DOI 10.5162/ettc2022/10.2

# How to get from MBSE to virtual product testing

*Gregor Staudte*
*Airbus Defence and Space GmbH, Rechliner Str. 85077 Manching, Germany*
*gregor.staudte@airbus.com*

**Abstract:**
Model-based system engineering (MBSE) is an important and useful approach to support very large system development. Especially within the defence aerospace industry. However, testing of system models is often performed only on a very static and abstract level. Its use often ends after a conceptual phase. Then the system models become detached from the real product.

The early and continuous verification and validation of the product against the MSBE models is a very important element to reduce program risks. The “Virtual Engineering” approach is our answer to enable testing of a virtual product as early as possible. A challenge is the relation to the MBSE world.

This paper outlines the recent experiences and the taken approach to couple the MBSE world with virtual testing. In fact, the environment is still in the setup phase were a special focus is put into this paper.

**Key words:** MBSE, virtual engineering, virtual product, virtual testing

## Motivation

Most of the mistakes are made in the design phase. In contrast, errors, malfunctions and misbehaviour due to these mistakes are discovered very late. The later they are discovered, the higher the costs of removal. A mistake that was made early in the design phase, but is found very late, for example after the product has been delivered to the customer, can cost many times more compared to detection right after the mistake was made [1].

Therefore, a virtual product testing methodology shall be applied within our programmes. It shall discover design errors as early as possible and therefore reduce risks. In our context, it is not intended, at least not today, to use virtual testing as means of compliance. The desired benefit is, to gain a higher system maturity when entering the formal verification phase.

An additional beneficial side effect should be the higher efficiency of the actual formal verification activities on the real target product. This can be achieved through test preparation and its verification on virtualized means. Another expected beneficial side effect is the reuse of the virtual testing environment for later training operations.

Due to the agile nature of the programme, a virtual test bed may also act as environment to demonstrate the increments to its stakeholders, to validate the system and to act as foundation for reflection.

Now to focus again on the core objective for virtual testing “to discover errors early”, the following question arises: What kind of mistakes are usually made and why? Many things can go wrong: There can be wrong tests. The implementation of a functionality can be wrong. A device might by integrated in the wrong manner. Often this is due to a wrong design. For instance, the interface definition is wrong, or the requirements are interpreted differently, maybe are not consistent. Anyhow, the mistake is often made by miscommunication. Two individuals are misaligned which leads to errors.

MBSE intends to address this misalignment. It is “the formalized application of modeling to support system requirements, design, analysis, verification and validation beginning in the conceptual design phase, and continuing throughout development and later life cycle phases.” [2]

The outcome of the design phase are documents based on a complex MBSE model, which describes the entire system. The MBSE model acts then as input for further development activities such as implementation, integration, manufacturing and verification.

Consequently, a very important goal is on one hand to verify the consistency and applicability of the design, and on the other hand to verify the

correct understanding and implementation within the preliminary work product. Since the first goal shall be covered by the MBSE framework itself, the second goal is the most important motivation for virtual product testing.

The focus of this paper is on the design and virtual product verification of avionics software. Mechanical or electrical engineering is out of scope.

**Context**

The systems addressed by this paper are in the field of military aircraft. Programmes that develop these systems tend to have an exponentially increasing complexity level. Both, on the system itself, and on the programme setup by multiple involved nations and companies in a partnership setup.

In order to meet customer needs better, and to stay on budget and on time, a semi-agile approach is used for our programme. The definition phase is split into several increments with a fixed duration, in which functional extensions are provided.

The design of such systems is structured into several layers, in which associated systems of layer n+1 act as subsystems of a system in layer n.

An illustrated, not to scale, timeline extract of our programme development phase is shown in the following Fig. 1. As remark, the design phase can be seen as phase spanning across concept & definition phase.

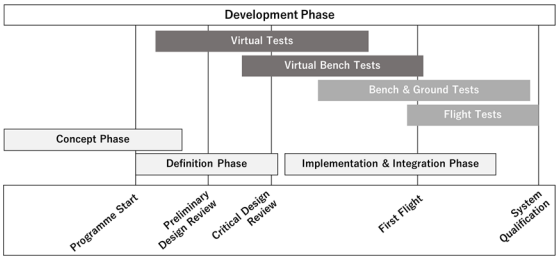


*Fig. 1 Programme development phase timeline*

**MBSE approach**

In the following, the focus is on how our programme applies MBSE. It should also be clarified whether the MBSE model is suitable for virtual product testing.

The MBSE approach relies on a framework, which provides the system engineering capability set to rely on common solutions from requirement, mission & operational analysis, architecture, safety and V&V, with a full digital continuity. As an overview, Fig. 2 visualizes our MBSE environment and how it is embedded into the overall engineering landscape.

The solution relies on the so-called R-MOFLT methodology [3]. It considers structural & behavioural views in both problem and solution space while keeping the focus on the system of interest and requirements along the entire development cycle:

- Mission analysis: Focuses on identifying the main purpose of the solution, characterizing the problem space, and determining possible solutions. Therefore, it describes what the problem is to be solved, and identifies potential solutions.
- Operational analysis: Focuses what the system does within missions. Therefore, it describes the system context and operation from user perspective.
- Functional analysis and architecture: Identifies the system functions to perform and their mutual relations to meet operational needs. Therefore, it describes how the system will work.
- Logical architecture: Describes logical system decomposition and clustering of functions into a logical structure in addition with their interfaces and corresponding behaviour.
- Technical architecture: Describes how to implement a logical architecture, by taking technological constraints into account, into a sufficient level of detail to support system implementation, integration and V&V.

The points listed are steps within the MBSE workflow, parts of the system model and perspectives to view the system model.

The MBSE solution contains a programme wide common visual modelling tool supporting SysML® [4] as the single modelling language for any MBSE activity. It also contains common access and data share for all programme partners. The integration, with respect to continuity and traceability with solutions outside MBSE scope, such as requirement, interface and test management, is ensured.

Within our programme execution, the actual MBSE practice is limited to static system modelling and interface, down to the system level of equipments & line replaceable units (LRU). The lower hardware / software level is in general not addressed by the MBSE model. Also not addressed are interface details, such as pin assignment or message formatting information of communication busses.

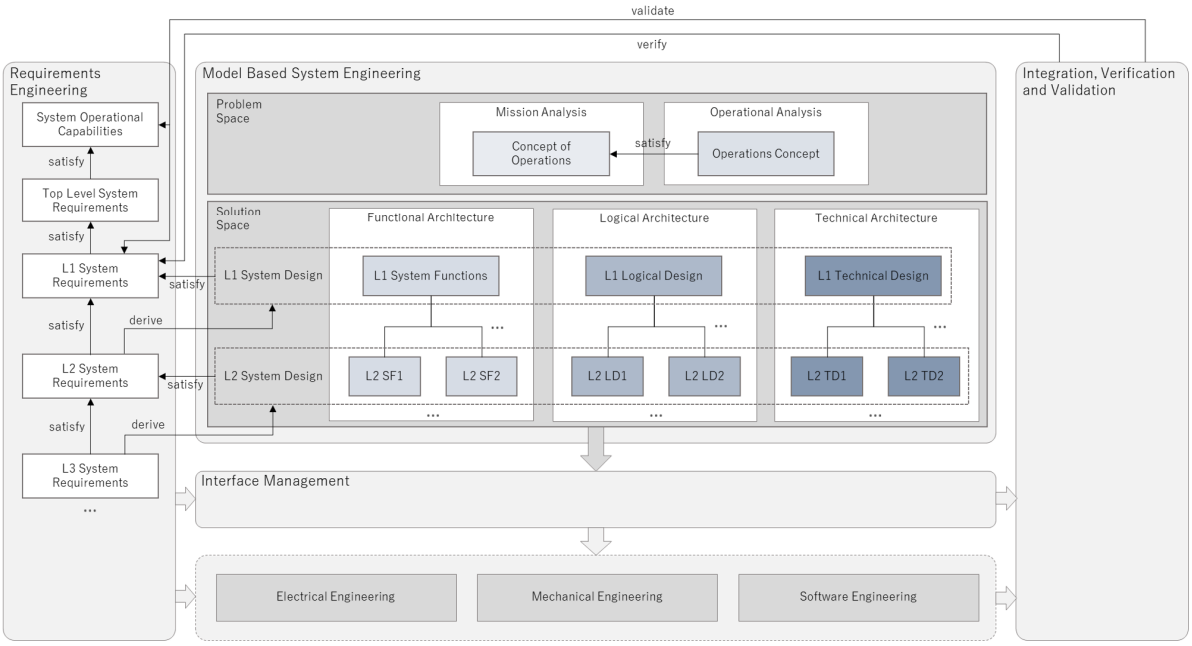


*Fig. 2 Overview MBSE environment*

Although the MBSE modelling tool offers capabilities for simulation, the MBSE model itself is not executable. The model is basically, a structured set of attributes and parameters. The fidelity level defines the possible simulation use-cases. Simulations based on our MBSE model are therefore limited to parameter evaluation. E.g. to calculate the overall system weight, out of weight parameters of subsystems.

These capabilities already offer some possibilities to verify the design. It can be checked automatically if the design complies with certain requirements, or whether certain parts are consistent. For the virtual product testing purpose, as addressed by this paper, these simulation capabilities are insufficient. There is the need for a simulated virtual product, which acts as the real target product.

**Virtual Engineering**

For that matter, our programme applies within the verification and validation (V&V) activities, the so-called Virtual Engineering (VE) principle. It is the structured and standardized end-to-end application of dynamic and functional / behavioural modelling and simulation of the entire system. Its purpose is mainly design verification and validation, but it also supports to product verification. Design verification aims to check whether a selected design results in a system implementation that meets the requirements. Product verification aims to verify the actual system implementation against the specification. Although formal certification and qualification activities are also a kind of product verification, those are not addressed by VE.

Its ambition is to have a virtualized product where testing can be performed as on a real product by a dynamic real-time simulation. The following two testing methodologies are addressed by VE:

- Model-in-the-Loop: Test setup within design verification, which uses simulation models as unit under test. Allows functional & logical verification of the unit functional chains and behaviour of the interfaces.
- Software-in-the-Loop: Test setup within product verification, which uses a re-targeted or re-hosted target software as unit under test. Allows verification of the unit implementation.

VE sets the focus on the avionics system. Computational mechanics simulations (e.g. computational fluid dynamics, computational structural mechanics) are out of scope for VE. However, simulation models based on such simulations may be integrated by simplification or connected by co-simulation. Goal is to achieve real-time execution capabilities of the simulation. Also not addressed by VE are tasks in context of high-level architecture exploration, operational analysis and parameter optimization.

The term “simulation” can be understood as the execution of simulation models over time. It is important to understand the difference of the term “simulation model” from the MBSE model. Both kinds of models represent a system by describing its key characteristics, behaviours and functions in a simplified version. The MBSE model is not executable. It is an abstract and

formalized system description. In contrast, the simulation model is executable. It represents the system behaviour over time, acting and reacting on input data. The simulation model is not limited to the actual system in development. It also can represent an external system, a physical component or a phenomenon that interacts with the system. The simulation model is not limited to a simplified representation of a system component. In context of VE, it can also be the actual avionics target software, either re-targeted, or re-hosted. But the real physical target device does not fit anymore into the concept of a simulation model. Its integration with the simulation (hybrid configuration) is also not addressed by VE directly. It is driven by the product verification activities, which VE supports by providing the remaining system simulation around the unit under test in a hardware-in-the-loop test environment to ease its setup.

Since VE is located, similar to MBSE modelling, on the left side of the V-model, it needs to be tightly integrated within the overall system development process. In this phase, the actual system design has a low maturity. It is not fixed and incomplete. That means, VE needs continuously to respond on changes, and support quick fixes of definition gaps within the design. A certain degree of flexibility and adaptability within the processes and tools is necessary.

In order to setup the virtualized product and to make use of the opportunities described, a processes, methods and tools environment is established. Following building blocks are an essential in it:

- Simulation Breakdown Structure: Describes the hierarchy of simulation models required to develop and integrate a full system simulation. It is derived from the system equipment list. It describes the context and related equipment for each simulation model. Therefore it defines which equipment the simulation model represents, together with some meta-data. The simulation breakdown structure is complemented with physics, environment and simulation-specific models.
- Functional Increment / Artefact Roadmap: The Functional Increment Roadmap (FIR) describes which functions shall be realized by simulation models in which order, to which extent, in which fidelity at which point in time. The Artefact Roadmap (AR) describes which actual simulation provides a certain functionality at which point in time. Both together ensure that a certain functionality needed by one component is provided at the right time.
- Integration & Execution Environment: A set of software tools supporting the creation and integration of simulation models into an executable simulation. In addition, the simulation & test execution runtime and additional software tools are part. The toolset also provides the necessary connections to ensure exchange with the programme's interface management and test management solutions.
- Initial Simulation: Consist of an initial and generic simulation model set, integrated into an executable simulation. It initially describes a generic system of the same nature the programme intends to build, integrated with a natural and tactical environment. It intends to be used as starting point for the functional growth and helps to decouple the deliverables of different suppliers from each other.
- Environment for cooperation: Ensures that work can be coordinated and that information and assets are shared between all participants. It includes databases & repositories, together with a version control-, issue tracking-, and collaboration system. A special focus must be given to our programme setup, with accessibility by multiple companies, in different nations, with specific military and national regulations.
- Laboratories: Provides the physical integration and simulation & test execution environment accessible for all VE participants. Since VE addresses only virtualized avionics equipment, this environment can be identified as a virtual test bench. The actual laboratories are built on top of dedicated computers, or hosted within a cloud environment to ease accessibility and availability to the users.
- Joint Model Office: It is an organisation, including a set of roles, processes, standards and guidelines, to ensure the concurrent development of simulation models across all involved suppliers and their integration into a common simulation. It deploys the simulation back to all participants and laboratories. It provides all assets as described

before, and the necessary helpdesk and support.

The clear goal is to harmonize avionic simulation activities across various programme stakeholders.

The following example shall outline a typical use case, addressed by VE:

*If supplier 'A' of component 'B' needs to have a simulation model of component 'Y', created by supplier 'X'. 'A' should get the simulation model of 'Y' from 'X'. 'A' must not do it by himself, since 'A' might have a different understanding of 'Y' than 'X'. Otherwise, 'A' creates a version of 'Y', which perfectly works with 'B', until it is integrated with the real component 'Y'.*

Integration issues shall be detected as early as possible.

**Interface Management**

In between MBSE and the VE, lies the interface management. The main purpose is to detail the interfaces between subsystems, at all levels underneath the system, in context of their specific nature. Since this paper addresses avionics, the focus is on data nature, which includes logical interfaces (information flow between systems) & corresponding electrical interfaces (e.g. physical avionic network busses). Other natures, such as mechanical interfaces, are not addressed in here. Although those are also part of overall interface management.

The interface management tooling provides exporting capabilities for software coding and load analysis. However, the main result is the Interface Control Document (ICD). It captures the detailed data characteristic to ensure that interfacing equipment is compatible and can be integrated and operated as specified. It is the obligation of the equipment supplier to detail the information in negotiation with interfaced parties. The information is hereby stored within a common database, which also ensures the consistency to a certain degree.

The database is split into two interconnected sections. One section describes the interfaces of the actual system. This includes the subsystems, the logical interfaces in between. This includes the product structure with all equipments and their detailed logical and physical interfaces, and the relation to the system structure. The other section is specific to VE. It describes the Simulation Breakdown Structure with the entire set of simulation models and their relation to the system & equipment structure. By that, a simulation model inherits the interfaces of the component it references. In addition, simulation specific interfaces are described entirely in this part. This could be e.g. a physical parameter like the real outside temperature. It is provided by an environmental simulation model and consumed by a sensor model, which then outputs the sensed value to an avionic network interface it inherits from the referenced equipment. In addition, prototypic interfaces can be described in this section, which are so far not part of the systems interface model.

Since in our approach, both interface management tooling and the associated databases are separated from the MBSE environment, interface relevant MBSE model data has to be imported to the Interface Management environment. This separation leads to a break in the Single-Source-Of-Truth paradigm. In order to ensure digital continuity, automated export/import is used on one hand, and on the other a clear information ownership and change management process. As consequence, high-level changes such as a new interfaces or equipment must be made within the MBSE model, which is the owner of this information. These changes will then be reflected automatically within the interface management environment. Low-level changes such as the message formats are made directly within the interface management environment. This kind of information is not present within the MBSE model.

**Workflow Description**

In the following, the VE workflow will be described. The Fig 3 illustrates it.

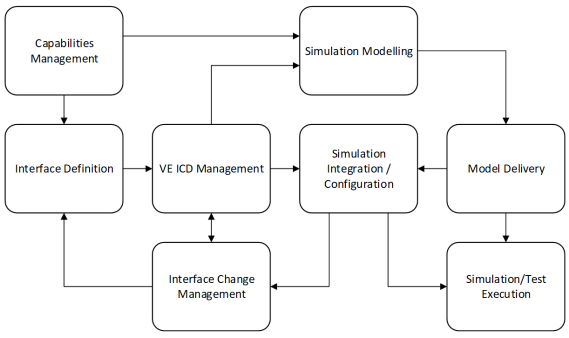


*Fig. 3 VE Workflow Overview*

First, as part of capabilities management the existence of the simulation model is defined, along with the avionic equipment's relationship, within the Simulation Breakdown Structure. In an iterative way, the Functional Increment & Artefact Roadmaps are defined based on the design and dependencies.

For each iteration, the avionic and simulation specific interfaces need to reflect the functional growth. The interfaces also need to be detailed enough to be usable within a simulation. This

means that data types and formats need to be defined, so that S/W programs can access it.

Since the Simulation Breakdown Structure is located in the same database as the data interfaces, the interface management environment acts as the Single-Source-Of-Truth for a model interface specification. It is exported and provided to the simulation model supplier together with the functional specification.

The supplier creates the simulation model. It can be hand-written code, or also auto-generated by using a Model-Based Engineering approach [5], which might use specific exports or generated templates out of the MBSE or Interface Management environment. The actual quality of a simulation is the better the more a simulation model relies on the same source code as the target software does. Therefore, the usage of re-targeted avionics software is also aspired.

The initial integration is done by the supplier before it is delivered according to the standards and guidelines the VE Joint Model Office has defined. In an automated process, the delivery is verified against those rules and the basic executability, before it becomes part of the simulation.

It is important to consider also specification changes made during the simulation model development and simulation integration. These changes are a result of either immature design or design mistakes. These changes need to be made at the actual data source. This can be the MBSE or interface management environment, with dedicated change management processes. But it is also important not to wait for the next iteration until a working simulation exists, maybe with other changes then necessary. Therefore a trade-off is made to quickly introduce changes within the simulation and in parallel trigger the actual change management process. An important goal is to have a running simulation in each iteration cycle.

Every participant has access to the simulation and can execute it to perform tests. These are equipment & model suppliers to perform unit tests (although most of these can be performed before the simulation model is delivered). These are also system and subsystem testers, which perform specific non-formal integration tests on their level. The test can be the same used later for the formal product qualification, but also specifically adapted tests for the virtual environment. It is important to understand that the virtual product-testing environment if not feasible for all kinds of tests. E.g., timing constraints cannot be meaningfully tested, since the actual real target environment has different execution times & performance. The tests execution is linked with the test management solution. Tests can be directly triggered from there, as well are test results uploaded to further process them there. Some tests require a deeper analysis and post processing of the test data produced during the execution. For this reason, a solution for test data analysis is integrated into the pipeline. It is foreseen to apply the testing toolchain in a continuous integration pipeline. Automatable functional and regression test shall be triggered automatically once simulation models have been updated.

Our current reality is that almost all system level tests require manual interaction. Either during the test via virtualized human-machine-interfaces, or within the post processing / analysis step. It is our ambition to make tests more automatable.

**Environment Setup**

Now a brief overview shall follow of our experience to setup the environment. In our programme it was identified that the MBSE and VE approach would require preparation before the programme development begins, when the major contribution is within the definition phase. There the entire environment with processes, tools and infrastructure would need to be fully ready. By the time the preparation started, the MBSE approach was not defined, nor was the VE approach fully outlined. Unfortunately, in parallel the majority of contracts with partners and required deliveries were defined without knowing the MBSE and VE needs in detail. Since both, MBSE and VE require commitment and contribution by all stakeholders, many re-negotiation effort had to be performed.

In our industrial environment, testing is often seen as something to be done at the end of the classical development process. This lead to not adequate priorities inside programme management to support the MBSE and VE definition. Since VE is also a step towards test driven development, which changes traditional working methods, anxieties and resistances were created in various engineering teams. Therefore, it is important to have a clear and proven concept even before start of the programme concept phase. That the link between MBSE and VE is done only by interface management, and the limited usage of MBSE are consequences that both approaches were not defined then.

**Summary**

Virtual product testing is performed on the means provided by Virtual Engineering to support design & product verification. Our MBSE and Virtual Engineering environments and

workflows are not directly linked with other. Implicitly both are connected through the interface management environment in a digital continuity. However, this is limited to structural and interface information. Although other aspects such as requirements and functional design are reflected in simulation models, the information does not have digital continuity. Manual lookup and transformation of design information during simulation model implementation is necessary. This is error prone. It can also lead to the decoupling of the MBSE model from the actual implemented reality. The more closely functional models are linked to target software, and the more the target software is based on auto-generation from MBSE, the more meaningful the virtual product testing.

Based on our experience, it is fundamental to define the detailed approach of “How to get from MBSE to virtual product testing” before the programme concept phase starts. Only then it is possible to involve all stakeholders appropriately and to steer the necessary change process. It may also lead to a deeper integration with a full digital continuity and a true single-source-of-truth paradigm.

## References

[1] NIST Planning report 02-3, The Economic Impact of Inadequate Infrastructure for Software Testing, (May 2002)

[2] International Council on Systems Engineering (INCOSE), INCOSE Systems Engineering Vision 2020, INCOSE-TP-2004-004-02 (Sep 2007)

[3] F. Bouffaron, Airbus MBSE Framework: Model Execution of System Architectures (MOFLT), *MBSE Cyber Experience Symposium 2021 – Japan*

[4] Object Management Group, OMG System Modeling Language, *OMG SysML® home page (Apr 2022*), https://www.omg.org/spec/SysML/

[5] Model-Based Engineering Forum, *home page (Apr 2022)*, https://modelbasedengineering.com/