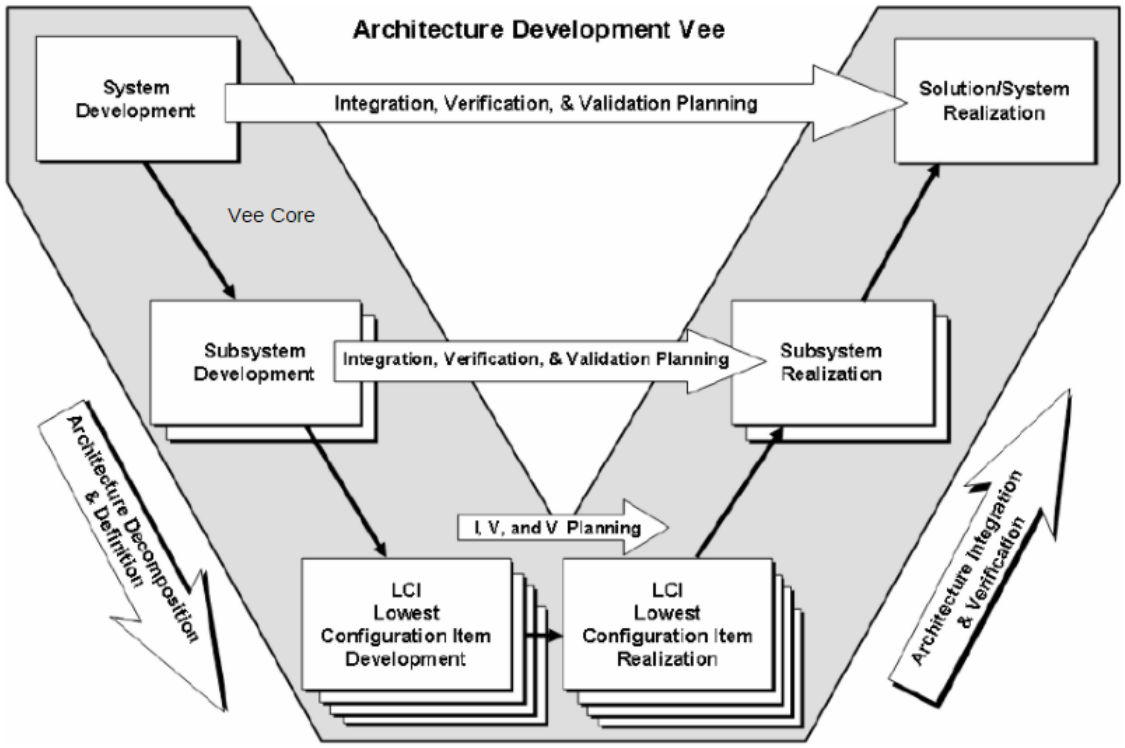

Figure 1: Dual Vee – architecture view[1]

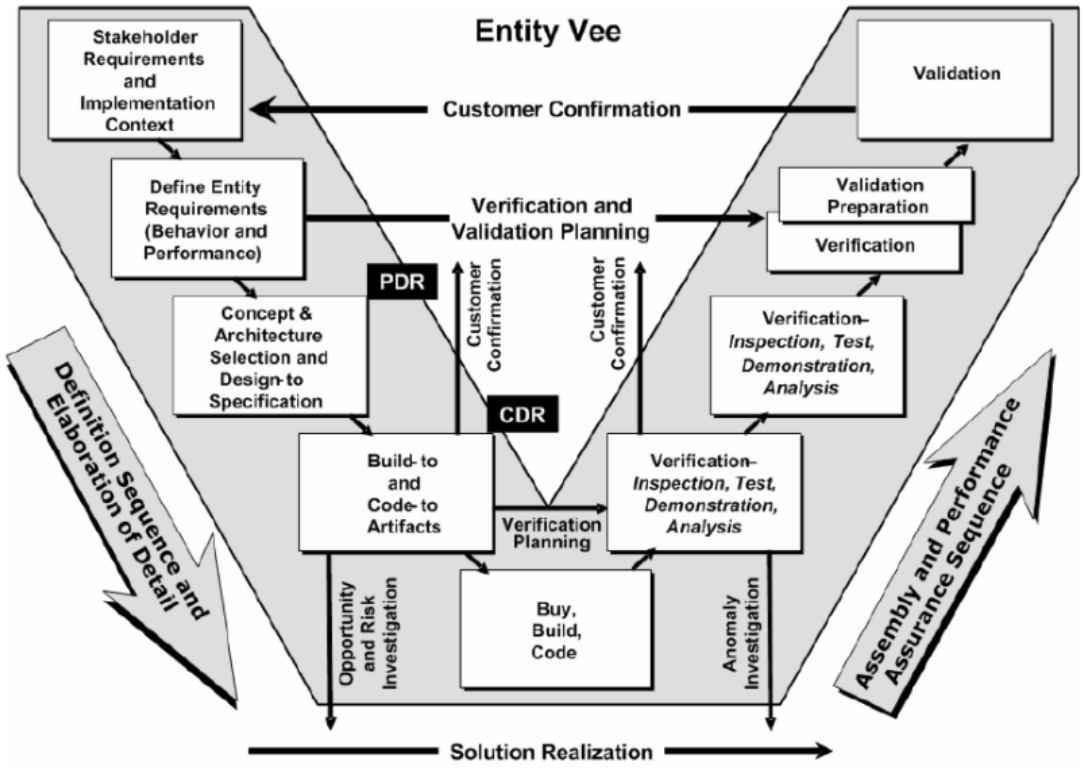

Figure 2: Dual Vee, entity view[1]

In Figure 2, note the inclusion of technical reviews as a part of the entity view of the Vee to indicate suitable interaction points with external stakeholders.

[1] Figures 1 &2 are copied from (Mooz, H. and Forsberg, K. (2006))

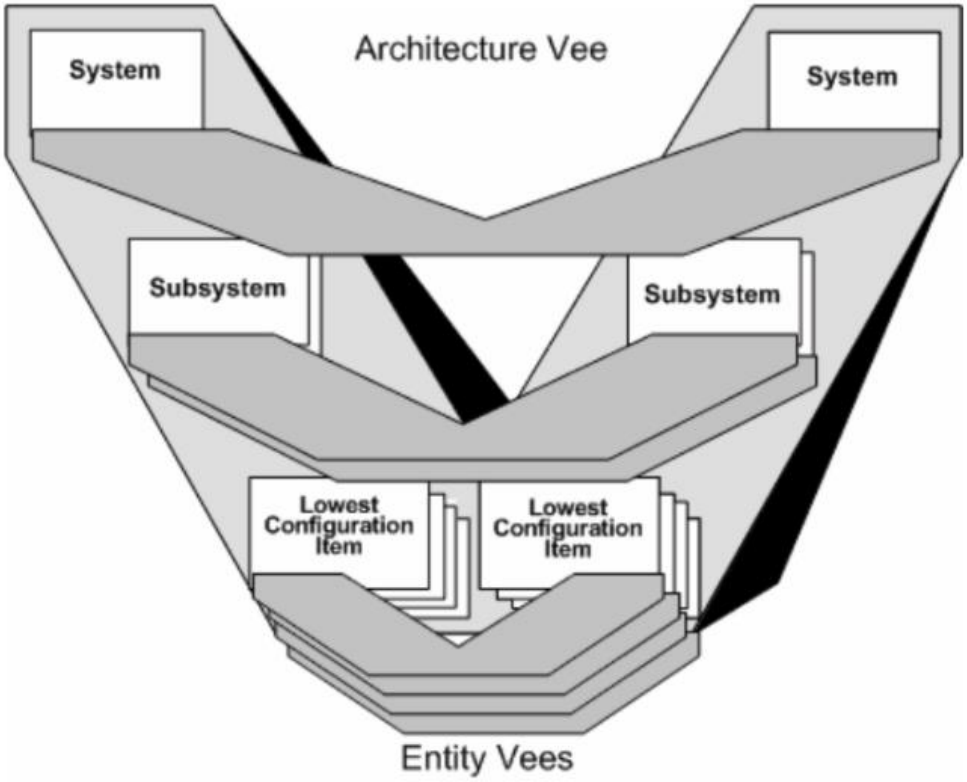

Figure 3: Combining the architecture and entity Vees[2]

When combining the Architecture and Entity Vee (see Figure 3) a comprehensive view of a systems structure and its associated development activities is created.

Forsberg and Mooz also provide examples on how the entity Vee model can be adapted to incremental development (Forsberg and Mooz 1995). The Figures 4 and 5 below capture the application of the Vee model in incremental development scenarios. Note the introduction of technical reviews within the incremental Vees to illustrate the importance of stakeholder interaction.

[2] Figure 3 is copied from (Mooz, H. and Forsberg, K. (2006))

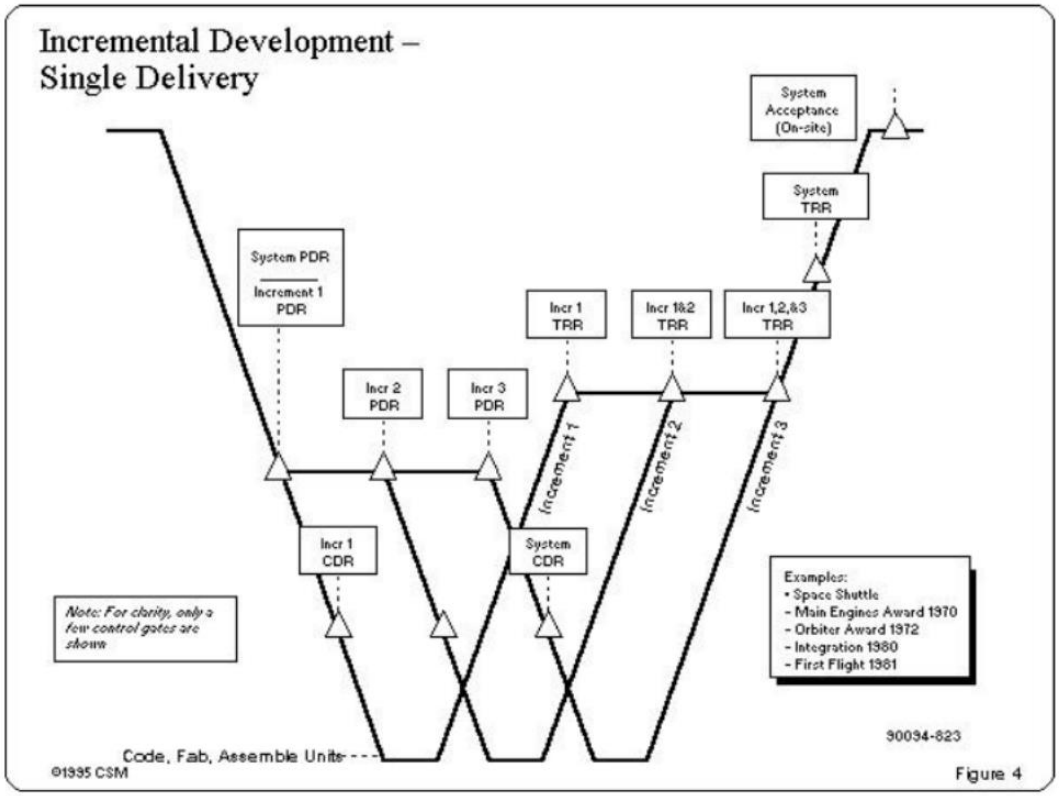

Figure 4: Entity Vee model, incremental development – single delivery[3]

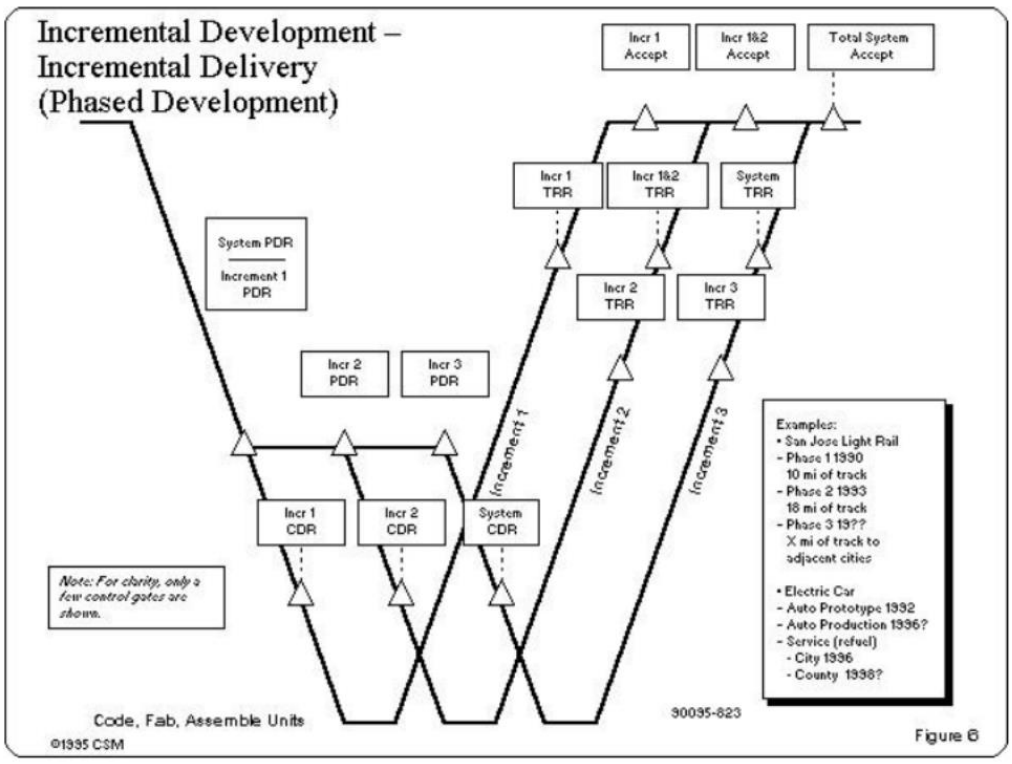

Figure 5: Entity Vee model, incremental development – incremental delivery[3]

[3] Figures 4 and 5 are copied from (Forsberg, K. and Mooz, H. (1991))

Vee shortcomings

Despite its elegance and the fact that it has aged well over the years since first published, the Dual Vee model is not a perfect match for systems development as performed by Saab Aeronautics. The items below describe the individual weaknesses that have led the organization to explore alternative models:

1. The Entity Vee has a strong focus on development phase, skipping earlier phases in the lifecycle. For the organization there was a need to clearly capture that many, if not all processes are initiated early in the lifecycle and the resulting work products are matured over time.

2. The Entity Vee provides guidance when activities end, but is weak in providing guidance on when activities area initiated. This is especially true for activities in early lifecycle phases. While the Entity view is logically correct, it is not a good tool for planning development activities.

3. With the advent of digital twins, virtual integration will occur in all system lifecycle phases and not only towards the end of development. For example, in Gripen E/F development, the development teams have access to numerous system wide simulators and test stations, ranging from software based simulators to complete "hardware in the loop" test rigs. This capability allows individual development teams to perform local integrations to ensure that the components under development will interface and interact correctly with the rest of the system. This raises the need for having a single integration process supporting virtual integration as well as the actual physical integration activities. In particular, we emphasize that procedurally, there is no distinction between physical and virtual integration. The process guidance shall be identical for virtual and physical integration.

4. The Architecture and Entity Vees can be interpreted such that the development activities started at a particular point in time will be integrated together, i.e. in the same Vee cycle. Initiating, say, four development activities, capability or component oriented at a point in time would imply that they are eventually integrated to form a new updated system configuration. This constraint is not problematic if an organization can reliably and consistently predict how long the development of a particular capability or components will take. With such a capability an organization would group and initiate development activities that take approximately the same time to realize and the overall process will be predictable.

   Within our home organization we pride ourselves for our professionalism. Despite this, delays and hiccups are commonplace. In fact we agree with the 3rd fallacy 'Our development plan is great; we just need to stick to it' in 6 Myths of Product Development (Thomke and Reinertsen 2012)[4]. Development of complex systems is notoriously difficult. There is a need for a model that clearly captures that development activities are performed continuously and allow realized components to be integrated as they become available. In other words, development is asynchronous to integration, verification and validation of realized product configurations (be they physical or logical). The integration function must be given the tools that allows it to quickly select what to include in a new configuration given the set of development artefacts (new or already in existence) available at a given point in time.

[4] In the paper Thomke and Reinertsen argues very convincingly that it is not possible to predictably foresee a long term in advance when individual teams will complete their development activities.

A basic development process – and its limitations

This section presents the process model resulting from analyzing weaknesses 1-3 as identified in the previous section. The process corresponds to the Entity Vee but extends to the lifecycle. Processes (largely compliant with ISO 15288:2008) are placed in parallel and over the lifecycle to illustrate that during large parts of the lifecycle, processes will interact and information will mature towards the final information set. Technical reviews are included, just as in the original Entity Vee view, to illustrate desired system maturity and stakeholder interaction. The set of reviews identified are identical to those defined in IEEE 15288:2 (IEEE 15288.2 2014) with the addition of:

- SPR (System Planning Review) held at the start of each lifecycle phase with the purpose of performing process tailoring such that each lifecycle activity is provided with adequate process support. A central element of the SPR is that the development team identifies the deliverables appropriate for supporting the coming activities, the technical reviews to be performed and most importantly the planned maturity of each deliverable at each review.
- StRR (Stakeholder Requirement Review) held in the Concept phase with the purpose of confirming that the stakeholder requirement set is complete and that stakeholder requirement conflicts and inconsistencies have been identified.

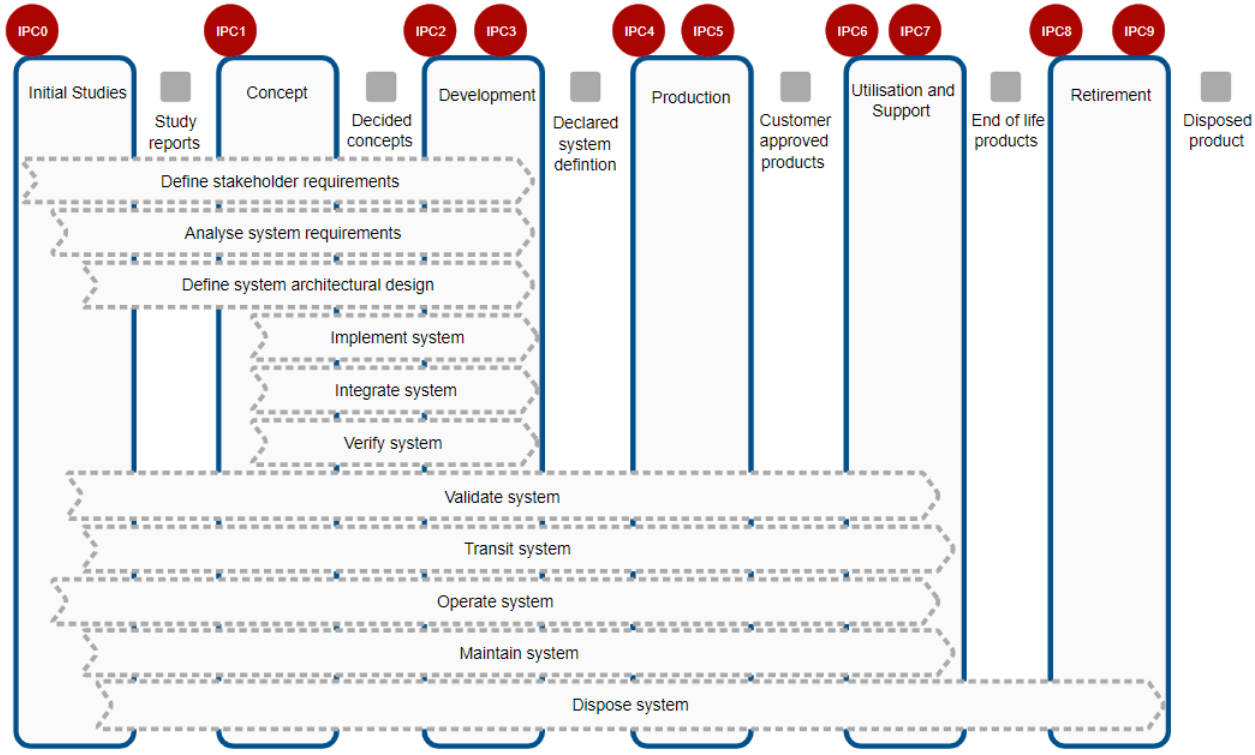

Figure 6: Process model overview

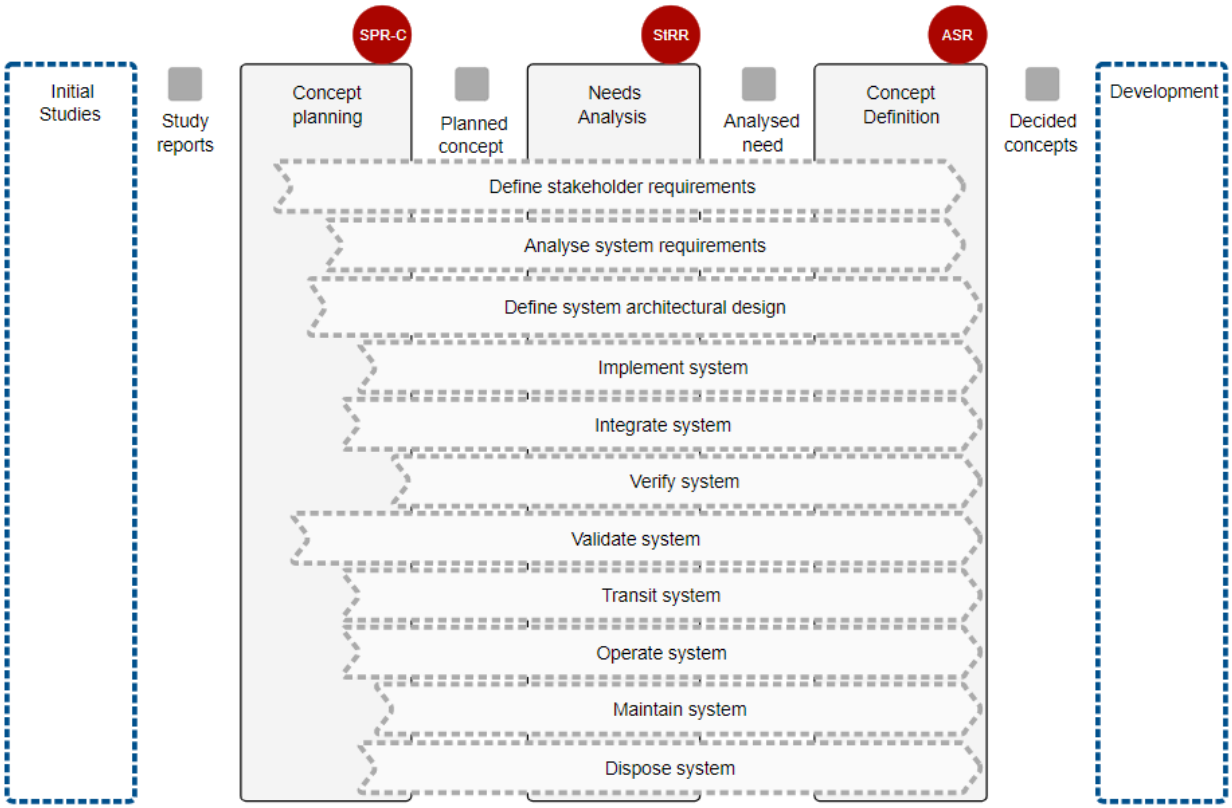

Figure 7: Process model – Concept phase

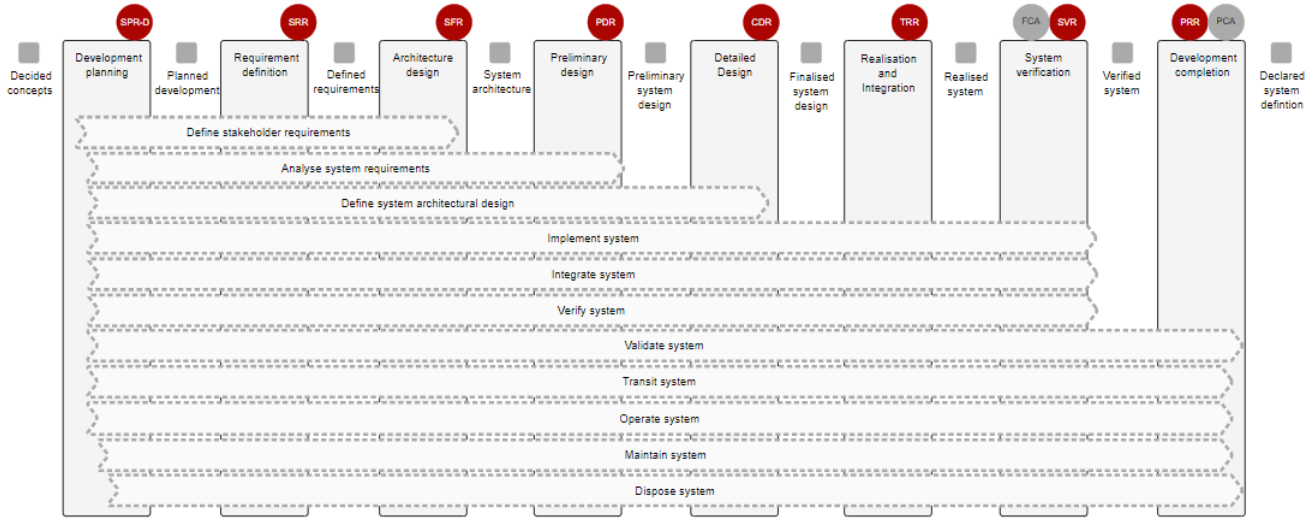

Figure 8: Process model – Development phase

We argue that the process view presented above addresses Vee shortcomings 1-3 as identified above as it:

- Provides a lifecycle view, explicitly including initial studies and concept phases with indication when activities are initiated early in the lifecycle and where information maturate over it.
- With the parallel process outline it is also clear that the same integration process can be used at any stage in development, be it for virtual or physical integration.

Overall, the model presented has proven itself well in operation. It conveys intuitively that activities in all processes will initiate early in the lifecycle and that the information created by each process will

mature over time. However, the model still conveys the impression that all development activities initiated at a particular point in time will terminate together, in the same Vee cycle. Hence further extensions are required.

The 4-box development model

This section introduces a development model that captures that development activities are asynchronous to integration. For the lack of better wording the model is simply called the 4-box development model. Before outlining the model a little additional background is provided.

The Gripen E/F weapon system is being developed using a Product Family approach. This means that instead of focusing directly on the development on customer specific solutions the focus is on seeking a common system core available in all customer configurations – *datum*. Optional capabilities in the weapons system are identified to enable configuring customer specific variants. Existing and future customers all share the *datum* realization, but are free to select from identified options. In a sense, the approach is similar to that of configuring a new car, except that the catalogue of options is smaller – and not public.

The Product Family approach depends on the existence of a Main Track, a warehouse that contains all versions of development artefacts (e.g., Subsystems, CSCI, HWCI and configuration information) that are available for assembling a Product Configuration.

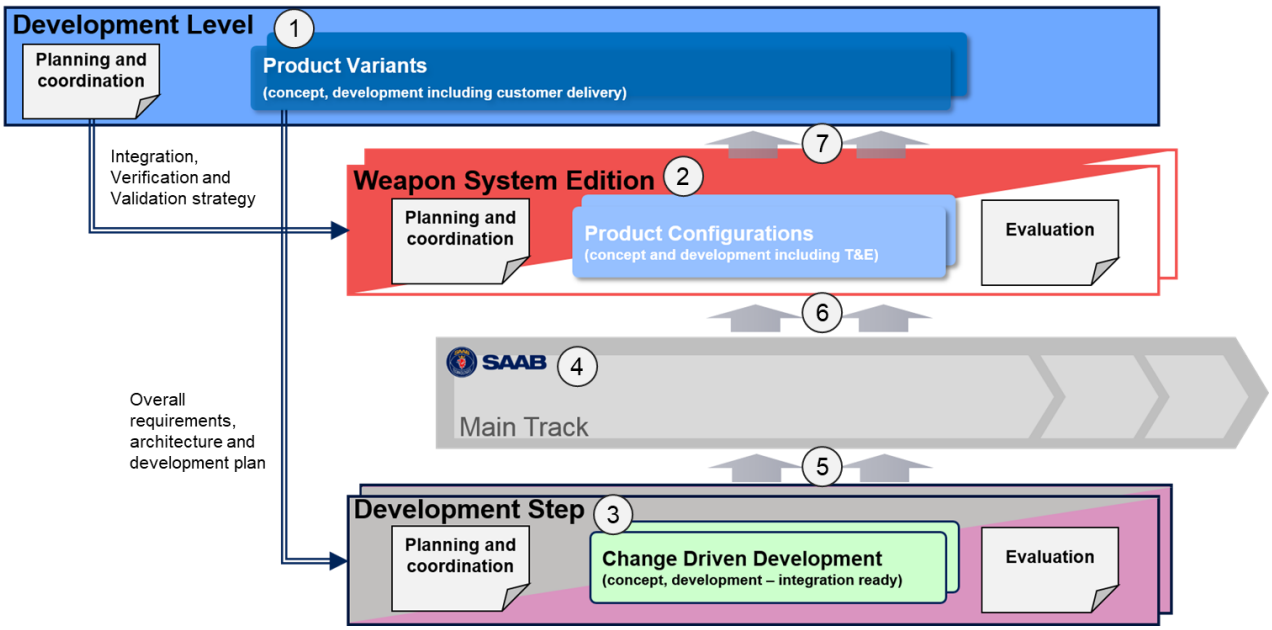

Figure 9: The 4-box development model

The 4-box development model defines three distinct views on development, see Figure 9. Note, that the Main Track is just a passive warehouse – no engineering activities are performed within it. The views are largely independent, but managed through Configuration Management activities, i.e., the usage of Change Requests (CR). Each view is executing in accordance with the process defined in the previous section of this paper. The numbering of the paragraphs below relate to the circled numbers in Figure 9:

1. Development Level: Development levels represent a contractual obligation under which the organization delivers one or more weapons system configurations – Product variants. This is the long term view including initial needs and concept studies and capturing what will eventually be delivered. Technical reviews are performed with focus on the final capability level. Requirements analysis and architecting is performed for each Product variant and the

overall Integration, Verification and Validation strategy is set. The Development level also identifies groups of capabilities for realization by the development teams as part of Development Steps. Additionally, the Development level serves as the commercial interface towards customers and external authorities. Consecutive Development Levels may be executed to incrementally deliver more advanced Product Variants. The main internal roles acting within the Development Level are Product Managers and Architects.

2. Weapon System Edition: Weapon System Editions are used to establish a set of aircraft or weapon system configurations by integrating developed configuration items (from the Main track) in accordance with the overarching Integration, Verification and Validation plan. Weapon System Editions define well-defined capability levels. This allows the integration organization the freedom to select a feasible integration sequence meeting the objectives of the Weapon System Edition. The integration organization have to ensure, by the execution of the technical reviews defined for the basic process that every product configuration created for flight testing is airworthy. A lessons learned and evaluation activity is performed for each Product Configuration created and evaluated. Development managers and Chief engineers are some of the main roles acting within the Weapon System Edition.

3. Development Step: The Development Steps are a defined set of capabilities in the Development Plan. Within the Development Step, the coarse development plan is broken down into small scale tasks suitable for incremental development. Hence, the Development Step captures what will be realized in the near future. Defined changes to the existing baseline are released for Change Driven Development to development teams. A development team is only allowed to deliver their realized system elements to the main track after successful integration testing in a complete system simulator. Enforcing this condition ensures high quality building blocks in the Main track. Technical reviews are executed to support quality and efficiency in each development activity. A lessons learned and evaluation activity is performed at the completion of each Change Driven Development activity. Individual development teams may adopt traditional or agile approaches based on the nature and magnitude of the development challenge. The individual development teams are the main actors within the Development Step.

4. The Main Track is, as stated above, the warehouse containing all versions and variants of all configuration items in the Gripen E/F system available for integration. Integration testing is performed prior to acceptance of items into the Main Track.

5. Delivery from Change Driven Development activities have to meet integration testing quality gates for acceptance into the Main Track. These activities are managed by the Integration Management team. The integration testing activity is performed in software based simulators and hardware in the loop test rigs allowing for testing the complete integrated system.

6. Selecting what to integrate from the main track is bound by the overall integration plan, and the current, in development, validation needs. A feature model, the Gripen Variant Master, is used to assist in the definition of valid product configurations. The Integration Management team is leading this this activity.

7. Weapon System Editions will deliver product configurations whose capabilities over time will converge towards agreed stakeholder requirements. Verification and Validation evidence collected from individual product configurations are used to build up for type certification process. Ultimately this fall under the responsibility of the Head of Development and Head of Airworthiness.

Effective communication between stakeholders is the key for successful application of the 4-box model. This is to ensure that changes to requirements, architecture or the emergence of unexpected integration problems are analyzed and resolved quickly.

Integration anatomies

This section outlines the use of integration anatomies (Taxén and Pettersson 2010, Taxén 2011) for understanding the relationships between the outcomes of development activities. An integration anatomy is a graph that captures how planned individual development activities build up to a complete functionality, see Figure 10. The leaves of the graph captures activities that place the foundation of the functionality, and the arrows indicate causal order. For instance, activities A and B, must complete prior the completion of C. As can be seen, an integration anatomy captures all potential integration paths.

It should be noted that the order captured in an anatomy need not coincide with the defined architecture. The related activities may affect different system elements.

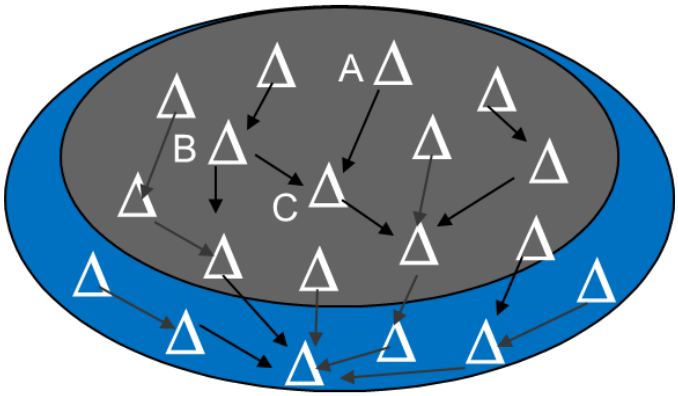

Figure 10: A sample anatomy

Successful implementation of the 4-box model requires an integration driven development approach. Keeping the integration aspects central (as opposed to e.g. the sequence of engineering activities) enables mapping of functions and capabilities to the different views of the 4-box model. In addition, when developing complex systems, frequent re-planning and reacting to unexpected events is inevitable. Keeping track of dependencies and striving for loosely coupled development efforts help mitigate that. Thus, integration planning in general and creating and maintaining (integration) anatomies in particular is crucial.

Anatomies come in different flavors depending on purpose, level of details and development phase. Three examples of anatomies on different levels are:

1. *Functional anatomies* outlining a specific function, sub-function or subsystem. These anatomies are used for validation and communication with customers and other stakeholders at the Development level. They are also useful for identifying external dependencies at a higher level and thus determining what needs to be in place (or mitigated) when integrating a particular function into the system. Two examples could be a high-level anatomy for the entire Electronic Warfare subsystem or a more detailed one for the Crash Survivable Recording function.
2. *Integration anatomies* that visualize the intended functionality planned for integration in a certain Weapon System Edition. Specifically, the integration anatomies help identify dependencies such as components that need to be integrated before other components and secondary effects of delayed or canceled integration of a certain component. For example, an

integration anatomy could reveal that an HMI interface must be integrated before a certain Electronic Warfare component can be integrated.

3. *Low level integration anatomies and integration plans* used for the short term detailed planning of software integration into the Main Track. These are necessary for detailed integration planning, not least in order to keep the Main Track clean and prevent broken or faulty software builds. For example, an integration plan might show that the Electronic Warfare component mentioned above is planned for integration three weeks before the HMI interface on which it depends, which would obviously be a problem.

Figure 11 illustrates the above anatomies and how they relate to the 4-box model.

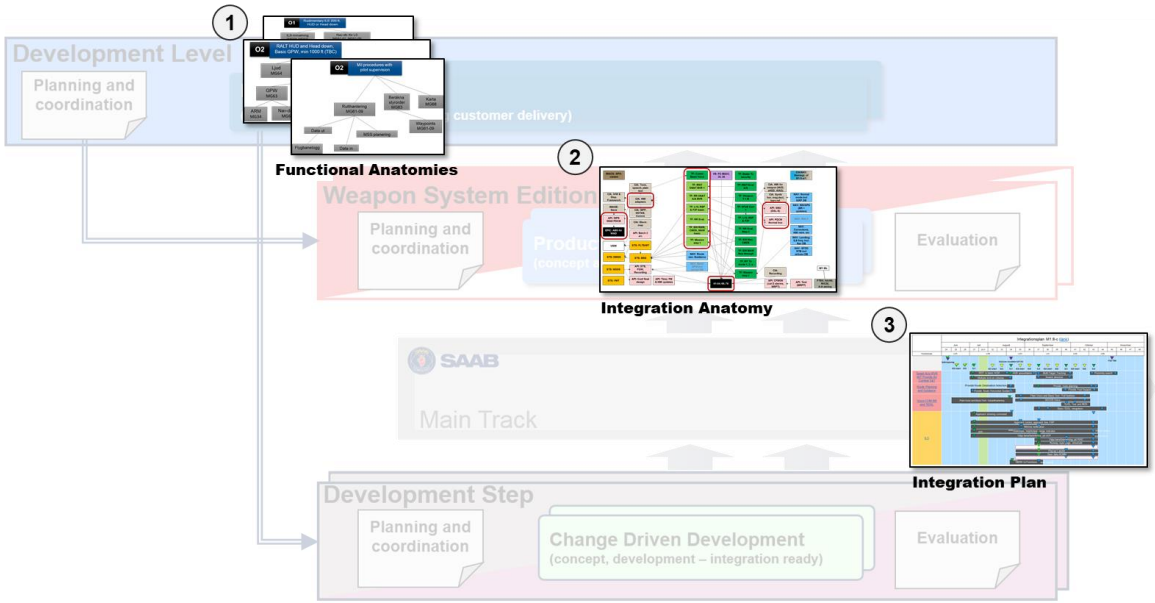

Figure 11: Different kinds of anatomies and how they relate to the 4-box model

Anatomies are also useful for illustrating and planning the systems engineering (integration) workflow, i.e. on a time scale rather than level of detail. For instance, the functional anatomies are typically created early in the systems engineering process. Since they outline a specific function (or subsystem) and its immediate dependencies, they can be created independently and early even if surrounding functions are not yet defined. They are then typically refined and modified as development progresses (or alternatively abandoned and replaces with other anatomies). For example, a functional anatomy for the Electronic Warfare subsystem may show that there is a dependency to an HMI component, but without specifying any details on the nature of that dependency.

Also, as integration into a Weapon System Edition is planned, an integration anatomy can be produced in order to give a bird's-eye view of the edition at hand. A Weapon System Edition may carelessly be defined as a bundle of (more or less finished) functions that happen to be mature enough for integration and verification during a specific time period. The integration anatomy captures integration relationships and constraints. The aforementioned functional anatomies are used as input when creating an integration anatomy. For example, an early integration anatomy may indicate that too much is planned for integration and/or that there are so many dependencies that the planned integration period will likely be too short.

Finally, the detailed, day to day integration into the software Main Track is visualized in a low level integration anatomy and/or an integration plan (which is in fact basically an integration anatomy where the different elements are also laid out in time). This plan may capture detail down to individual days. For example, that the Electronic Warfare s/w team are planning to integrate a component next Tuesday. Figure 12 illustrates how the different kinds of anatomies fit in an integration work-

flow, from the Systems Engineering work on individual functions/subsystems to the software integration in the Main Track.

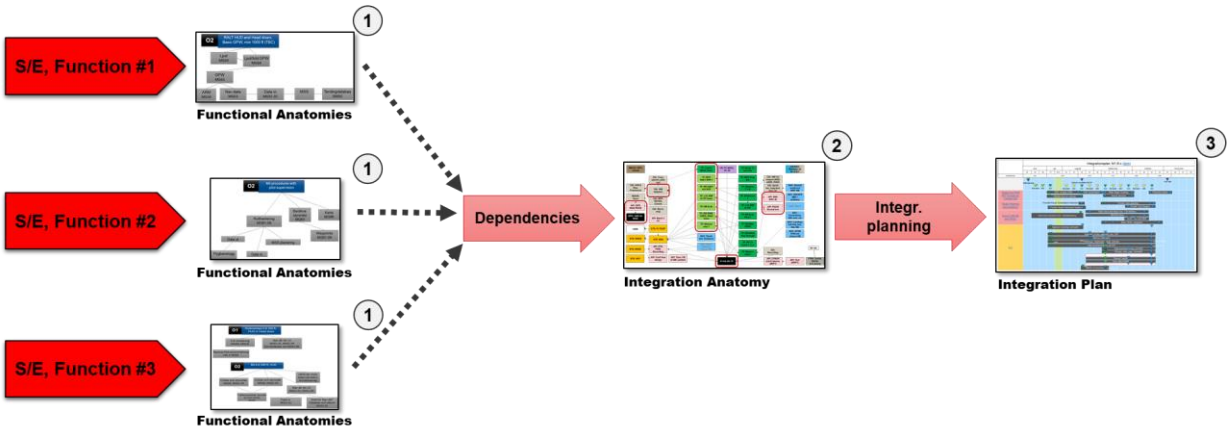

Figure 12: Different kinds of anatomies and how they fit in an integration workflow

Discussion

In this paper we have argued that the traditional Vee model is too rigid to be used as a model for explaining development of military aircraft within Saab Aeronautics. The process view over the lifecycle and 4-box model presented in previous sections of this paper offers a better match to the way the development planning and integration is performed within the organization in that:

- It provides a view where processes are outlined clearly over the lifecycle, providing a clear planning template for development activities.
- It emphasizes that processes are executed in parallel and there is no need to make any distinction of activities performed on the left side and on the right side of the Vee. This is particularly important for integration of virtual products for use in simulators and test rigs.
- The 4-box model also clearly decouples component/capability development activities from integration activities. This enables integration teams to select what is available for integration at any point in time, regardless of the state of development activities. Moreover, the Development Level component in the 4-box model provides an explanation for how incremental development can be combined with a traditional entity Vee as a front-end towards external stakeholders.

Reconciling the iterative approach to development in the 4-box model with traditional program level technical reviews poses another challenge in the dialogue with external stakeholders. A Critical Design Review (CDR) using traditional definitions will focus on reviewing the soundness of the ‘paper’ product. When the 4-box model is applied, substantial parts of the product will already be realized at the time of a CDR, while some other parts might not yet be fully specified. The question is when to schedule the review in order to maximize stakeholder value and minimize risk, see Figure 13. If scheduled too early the review will provide insufficient insight into the final product and if scheduled too late it will result in an inability to address stakeholder feedback (as the individual function that prompted the feedback may already be realized). The current approach to manage this dilemma is to schedule periodic reviews for a Product Variant and focus on the progress and status at relevant capabilities under development.

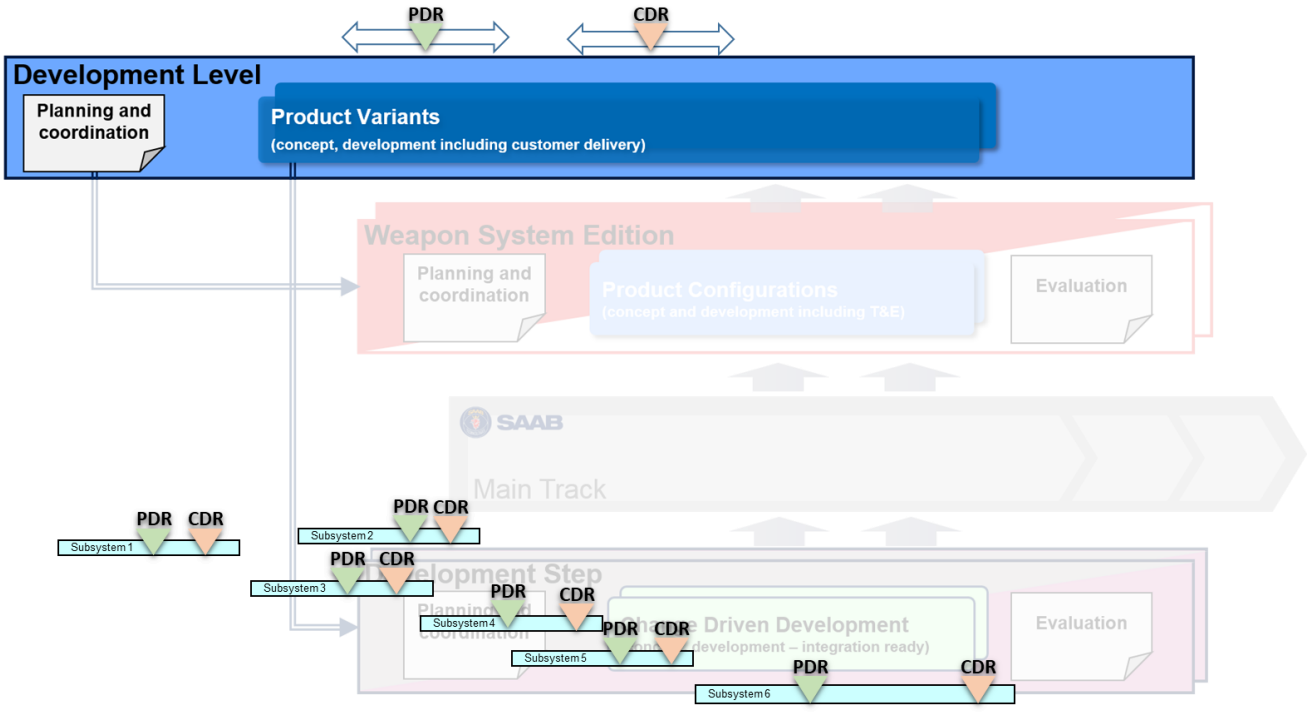

Figure 13: Mapping Development Level reviews to incremental development progress

Another observation is that the 4-box model approach is most efficient when there is loose coupling between individual development activities. There is little value in decoupling development and integration activities if all planned changes are tightly coupled. Hence, when in such a situation it is wiser to revert to the traditional Vee-model or in the context of Saab Aeronautics to execute the standard development process – without the 4-box model extensions.

At the time of writing this paper the 4-box model has been used for about 5 years within Saab Aeronautics. Overall, we are satisfied with the expressive power it provides and its ability to capture how the organization desires to perform Systems Engineering. Objectively, it has not changed the structure or roles in the organization. But, it has provided us with the tools allowing us to explain how we desire to operate, clarified roles, responsibilities and communication patterns. With the 4-box model we can explain with clarity, internally and externally, how the individual parts of the organization need to interact. The only ‘drawback’ is that it highlights the complexity of fighter aircraft development. We are unable to provide an oversimplified “tell all” model that works in all situations. After implementing the 4-box model we have realized that there is a lot of overlap in terms of objectives and structure with the extended agile frameworks, e.g., the Scaled Agile Framework (SAFe) (Leffingwell et al., 2017). We will investigate the nature of the overlap in future work. However, the 4-box model allowed the organization to enhance existing practice using existing principles and terminology without the need to take on an entirely new framework.

That said, the models presented herein do not completely replace the Architecture Vee model. It is still in use for explaining the fact that systems are architected and then eventually integrated, verified and validated. Moreover, the presented 4-box model admittedly lacks the simplicity and elegance of the traditional Vee model.

Conclusions

In this paper we have presented the set of models used within Saab Aeronautics to guide and explain the development of fighter systems, such as the Gripen E/F. George Box famously stated ”All models

are wrong – but some are useful" (Box and Draper 1986). The models presented in this paper are no exception – they are wrong. There is no way that they can capture the details and peculiarities of complex systems development. However, we feel that they provide value in that they provide mechanisms for capturing:

- That development processes are performed in parallel over the lifecycle, while the maturity of the information increases. Hence, they provide the basis for development planning.

- Implicit support for reasoning on integration, verification and validation in identical terms regardless of whether such activities are performed using digital twins or on the actual realized system.

- That development activities are asynchronous to integration activities. This extension was created to allow for the fact that it is very difficult to predict the duration of individual development activities. Integration schedules are not dependent on the most delayed team. Anatomies are used to capture the relationship between individual development activities and help system integrators deduce the completed development artifacts that can be used to create a value adding integrated configuration.

The Future extensions to the presented models include a greater focus on change and configuration management to improve communication. Moreover, it will be used to support other product development programs beyond Gripen E/F.

References

Boehm, B., A Spiral Model of Software Development and Enhancement. ACM SIGSOFT Software Engineering Notes. 11 (4), 1986

Box G and Draper N., Empirical Model-Building and Response Surfaces, p. 424, Wiley–Blackwell, 1986

IEEE 15288.2, Technical Reviews and Audits on Defense Programs, IEEE 2014

ISO 15288 Systems and software engineering — System life cycle processes, ISO 2015

Forsberg, K. and Mooz, H. (1991), The Relationship of System Engineering to the Project Cycle. NCOSE International Symposium, 1: 57-65 https://doi.org/10.1002/j.2334-5837.1991.tb01484.x

Halligan R., Description of the Wedge Model, https://www.ppi-int.com/the-wedge-model/, visited October 27th, 2021

Leffingwell D, Yakyma A., Knaster R., Jemilo D. and Oren I.,SAFe Reference guide. Addison Wesley, 2017

Mooz, H. and Forsberg, K., The Dual Vee – Illuminating the Management of Complexity. INCOSE International Symposium, 16: 1368-1381 https://doi.org/10.1002/j.2334-5837.2006.tb02819.x

Royce W. Managing the Development of Large Software Systems. In Proceedings of IEEE WESCON 26, 1970

Taxén, Lars & Pettersson, Ulrik. (2010). Agile and Incremental Development of Large Systems. EuSEC 2010

Taxén, Lars & (Ed, L.. (2011). The System Anatomy – Enabling Agile Project Management. Studentlitteratur

Thomke S. and Reinertsen D., Six Myths of Product Development, Harward Business Review, May 2012

Biography

Erik Herzog is a Technical Fellow at Saab Aeronautics. Dr. Herzog received his Ph.D. at the Department of Computer and Information Sciences at Linköping University, Sweden. His professional interests include development and introduction of Systems engineering processes, specification methods, information modelling, tool integration techniques and change leadership. Within the INCOSE sphere, he has contributed to events management in areas like publication management, events management including invention of the Systems Engineering tour concept.

Åsa Nordling Larsson has worked professionally with Configuration Management and Product Data Management for more than 20 years and currently works in the Design Engineering Team of the Gripen E fighter program. Her special interests is Business development related to Configuration Management, Product Data Management and Systems Engineering. She has a bachelor's degree in Systems Analysis from Linköping University, Sweden

Olof Sundin is a Distinguished Engineer at Saab Aeronautics and currently works in the Engineering Management team of the Gripen E fighter program. With a background in software engineering, Olof has many years of experience in the development and integration of complex systems and frequently holds lectures and courses, primarily on the subject of Integration Driven Development.

Anna Forsgren Goman is a Principal Engineer at Saab Aeronautics and have worked with systems development and integration management at Saab for almost 20 years. She has a master's degree in Applied Physics and Electrical Engineering from Linköping University, Sweden. Her professional interests include Systems Engineering of Complex systems, Technical Leadership and Verification and Validation processes.