RESEARCH ARTICLE OPEN ACCESS

A MBSE Approach for the Development of Complex Technical Systems

Grischa Beier*, Asmus Figge*, Stephan Marwedel**
**(Fraunhofer-Institut für Produktionsanlagen und Konstruktionstechnik (IPK), Pascalstraße 8-9, 10587 Berlin, Germany, grischa.beier@gmx.net)*
***(Airbus Deutschland GmbH, Architecture and Integration, Kreetslag 10, 21129 Hamburg, Germany)*

ABSTRACT
Model–-based development techniques have been successfully introduced into the design process of aircraft systems. Examples of these techniques are performance models for digital systems and networks or 3D–models for equipment allocation and weight assessments. However, current systems development processes usually do not allow for a continuous usage of models reducing the potential benefits of a *Model-based Systems Engineering* (MBSE) approach such as an interdisciplinary, low–effort evaluation of different design alternatives. The paper presents a novel MBSE approach that focuses on the early development phases. It suggests a sequence of models to be developed, provides required activities as well as data formats and considers the complexity of modern mechatronic systems by distinguishing the required activities by the respective abstraction level. An aircraft systems example is used to highlight the main features of the approach.

Keywords - Design quality, Modeling, Simulation, Systems Engineering, Traceability

Date of Submission: 25-08-2017 Date of acceptance: 17-09-2017

I. INTRODUCTION

The development of complex mechatronic products requires the contribution of a multitude of development departments creating a number of information artifacts from different disciplines [1]. Most of these artifacts describe the to-be-developed product from different perspectives (overall requirements on the product, geometric parts, electronic devices etc.) and are elaborated in several different authoring tools. A comprehensive development approach has to address this diversity and support developers in efficiently creating a reliable product that meets the initial requirements. The aircraft manufacturer Airbus and the Fraunhofer institute IPK jointly elaborated a model-based development approach that considers the state of the art in product development and allows for the efficient development and verification of complex mechatronic products involving a multitude of third parties.

In this paper a novel Model–Based Systems Engineering (MBSE) approach is presented. It focuses on early development phases and allows for an efficient and consistent information generation as well as early evaluation of generated solutions. One of the main advantages of models is simplifying the real world's complexity so that different solution alternatives can be generated with comparatively little effort. Being more simplistic, these alternatives additionally can be interpreted more easily by experts from different disciplines allowing for a more holistic evaluation. The model-based characteristics of the presented approach therefore facilitate an involvement of experts with different backgrounds in early stages of the development. This early evaluation helps avoiding late and therefore expensive design changes [2], being a major difference in comparison to other development approaches such as presented in [3].

The paper starts with a brief state of the art analysis regarding already existing MBSE approaches. It continues with a stepwise introduction of the main processes that comprise the MBSE approach, each explained by a common product example. A concept on how to apply the described MBSE approach for the development of complex products is presented. The paper concludes with a summary and outlook.

II. EXISTING (MODEL-BASED) SYSTEMS ENGINEERING APPROACHES

The aim of this paper is to define a practical yet comprehensive approach to Systems Engineering applicable to civil aerospace systems. Although the focus is primarily on civil aerospace systems its principles should be applicable to other areas as well. The basic idea is to take the standard Systems Engineering (SE) practices of developing the functional, physical and operational architectures according to a V-model as the base. See e.g. [4] for a description of the general approach to SE. This general approach has been discussed and extended by the SE community intensively and a lot of system development standards have been derived from it, e.g. [3] and [5]. Furthermore, some organizations defined their own version of the application of SE principles in form of a handbook, see [6] and [7] as examples.

The general idea of the proposed approach to MBSE is to define a set of generic activities that are repeatedly executed at each level in the hierarchy of the system development process, similar to the Incremental and Iterative Development (IID) described in section 3.4.2 in [7]. Each activity is supposed to be supported by a set of models which are refined throughout the process. See Figure [1]1 for an overview of the process steps and iterations iterations for three hierarchical levels: Product, System, Subsystem.

Although the type of the model used in each step is dependent of the exact type of the system under design a generic structure for the models can be defined.

Some of the aforementioned standards and adaptations do mention the use of models for verification purposes. However, their use is not generally advocated throughout the complete development cycle. [7] does mention the use of models in section 4.3.2.6 and encourages the use of functional models based on the *Integrated Definition of Function Modeling* (IDEF) in section 4.12.2.2 and SysML in section 4.12.3.2 for the description of the architecture, but does not define executable models as an integral part of the overall system architecture development effort.

[5] mentions modeling and simulation in section 6.4 as possible methods for the analysis of alternatives during the systems analysis and implementation phases. Furthermore, models are also referred to as a means for validation and verification. However, there is no consistent definition concerning the model structure and the use of the analysis models in later phases.

Section 3.2 of [3] elaborates on model-based design. It describes a procedure for building and analyzing models and lists a set of commonly used tools for model creation. This procedure describes the main activities concerning the use of models in a Systems Engineering context. It does not address multiple hierarchical levels of a product though. It is only suggesting to go repeatedly through the macro-cycle with increasing product maturity. Despite its focus on embedded systems development in the automotive sector it is a good starting point for the development of a more general approach that can be applied to aerospace systems.

In addition to the proposed methodologies formal modeling languages have been developed to formally specify technical systems. The most well-known of which is the *System Modeling Language* (SysML), a graphical modeling language derived from a subset of the *Unified Modeling Language* (UML) introduced by the *Object Management Group* (OMG). SysML provides a graphical notation and an information model. Another example of a formal modeling language is the *Object-Process Methodology* (OPM) [8]. In contrast to SysML the OPM allows for the modeling of resources and the representation the system structure with the corresponding functional allocation in a single diagram. For further details regarding Systems Engineering methodologies or languages and Model-based Systems Engineering see also [7, 8, 9, 10, 11, 12, 13].

The MBSE approach presented in this paper combines different elements from the approaches discussed in the state of the art section before. Some mechanisms that are of great importance in modern Systems Engineering but not sufficiently considered in those approaches are added in our approach. Our approach therefore contains proposals how to deal with a complex system of multiple hierarchical layers and how to capture trace links between the proposed models.

III. STEPWISE INTRODUCTION OF MBSE ACTIVITIES

One of the main goals of Systems Engineering is to ensure a consistent process that leads from high level product requirements to a detailed system architecture. In the approach presented in this paper, the scope of Systems Engineering is extended by the notion of model–based development. That means that all elements, starting from the early high level requirements, down to the detailed requirements for each component in the system architecture, are documented by models that allow for discovery of interdependency, and for analyzing effects of changes or alternative designs. The approach presented in this paper can be regarded as model-based for various reasons. Most importantly, all information requested in the suggested process phases being introduced in this section can be represented with existing modeling tools. Suggestions for adequate exchange formats to share these modeled data is provided in the appendix. Additionally, all introduced models do comply with the prevalent criteria for models: mapping, reduction and pragmatic [12, 14]. While the presented approach does not focus on the actual implementation of system components, such as

[1] Please see all Figures at the end of the manuscript.

Computer Aided Design (CAD) modeling and software programming, it emphasizes the importance of synchronizing the implementation results with higher-level simulation models. Hence, the MBSE approach partially addresses the implementation phase and also extends the traditional Systems Engineering approach with respect to the development phases that it covers. Virtual *Validation & Verification* (V&V) activities are considered to be performed during the early phases of the development and are therefore covered by the approach presented, while physical V&V activities requiring a real implementation of the later product are not covered.

The presented MBSE-approach consists of multiple phases each creating, modifying and/or consuming a set of models, such as a requirements model, a system structure model, various simulation models, etc. (see Figure 2 for the V–model according to the proposed MBSE approach). These models are refined and their maturity increases during the development. The single phases of the MBSE approach will be introduced in the upcoming sections and elaborated by means of the following example:

Example 1: *Consider the electrical network architecture of a civil aircraft*[2]*as an example. An aircraft usually features two electrical networks: One for electrical power supply and distribution and one for data communication.*
Most modern aircraft are equipped with two engines, each of them driving an electrical generator which supplies electrical power to the aircraft systems. Current aircraft systems have different power supply characteristics, e.g. some systems are supplied with 28V DC, while others need 115V AC. A safe flight is only possible if electrical power is supplied to the essential aircraft systems. This means that the electrical power supply and distribution network must comply with stringent safety objectives.
Modern aircraft are controlled and monitored by digital computers and software provides most of the essential control and monitoring functions, e.g. for flight control, navigation, communication, environmental control and power management. Current aircraft architectures make use of a common type of computer, called Core Processing Module *(CPM), that is combined with other computers of the same type into a network. This network provides a common computing and communication resource for the different aircraft systems. Additionally, other networks, e.g. for cabin system communication and in-flight entertainment may be installed on an aircraft.*

Each of these networks require complicated wiring within the aircraft. A comprehensive set of rules about how to define a cable route within the fuselage or wing assembly exist and must be obeyed, e.g. for the separation of essential and non-essential networks parts and to prevent electromagnetic interference between sensitive equipment and an electrical conductor.

1.1 Product Definition

The process starts with the definition of the high–level requirements for the product to be developed. This is done in the Product Definition phase where the general product is defined based on customer demands (e.g. via sales), mature technological innovation and management expectations. The demand is usually obtained by analyzing the market, associated challenges (competitors, legislative regulations, standards, risks) and different product ideas. Product ideas can also include the use of different mature technologies and strategies to put them into the market. In this first stage of the development process the high level requirements of the product are defined in an unstructured manner and passed to a project responsible. The result of this phase usually is a set of documents and not necessarily a model. This phase involves the integration of technical innovations as a result of research and development, see Table 1 an Appendix.

Example 2: *Aircraft manufacturing and configuration capabilities are heavily impacted by the electrical network design and interface technology used for connecting equipment to the network. Once the network has been designed and installed into the aircraft it cannot easily be changed or adapted to specific customer needs. This creates a demand for a new approach to electrical networking allowing for increased flexibility and a reduction in the number of interface and cable types. Furthermore, the replacement of a metallic cable by other transmission means, such as wireless technology or optical fibers, may significantly reduce the weight impact of the network installation.*
High-level product requirements:
- *HL-R1: Keep systems flexible to easily meet specific customer needs*
- *HL-R2: Reduce weight of the aircraft*
- *…*

[2] The following levels are typically defined in civil aerospace system development:

1. Aircraft level
2. System level
3. Subsystem or equipment level

1.2 Product Requirements Analysis

Since the high-level requirements that are documented in the *Product Definition* phase do usually not adhere to the high formal quality standards for proper requirements documentation and address the entire product to be developed, they need to be reviewed and validated, detailed and unambiguously refined, formalized and structured. This is done in the *Product Requirements Analysis* phase. Furthermore, the requirements are transformed into a model in this phase. That means that they are modeled using a tool environment that allows for storing them based on a formal data model. That data model enables the developers to structure, analyze and query the stored requirements in an efficient way. If a pair of product requirements influences each other a link has to be modeled between them to ensure traceability[3].

Example 3: *In the future the multitude of power supply networks shall be reduced and the general DC supply voltage shall be increased from 28V to 240V. Furthermore, the network configuration flexibility shall be enhanced to ease the implementation of changes to the network after the aircraft has entered service.*

- *P-R1: Use standardized interfaces [HL-R1]*
- *P-R2: Reduce the MEW (manufacturer's empty weight) of the aircraft of by 500kg [HL-R2]*
- *P-R3: Introduce new types of transmission technology (e.g. wireless technology) [HL-R1, HL-R2]*
-

Finally, in addition to the already modeled functional requirements, a comprehensive set of non-functional requirements will be defined. These requirements address the so-called −ilities, i.e. availability (usually defined as a safety objective), reliability, maintainability and supportability.

1.3 Functional Architecture Definition

The requirements model from the *Product Requirements Analysis* phase describes requirements for the entire product. Based on these requirements and in order to break them down the required product functionality is identified. In this *Functional Architecture Definition* phase a functional architecture model is derived by defining necessary product functions and their functional parameters, breaking them down hierarchically and modeling their dependencies and relations. The understanding of functional dependencies in the product to be developed is a basis for the definition of system interfaces conducted later and can be used for designing an optimized system structure (reducing the number of interfaces). The last step in this phase is the creation of links between the functional architecture model and (product) requirements model. This allows for the later identification which requirements a specific function has been derived from, and it allows for verifying whether all functional requirements have been covered in the functional architecture model. See Figure 3 for an excerpt of a functional architecture.

1.4 System Structure Definition

In the *System Structure Definition* phase a system structure model is created, based on the functional architecture model. Therefore, the functional model is the informational basis to decide which systems and subsystems need to be included in the system structure model. Further input comes directly from topological parameters within the requirements defined during the requirements analysis, which is documented through a link between the respective requirement and systems structure element. The hierarchical system structure model describes topological structure and parameters of the system and all its elements, which are needed to perform the required functions. The system structure model consists of blocks representing its elements and can be enriched with attributes. They do not contain the detailed inner workings of the subsystems. The amount of parameters attached to the elements and their level of detail increases during the development process. Later in the process, once a behavior simulation model is defined, the initially defined/estimated topological parameters of the system structure are refined and completed with behavior simulation results.

Example 4: *The top levels of a system structure of an aircraft are pre-defined by so called ATA chapters, which are the same for all aircraft manufacturers. Usually the entire system structure is reused from legacy projects and contains elements like harnesses and connectors. Figure 4 shows a reduced example of an aircraft system structure.*

The system structure is continuously updated during the development process. For example, if parts of the wired communication network are replaced by wireless technology, new elements have to be introduced, such as transmitters, receivers and antennas.

1.5 System Architecture Definition

Once the system structure model has been set up, functions (that are specified in the functional architecture) are allocated to the subsystems (i.e. the elements of the system structure model) by creating

[3] Model elements followed by identifiers in square brackets indicate that a trace link has to be modeled from this element to the elements with the respective identifiers' text

links between both models. This is done in the *System Architecture Definition* phase. Links between both models allow for verifying whether all product functions are covered by elements of the system structure and it allows developers of a subsystem to easily identify which functions they have to implement. The links between the system structure model and the functional architecture model are stored in the system architecture model, which may consist of multiple layers/levels of detail. The system architecture can be understood as an integration model since it does neither contain nor copy the other models but rather integrates them by adding links without modifying the source models as such. See Figure 5 for an exemplary system architecture mapping functions to elements of the systems structure

1.6 Requirements Analysis

Since the requirements from the product requirements model are only specified for the entire product, they need to be broken down further and allocated to the system structure elements. The *Requirements Analysis* phase covers the process of breaking down high-level requirements and deriving more detailed ones (and linking them to elements of the system structure and to functions from the functional architecture model).

Example 5: *Regarding the electrical network example the product requirements (e. g. use standard interfaces, reduce weight) need to be broken down into more detailed requirements:*

- *R1: The total weight of harnesses shall be reduced by ca. 15% [P-R2]*
- *R2: The number of power supply networks shall be reduced and the general DC supply voltage shall be increased from 28V to 240V. [P-R1, P-R2]*
- *R3: Reduce number of electrical and data interfaces by 30% [P-R1]*

An in-depth analysis shows that the requested weight reduction cannot be achieved with the current technological concept. For that reason alternative technological concepts for the function distribute data are evaluated (such as WLAN or optical transmission).

The three phases *System Structure Definition, System Architecture Definition* and *Requirements Analysis* are highly interdependent as indicated in Figure 2.

1.7 Concept Selection

Once the system structure model, the requirements model and the system architecture model are complete, it must be considered how new functions from the functional architecture model can actually be implemented by the elements of the system structure model. This is done in the *Concept Selection* phase, see Table 7. In this phase alternative solution principles are identified for each considered function. These solution principles are evaluated and the most suitable one is selected to fulfill the function under consideration. As soon as a solution principle is selected, its functional and topological parameters must be refined, e. g. to allocate space within a digital mockup.

Example 6: *The concept selection is a crucial step on the way to a mature system architecture, as it reflects the main design choices and trade-offs. In case of the electrical network, the exact application of new technologies, such as optical communication technology will be determined, i.e. what part of the data communication network will be implemented using optical components and how the optical components shall be interfaced with the other parts of the network. Furthermore, the architecture definition includes important non–functional design choices regarding performance, resilience and reliability. The most important choices for the data communication network concern the replacement of existing copper networks by optical fibers or wireless communications. For the power supply and distribution part of the network concepts to be considered are to use higher supply voltages for the DC network and to consider passive network extensions to allow for load balancing between parts of the network that may be idle during specific operational phases, such as take-off, climb or taxi.*

1.8 Behavior Modeling and Simulation

The behavior modeling and simulation phase is the core activity of the proposed approach. Up to this phase all models that have been created were either purely functional models or static structural models as described in sections 3.3 and 3.4. Typically a standardized description is used for these two types of models in order to facilitate information exchange between different stakeholders. SysML is the most widely used modeling language for this purpose.

However, this is not sufficient for a real architecture analysis and trade-off study of a given system design. In order to perform such a study, a comprehensive executable model of the system under design must be build, validated and executed with a variety of parameter sets that allow for a thorough analysis of the system behavior. In contrast to purely functional models this model must include all resources required by or pertinent to the system. This is especially important as modern system architectures usually represent cyber-physical systems, i.e. mechanical, electrical or chemical systems digitally controlled and monitored by networked electronics and software [15]. In order to evaluate such architectures, an executable model of all parts of the architecture is needed to analyze the complete dynamics of the coupled system, i.e. the mechanical, electrical or chemical process plus the digital network and the transmission protocols employed for control and monitoring. This means simulating a hybrid system, i.e. a system with continuous and discrete parts in a joint model which allows for understanding the complex dynamics of the overall system. See [16] for details on hybrid systems modeling.

The Object Process Methodology (OPM) mentioned in section 2 presents an interesting approach for describing the system behavior in a way that is usable for developing a simulation model. This description can be used as the base for creating the comprehensive system simulation model [17].

The comprehensive executable system model requires thorough validation before it can be employed for architecture studies. Validation usually relies on either the availability of a similar system supplying data, e.g. from a measurement campaign of a mechanical, electrical or chemical system or a well defined mathematical algorithm for scheduling and transmission protocol state events. Once validation has been performed the model can be regarded as a virtual prototype of the system under design.

During the equipment test and integration phase one usually performs accelerated life testing in order to find any flaws in the real system design. This can be prepared with the validated executable system model by performing system boundary behavior studies. For such a study the simulator is supplied with a set of parameters that are deliberately close or even beyond the specified design limits of the system. The simulation results then show how the system would behave if any of the parameters or a combination of those will be out of the specified range during operation. This might either directly show unapparent design flaws or serve as a sensitivity analysis that shows possible limitations of the system design that might require further investigation.

It is important to emphasize that a system simulation model must include all system elements at a level of detail that are required to understand the system dynamics. At the top level, i.e. the system under design, a comprehensive system model would include the system itself and all the interfaces to other systems that are relevant for the overall dynamics. When performing iterations as depicted in Figure 1 it must be ensured that all simulation models created at a lower level of the architecture are consistent with the models one layer above. This requires the definition of an interface between the models.

For example, when developing a networked control system, such as a flight control system of an aircraft, the top level model would include the complete network with its communication channels, the underlying protocols and all components communicating over that network together with their behavior. This allows for studying overall system dynamics and emergent behavior. Each component connected to the network may be a complex system by itself. When refining the simulation model of such a component it is important to retain the interfaces of the component model to the overall system model. This permits for validation of the refined model in the context of the overall architecture and allows for early detection of any unwanted emergent behavior. In order to ensure consistency across all architectural levels, it is vital to choose the right level of abstraction for each simulation model. At higher architectural levels models can be more abstract while they will cover more details at lower levels. However, at each level sufficient detail must be provided to fully understand the respective dynamics.

The results of the behavior simulation are used to update the parameters in the functional architecture and the system structure. Furthermore, the models themselves need to be linked to the other existing models in order to ensure integrity. The updated system architecture model then allows for performing an integrated verification of the overall architecture. Since the *Behavior Modeling and Simulation* phase affects the models created in prior phases this may lead to iterative corrections of those phases.

Example 7: *The behavior model has to cover the relevant properties of the architecture and thus permit a sound design decision. This includes models for performance evaluation of the digital protocols to be used for data communication, the physical behavior of the transmission channels, such as wave propagation in an optical fiber or short distance microwave transmission. Furthermore, reliability models must be created to assess, if the architecture is capable of fulfilling the necessary safety and supportability objectives. The challenge in case of the new electrical architecture is that previous experience with a similar design does not exist. This makes it difficult to validate the simulation models. Thus, model development is an incremental process during this phase. Although the functions of the system will essentially remain the same the system structure has to be adapted. For example, consider a replacement of parts of the wired communication network by wireless links. This will introduce new elements into the system structure, such as transmitters, receivers and antennas. Alternatively, an optical fiber link may replace copper wire for some parts of the network,* *also introducing new elements, such as light sources, optical transducers and optical switches.*

1.9 Subsystem Requirements Analysis and Specification

When the overall system architecture and all other linked models have been verified and validated successfully, the specifications containing the expected characteristics and parameters for the system to be implemented (i.e. subsystems) must be created. This is done in the *Subsystem Requirement Analysis and Specification* phase. The resulting specifications can be handed over to different development parties that can also be external suppliers. A specification for one subsystem contains detailed requirements, parts of the system architecture (i.e. system structure elements and assigned functions), behavioral models for this specific subsystem and a simulation model template. The simulation model template is an executable simulation model comprising a mock-up of the subsystem to be developed (i.e. a black-box with a simple behavioral function) and the interfaces of that subsystem to other systems and subsystems (modeled as black-boxes). The resulting simulation model can then be used for validating the functionality of the subsystem. If a physical subsystem (and not software) is to be developed then that specification also contains a space allocation model.

Example 8: *For the described subsystems, requirements specifications are created based on the previously defined models (requirements, behavior, system architecture, space allocation). Especially important in the specifications are the interface requirements, as they address the interfaces of the aircraft system components with the newly designed network.*

In order to ensure that the subsequently developed components comply with the specifications and the overall system behavior is as expected, results from the component development phase are used to successively refine and complement the behavior simulation model and (indirectly via this model) the functional and topological parameters in the system architecture. This iterative routine should be defined in the specification in order to make sure that models developed by suppliers or other organizations are compatible with those used in the *Behavior Modeling and Simulation* phase.

Example 9: *In order to start validation early, a simulation model at product level, i.e. aircraft level, covering all relevant components of the aircraft is used at Airbus. First, each component is modeled according to the original specification with an abstract model. As specifications are passed to suppliers responsible for the detailed component design, more detailed models are developed. These detailed models are subsequently delivered by the suppliers to Airbus for integration into the overall simulation model. This permits for constant monitoring of both supplier design quality and behavior of the overall system architecture at product level. At the final stage real hardware is connected to the overall model replacing the component simulation models retaining only the simulation of the aircraft environment. This approach requires an interface between higher level simulation models and detailed component models that are shared between Airbus and their suppliers. To this end, Airbus has developed such an interface definition and provides a toolbox supporting common simulation tools together with an integration process definition to system designers and suppliers.*

IV. APPLICATION OF MBSE APPROACH ON COMPLEX PRODUCTS

For easier understanding complex products are designed in hierarchical structures [18]. Their system structure is usually defined by multiple abstraction levels ranging from the overall product via systems, subsystems and other levels to components. Depending on the complexity of the product some of the presented MBSE steps should be repeated subsequently for each abstraction level until a precise specification of a component can be achieved. For a product with the aforementioned four abstraction levels Figure 2 indicates the sequence of process steps that need to be executed on the respective abstraction level. Information models from prior abstraction levels can be used and refined in later stages. It is not necessary to build up separate models for every abstraction level.

To ensure consistency between all modeled data two main mechanisms have been applied. On the one hand, data from subsequent phases are used to refine and complement models having been created in prior phases (see section 3). Additionally, do entities of two consecutive models get linked through trace links to ensure full coverage of requirements or functionalities. For further insights into mechanisms for traceability capture, usage and visualization see [19, 20, 21]. However, it might be helpful to add further procedural mechanisms or algorithms to check the consistency of models when implementing this approach in real development projects.

Example 10: *The challenge concerning subsystem requirements analysis for the aircraft electrical architecture is to give any developers of an equipment or subsystem enough freedom to make their own design choice while at the same time retaining a valid overall architecture. This can only be achieved if an overall architectural model was* *created before subsystem development starts. This model is then subsequently used as the central point of integration during the complete development process. For example, developers of an optical transceiver might choose specific hardware for this type of equipment. Only when their design is documented by an executable model and this model can successfully be integrated into the overall architecture model, the design can be regarded as valid. The same applies to all other components of the architecture, such as wireless transceivers, semiconductor power switches, network control software and digital communication protocol designs.*

V. CONCLUSION

The paper presents a novel MBSE approach that focuses on the early development phases and takes the complexity of modern mechatronic systems into account by distinguishing the required activities by the respective abstraction level. It has been initially evaluated by experienced developers at Airbus, whose feedback was used to adapt the first drafts of the approach. Furthermore, the approach was applied on the development of a simple exemplary product for verification purposes.

However, this MBSE approach has not yet been applied in practice on highly complex technical systems. This clearly limits its validity. In a next step the approach should be validated with the help of a more complex technical system to gather lessons learned and to verify its suitability for Systems Engineering.

Additionally, future research should investigate if the feedback loops can also be applied to later life cycle phases of the product, such as production and usage. The product-related information gathered in these phases could be used to continuously improve the digital model of the product. Production engineers as well as users gain a valuable detailed knowledge by spending a lot of time with the product. Knowledge obtained by maintenance staff is especially important, as they have expertise to easily map their knowledge of product flaws onto the corresponding data set in the digital product model.

Therefore, the presented MBSE approach not only provides a contribution to the Systems Engineering community but can also be seen as a sound starting point for future research.

REFERENCES

[1] G. Beier, U. Rothenburg, R. Woll and R. Stark, Durchgängige Entwicklung mit erlebbaren Prototypen - Modellbasiertes Systems Engineering. *Digital Engineering, 3/2012*, 14-17.

[2] NIST, *The Economic Impacts of Inadequate Infrastructure for Software Testing*, Technical Planning Report 02-3 (National Institute of Standards and Technology, Gaithersburg, USA, 2002).

[3] VDI 2206, *Entwicklungsmethodik für mechatronische Systeme*, Technical Report ICS 03.100.40; 31.220 (Verein deutscher Ingenieure, Düsseldorf, Germany, 2004).

[4] D. Buede, *The Engineering Design of Systems: Models and Methods* (John Wiley & Sons, 1st edition, 2000).

[5] ISO/IEC/IEEE, *Systems and software engineering — System life cycle processes*, ISO/IEC/IEEE 15288 (2015).

[6] NASA. *Systems Engineering Handbook*, Technical Report SP-2007-6105 Rev1 (National Aeronautics and Space Administration, Washington, USA, 2007).

[7] INCOSE, *Systems Engineering Handbook. A guide for system life cycle processes and activities*, version 3.2 (International Council on Systems Engineering, 2010).

[8] D. Dori, *Object–Process Methodology. A Holistic Systems Paradigm* (Springer Verlag, 2002).

[9] B. S. Blanchard and W. B. Fabrycky, *Systems Engineering and Analysis* (Prentice Hall, 4th edition, 2005).

[10] INCOSE, *Survey of Model-Based Systems Engineering (MBSE) Methodologies* (International Council on Systems Engineering, 2008).

[11] OMG, *Systems Modeling Language 1.3* (Object Management Group, 2012).

[12] A. L. Ramos, J. A. Ferreira, and J. Barcelo, Model-based systems engineering. an emerging approach for modern systems. *IEEE Transactions on Systems, Man and Cybernetics, 2012*, 101–111.

[13] R. E. Thompson, J. M. Colombi, J. Black and B. J. Ayres, Disaggregated Space System Concept Optimization: Model-Based Conceptual Design Methods, *Systems Engineering, 18(6)*, 2015, 549-567

[14] H. Stachowiak, *Allgemeine Modelltheorie* (Springer Verlag, 1973).

[15] E. A. Lee and S. A. Seshia, *Introduction to Embedded Systems. A Cyber–Physical Systems Approach* (MIT Press, 2017).

[16] R. Goebel, R. G. Sanfelice, and A. R. Teel, *Hybrid Dynamical Systems: Modeling, Stability, and Robustness* (Princeton University Press, 2012).

[17] M. Schulz, V. Zerbe, and S. Marwedel, Using the object process methodology to build simulation models. *Proc. 3rd International Conference on Model–based Systems Engineering*, Fairfax, Virginia, United States, 2010.

[18] M. E. Sosa, A. Agrawal, S. D. Eppinger, and C. M. Rowles, A Network Approach to Define Modularity of Product Components. *Proc. 17th ASME International Conference on Design Theory and Methodology*, 2005, 435–446.

[19] A. Figge, *Effiziente Erfassung und Pflege von Traceability-Modellen zur Entwicklung technischer Systeme*, doctoral diss., Technische Universität Berlin, 2014.

[20] G. Beier, *Verwendung von Traceability-Modellen zur Unterstützung der Entwicklung technischer Systeme*, doctoral diss., Technische Universität Berlin, Germany, 2014.

[21] E. Brandenburg, A. Figge, S. Zander and G. Beier, Recommendations for Tracelink Decisions – An Empirical Investigation of Visualization Methods, *Proc. 5th International Conference on Applied Human Factors and Ergonomics*, Krakow, Poland, 2014.

Figures

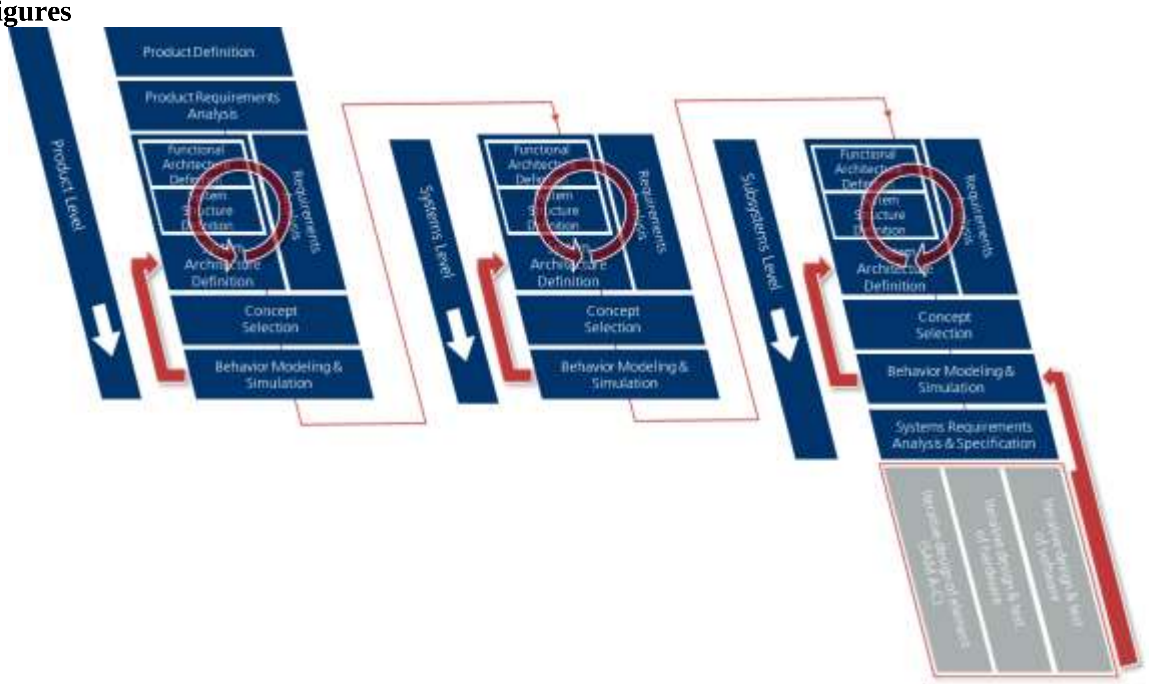
Product Level
Product Definition
Product Requirements Analysis
Functional Architecture Definition
System Structure Definition
System Architecture Definition
Requirements Analysis
Concept Selection
Behavior Modeling & Simulation
Systems Level
Subsystems Level
Systems Requirements Analysis & Specification
Iterative design of components (SAMA-C)
Iterative design & test of hardware
Iterative design & test of software

Figure 1: Process iterations for three hierarchical levels: Product, System, Subsystem

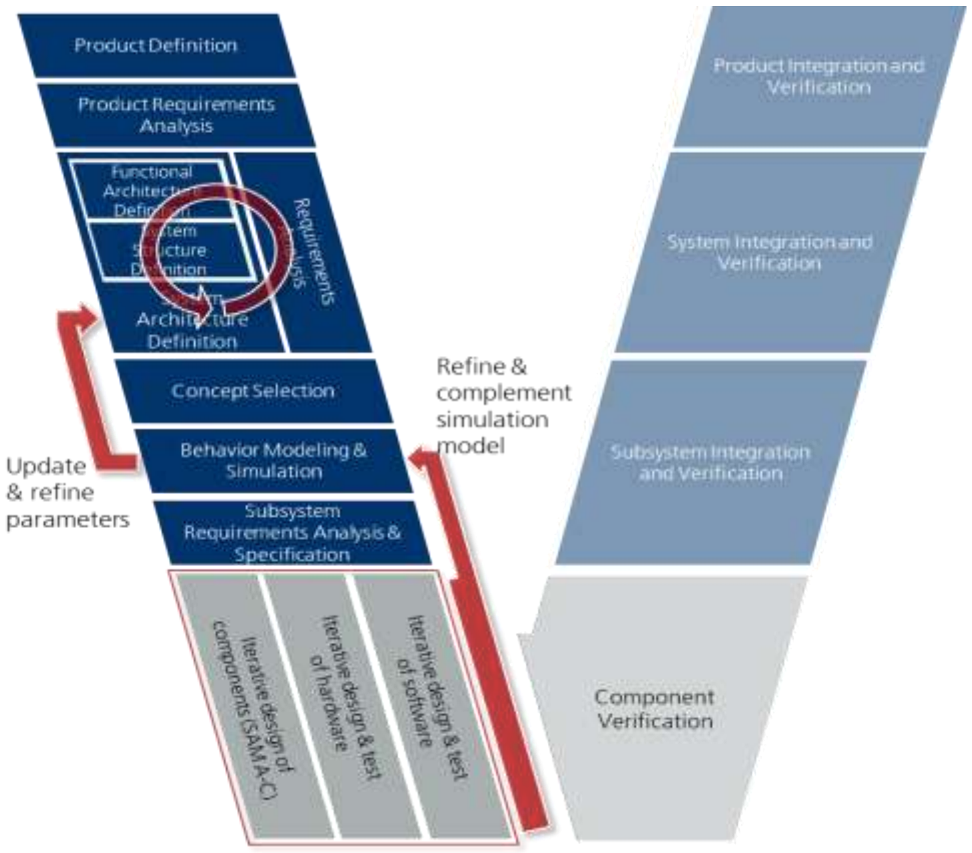
Product Definition
Product Requirements Analysis
Functional Architecture Definition
System Structure Definition
System Architecture Definition
Requirements Analysis
Concept Selection
Behavior Modeling & Simulation
Subsystem Requirements Analysis & Specification
Iterative design of components (SAMA-C)
Iterative design & test of hardware
Iterative design & test of software
Update & refine parameters
Refine & complement simulation model
Product Integration and Verification
System Integration and Verification
Subsystem Integration and Verification
Component Verification

Figure 2: V–model according to the proposed MBSE approach

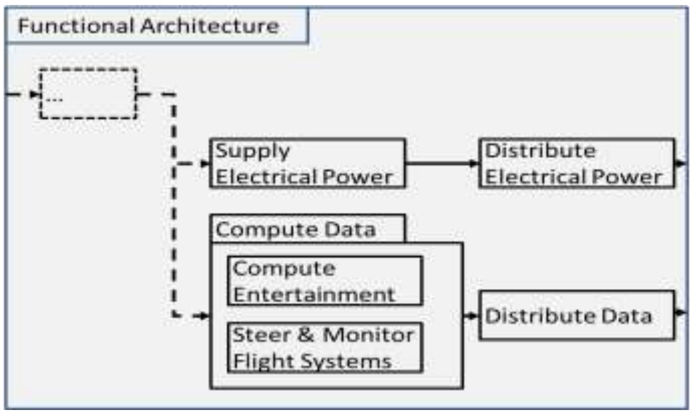

Figure 3: Excerpt of a functional architecture

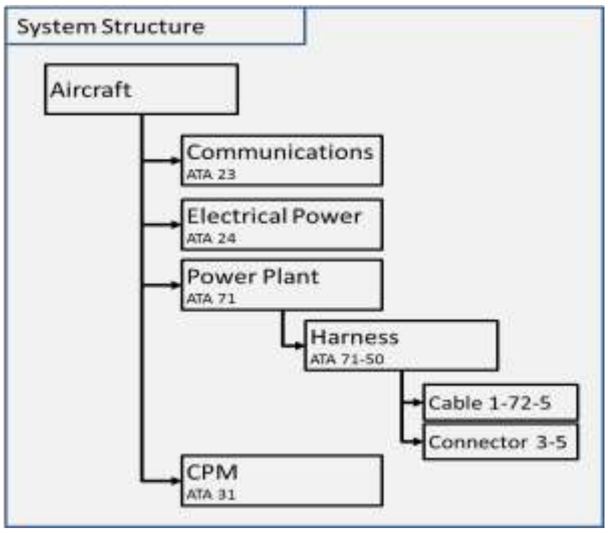

Figure 4: Excerpt of a system structure

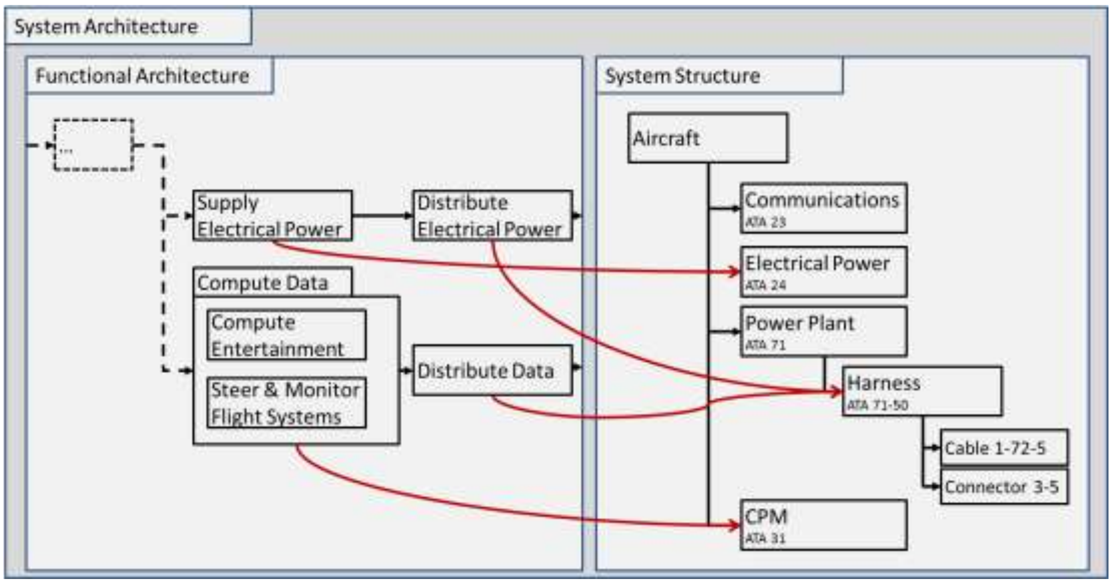

Figure 5: Exemplary system architecture mapping functions to elements of the systems structure

Appendix: Tables providing Characteristics for respective Process Steps

Table 1: Product definition phase characteristics

Input	Output	Exchange format	Acting roles
• **Product ideas, innovations** • **Customer needs and demands, market analysis** • **Management expectations** • **Technology and trend reports** • **Legal constraints, standards**	• Unstructured high level product requirements	• Requirement Interchange Format (ReqIF)	• Management, • Sales, • Research & Development

Table 2: Product requirements analysis

Input	Output	Exchange format	Acting roles
• **High-level product requirements**	• Product requirements model	• Requirement Interchange Format (ReqIF)	• Requirements Analyst, • Business Expert, • Project Manager

Table 3: Functional architecture definition

Input	Output	Exchange format	Acting roles
• **Product requirements**	• Functional architecture • Mapping of functions on corresponding product requirements • Prioritized functions	• Systems Modeling Language (SysML)	• System Engineering or Architect • Function Modeler

Table 4: System structure definition

Input	Output	Exchange format	Acting roles
• **Functional architecture** • **Requirements model (incl. topological parameters)** • ***Later:* topological parameter values from the behavior model**	• Functional architecture • Mapping of functions on corresponding product requirements • System structure model with topological parameters	• Systems Modeling Language (SysML)	• System Engineering or Architect

Table 5: System architecture definition

Input	Output	Exchange format	Acting roles
• **Functional architecture** • **System structure** • **System requirements**	• System architecture	• Systems Modeling Language (SysML) • Traceability tools	• System Engineering or Architect

Table 6: Requirements analysis

Input	Output	Exchange format	Acting roles
• **Product requirements model** • **Functional architecture model** • **System structure model**	• System requirements model	• Requirement Interchange Format (ReqIF)	• Requirements analyst • System Engineering or Architect

Table 7: Concept selection

Input	Output	Exchange format	Acting roles
• **System Architecture** • **System Requirements** • **Existing evaluation criteria**	• Evaluated solution principles • Allocation of documented solution principle to architecture	• Requirement Interchange Format (ReqIF)	• Requirements analyst • Design Engineer • Software Architect

Table 8: Behavior modeling and simulation

Input	Output	Exchange format	Acting roles
• **System Architecture** • **System Requirements** • **Selected Concepts**	• Behavior model • Simulation results • Updated and revised parameters	• *Depends on problem definition*	• Simulation Expert • Software Architect

Table 9: Subsystem requirements analysis and specification

Input	Output	Exchange format	Acting roles
• **System Architecture** • **System Requirements** • **Behavior Models**	• Subsystem requirements specification • *Optional:* Simulation model template	• Requirement Interchange Format (ReqIF)	• Requirements Engineer • Simulation Expert • Purchasing department

Grischa Beier. "A MBSE Approach for the Development of Complex Technical Systems." International Journal of Engineering Research and Applications (IJERA), vol. 7, no. 9, 2017, pp. 24–36.