Digital Twin Validation - Methods and Techniques

[WP2; T2.5; Deliverable: D2.5]

Public

AI training using Simulated Instruments for Machine Optimization and Verification

Version	Status	Date	Page
1.0	Public	2024-01-31	1/46

Document Information

Project	ASIMOV
Grant Agreement No.	20216 ASIMOV - ITEA
Deliverable No.	D2.5
Deliverable No. in WP	WP2; T2.5
Deliverable Title	Digital Twin Validation - Methods and Techniques
Dissemination Level	Public
Document Version	1.0
Date	2024.01.31
Contact	Michael Wild
Organization	DLR e.V.
E-Mail	michael.wild@dlr.de

The ASIMOV-project was submitted in the Eureka Cluster AI Call 2021
https://eureka-clusters-ai.eu/

Task Team (Contributors to this deliverable)

Name	Partner	E-Mail
Michael Wild	DLR	Michael.Wild@dlr.de
Mehrnoush Hajnorouzi	DLR	Mehrnoush.Hajnorouzi@dlr.de
Niklas Braun	AVL	Niklas.Braun@avl.com
Arjan Mooij	TNO-ESI	Arjan.Mooij@tno.nl
Bram van der Sanden	TNO-ESI	Bram.vandersanden@tno.nl
Mustafa Majar	TrianGraphics	Mustafa.Majar@triangraphics.de
Jan Willem Bikker	CQM	Janwillem.Bikker@cqm.nl
Andreas Eich	LiangDao	Andreas.Eich@liangdao.de
Narges Javaheri	TFS	Narges.Javaheri@thermofisher.com
Roy van Zuijlen	TU/e	R.A.C.Zuijlen@tue.nl

Formal Reviewers

Version	Date	Reviewer
1.0	26.1.2024	Ezra Tampubolon (Ezra.Tampubolon@norcom.de)
1.0	26.1.2024	Jan van Doremalen (jan.vandoremalen@cqm.nl)

Change History

Version	Date	Reason for Change
0.3	2023.12.07	Initial Version. Ready for internal content review
0.6	2024.01.23	Ready for formal review
1.0	2024.01.31	Formal review done

Abstract

The ASIMOV project and the proposed ASIMOV Solution use a digital twin (DT) instead of the real system to provide training data for an artificial intelligence (AI) algorithm that shall optimize the cyber physical system. More specifically a Reinforcement Learning Agent (RLA) is used, and it's paramount to address the validity of the DT to ensure that the RLA learns the right policy. The present document is concerned with verification and validation (V&V) topics that are of special relevance in the ASIMOV Solution and put the more traditional V&V approaches for digital models into context. These topics include having different purposes of the DT during the four steps and three phases that occur in the development of an ASIMOV Solution, as well as the existence of a physical counterpart and (in later phases) the existence of a fleet. The DT is interpreted as a subsystem of the ASIMOV Solution, for which functional and non-functional goals are derived. For each of the derived functional goals, a validation metric is proposed. For each of the two main use cases of the project, the Electron Microscope, and the Unmanned Utility Vehicle, some V&V specific topics are highlighted, and some concrete validation metrics are applied.

Contents

Table of Figures

Table of Tables

List of Abbreviations

Abbreviation	Meaning
AI	Artificial Intelligence
CBED	convergent beam electron diffraction
CPS	Cyber-physical System
DT	Digital Twin
EM	Electron Microscope
FGAS	Functional goal of the ASIMOV Solution
FGDT	Functional goal of the DT
KPI	Key Performance Indicator
NGDT	Non-functional goal of the DT
NGAS	Non-functional goal of the ASIMOV Solution
NN	Neural Network
ODD	Operational Design Domain
PT	Physical Twin
RL	Reinforcement Learning
RLA	Reinforcement Learning Agent
RS	Real System
SOTA	State of the Art
SOTP	State of the Practice
(S)TEM	(Scanning) Transmission Electron Microscope
TG	TrianGraphics
UC	Use Case
UUV	Unmanned Utility Vehicle

1 Introduction

The task to validate a digital twin (DT) within the ASIMOV Solution differs in some key factors, e.g. because of the fleet aspect of DTs, or the specific purpose of the DT to provide useful training data to an Artificial Intelligence (AI), from traditional verification and validation (V&V) tasks developed for digital models. This document gives a brief overview of existing V&V methods, highlights challenges when applying them within the ASIMOV Solution, and proposes methods to address these challenges.
In our understanding of the term DT, we mainly orient ourselves on the works of Grieves et al. [1], [2], Wagg et al. [3], and the survey from Jones et al. [4]. Further, we build on the findings of the ASIMOV Working Group [5] e.g. the four steps in the development of the ASIMOV Solution. They are the initial development, the connection of the DT and the AI, the connection of the AI and the cyber physical system (CPS), and the connection of the AI and other CPSs. We include topics related to twinning (synchronization) (see [6]) and end up with different loops between the DT, the real asset, which we will call physical twin (PT), and the reinforcement learning agent (RLA) which takes the role of the AI in the ASIMOV Solution.
This deliverable is structured the following way: First we give an overview of the state of the art in V&V of digital models. Next, we explain the ASIMOV Solution and the use cases, focusing on the DT and its purpose within the ASIMOV Solution. Building on that, we describe the validation efforts within ASIMOV, focusing on DT specific topics (which extend the concept of a digital model) and on the purpose of the DT which is to provide useful training data to a RLA.

2 SOTA Verification and Validation of Digital Models

In this section we will give a short overview of the state of the art of V&V for digital models. In order to delineate "verification" from "validation" activities we build on the ISO standard 9000:2015 [7], where *verification* is defined as "confirmation, through the provision of objective evidence, that specified requirements have been fulfilled" and *validation* as "confirmation, through the provision of objective evidence, that the requirements for a specific intended use or application have been fulfilled". In other words, verification can be thought of as "building the model right" and validation as "building the right model" [8]. Additionally, the System Engineering Body of knowledge highlights that

> *"within verification, comparison between the expected result and the obtained result is generally binary, whereas within validation, the result of the comparison may require a judgment of value regarding whether or not to accept the obtained result compared to a threshold or limit. Verification relates more to one element, whereas validation relates more to a set of elements and considers this set as a whole. Validation presupposes that verification actions have already been performed. The techniques used to define and perform the verification actions and those for validation actions are very similar" [9].*

2.1 Guiding Principles

Computer-based models have existed since the 1950s and V&V has been a topic from the beginning. Therefore, the body of methods and research is quite large. Even though a lot has changed over the years [10], [11], the fundamental principles are still valid. Balci [12] gives 15 such guiding principles out of which we list three that are of special importance for the ASIMOV approach and give a short explanation why.

V&V must be conducted throughout the entire modeling and simulation (M&S) life cycle. The ASIMOV Solution has steps and phases and the purpose of the DT changes during those (see Sec. 4.2.1 for details). This principle is the base for a hierarchical exploration of functional and non-functional goals of the DT in Sec. 4.2. In order to have a reference for a M&S life cycle, we orient ourselves on the one given by [8] which is shown in Figure 1. The steps in the green box labeled "Conduct Verification, Validation, and Accreditation" are closely mapped to the steps in the light blue box labeled "Develop New M&S". Both start with determination of requirements. This is a relevant topic for validating the DT within the ASIMOV Solution, because since the quantification of how a black box RLA learns its policy is not understood completely and is actively researched in the field of explainable reinforcement learning (XRL)

[13] and explainable AI (XAI) more generally. It is also not clear how to give the required minimal accuracy for the output of the model. We address this in Section Validation on Raw Output vs Validation on Postprocessed Data 4.4.2.

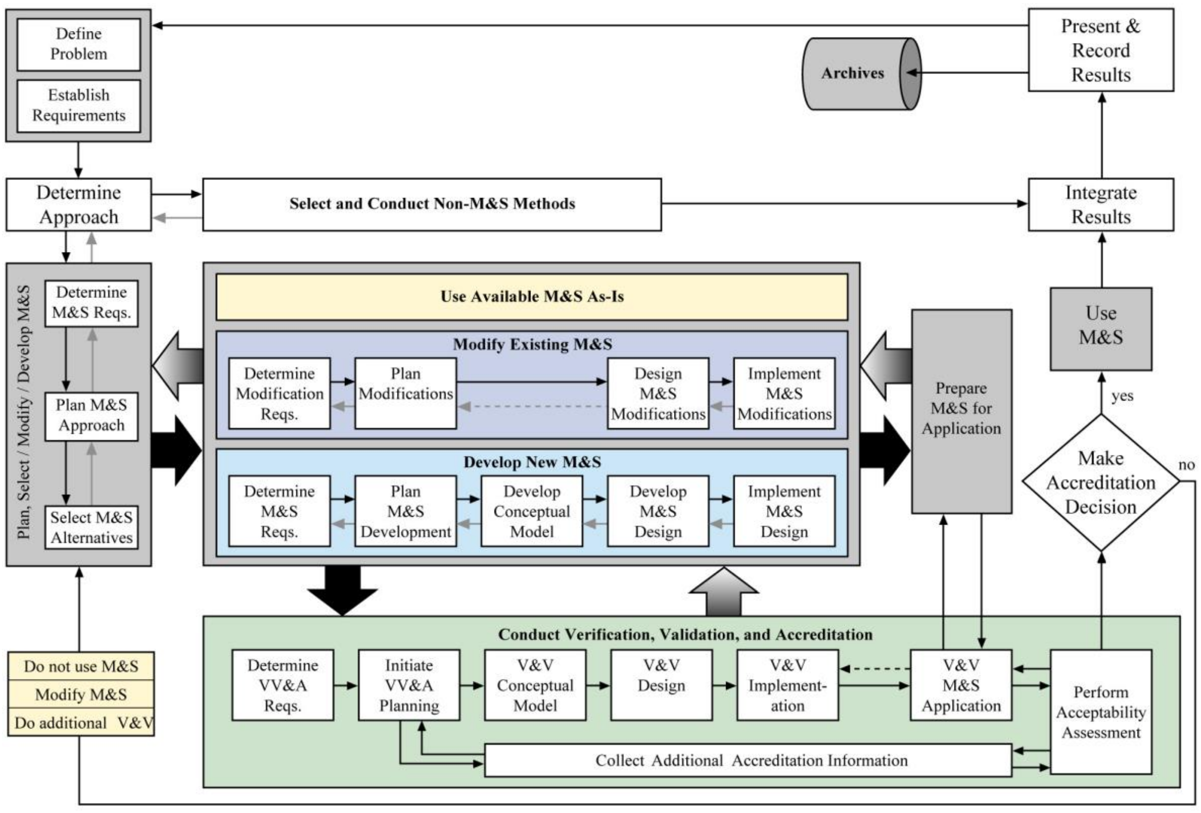


Figure 1 VV&A activities in the M&S life cycle ([DoD96, p. 3-18]) (taken from[8])

The outcome of Verification, Validation and Accreditation (VV&A) should not be considered as a binary variable where the model or simulation is absolutely correct or absolutely incorrect. This principle is relevant here, since e.g. in the scanning transmission electron microscope (STEM) use-case images (so called ronchigrams) created by the model and the real STEM will never be exactly identical. There, validating the correct trend behavior of lower dimensional features is a feasible approach.

A more simplified abstraction of a model development process is shown in Figure 2. The *Problem Entity* is the system to be modelled. The *Conceptual Model* is the mathematical or logical representation of the problem entity developed for a particular purpose. The *Computerized Model* is an implementation of the conceptual model that can be executed. In the resulting triangle, four validation and verification activities are given:

1. *Conceptual Model Validation:* determine that the theories and assumptions underlying the Conceptual Model are correct, and that the model aligns with the intended purpose;
2. *Computerized Model Verification;* assure that the implemented Computerized Model is correct;
3. *Operational Validation;* determine that the model's output behavior has the required accuracy given the intended purpose and application domain;
4. *Data Validity;* ensure that the relevant data for model creation, evaluation, testing, and experiments is adequate and correct.

In Section 0 we use this triangle as a framework to position the validation activities performed in the ASIMOV context.

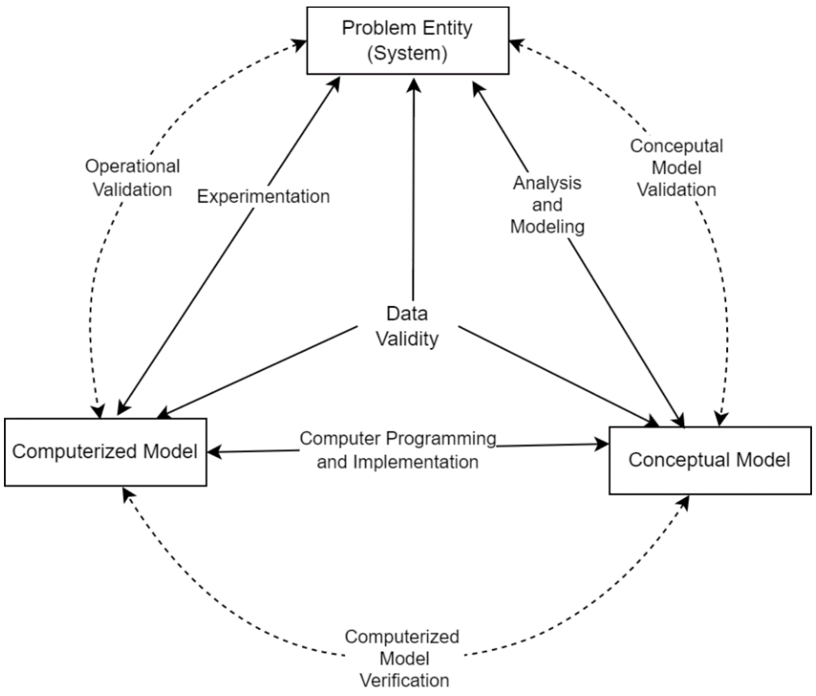


Figure 2 Validation and verification tasks in a simplified modeling process. Reproduced from [14].

Another guiding principle shall be emphasized in the context of data validity.
The double validation problem must be recognized and resolved properly. What is meant by this is that the approach of presenting both the actual system and the model with the same inputs and comparing the outputs to judge the validity of the model is only valid if it has been validated that the model and system inputs match each other with sufficient accuracy. This test is also referred to as input data model validation.

2.2 Concrete Model Validation Topics for Specific Models

2.2.1 Validation of Emergent Behavior

Emergent behavior refers to phenomena that arise from the interactions of different system's components and leads into patterns or behavior that cannot easily being predicted from the behavior of individual components. Grieves et al. call this kind of system behaviors Unpredicted Undesirable (UU).

> *"The UU category holds the potential for serious, catastrophic problems. It is this category of emergent behavior that we need to focus on. We need capabilities and methodologies to mitigate and/or eliminate any serious problems and even reduce unforeseen problems that are merely annoying. This is the purpose of the Digital Twin model and methodology" [1].*

In general, a single system does not generate emergent behavior. For instance, it is a system of systems which can create an emergent behavior [15], where the output is more than the sum of added component systems and it is mostly non-proportional to the inputs. Axelsson has a different taxonomy for phenomena which uses strong and weak emergence instead of Unpredicted and Predicted Undesirables. The common idea is, that a class of phenomena exists (strong emergence), that cannot even in principle be deduced from the element-level phenomena and structures [16].
Within the context of DT, these behaviors are not explicitly encoded (or programmed) in the model of components. Instead, they emerge from the dependencies among the interconnected models. The idea is to build a DT that is a "good" replica of the PT. Therefore, DT should be able to generate (or predict) emergent behaviors too. To this end, the DT should encompass rare and critical states within its state

space and the transitions that drive the DT into those states. Here, it is important to insert uncertainties in the modeled transitions to enhance the DT's capability to capture unpredictable emergent behaviors.

2.2.2 Validation of State Transition Towards Behavior Validation

The Markov property is a key assumption in reinforcement learning to simplify the model of interactions between agent and environment. It ensures that the state contains enough information for the RLA to make optimal decisions and learn effective control policies. This property implies that the agent does not need to maintain a detailed history of past events, which is crucial to make an optimal decision.
In the ASIMOV approach the DT is the environment of the RLA or the external system with which the RLA interacts (during the training phase). The DT provides feedback to the RLA in the form of a reward and a transition to the next state, for the action it has decided to take. One aspect of the validation of DT is concerned with validating the Markov property, which requires an iterative analysis and tuning process of state space and transition function.
In the following, we first briefly discuss the aspects of interaction between the DT and the RLA.

- **State space.** The DT should be able to encode the state of the actual system. Therefore, we ensure the defined state space of the DT fully characterizes the actual system and captures all relevant information and features that are necessary for the RLA to make decision and select (i) an action and (ii) a transition to the next state to occur. The state representation should include sensor data, environmental conditions, internal variables and other factors that can influence the behavior of the actual system. Every state of the actual system that is relevant to the optimization goals should be covered by a state in the state set of the DT. In addition, all states of the DT should be reachable from at least one of the other states and consequently from the set of applicable initial states (ergodicity property).
- **Markovian transition function.** The conditional independency of the Markov property implies that the probability of transition from one state to another in one time step depends only on the current state and the action taken and not on the events that occurred before. The integrated models of a DT are developed on the basis of physics principles, differential equations or data-driven approaches. However, they should represent the dynamics of the system as accurately as possible for the purpose of optimization. The overall transition function between states should incorporate adjustable parameters to be tuned over time to constantly match the behavior of the DT with the actual system. In addition, the probabilistic transition functions should be checked against the frequency of reaching different states.
- **Reward function.** The reward function quantifies how well the RLA is performing the desired optimization task. The reward should be shaped to be based on immediate state-action pairs and expected future rewards and not on the history of those. Moreover, the reward function should closely mimic the rewards that the RLA would receive from the actual system in order to effectively guide learning.

It should be ensured that the DT accurately represents the underlying system dynamics. Furthermore, as a Markov model, the state transition should only depend on the current state and the action applied to the system, and not on the sequence of states that preceded the current state. In order to design a framework for the validation of a DT, it is essential to specify the state space and transition function in advance. The state transition events (e.g. from state A to state B) are defined by considering the dynamics of the system. The result is a state space together with the set of transition events between the states. The state space of the DT comprises the possible states that DT can reach. The states are expressed by the valuation of the relevant parameters that are identified (for more details refer to [17]).

Complex systems are typically constructed by integrating smaller systems or components. Similarly, the DT here represents a complex model incorporating several smaller models, each of which has its own state space. If the smaller systems behave independently, the state space of the overall system can be a direct product or a vector resulting from the concatenation of the state spaces of the smaller systems. However, the assumption of independent components is not relevant for most of the complex systems, including our use cases. Given the fact that the smaller systems interact with each other, defining the state space of the complex system becomes a non-trivial task. To this end, we require to apply methods to define the DT state space, such as Bayesian hierarchical models to construct a hierarchical state space from the state spaces of the smaller systems [18], MCMC (Markov Chain Monte Carlo) methods [19], or by applying model-learning techniques on the generated traces of simulated DT.

By simulating the DT with a concrete scenario, we can collect data of pre- and post-states of the successive steps of the scenario. The simulation of DT by using a number of different scenarios, leads us to construct a formal specification model of the DT behavior which can later be used for the correctness checks. The simulation results can be compared with real-world data using statistical tests and validation metrics to evaluate the accuracy and predictive power of DT. It is an iterative process where the state space and transition function need to be refined based on the validation results to improve the model's fidelity. Meanwhile, sensitivity analyses are helpful to understand how variations in transition function and initial conditions may affect the outcome behavior.

Before designing experiments for the model validation, we should define the objectives of validation in terms of specific aspects of DT, such as the accuracy of transition probabilities, the prediction of system behavior, or the robustness of the model in different scenarios. The selection of validation metrics should be aligned with the validation objectives. On this basis, we can establish a set of validation criteria that must be met for the model to be considered valid.

The development of a DT for a complex system demands several phases, with each phase requiring specific validation techniques. By integrating these validation techniques in different phases, we build confidence in the DT. During the design phase, it is important to validate that the DT is able to capture the underlying dynamics of the system. For the data-driven models, it is essential that the synthesized data by running various scenarios mimic the behavior of the system. The technique K-fold cross validation [20] can be helpful in this case. While simulating the DT, it is important to understand its behavior under different conditions. To introduce variation in input parameters, Monte Carlo simulation is helpful. By simulating the DT with the resulting different scenarios and comparing the simulation results with the expected behavior, we can evaluate the DT. The extreme or unexpected scenarios are useful to assess the robustness of the DT and further validation with respect to the prediction accuracy of the DT. Sargent defines the operational validity as the determination that the output behavior of the model has sufficient accuracy for its intended purpose over the domain of its intended applicability [14].
NASA technical standard for models and simulation, STD-7009A defines validation as *"The process of determining the degree to which a model or a simulation is an accurate representation of the real world from the perspective of the intended uses of the M&S"*.
For the purpose of validation, we can assume three different situations depending on data availability:

1. No data available from the real system
2. Available data only on the output of the real system
3. Available data on the output, the corresponding input and the trace of the real system

Depending on the situation listed, the validation strategy or techniques must be selected. While in the first situation, expert knowledge or the verification of the model internal structure are useful, in the second and third situations we can conduct experiments and compare the resulting behavior with that of the real system. Different types of models can benefit from different validation techniques to ensure their robust and accurate performance. The validation of a "white box" model, i.e., causal- or theory-descriptive, is different from the validation of a "black box" model, e.g. data-driven. While in black box modeling, the model is considered valid if its output behavior matches the observed output of the real system within some specified range of accuracy, a valid white box model should produce/predict the output of the system and explain how the output behavior is generated [21].

The system dynamics models are mainly white box models, and their simulations should be validated to accurately represent the interactions between parameters. The validation of system dynamics models is the process of establishing confidence in soundness and usefulness of the model, and thus the validity becomes a relative concept.
In a relativist/holistic view, a valid model is one of many possible ways to describe a real system/event. "Models are not true or false but lie on a continuum of usefulness." [22]
The confidence in a system dynamics model is gradually accumulated as the model passes the behavioral and structural tests introduced by [23]. These tests are used to compare the model behavior

with descriptive knowledge about the system, and to compare the model behavior with the observed system behavior. There are three main categories of tests:

1. Direct structure tests: They evaluate the validity of individual model equations by comparing them with available knowledge, either from the real system that is modeled or a generalized knowledge from literature. The simulation of the model is not required for direct structure tests. The examples of this type of test include structure and parameter verification tests, direct extreme condition test and dimension consistency tests.
2. Structure-oriented behavior tests: They indirectly evaluate the validity of the model's structure by analyzing the behavior patterns generated by the model. The examples are extreme condition (behavior) test by assigning extreme values to a set of selected parameters, the behavior sensitivity test and modified-behavior prediction.
3. Behavior pattern tests: They focus on how well the model can replicate the major behavioral patterns that are observed in the real system, i.e., the emphasis is on predicting patterns such frequencies or trends rather than point prediction. The example of this type of test includes the multiple-test procedure.

2.2.3 Validation of Data-Driven Models

A model that is derived from a set of empirical data is a type of black box model. Model validation of a black box model should check whether the model is a good approximation to the data.
There are various validation techniques described in literature, including:

- **Visual inspection of the output of the model (also called graphical comparison** [12]**):** it is very important to perform visual inspections, to see whether the model approximates the data in a good way. A classic example stressing the importance of these inspections is Anscombe's quartet [24]. In this example, shown in Figure 3, the black box model is a linear regression line. All four data sets have the same values on all typical summary statistical properties like mean, sample variance, correlation, and coefficient of determination of the linear regression. However, a visual inspection immediately shows that the data sets vary considerably, and the model is not a good representation for these data sets. The method of graphical comparison is valuable, but essentially qualitative [25].

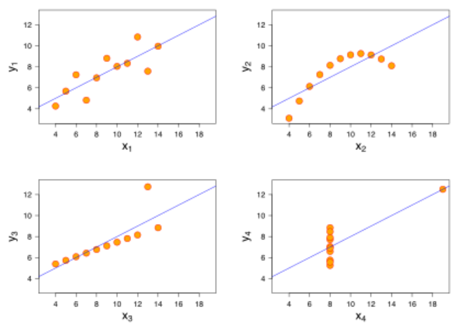


Figure 3 The four sets of Anscombe's quartet. Figure taken from [26].

- Related to this visual inspection in case of a statistical model (such as linear regression) is checking the underlying modelling assumptions. A notable example are the residuals (differences between observed and predicted values). In regression, they should be identically and independently distributed. Certain visual inspections can reveal problems, e.g. residuals in observation order (are residuals of nearby observations more similar to each other, do they have constant variance), outlying residuals, and the distribution of the residuals (severely non-normal?). Also, residuals versus a model predictor, or versus another variable not in the model (day-in-the-week) may reveal problems.
- **Sensitivity analysis:** determine how sensitive the output values of the model are to changes in the input values. Sensitivity analysis is closely related to determining stability of the model, whether a small change in an input value causes a problematic large change in the output.

Techniques for sensitivity analysis are found in design of experiments and regression analysis [27], [28]. Insights in sensitivity, a.k.a. effects of predictors on output, also allow interpretation from an expert view of the system; for instance, do the sensitivities have the correct sign, are the sizes as expected. The concept is also linked to causality: what if a single input is changed while holding all others constant, whereas in the training data usually change simultaneously.

- **Independent test set and cross validation**. In machine learning / supervised learning, a common method is to split the data at hand into a training set and a test set randomly, e.g. in a 70-30 division. Then the model is built (“trained”) just using the training set, and evaluated on the test set, resulting in a value of some error metric. However, often the dataset to start with is small, so that the test set is small. This means that the error metric is actually uncertain as an estimate of the true error on all future observation, or some imagined large test set. For this reason, often a scheme different from a training-test split is taken, e.g. k-fold cross validation. In addition, confidence intervals can be used to express the uncertainty of the error metric, see [28].

3 Description of the ASIMOV Solution and the Use Cases

The ASIMOV Solution is explained in the ASIMOV Reference Architecture Document [5]. We will repeat and highlight some aspects that are relevant for the validation task. Then we'll do the same for the two use cases, STEM and UUV. Also, relevant aspects of the state of the practice (SOTP) for V&V in different domains will be described. We will give weight to the purpose of the DT, since it is the base for validation [29]. Looking at Figure 4, we can see that the Validation of a DT cannot be put at a specific step in the DT development process but has to happen at every step.

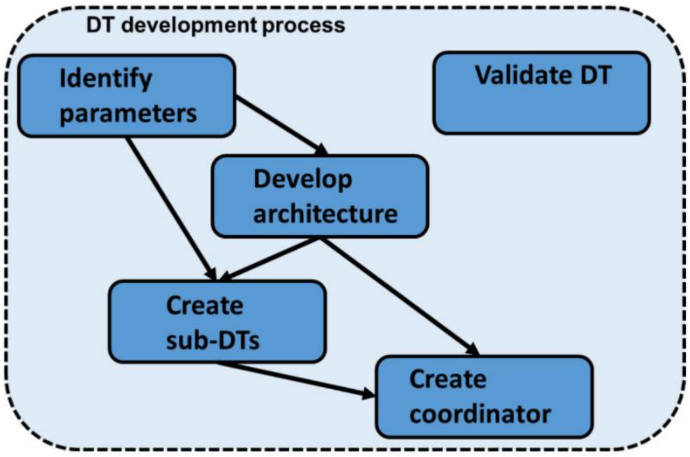


Figure 4 Simplified development process of a digital twin. The blue boxes indicate the development steps, and the black arrows indicate the dependency / order. Figure copied from [5].

Considering the architecture in Figure 5, we can see the connection of the System and the RLA. Here it is left open if we mean the PT or the DT as System. In theory the RLA could be trained on either, but to circumvent long response times, limited availability, and potentially harm to the PT, there is a strong preference to use the DT. The figure also covers the pre- and postprocessing, which are discussed in [30].

We identify some functions the System must be able to execute:

- Provide an **Output y**, from which a **State S** and a **Reward R** can be derived.
- Accept a **Control Input u**, which is derived from an **Action A**.
- Accept **Disturbances d** as further input.

Further details are given in [30]. **System parameters c** influence the behavior of the system.

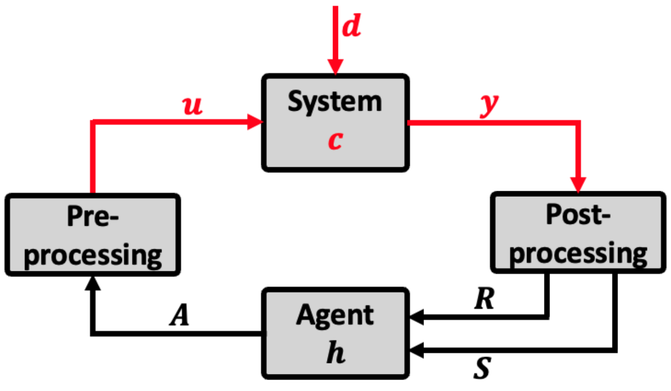


Figure 5 AI-training architecture. Description of the letters in the text. Figure taken from [30]

Before we dive into the validation of the DT in ASIMOV in a later chapter, we pick up on the conceptual layers identified in [5]. There, the ASIMOV Solution is seen as a system, which has its own purposes (or qualities, goals). These include performance, scalability, etc. Ideally, we can provide traceability from these purposes to the ones of the DT, which can be seen as one of its constituents (sub-systems).

3.1 UC UUV

The use case is described in [31]. We will briefly reiterate and focus on V&V topics here.

3.1.1 Explanation / Framing

In this use case the optimal set of scenarios, which shall be used to calibrate a UUV in its operational design domain (ODD), shall be found. Optimal with two objectives: The scenarios shall be novel (no unnecessary tests) and critical. In order to quantify novelty of a scenario an autoencoder is used. It is trained on previously considered scenarios and then presented with a scenario whose novelty shall be estimated. A high error in reproduction indicates novelty [32]. As for the criticality of the scenario, the method described in [33] is being applied to derive a suitable set of criticality metrics.
The basic loop between the RLA and the DT is shown in Figure 6.

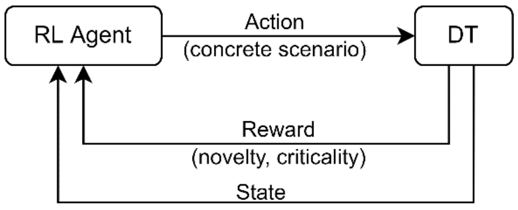


Figure 6 The basic RL loop between the DT and the RLA in the UUV use case.

The system in this use case can be characterized as complex, because some of its properties cannot be explained solely by analyzing its subsystems but are emergent [34]. To elaborate on this notion, we deconstruct the system. The system (same for DT and PT) itself contains an environment. This environment, whose main purpose is to provide stimulus for the sensors (also as a function of the actuators), needs to be abstracted when it comes to the topic of its validation. The green/yellow components in Figure 7 can be validated directly because they exist in both the PT and the DT. Also, the orange component (the environment) though it is completely on DT side, can be validated both internally and in terms of its output to the other system components. However, the driving function (gray block) is explicitly out of scope, since it is the system that shall be tested/validated for its operational design domain (ODD) with the resulting scenarios.

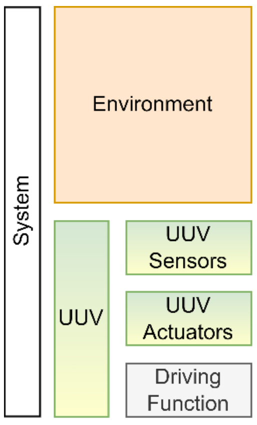


Figure 7 Internals of the System in the UUV use case.

3.1.2 Pinpointing the PT

Since one of the characteristics of a DT is the existence of a PT, we want to pinpoint what the PT is in this use case. We start by recognizing that the DT delivers the criticality and novelty of a scenario that was proposed by the RLA (see Figure 6). The corresponding PT would be the PT of the UUV existing in the PT of the environment (which is the real open world). This is impossible or at least highly impractical, since the actions of the RLA are the scenarios, which would need to be enacted in reality.
Therefore, in the first abstraction of the PT (abstraction_pt_01), the open world is removed from the equation, and abstracted in such a way that part of it is realized as the (PT of the) testbed, and part of it as the virtual environment.

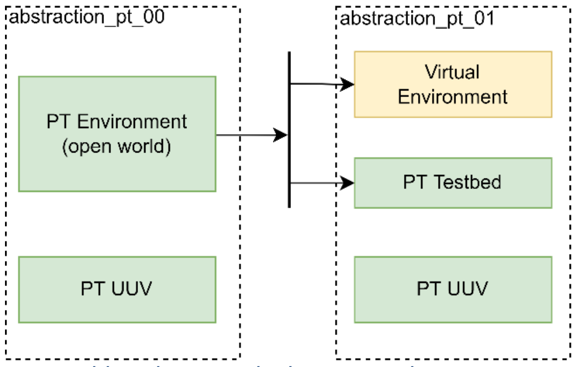


Figure 8 Abstraction of the open world environment in the PT. Real components are green, virtual components yellow.

The DT can be abstracted in a similar way (see Figure 9). The DT of the environment can be compared to an open world video game in abstraction_dt_00. This is especially in contrast to the process described in [35], where a real-world scenario is recorded in the real open world with a real car. This is then replayed in the simulation and is the base for the sensor validation.

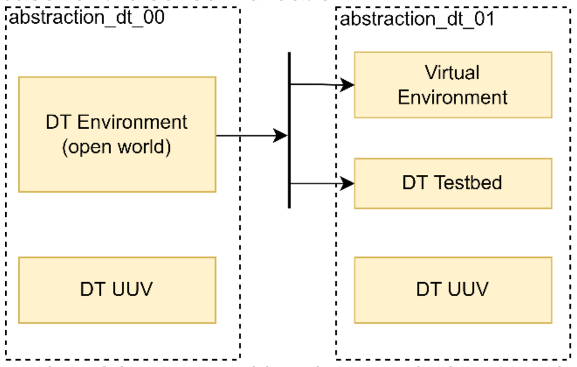


Figure 9 Abstraction of the open world environment in the DT. Colors like above.

With the proposed abstraction of the PT and the DT of the environment, the DT and PT of the testbed can clearly be identified. Twinning and operational model validation is possible. Figure 10 shows the updated Figure 7 with the proposed abstraction for both the DT and the PT. This is in line with the modularity principle, described in [36], Sec 3.4.2.

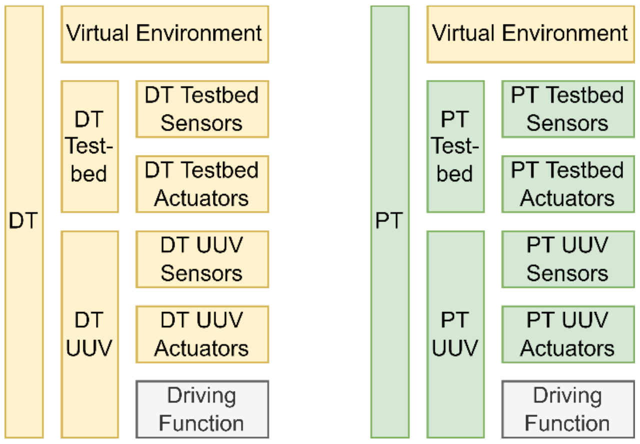


Figure 10 Internals of the System in the UUV use case, updated.

Procedures to validate the sensor and actuator models, as well as the virtual environment, already exist and will be summed up in the next sections.

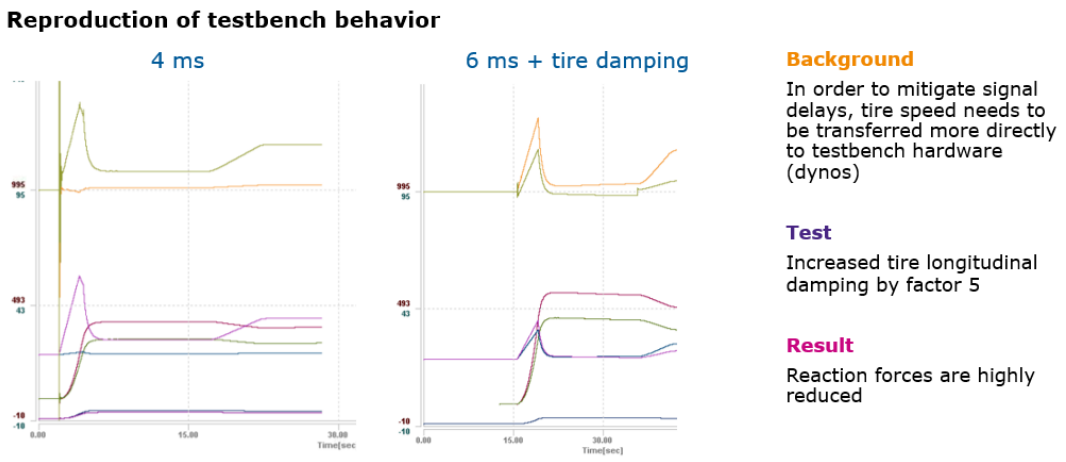
3.1.3 Testbed Models Validation SOTP

The ideal testbed is able to handle even the most dynamic longitudinal driving maneuvers of the vehicle, while being able to provide accurate wheel speeds and force/torque measurements without any latency. In reality, there are influences that make the testbed deviate from this ideal situation.

- Mechanical Inertia: The rotational mass of the dyno itself (even more relevant on chassis dynos than on powertrain testbeds) leads electrical power being used or released, during speed changes, even without any vehicle on the testbed. This has to be measured/calculated beforehand to be able to compensate these effects in a test run.
  This aspect of the model can be checked by measuring the power that is required to spin up the (virtual) testbed with a specific speed gradient.
- Electrical Power: The electrical power and torque of the electric motors of the testbed have their limitations. Especially highly dynamic maneuvers can bring the electric motors near these limits. It is therefore helpful to have a model of the testbed, to check these dynamic limitations beforehand.
  This aspect of the model ideally uses the characteristic power curve of the testbed manufacturer. Otherwise, it can be measured and validated by running tests that exceed the capabilities of the testbed where the dynamics are therefore limited by the electrical power of it.
- Controller Delay: The fact that torque is being measured, a simulation step is being executed, a new target velocity is computed and then a speed controller dials in the new speed, has effects on the response of the testbed. A real road would have an immediate response, while the testbed needs to process the signals and is therefore limited by computational power and the frequency of operation. There are possible countermeasures to reduce the effect of these delays, but these also require a model of the testbed, to estimate how strong these effects are in the first place.
  This aspect of the model can ideally be implemented by transferring the delays due to signal transmission and processing steps to the virtual model. These delays are typically given in the

form of refresh frequencies of signals and controller functions. Figure 11 shows the effects of delayed signals next to a well-tuned system with stiffer tire parameters.

-



-

Figure 11 Effects of delayed signals

-

3.1.4 Vehicle Dynamics Models Validation SOTP

The validation of vehicle dynamics models depends on the intended purpose. An optimization of the vehicles driving dynamics from a driving performance point of view might require a detailed and complex model, that allows for very accurate kinematics and force reproduction in simulation. Suspension setup testing for racing applications would be a situation where such a very accurate dynamics model is needed. A less complex dynamics model, as required for ADAS function testing, might still produce similarly accurate results in scenarios with moderate dynamic behavior.
The validation of the vehicle dynamics model therefore starts with the exact definition of the required dynamics. Lower dynamics are typically easier to replicate with less complex models, as non-linearities are more common in highly dynamic scenarios. Dynamics can then be further divided into lateral or longitudinal scenarios. Although the underlying dynamics model is typically a common one, longitudinal and lateral dynamic maneuvers are stressing different parts of the model. This can be used to target different groups of model parameters in validation with different driving maneuvers.
A good example for validating the longitudinal dynamics of a vehicle is by comparing the results of a virtual and real coast-down test, where the vehicle is decelerating from a certain starting velocity up until stand still. As the vehicles speed changes during the experiment, it is possible to identify velocity dependent and independent driving resistances. These can then be compared between virtual and real, to guide tweaking of the parameters to match the results. Comparison can be done by comparing time series data by visually inspecting the respective curves. Automatic approaches are also available. Most of the parameter-tuning can hereby be done in a single shot, while finer details may require an iterative approach.
Similar tests can also be done for acceleration and sine wave or circle driving. These validation methods already build the foundations for a sufficiently accurate ADAS Model.
Detailed experiments and crosschecks are of course necessary, if accuracy for certain components is required. The general procedure is, however, similar. The dynamic limits of the simulation model are defined. Experiments around these limits serve as ground truth data, while the accurate reproduction of the models' outputs can then be evaluated by comparing simulation and reality.
It is also important to note, that most vehicle dynamics tools are not a blank sheet of paper, but rather offer templates for specific vehicle types, suspension and steering geometries, kinematics, etc. This leads to specific physical parameters being already requested by the software.

3.1.5 General Sensor Model Validation SOTP

Magosi et al. provide a survey on models of radar sensors for virtual test and validation. Their considerations are also valid for different kinds of sensors. They distinguish between operational, functional, technical, and individual sensor models and highlight where they can be used in the V-Model [37]. Further, sensor models are distinguished between physical sensor models (also called white box models) which incorporate the underlying physics of the respective technology, and data-driven (or black box) models that learn their operation based on recorded data.
In the field of sensor model validation, it is paramount to compare the simulation results with experimental data. The latter can be easily collected by logging the data produced during a real test drive. Synthetic data can be produced by following the 6-step process as described in [38] and [35]. The steps will be summed up briefly:

1. An experimental drive is performed and recorded
2. A virtual scenario is generated
3. Virtual sensor models are parameterized
4. Execution of virtual test drive
5. Recording of synthetic sensor data
6. Comparison of data from 1. and 5.

According to [39] the comparison from step 6, (which uses validation metrics) is only one of three steps that are needed. The other two are “interpolation or extrapolation of the computational model to conditions corresponding to the intended use of the model, and [...] determination if the estimated accuracy of the computational model, for the conditions of the intended use, satisfies the accuracy requirements specified.”

3.1.6 Lidar Sensor Model Validation SOTP

The purpose of the sensor model, around which its validation shall be centred, is to provide data that helps (in fusion with other sensors, e.g. cameras) to construct an object list (and velocities of these objects). These objects in turn are the input for the driving function, which governs the trajectory of the ego vehicle. The trajectory is the base for the calculation of the criticality metrics, which are used (together with the novelty of the scenario) to calculate the reward R for the RLA. From this train of thought, we can derive the purpose of the simulation environment which is to provide stimulus for the sensor models.
As the technology itself, the validation of Lidar sensor models is in the development stage, where concepts proposed for a validation in real case scenarios still need to be evaluated for practice. Furthermore, the requirements concerning the validation highly depend on the requirements of the task at hand. However, there are certain basic principles that any validation must consider. And validation within a real case scenario will only be possible after several iterative validation steps in more controlled, simpler environments [40].
The initial step of any validation is to choose a simulation model that allows a proper representation of the physical system. Generation, propagation, and intrinsic properties of the initial laser beam need to be modelled, as well as the properties after reflection from a given surface and the detection. The relevant information needed, include position and type of the emitter, e.g. the number of scan lines and active phases, the wavelength and divergence of the beam, which are usually modelled via ray-casting or ray-tracing, the position and reflective properties of targets, noise depending on distance, and finally, the properties of the sensor, e.g. its field of view, threshold for signal-detection or intrinsic noise-level [41] [42]. Depending on the scenario, also relative velocities of the sensor and targets need to be considered, as well as weather conditions or higher order reflections of the laser beam [43].
First, simple scenarios focus on static objects in a controlled environment, correct relative positions, and the reflective properties of sample surfaces. Proper positioning and size representation can be validated by an overlay of real and synthetic results, reflective properties by comparing the number of points representing an object and their intensity. The second step in validation may include defined, slow-moving objects in a public environment, before scenarios occurring in a public test drive are considered [40], [42]. The latter steps will need automated validation. Figure 12 sketches one of the proposed procedures based on general indicators as described in [42]. Occupancy grids (OG) are derived from real and

synthetic datasets and are then compared. The deviation is quantified by overall error, Barons- and Pearson-correlation.

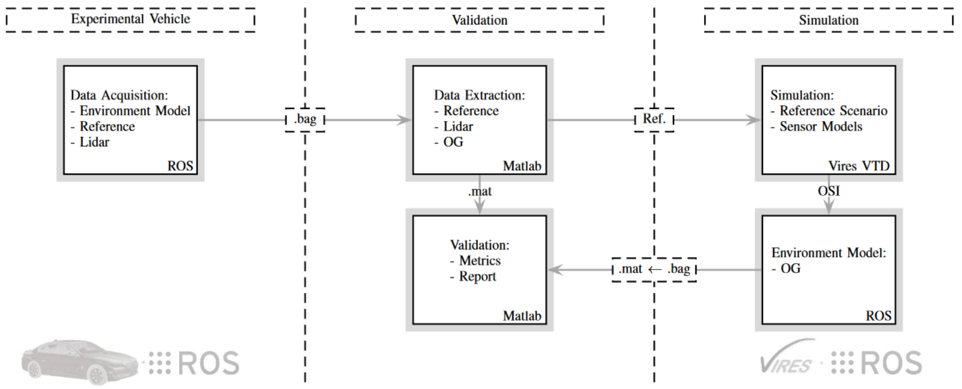


Figure 12 Schematics of proposed Lidar validation as described in [42]

Specific validation details, such as precision, depend highly on the purpose of validation. E.g., is an exact representation of the probed system down to single point level needed? Or does it suffice to capture the shape and size of certain objects? For an example see Sec. Operational Validation in UC UUV5.1.

3.1.7 Simulation Environment Validation SOTP

As for the sensor model validation we want to pinpoint the purpose of the simulation environment.
As detailed above, the task of the simulation environment can be stated as “provide stimulation for the sensor models”. So, validating the environment is to be reduced to validating the stimulating elements inside the environment and their relative and absolute positions. Because of the lack of numerical outcome, the standard validation procedure, which is followed by TrianGraphics internally, typically entails the application of informal techniques, such as visual inspection (face validation). Visual inspection on the generated scene can be conducted by human expert or screenshots comparison system which emulates the behavior of the human expert in identifying simulation results that don’t correspond to a reference screenshot, such as floating buildings
In the following, V&V methods that are typically followed internally are given and they are shortly explained.

- Tests like checking the existence of files are done automatically. This can be seen as a static validation method [8].
- Screenshot automated comparison with a reference image is done.
- Visual inspection by experts to identify illogical simulation results, like floating buildings.
- Validate at different levels: OpenDrive and virtual environment
    - (Static) Consistency
        - Environment (= visible 3D model) + scenario + OpenDrive (= virtual 3D road network) logically aligned.
            - 3D models in environment and OpenDrive (offsets should not exist)
        - Logical structure of the environment (no tree on road)
        - Street directions (One-way Street towards each other)
            - Logic of the road network
        - Check possible collisions of objects
    - Completeness
        - Road import
            - contain all elements to be tested (are all signs there, which were defined in OpenDrive?)
        - Are all visual elements of an object class included (e.g., road markings)?
        - Are all vector data, elevation data, satellite data, texture references, … included?

- Suitability (related to road network logic)
  - Drivable
    - Is it possible to get from A to B?
    - Are the curvatures of the roads realistic?
  - Road traffic regulations
    - Speeds (signal<-> lane)
    - Traffic sign logic (e.g., main and side roads at intersection).

The next section describes the validation mechanisms, which are applied in context / developed for ASIMOV.

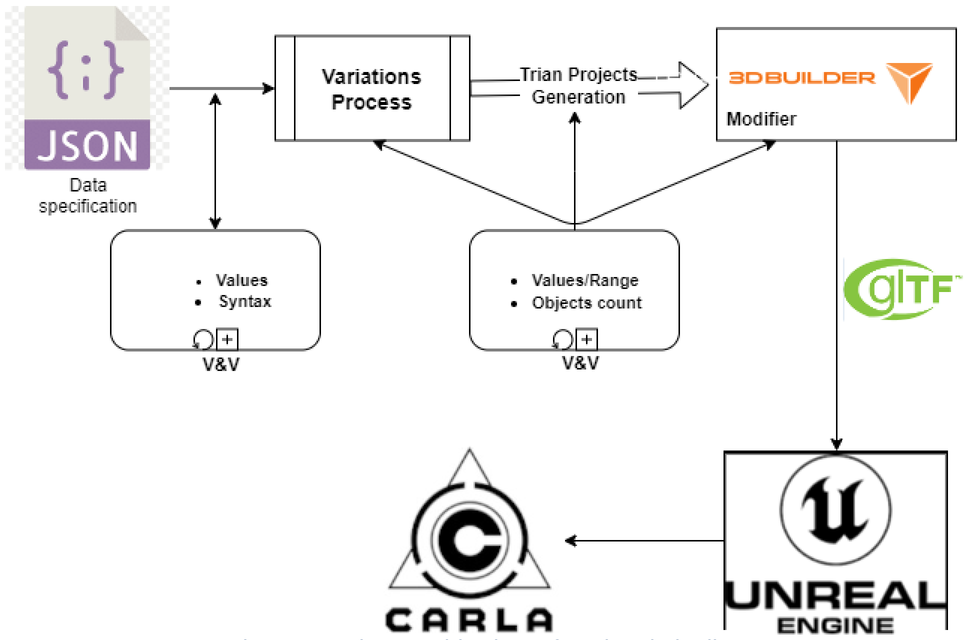


Figure 13 TrianGraphics base functional pipeline

The functional pipeline of the creation of the simulation environment is shown in Figure 13. It starts with a static scene with a parking car in the street, which was created. The other items are the following (Tasks that need to be done for verification and validation are highlighted with V&V):

- The Json file describes a set of control parameters values/ ranges.
  V&V: the Json data values and syntax is to be validated.
- Following the rules described by the input control parameters and depending on the values/ranges AI agent, the variation generator creates several scenarios of the environment and objects located in it. Trian projects are generated to describe the created environment scenarios and processed in Trian3DBuilder Modifier unit.
  V&V: are objects logically and probably located in the scene e.g., size, position, count:
  - A validation Text will be dumped by Trian after generation step. It consists of information about the placed objects and will be validated by comparing the intended objects count (from JSON Data) with the count of the resulted objects that are actually placed in the generated variation and the system will yield an information accordingly.
  - The dumped text contains information about intersected / collided elements in the scene with numerical output that describe to which extent are these elements intersected and will be passed to the AI agent as some sort of penalty to be considered in the next iteration.
  - Concrete implementation of this validation mechanism includes two important criteria:
    - The “intersection value” will be expressed as percentage value describing how much are bounding boxes of these specific objects are intersected so that the generated value can be independent from differentiation in sizes and/or shapes of the intersected scene element.

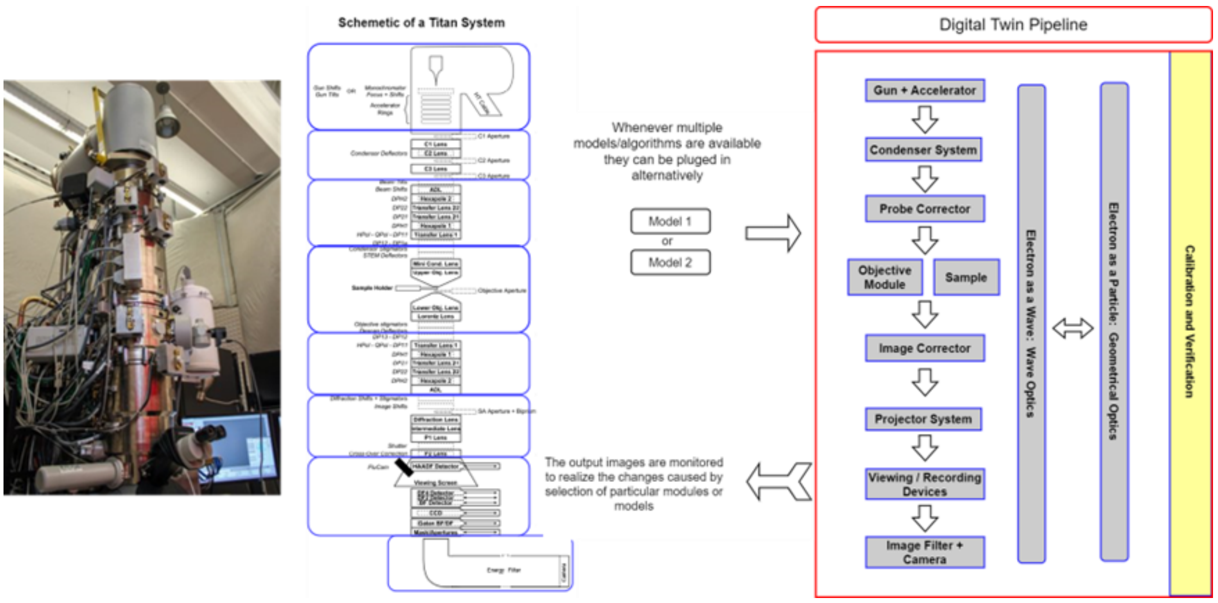
    - The severity of the intersection: to what degree should the agent be "penalized" as a result of this intersection.
- Regarding modelling sensors in 3d scene, in case the sensors are to be simulated, creating the sensors models will be dependent on research results of other parties in the project and/or possibly AI agent feed to specify the exact position of the sensor model e.g., location, rotation and so on.
- The created 3D scenes (variations of environment) are exported to Unreal Engine as Graphics Library Transmission Format (GLTF) data.
  V&V: Trian3DBuilder software internally validates the exported GLTF data.
- The Unreal environment scenarios are processed inside Unreal with CARLA interface/plugin. LUA script controls the variation process, accompanied with the data specification json instead of controlling them with TrianBuilder modifiers in the previous version of the pipeline.
- Ongoing work includes judging if it is realistic, what can be seen in the scene.
  - E.g., is the car too fast?
  - Glitches can potentially be validated.

3.2 UC (S)TEM

The use case is described in [44]. We will briefly reiterate and focus on V&V topics here.

3.2.1 Explanation / Framing

The (S)TEM use case aims at automatic aberration correction in a TEM system by an artificial intelligence (AI) model which is trained on the data generated by a TEM digital twin (DT). The DT consists of physics-based models describing different modules of a TEM system. Figure 14 shows the schematic of a Thermo Fisher Scientific TEM system (Titan) and the breakdown to its modules. Ideally one or more computational models represent each module. The outputs are collected at the bottom of the column on detectors and camera. The outputs of the DT are compared to the outputs of the real TEM. It is worth noting that an electron which is used for the imaging in a TEM can be described as a particle or as wave, giving rise to alternative modelling approaches in the digital twin.


Schemetic of a Titan System
Whenever multiple models/algorithms are available they can be pluged in alternatively
Model 1
or
Model 2
The output images are monitored to realize the changes caused by selection of particular modules or models
Digital Twin Pipeline
Gun + Accelerator
Condenser System
Probe Corrector
Objective Module
Sample
Image Corrector
Projector System
Viewing / Recording Devices
Image Filter + Camera
Electron as a Wave: Wave Optics
Electron as a Particle: Geometrical Optics
Calibration and Verification
Figure 14 The schematic of a Thermo Fisher Scientific TEM system (Titan) and the breakdown to its modules

Often when DT is mentioned it refers to a certain collection of models that produce the outputs required for the application of interest. For the aberration correction use case we chose to detect the convergent beam electron diffraction (CBED) patterns, also known as the Ronchigram. CBED pattern changes according to the microscope aberration status and is currently used by a human operator to correct aberrations. In ASIMOV we aim at automating this process. To generate realistic CBED patterns we use several models in the DT. Our main calculations are following the wave optics. We model the sample-electron interaction using the multi-slice method. Furthermore, we model the camera effects in the DT. Figure 15 shows a simplified view of the CBED pattern formation. After each step of the calculations in DT the wave is inspected to ensure that the outputs agree with the theoretical expectations. For example, we have verified that our simulation outputs agree with a textbook which describes the multi-slice and wave-optics approach [45]. This is the first essential verification of the DT.

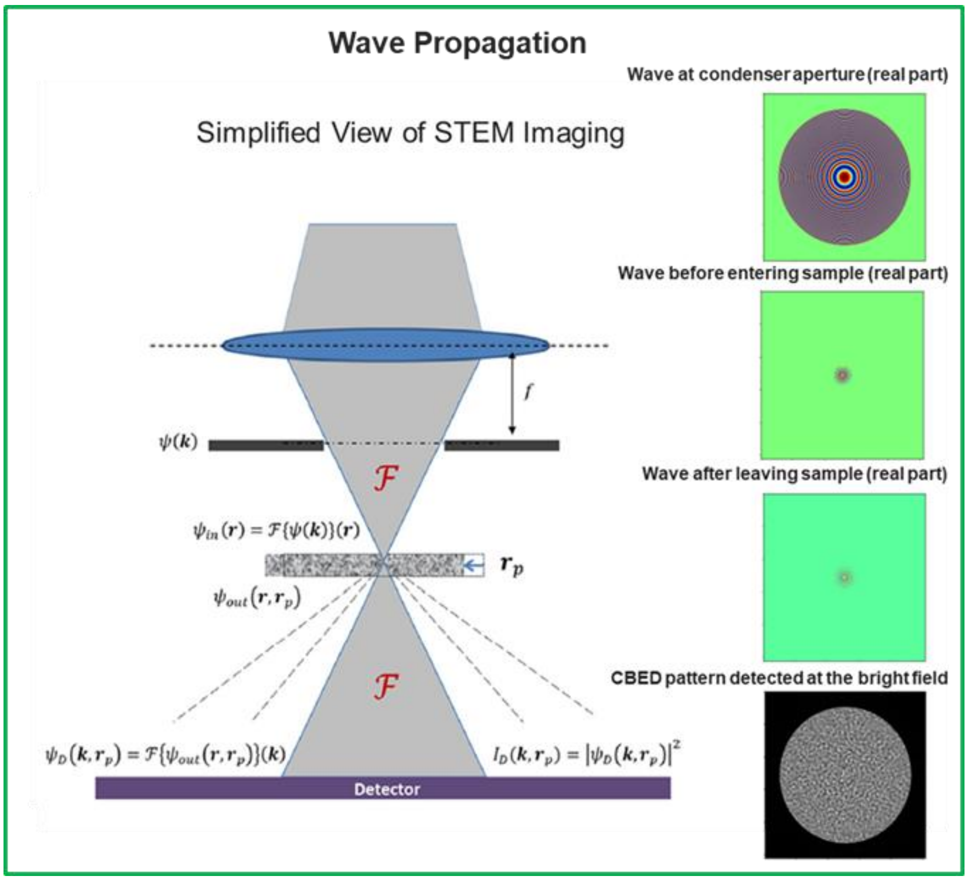

Wave Propagation
Simplified View of STEM Imaging
Wave at condenser aperture (real part)
Wave before entering sample (real part)
Wave after leaving sample (real part)
CBED pattern detected at the bright field
f
$\psi(k)$
$\mathcal{F}$
$\psi_{in}(r) = \mathcal{F}\{\psi(k)\}(r)$
r_p
$\psi_{out}(r, r_p)$
$\mathcal{F}$
$\psi_D(k, r_p) = \mathcal{F}\{\psi_{out}(r, r_p)\}(k)$
$I_D(k, r_p) = |\psi_D(k, r_p)|^2$
Detector

Figure 15 A simplified view of the CBED pattern formation Figure adapted from [46]

3.2.2 V&V Aspects of TEM Digital Twin

The validation of a DT has two major aspects:

- The physics-based models should produce the expected outcome, for example, according to the theory, and the outputs must be robust.
- The output of the models should agree with the outputs from the real instrument under same settings. To enable the AI model to perform on the real instrument, the data generated by the digital twin should give a reliable representation of the real instrument for the use case of interest. Therefore, this aspect is particularly important in the ASIMOV project.

In Sec 5.2 we will introduce the verification methods that so far have been used in TEM DT and will provide more details on a few of these methods as examples.

4 Verification and Validation (V&V) at ASIMOV

Approaches like the one described in [47] deconstruct the Digital Model into atomic parts and conduct V&V on them individually. A similar approach is done for the simulation algorithms. This helps to structure V&V efforts, but an argument why the total is valid when its parts are valid is missing. Similarly, methods for uncertainty propagation like [48] exist, but are not applicable to every use case for which the ASIMOV Solution can be applied. Therefore, we want to highlight some differences opposed to traditional model validation. Those differences shall be listed here. We will use the findings of the ASIMOV Working Group [5] as our base. There, the main difference was identified as a model being a representation of a system which is (more or less) static, and a DT evolving dynamically with its physical counterpart by updating within some interval. This leads to the concept of the adaptive and adaptation model which will be described in Section 4.4.1. We identify two levels in validation: the digital models' validity, which results in a higher confidence on the credible model that replicates the "event" or "behavior" that happens in reality under similar conditions, and the digital twin validity which requires a predictive validity. Assuming the digital models are validated, at this level the predictive properties need to be addressed, such as predictive bias and statistical significance, autocorrelation components, etc.

4.1 Specification of V&V Activities Within the ASIMOV Solution

We give an overview of the components of the ASIMOV Solution that extend the methods described in Sec. 2. This is shown in Figure 16, where the fleet aspect of DTs is highlighted through the introduction of prototypes and instances, the reason being that the DT paradigm should allow to easily instantiate DTs for other physical systems.
Different entities are shown, namely the instantiated DTs and the System Configuration Variations (yellow), the PTs (green) and the RLAs (blue). Dotted arrows depict instantiations, solid arrows data flows. Bidirectional arrows indicate twinning. The System Configuration Variation can be seen as a DT without a PT, where the system configuration was derived in an ‘artificial’ way.

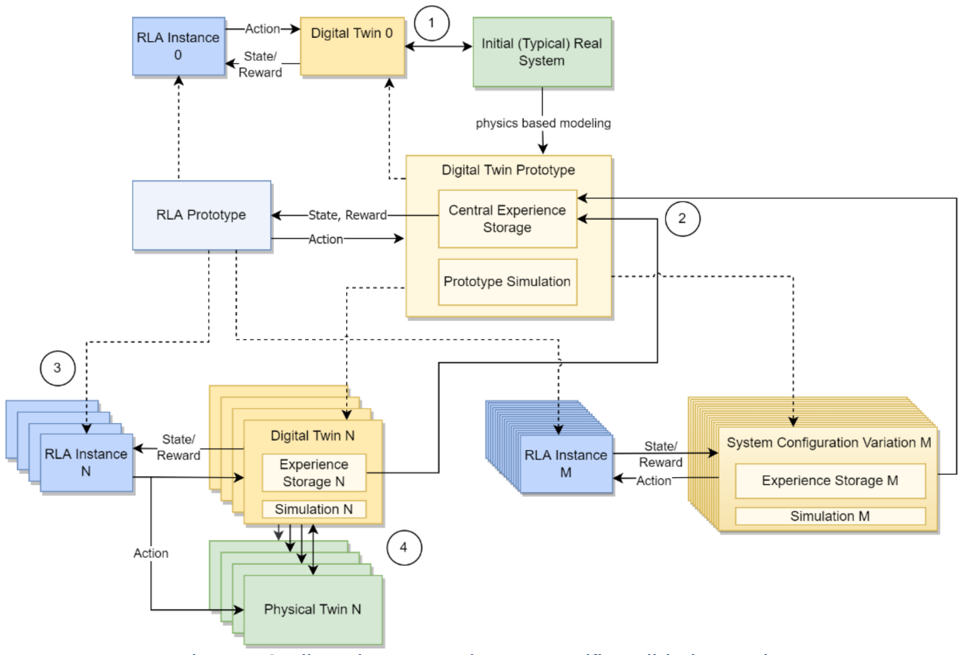


Figure 16 Allocating DT and RLA specific validation topics.

The development of the DT starts with an Initial (Typical) Real System. Through physics-based modeling, a DT Prototype can be created. From this, the DT instance 0 which is linked to the initial system is derived. This DT is linked to an instance of the RLA Prototype. The RLA Prototype provides the “zero-shot” policy for the optimization of novel CPSs. Instantiation from the DT Prototype can happen in two ways: Through “artificial” variations (System Configuration Variations) and based on actual differences in the PTs which are then twinned to the DT.

The identified validation tasks are marked with numbers from one to four. We map these topics to “Sargent’s triangle” which was introduced above (see Figure 2). Topics 2, 3 and 4 cannot be mapped to this framework directly.

1. Traditional model validation, where one executable model is compared to one physical asset. See “Step 1: Initial development” in the ASIMOV Solution below.
2. Data validity. The Experiences from the instances are used to alter the prototype. The quality of this data has to be validated.
3. Zero-shot approach. The policy from the RLA prototype is shipped for newly instantiated DTs. The validity of this zero-shot needs to be examined. If it’s not good enough, a finetuning phase may happen.
4. Twinning / real time validation, addressed in [6].

4.1.1 System Boundaries

To provide a clear scope for the validation task it is necessary to establish boundaries for the DT from its environment. We build on the framing developed in [30] which is also shown in Figure 17. We see that pre-, post-processing, and variations are technically out of scope. However, when we later derive the functional goals for the DT, some of them (like FGDT-07 (Robustness)) require us to consider topics from the preprocessing. This again highlights that validation cannot happen in isolation. The variations block is related to the fleet aspect of the DT paradigm.
In [30], Sec. 3.1.2, constraint handling is discussed and linked to the field of safe RL. One example is a predictive safety filter (PSF) as part of the pre-processing. This only makes sense in the operating phase. In the training phase the agent needs to get negative reward for proposing invalid actions. Other techniques are available as the authors in the survey on safe RL show [49].

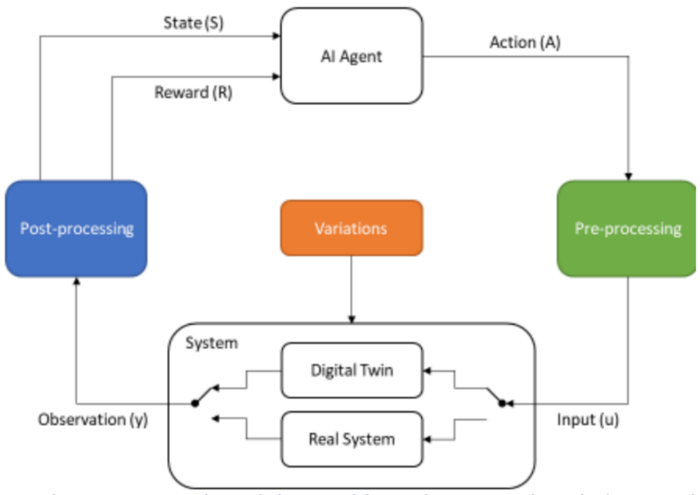


Figure 17 Pre- and Post-processing of data to / from the DT and Variations. Taken from [30].

4.1.2 V&V in the different Steps and Phases of the ASIMOV Solution

System level development of an ASIMOV Solution can be divided into four steps and three phases, as described in [5]. We will repeat the steps and phases here and highlight some validation-specific aspects.

Four Steps:

- **Step 1: Initial development.** This is where classical model validation techniques can be applied. Physics-based modeling is especially important in this step. Dynamic models are developed that are the base for the predictive capabilities of the DT. Reduce-order models are critical enablers for predictive digital twins [50]. Once the DT Instance 0 is derived from the DT Prototype, operation model validation can happen.

- **Step 2: Connecting the DT and AI.** The DT is connected with the RLA, exchanging observations, rewards, and actions. An infrastructure (see Figure 17) is created, that allows for training and fine tuning. Here, an important step is the preprocessing of actions coming from the RLA, and post-processing of data coming from the DT. Both are described in [30]. Note that no explicit DT validation happens in this step. details on validation of the AI are the topic of the ASIMOV deliverable D3.4.
- **Step 3: Connecting the PT.** In this step, the RLA controls the PT directly. Note, that this direct control could be circumvented via virtual to physical twinning, which means that the RLA can control the DT directly, without putting the PT at harm, if the RLA operates only on the DT. After an optimal setting is found the RLA is twinned to the PT. Details in [6].
- **Step 4: Connecting to other CPSs.** A policy learned for one CPS can be applied to other CPSs in this step. We see this as a zero-shot approach, understood as in [51]. Since the RLA is trained on a fleet of DTs, data validity is paramount in this step. One potential issue is the question if there are "ripple down effects" for previously working DTs. This is relevant only if previously rolled out DTs shall stay synched with the DT Prototype. In this step, the view of the DT as a single instance that needs to be validated is extended to the DT as a broader concept which includes information from different CPSs. A central topic is variations of the DT. This is linked to federated learning. Details in [30].

Three Phases:

- **Phase 1: Training phase.** The RLA (which is another subsystem of the ASIMOV Solution) is connected to a DT. The purpose of the DT is to provide training data for the RLA, so that the goals of the ASIMOV Solution can be fulfilled.
- **Phase 2: Operational phase.** The RLA is connected to a PT and optimizes it. Operational model validation can be performed by presenting the DT and the PT with the same actions of the RLA and comparing the output.
- **Phase 3: Fine-Tuning phase.**
  - **Phase 3.1: Local Fine-Tuning phase.** The RLA has information from the respective PT. This phase is only accessible after Step 3 has been reached.
  - **Phase 3.2: Global Fine-Tuning phase.** The RLA has information from multiple CPSs. This phase is only accessible after Step 4 was reached.

4.2 Derive Functional and Non-Functional Goals for the Digital Twin

As mentioned above, the DT is a subsystem of the ASIMOV Solution. Different goals of the DT are linked to different steps and phases. The achievement of certain steps are prerequisites for certain goals of the DT. This will become clearer once we list concrete metrics in Sec. **Error! Reference source not found.**.

4.2.1 Functional and Non-Functional Goals of the ASIMOV Solution

These goals shall be repeated here. They are labeled as **F**unctional **G**oals of the **A**simov **S**olution (FGAS) and **N**on-functional **G**oals of the **A**SIMOV **S**olution (NGAS). Note that FGAS-08 Safety was added here.

FGAS-01: Accuracy of Result. Does the result achieve the required accuracy? Here, we understand accuracy as how close a given set of measurements (observations or readings) are to their true value. Since the result of the ASIMOV Solution is the optimal setting of a CPS, we see the calibration as the result and the concrete calibration an expert did as the true value.
FGAS-02: Robustness. Ability to handle disturbances, extreme inputs, etc.
FGAS-03: Reliability. It shall be the norm (e.g., 90/95% of the time to obtain a good result. Otherwise, feedback shall be given on what went wrong.
FGAS-04: Reproducibility. Ability to always obtain the same result. This is closely linked to *precision,* understood as providing information on how close the measurements are to each other.
FGAS-05: Time to result. Execution time in operational phase to reach the result.
FGAS-06: Scalability. Ability to be usable in a product family, and for a variety of usage scenarios.

FGAS-07: Explainability. Ability to give insight into how the solution works, and plausibility of the result.

> **FGAS-07a: Explainability of the behavior of the RLA itself.** This is specific to the RLA, and not a focus for DT validation in this document.
> **FGAS-07b: Explainability of the process to get the final AI behavior.** In other words, the RLA is assumed to work as intended, and the steps taken to allow it to come up with its behavior need to be explainable.

FGAS-08: Safety. The physical twin shall not be put into harm.

NGAS-01: Footprint. How much space/energy does the solution need?
NGAS-02: Integrability. How well can the solution be embedded into the existing product?
NGAS-03: Maintainability. How well can the solution be upgraded or evolved in the field?
NGAS-04: Development cost. How much does it cost to develop the solution?

4.2.2 Functional Goals of the DT

The goals of the ASIMOV Solution cannot be directly mapped to the goals of the DT. As an example, FGAS-01 (Accuracy of result) does not solely depend on the accuracy of the DT, but also on the algorithms that are deployed in the RL system and the interplay between the two systems. We will formulate functional and nonfunctional goals for the DT and map them to the goals of the ASIMOV Solution, wherever possible.

It is differentiated between functions of the system (ASIMOV Solution) in the training phase and functions in the operational phase. This split should also be considered for the DT. Starting with the training phase, the major function of the DT was identified as F1: Execute [DT] behavior. Here the DT creates the system ([DT]) behavior. In the operational phase, the only difference is that the PT creates the system ([PT]) behavior. This is the phase when operational validation techniques can be applied. We compare how the DT behaves in contrast to the PT, when both are confronted with the same inputs, but have to keep the input data model validation in mind.
Next, we'll list purposes of the DT in different phases and come up with derived functional goals, labeled as FGDT, and non-functional goals, labeled as NGDT. This will be the base for DT validation metrics.

4.2.2.1 Deriving Functional and non-functional goals from the purpose(s) of the DT

Here, we iterate through the different purposes the DT has within the ASIMOV Solution. We collect the FGDT and NGDT. This hierarchical presentation (goals inside purposes) is useful for exploration. Later the goals will be listed and linked to the higher-level goals of the ASIMOV Solution.

- **Purpose 1 (Phase 1: training phase):** One DT instance is linked to one RLA instance. The purpose of the DT is to provide useful data to the RLA instance in an efficient way. Useful data is to be understood as data that allows the RLA to learn a valid policy.
    - **FGDT-01: Usefulness**. FGAS-01 (Accuracy of the result) cannot be directly translated into the goal of a high accuracy of the DT, since the interplay with the RLA is also relevant. Further, e.g. Kober et al. emphasize that it's necessary to elaborate suitable fidelity levels for DT based on its purpose and not aim for the highest fidelity (accuracy) possible [52]. Therefore, we soften this goal to usefulness of the produced data for the RLA. It should be made explicit what quality (e.g., trend (i.e., sign of first derivative of input (e.g., knob setting) and output (i.e., quality of output))) from the data is enough to train the RLA. An important aspect to consider is how to differentiate between zero shot quality, the fine-tuning policy, and application of policy on PT directly.
    - **FGDT-02: Interpretability**. Considering FGAS-07 (Explainability), a solution derived with DTs that behave significantly differently to the PTs, cannot be explained very well and does not produce trust in the solution. Considering FGAS-03 (Reliability) feedback from the DT is needed if no good result could be obtained. To be able to provide this feedback, interpretability is needed. This also links to FGDT-06 (Architecture) below. For interpretability, it is very important that the underlying dynamics of the model can

be understood. This can be achieved by basing the DT on a first-principle model that reflects physical laws and validating whether the DT is in accordance with these laws using test cases. Alternatively, for data-driven DTs, surrogate models can be used to explain the behavior of the DT at a more abstract level. Note that high accuracy also typically means a higher model/DT development cost. This FG is therefore in competition with NGAS-04 (Development cost) and NGAS-01 (Footprint).

  - **FGDT-03: Low execution time.** The DT should accurately mimic the behavior of the PT. A high DT accuracy means that the DT needs to be more detailed. This leads to additional input data that is needed, and more analysis to process this data. To generate sufficient training data in the available time, the DT should have a low execution time. Related to FGAS-05 (Time to result), since DT calculation time is part of the total training time.

- **Purpose 2 (Phase 3.2: RLA has information from multiple CPS (fleet)):** A second purpose of the DT is to provide data that enables the RLA Prototype to be able to learn a good zero shot policy for new CPSs, or a zero-shot policy that facilitates fast fine-tuning for new CPSs. This second notion extends the purposes of traditional simulation models (and of single instances of DTs, like in Phase 1 of the ASIMOV Solution), since it involves information from a whole fleet of DTs. It can be linked to FGAS-02 (robustness), FGAS-05 (time to result), and FGAS-06 (Scalability).
  - **FGDT-04: Variations (to provide good zero shot)** The DT shall provide data that is varied enough to enable the RLA Prototype to learn a policy that is applicable to novel CPSs. A policy that was derived from a DT that was mimicking a specific PT with all its unique distinguishing marks, cannot be expected to work as well on another PT with other unique distinguishing marks. Note that this FG extends a single instance of a DT and is concerned with qualities that emerge from the fleet aspect in the DT paradigm.

- **Purpose 3 (allow quick fine-tuning) [extension of purpose 1 and purpose 2]:** If the zero-shot approach is not good enough, a fine-tuning phase (Phase 3.1) needs to happen. We want to shorten the time this fine-tuning takes. To make this time more concrete, we deconstruct the process at hand into its phases as shown in Figure 18. There, three times are introduced: **t_twin**, which is the time the physical to virtual twinning takes, **t_sim**, which is the execution time of a simulation (this is linked to FGDT-03), and **t_apply**, which is the time it takes to apply the policy to the DT. The resulting total time for the fine-tuning is t_twin + N*t_sim + t_apply. Note, that the number N of samples needed to adjust the policy of the RLA is a goal for the RLA and shall not be the focus here. It is a known problem that current RL techniques have a low sample efficiency and require a huge number of interactions with the environment [53]. Also note, that fast twinning is a goal for the CPS (DT+PT) as a whole, and therefore won't be formulated here. A need for low execution time (t_sim) was already introduced above (FGDT-03)

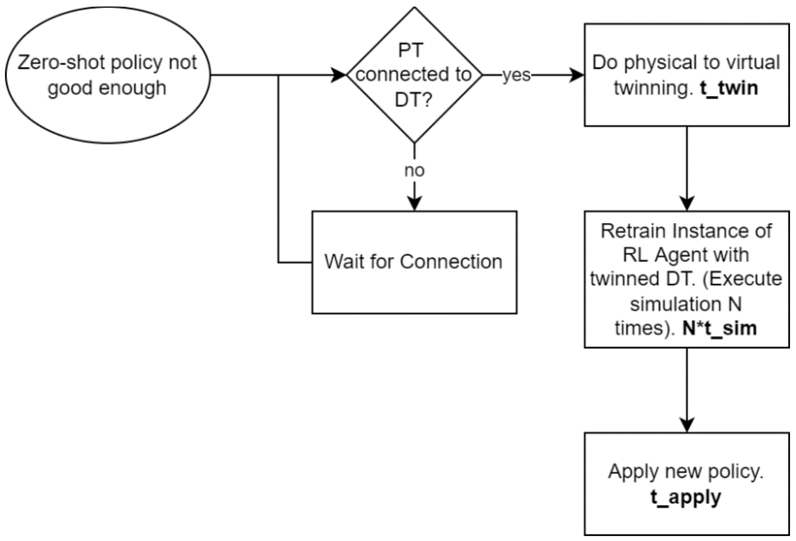

Zero-shot policy not good enough
PT connected to DT?
yes
no
Do physical to virtual twinning. t_twin
Wait for Connection
Retrain Instance of RL Agent with twinned DT. (Execute simulation N times). N*t_sim
Apply new policy. t_apply

Figure 18 Deconstruction of the fine-tuning process

- **Purpose 4 (Phase 2: RLA connected to PT. DT as safety mechanism):** The DT can be used as a "safety mechanism" to test-out actions proposed by the RLA prior to applying them on the PT directly. This is one solution to the problem that most exploration methods in RL are blind to the risk of action. This also links to topics in safe RL, e.g. modifying the exploration process [49].
  - **FGDT-05: Safe Approximation.** For this purpose, it is not so important that the DT is as accurate as possible, but rather that its predictions for safety-related aspects are conservative (e.g., we could imagine that output images do not need to be accurate for predicting safety, but currents that control magnetic lenses cannot be too high).
- **Purpose 5 (Phase 2: RLA connected to PT. Twinning.):** The optimization is done using the DT, then the result is mirrored / twinned / copied to the PT. This virtual to physical twinning is in contrast to only physical to virtual twinning, which is a characteristic of the Digital Shadow (DS) [54].
  - **FGDT-06: Architecture:** the DT needs to consist of the same submodules as the PT. This also is essential for the method of uncertainty propagation through a system explained in [48]. This also helps user understanding of the model (FGAS-07 (explainability)) and providing feedback, when the ASIMOV Solution did not provide a good result (FGAS-03). This FGDT might be in conflicting with NGAS-04(development costs), since it disallows a data-driven black box model. The guiding question here can be "does the DT fulfill the reference architecture?"
- **Purpose 6 (Status monitoring):** Get insight into PT (e.g., for predictive maintenance when sensors are missing in the real-world). Observation on some sub-models may not be available at runtime in the PT, but in the DT. This links to FGDT-02 and FGDT-06. Note that this purpose is outside of the scope of ASIMOV.

To assure the derived FGDTs are complete, we consider the FGAS-02 robustness, FGAS-03 reliability, FGAS-04 reproducibility in more detail and derive additional FGDTs.

FGAS-02 (robustness). have clear ranges on input parameters of DT, and clear output ranges. Ensure graceful degradation of the DT if the input range is violated. Note that these topics are located outside of the DT (see Figure 17). The existence of certain filters that guarantee that the DT is only presented with valid inputs is part of the context in which the DT can be used.
FGAS-03 (reliability). We frame this FGAS in the following way. The DT should be able to deal with uncertainty in the inputs. → Add FGDT-07 Robustness.
Further details: There are two types of uncertainty:

- **Aleatoric/statistical uncertainty:** notion of randomness. Variability is caused by inherently random effects, e.g., unknowns that differ each time we collect inputs.

- **Epistemic/systematic uncertainty:** caused by lack of knowledge, things we could in principle know, but do not in practice (inaccurate data, DT neglecting certain effects, unobservable parameters).

There are various techniques to deal with uncertainty, see e.g. [55]. For example, forward propagation to predict the overall uncertainty in the DT outcome, using techniques like Monte Carlo simulations, surrogate-based methods, and uncertainty propagation.
FGAS-04 (reproducibility): reproducibility requires a deterministic DT, as well as reproducible training in the context of collecting input data, running the DT, and getting the required training data. → Add FGDT-08 Reproducibility.

4.2.2.2 Resulting functional goals of the DT

Table 1 shows from which high-level goals the goals for the DT were derived. The asterisk (*) indicates that the respective goal is technically outside of the system boundary of the DT.

Table 1 Functional Goals of DT

Functional Goals DT (FGDT)	Short description	Aligned with	Conflicting with
01 Usefulness	Data produced by DT shall be useful for RLA	FGAS-01	
02 Interpretability	First principle modeling. High accuracy	FGAS-01, FGAS-03, NGAS-03, FGAS-07	NGAS-01, NGAS-04
03 Low execution time	Sufficient training data in available time	FGAS-05	FGAS-01
04 Variations (zero shot)	Fleet of DTs shall be diverse enough to enable the learning of a valid zero-shot policy	FGAS-02, FGAS-05, FGAS-06	
05 Safe Approximation	Safety mechanism to not put the PT in harm	FGAS-08	
06 Architecture	Same sub-modules as the PT	FGAS-03, FGAS-07, FGAS-08	NGAS-04
07 Robustness	Deal with uncertainties in the inputs	FGAS-02	
08 Reproducibility	Deterministic models	FGAS-04	

4.2.2.3 Resulting non-functional goals of the DT

We derive NGDTs in a similar way to the FGDTs above.
The NGAS-04 (Development cost) leads to:

- **NGDT-01: Efficiency.** The development of the DT shall be efficient in the time it takes and the money it costs to develop it. This conflicts with FGAS-01 (accuracy), which also typically means a more detailed model with more input data crunching, affecting compute load, and therefore also computing resources needed. This links to NGAS-01 (footprint). Extensive DT models and computations require significant resources.

NGAS-02 (integrability) can be mapped to:

- **NGDT-02: Clear description.** For the DT to be integrable, there must be a clear description of the needed resources, interface definitions, deployment approach, and IP protection. This might be in conflict with NGAS-04, since documentation costs money, but is well-aligned with NGAS-03 (Maintainability).

NGAS-03 (maintainability): Understandability of DT models is crucial for maintenance. Ideally the DT allows for a plug-and-play strategy to integrate new versions of DT sub-components. (this also links to FDGT-07 (architecture)).

- **NGDT-03: Standards.** The DT should adhere to standard interfaces and connectors. Note that this non-functional goal introduces a trade-off with some important functional goals. This is a key challenge to the ASIMOV Solution

NGAS-01 (Footprint) leads to:

- NGDT-04: Footprint. The models and simulations shall only use as much space and computations as needed. Also, the explored variations shall be limited to expected parameter space.

Table 2 Non-functional goals DT

Non-Functional Goals DT (NGDT)	Aligned with	Conflicting with
01 Efficiency	NGAS-01	FGAS-01
02 Clear Description	NGAS-02, NGAS-03	NGAS-04 (short term)
03 Standards	NGAS-03	
04 Footprint	NGAS-04	

4.3 Validation Metrics

In Step 1 (Initial Development), the initial versions of the DT prototype and the RLA prototype are created. It is a process where domain knowledge, physics (first principles) and measurement data are used to create a digital representation of a typical instance of the system.

Conceptual validation. The initial development of the DT prototype can be seen as the conceptual model creation. Since no one-to-one mapping from the prototype to an executable instance exists, operational model validation cannot be done the same way as for the validation of the first instance of the DT. Therefore, verification is paramount at this stage. Also, when the sub-models in the prototype are created in a data-driven way, data validity is important. In step 4 of the ASIMOV Solution, it is applied to a fleet of real systems. Then each real system gets a DT derived from the prototype. This enables continuous improvement of the prototype, e.g., when the solution does not work for one specific asset and the problem is fixed locally. This information can be used to improve the DT prototype. This makes conceptual validation of the DT within the ASIMOV Solution different from traditional conceptual validation, because the view of the DT as a single instance that needs to be validated is extended to the DT as a broader concept which includes information from different CPSs from step 4 onwards. One potential issue is the question if there are “ripple down effects” for previously working DTs when adjusting the prototype. This is relevant only if previously rolled out DTs shall stay synched with the DT Prototype.

Operational validation. Once the initial DT instance is derived from the prototype, the first operational validation can happen. Boschert et al. list requirements that enable the operational model validation to happen. They include that data from the real asset and executable models exist and high-level requirements are given [56]. Those are our FGDT. Different metrics can be applied once certain steps in the ASIMOV Solution are achieved. If for example the DT is not connected with the PT yet, we depend on previously recorded empirical data and need to think about extrapolating the validation assessment into regions where no empirical data is available. In the following we list validation metrics for operational validation. These metrics can be applied to check whether the FGDTs are fulfilled. Since fleet aspects are not relevant at this point, traditional model validation methods can be applied.

In Table 3 the FGDTs are listed and linked to a fitting metric. In the following those metrics will be described in detail, including preconditions, so they can be applied.

Table 3 Overview validation metrics

FGDT	Metric
01 Usefulness	1: Functional indirect 1.1: Compare final solution 1.2: Compare steps
02 Interpretability	2: Similarity Metrics
03 Low execution time	3: Execution time.
04 Variations (zero shot)	4.1: Evaluate robustness of fine-tuned policy 4.2: Evaluate zero-shot
05 Safe Approximation	
06 Architecture	6: Architecture
07 Robustness	7: Robustness
08 Reproducibility	8: Reproducibility

Preconditions
These are requirements on a DT that need to be fulfilled, so that a DT can be used within the ASIMOV Solution.

Static:

- Interfaces
    - Must: Interfaces for the RLA, or rather the pre- and post-processing blocks need to exist.
    - Should: The DT should fulfill the reference architecture.

Dynamic:

- For some metrics: The DT needs to be connected to the PT.
- For some metrics: The DT needs to be connected to the RLA.

Thresholds for metrics (pass/fail criteria):

- We can derive the FGDTs, and link them to quantitative metrics, but the threshold for acceptance needs to be given externally.

Baselines:

- When measuring the execution time, a reference scenario needs to be given.

Metric 1: Functional indirect. To assess FGDT-01 (usefulness), we introduce Metric 1: Functional indirect. The validation strategy here is *"The DT is valid, if it produces data which is suitable to train an RLA".* This means we take the role of the RLA into account. We apply the RLA that was trained using only the DT on the PT. We call the metric "indirect" since the DT is validated based on the performance of the RLA, not via later described similarity of its output to the output of the PT. We introduce Metric 1.1 and Metric 1.2 as two variants of a functional indirect metric.

Metric 1.1: Compare final solution. This means that the outcome of the ASIMOV solution is compared with some oracle. The oracle can either be a human expert, or another automated system. This assessment of the quality of the solution is done by a domain expert and won't be quantified or detailed further here, since it's highly use-case specific. Another feasible way is to collect feedback from the field, once a minimal viable product (MVP) was rolled out. The benefit of the latter strategy is to get more diverse feedback. However, it requires a certain openness of the customers.

Metric 1.2: Compare steps to arrive at final solution. Another (less abstract) notion for a functional indirect metric is to compare the actions from the RLA (trained on DT only) when presented with a PT in a certain state (note, that we do not necessarily expect this exact state to be available for the DT, since we don't expect to model every aspect of the PT). This action will be compared with the action a human

expert (or a reference automated system) would take. The downside here is, that the RLA might be able to find optimization strategies that, interpreted step by step by a human expert, are against the expert's intuition and will be discarded as invalid. Something similar was famously observed in the game of go between AlphaGo [57] and world champion Lee Sedol, where AlphaGo's 37th move in the match's second game surprised the commentators, experts themselves, thinking "it was a mistake" [58].
Other approaches which involve training of an RLA (RLA_PT) only on the PT and another RLA (RLA_DT) purely on the DT and then somehow comparing their policies can be considered but are not feasible since a lot of samples are needed for training and the data collection using the PT can only be done in real-time.
Problems with those indirect validation metrics are that they are sufficient, but not necessary for the validity of the DT. When applying this metric (and it fails) it is not clear where the error happened; in the output of the DT or in the RLA itself. In other words, we could prove the validity of the DT with this metric, but not that it is invalid.

Metric 2: Similarity Metrics. Applying traditional operational model validation metrics the validation strategy is *"The DT is valid when it produces similar output as the PT"*. We measure the closeness of data produced with the DT and data from the PT using similarity metrics. We'll give a short overview here. Since the topic of comparing and quantifying the differences between the data generated by the digital twin and the physical system was a topic of [6] (Sec 2.3), we won't repeat it here but briefly summarize it. The output of a system (y) can contain multiple different data types; therefore, several different comparison techniques are discussed. They include techniques for images, scalar values, and time-series data. The acceptable threshold for acceptance is defined either statistically or through expert knowledge. As an example, the difference between two timeseries, one produced by the DT (y_sim(t)) and one by the real system (y_real(t)) can be quantified using the mean absolute error (MAE).

$$MAE = \frac{1}{n}\sum_{T}|y_{sim}(t) - y_{real}(t)|$$

Further metrics that are often used to compare outputs include Pearson's correlation, Spearman's rank correlation coefficient, Kendall's Tau, Cosine similarity, and Jaccard similarity. If the physical system is not deterministic stochastic properties like means, variances, confidence intervals need to be considered. If Step 2 of the ASIMOV Solution has not been reached (that means the DT is not connected to a PT), we rely on empirical data. The range of inputs that need to be checked needs to be addressed. In case of empirical data input validation and extrapolation is part of the validity assessment. Validation shall happen in both normal and extreme scenarios.
It might not be necessary to judge the similarity of the raw (potentially high-dimensional) raw output of the DT but be sufficient to compare lower-dimensional derived features. This is the topic of Sec. 4.4.2.
Validation using similarity metrics is not limited to the output y but can be done for internal parameters that are observable in both the DT and the PT in a similar way as for the output y [6] (Sec 2.3). The great benefit there is that misaligned sub-modules of the DT can be identified this way and aligned specifically. This is of special importance, when the ASIMOV Solution did not provide a good enough solution and the submodule that caused the problem shall be identified.

Metric 3: Execution time. To assess FGDT-03 (low execution time), we introduce Metric 3: Execution time, that measures the execution time *t_exec_dt* of the DT. We assume that a requirement *t_exec_req* is given. An upper bound for this could be the execution time of the actual system to perform the same operation, as we would like to reduce the total training time by using the DT. Note, that time can vary a lot based on inputs, so we need to have some kind of reference scenario.

How this metric is used:

- Derive a requirement t_exec_req on the maximum execution time for the DT for the given scenario.
- Run DT to yield its execution time t_exec_dt
- Compare t_exec_dt with t_exec_req

Metric 4: Variations-related. In order to measure how well the DT fulfills FGDT-04, which is to enable the RLA to learn a good zero-shot policy, we introduce Metric 4. We assume a fleet of N DTs and N RLAs that were trained 1-on-1 on these DTs. This requires that Step 4 of the ASIMOV Solution was reached.

This situation is closely related to the one in federated learning (FL). In FL the parties performing the local training are called *clients*, and the instance that orchestrates the training is called *aggregator*. This is described in more detail in [30] (Sec 3.3). Farebrother et al. propose a protocol to evaluate generalization in reinforcement learning [59]. If the policy that an RLA (RLA1) has learned on a specific DT (DT1) is *robust*, we expect the policy to also work on a different (slightly varied) DT (DT2). There are three levels of variations within the ASIMOV Solution; disturbances, system configuration variations, and domain variations [30].

Metric 4.1 Evaluate robustness of fine-tuned policy. To quantify the generalization of the policy from RLA1 which was trained using a_1 samples from DT1, we measure the cumulative reward averaged over some (ne_1_2) episodes when applied on DT2. This can be done on the remaining N-1 DTs as well followed by calculating the average again to get one number. We expect this score to be higher, when the variations of the DTs come from disturbances (minor variations), and lower when the DTs come from a different domain (higher variations).

Metric 4.2 Evaluate zero-shot. Another related metric is to test the quality of the zero-shot approach. Here, a policy for the RLA prototype is calculated from experiences from the instances. This policy is then used to again calculate the cumulative reward averaged over some (ne_p_i, the p stands for prototype) episodes on DT_i. Averaging those cumulative rewards for all N DTs we get a measure for the quality of the zero-shot approach. The quality of the zero-shot can also be evaluated by applying the RLA prototype on a DT that was not part of the N DTs.

Metric 5: Safe Approximation. Here it's not so easy to find a metric, because it might not be possible (or financially viable) to produce edge cases (unsafe parameter ranges) in reality (e.g. (near) crash scenarios in the UUV use case, or actions that potentially destroy millions worth of equipment in the EM use-case.)

Metric 6: Architecture. This can be thought of as a binary outcome. Does the implementation follow the reference architecture, yes or no. Assessment requires access to the actual implementation.

Metric 7: Robustness. To assess FGDT-07 (robustness), we introduce Metric 7: Robustness. Note that here we soften the reliance on the preprocessing module. This metric considers how the DT behaves and deals with error handling in cases when extreme/unexpected inputs are provided:

- How does the DT respond if one or multiple internal DT sub-systems fail for specific values in the input parameters' range?
    - Does the user get informed that an error happened within the DT?
    - Does the DT still generate a valid output in case of unexpected inputs?

How this metric is used:

- Identify extreme inputs outside of the acceptable range of accuracy for the inputs, as well as corner cases that are on the boundary of the acceptable range.
- Present the DT with the corner cases and measure that the DT goes to a safe state.
- Present the DT with the extreme inputs, measure that the DT goes to a safe state, and check that the user is informed when errors occurred within the DT.

Metric 8: Reproducibility. To assess FGDT-08 (reproducibility), we introduce Metric 8: Reproducibility**.** The goal of this metric is to check whether the DT produces the same result when used repeatedly in the same circumstances. Reproducibility requires a deterministic DT, as well as reproducible training in the context of collecting input data, running the DT, and getting the required training data.
How this metric is used:

- Run the DT multiple times with the same input and control parameters
- Assess whether the DT generates the same output in all iterations.
    - If not, then the internal state of the DT is likely different, or the DT is nondeterministic.

Since the DT paradigm also accounts for models changing over time (without changed inputs), we need to consider this for the validation. This will be the topic of Sec. 4.4.1.

4.4 ASIMOV Specific Validation challenges

4.4.1 Validating an adaptive model

We adopt the architectural view from [36] and [6] (see Figure 19) which makes the adaptive nature of the DT explicit. It illustrates the different aspects of a DT that have to be validated for use of the DT in the operational phase. Wright et al. highlight the need to have a data-driven component in the DT, so it can adapt to changes in the PT [60]. This is in line with Bochert et al., where it is stated that “The Digital Twin evolves along with the real system along the whole life cycle” [56].

The DT can be seen as composed of two parts, the Adaptive Model and the Adaptation Model. This can be seen as a part of the model management system in [56], which shall keep the single models up to date as changes occur.

- The **Adaptive Model** has some static configuration (c_fixed) and a dynamic part, that can be adapted by the Adaptation Model. Its purpose is to mirror the behavior of the PT as closely as possible.
- The **Adaptation Model** compares the DT and PT behavior and suggests new settings c_tune(t). Its purpose is to govern the behavior of the adaptive model, so that the DT behaves like the PT.

The controllable system input u(t) as well as the uncontrollable disturbances d(t) are provided to both the DT and the PT. Parameter c_PT(t) is the configuration of the PT, which is in general only partly observable and is not necessarily static. y_PT(t) is the resulting output of the PT.
Ideally, providing the same u(t) and d(t) of the physical system as input to the Adaptive Model, it outputs a similar y(t). In practice, there will be some difference between the output of these systems and a static initial fixed system configuration c_fixed is unlikely to be found to work immediately in all operational stages of the system. Because of that, there is the possibility to tune the Adaptive Model via the Adaptation Model, which is parameterized using the parameter c_adaptation, similar to c_fixed for the Adaptive Model. It can tweak the parameters c_tune(t) and therefore adapt the DT to the PT by comparing their system behavior and changing the DTs behavior. Using the right methods to adapt the DT during runtime is the focus of [6].

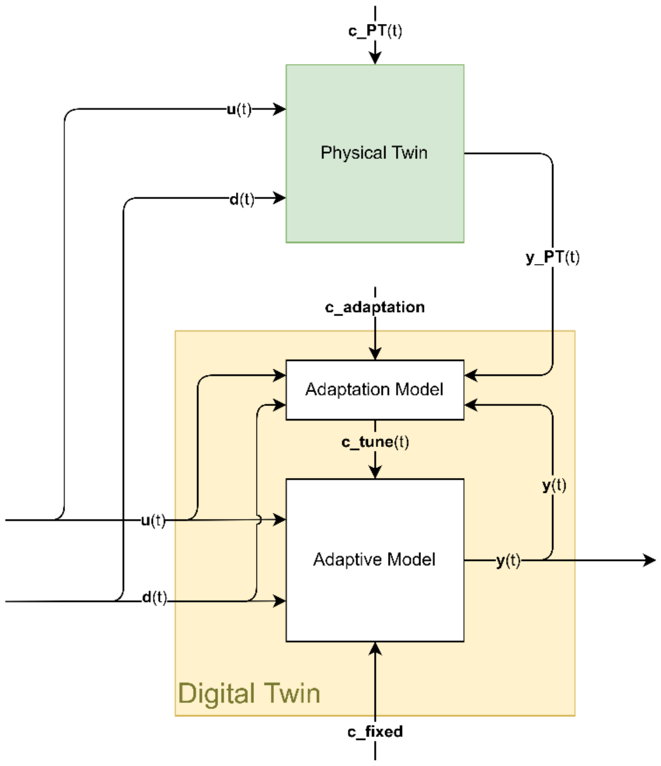


Figure 19 Adaption of the DT

4.4.2 Validation on Raw Output vs Validation on Postprocessed Data

We want to highlight a commonality between the use cases. Validation can happen on the data produced by the DT directly (Ronchigram for STEM, resp. raw image for camera model / LiDAR point cloud for LiDAR), or on the objects derived from this raw data (features in the Fourier transformed space, resp. objects found in the raw outputs of the sensors / sensor models.
Figure 20 shows these two levels for operational model validation as double arrows. If the post-processing is assumed to be the same in the physical and the digital realm, and the purpose of the model can be fulfilled with the features alone, then the validation on feature level is sufficient.
If the post-processing is robust enough against small perturbations in the raw data, noisy sensors do not need to be modeled in the DT.

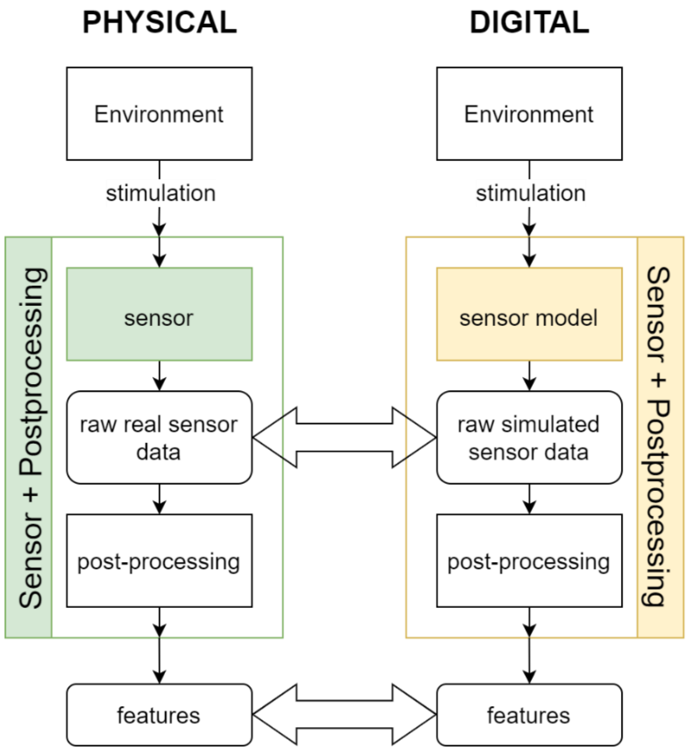


Figure 20 Comparing similarity on raw sensor output level vs feature level

If the AI relies only on the features extracted by post-processing, not on the raw sensor data, then the requirements on the DT can become weaker. No longer does the raw sensor data need to be identical or similar, but only the extracted/derived features. This has a direct impact on what needs to be validated for the AI training: the validation of the DT gets replaced by validation of the DT combined with post-processing. This links to the topic of validation of a DT with different fidelity levels in [17]. The notion here is that fidelity can be decreased until the derived features change. The granularity/fidelity of the chosen model shall be just fine enough to allow the fulfillment the purpose, but not finer [56].
If the physical and digital post-processing are identical functions, then equivalence of features is formally easier (at least not more difficult) to achieve than equivalence of the real sensor data. The notions of equivalence and "good enough" may however differ considerably between raw sensor data and features.

5 Additional Examples for V&V in the Use Cases

After having discussed the state of the practice for V&V for relevant topics in the use cases in Sec. 3 and having discussed some V&V approaches specific to the ASIMOV Solution in Sec. 4, we will highlight some additional V&V examples from the use cases here.

5.1 Operational Validation in UC UUV | Operational Sensor Model Validation on Object Level

We want to exemplify the process from above with the LiDAR sensor validation. There, the purpose of the sensor in combination of perception software is to provide an object list based on which the driving function acts. Experiments show that the points in the point cloud vary in their intensity, and their exact position. Especially noticeable is that the noise is not reproduced in the simulation with Carla. There the points are in a clean lattice. Examples in Figure 21, where a target plate is probed by a LiDAR.

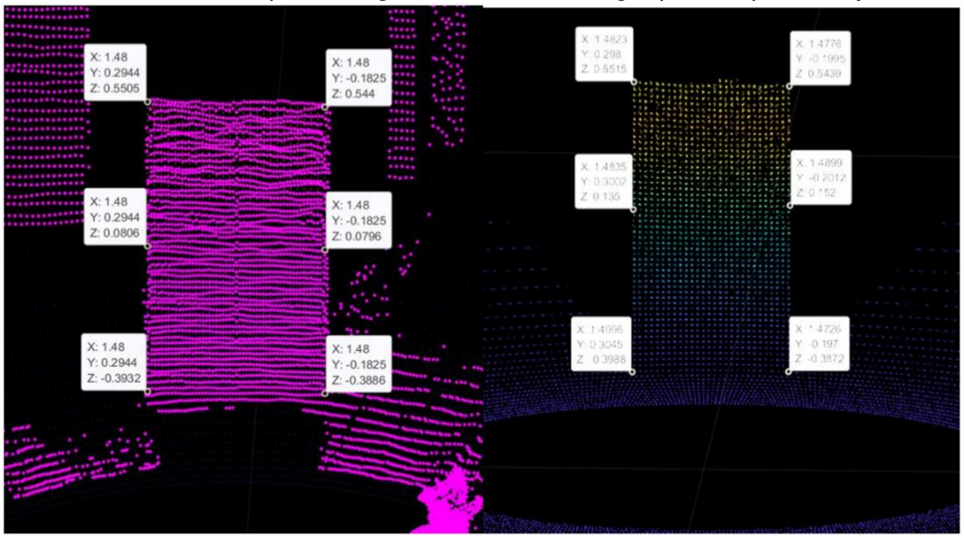


Figure 21 LiDAR Measurement of a target plate in a physical setup (left) and in Carla (right) © 2022 LiangDao GmbH

If we were to validate the raw point clouds, we would also need to include a detailed noise model in the simulation. The model would need to consider all potential sources of perturbations, including intrinsic mechanical perturbations of the sensor, which can be highly specific, road bumps that shake car and sensor, perturbations of the car itself, but also environmental perturbations due to strong winds. Such a detailed representation would consume an extensive amount of resource, but it can be avoided.
The perception software in between sensor and driving function is supposed to detect objects consisting of point groups and not single points. See again Figure 21. While intensity and position of single points vary between the physical and digital system, the shape of the probed plate is well reproduced. Compared to the size of the point group representing the target plate, the errors on point level are small.
Additionally, already in the physical system the perception software has to cope with changing appearance due to noise, see Figure 22Figure 22. The scene shown in this figure was analyzed by perception software. Identified objects have been marked with bounding boxes, which have been annotated with additional information (class, e.g. car, and speed). The cut-out (top-right) zooms in on the car bottom left. Note the varying intensity of the points representing the car in the cut-out, even though the car itself has a smooth surface (not shown). The point intensity will also change for each new frame and the change appears chaotic.

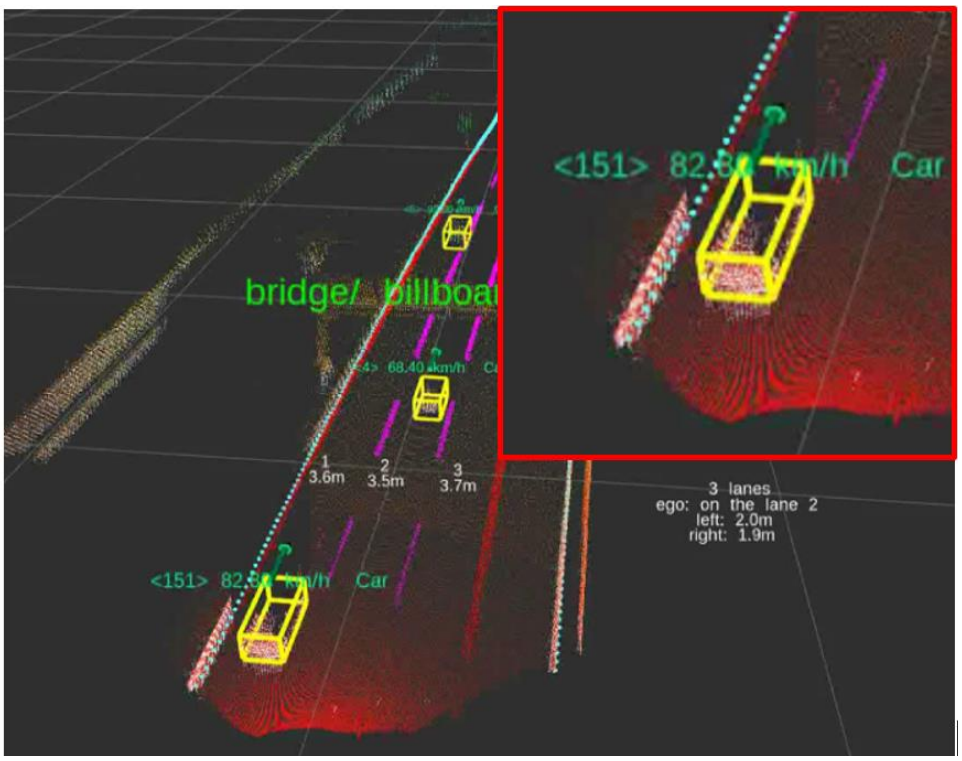


Figure 22 LiDAR 3D Point cloud of an Autobahn scene. © 2022 LiangDao GmbH

Despite the strong variations on point level the perception software is able to detect objects, because on point group level the perturbations are small. Therefore, since the perception software can be identical in the physical and digital counterpart of the system, the precise modeling of the noise in the digital system can be simplified (if not neglected, depending on the specific use case) except for corner cases where the point group size approached the level of single points. However, also the perception itself will start to fail when the point-group size reached single point size.

5.2 Validation in UC (S)TEM

5.2.1 Validation of Digital Models of (S)TEMs SOTP

Here we introduce several methods that have been use in TEM DT verification.

5.2.1.1 Visual inspection on outputs and trends:

1. Finding new representation of the outputs: Our analysis showed that using the Fourier transform (FT) of the CBED pattern gives an easier interpretable idea about the lower order electron beam aberrations, while the CBED itself carries a more combined information about the aberrations and the sample, and thus is more complex for a human to inspect. Therefore, often we use such processed images in our verification process.

2. One-to-one comparison of CBED for the same settings between real TEM and the DT outputs: See Figure 23 for an example.

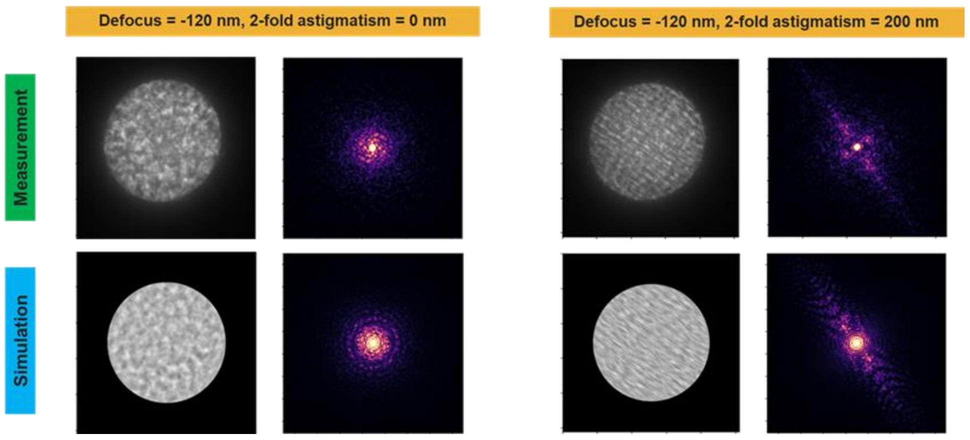

Defocus = -120 nm, 2-fold astigmatism = 0 nm
Defocus = -120 nm, 2-fold astigmatism = 200 nm
Measurement
Simulation

Figure 23 The CBED pattern and its Fourier transform. A comparison of real and DT data for specific settings of aberrations.

3. Trend in CBED changes due to changes in aberration coefficients: Figure 24 shows that even though the details of the images between real and DT are different, the trend in changes of aberrations are similar. Here amorphous Carbon was used as sample in DT. In real data some impurities were present. Figure 25 presents the same idea for a sample of crystal Silicon. In case of a crystalline sample the correct crystal orientation should be used in the sample description in DT.

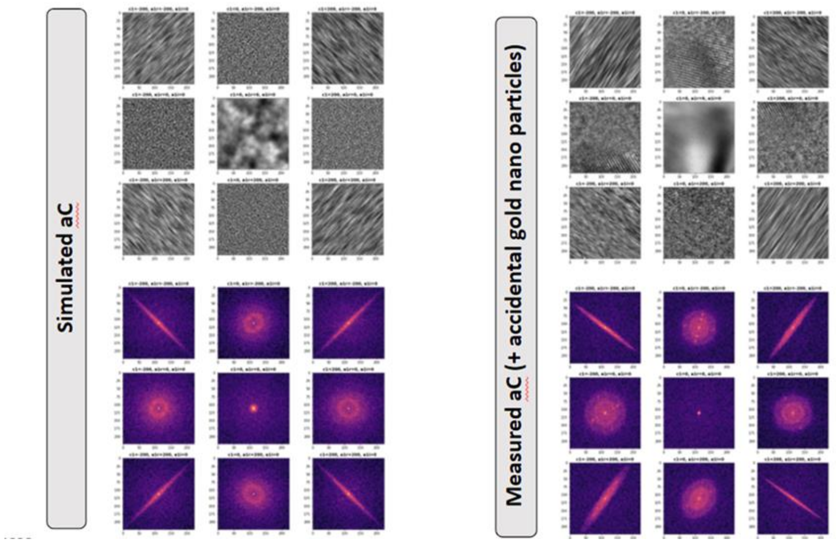


Figure 24 Amorphous Carbon, Ronchigram and its Fourier Transform

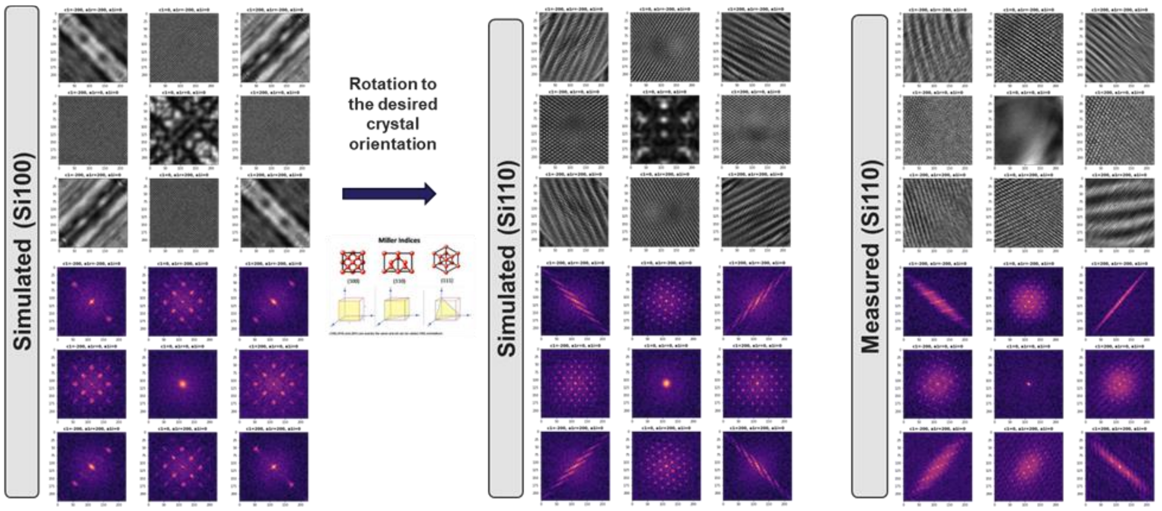


Figure 25 Silicon crystal, Si100 and Si110 orientations, Ronchigram and its Fourier Transform

4. Intensity distribution of real and DT datasets and sensitivity to DT parameters: We inspect the datasets generated by DT and by real instrument to ensure that the difference on the entire dataset is as small as possible. For example, we can use the intensity distribution of the images for all settings available in the dataset and identify if a clear structural difference is observed. Figure 26 shows such investigation. Here we concluded that for a certain set of parameters we can get closer to the real data in terms of intensity distributions.

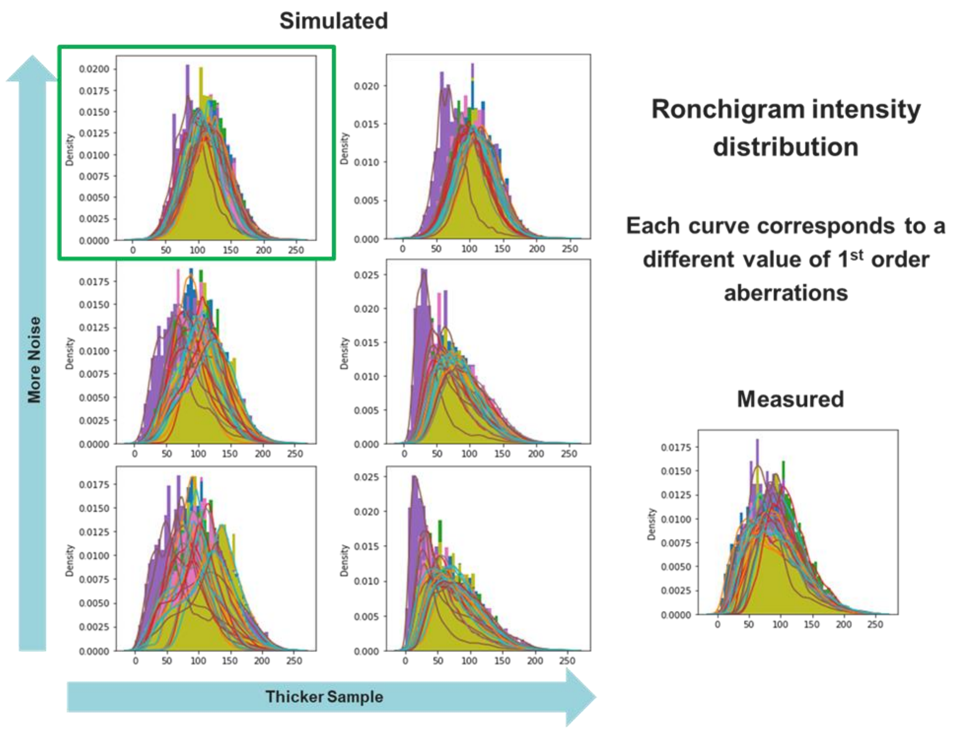


Figure 26 Comparing the intensity distribution of the entire datasets. The green box shows a specific simulation parameter which resulted in a better match between the real and the DT datasets.

5.2.1.2 Quantitative comparison

1. One-dimensional FT profile of CBED: We try to identify best simulation models and model parameters to find a closer profile to the real data profiles. The one-dimensional FT profile for the regime of large defocus is a good representation for estimation of the defocus value. See Figure 27 for an example.

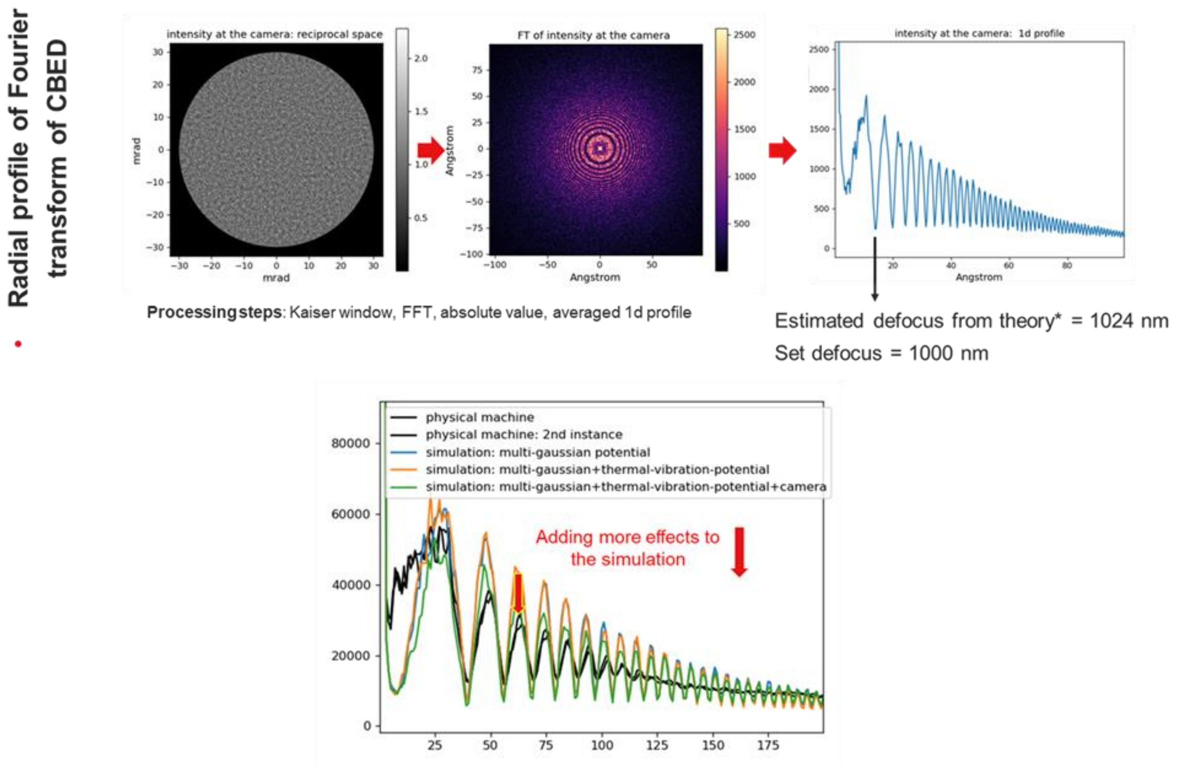


Figure 27 The comparison of the 1d profile of the FT of CBED for different model selection in DT and the real data. As a side benefit defocus value can be estimated for the oscillations of the 1d profile using the method described in [61]

2. A metric of comparison of image similarity: This metric is rather an enabler for verification. Additionally, it can be used for developing automatic calibration of the DT. Figure 28 shows an example. There, in each case one image from real TEM is compared to the simulated images with all settings presented in the grid. The colour bar shows the value of the metric. When camera noise is added to the simulation some metrics do not show a smooth change with changing the aberration settings (c1, a1), which indicates a poor metric regarding robustness to noise. While some other metrics are robust to noise to a level that is expected in real

camera. Moreover, a good metric also finds a best simulated image which is visually like the real image (shown in the lower row).

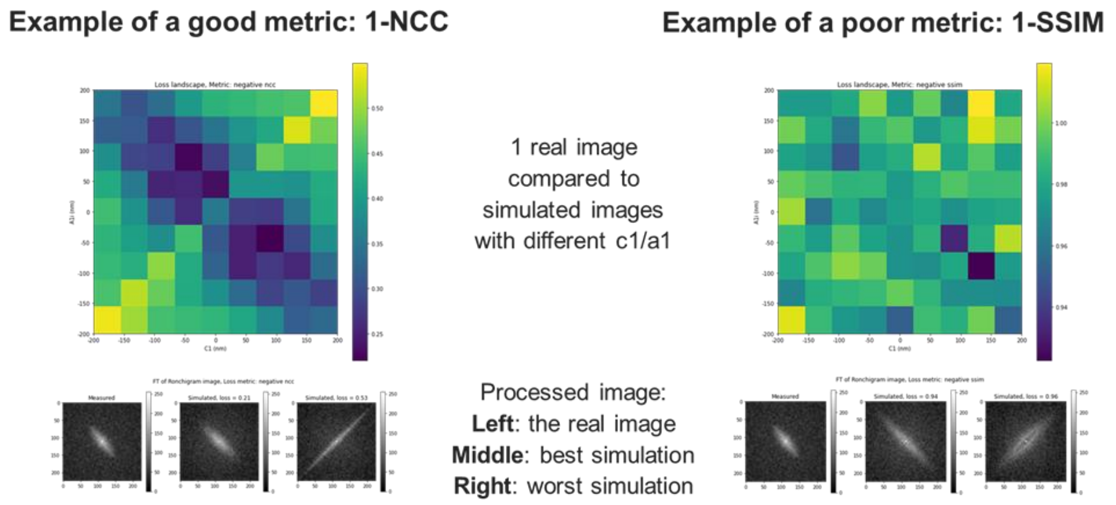


Figure 28 Example of good metric (Normalized cross correlation) and a poor metric (Structural similarity).

In our use case, a good metric should have the following properties:

- Showing a distinctive map: A completely noisy map or a map which is flat in big regions so that many images appear the same is not desired.
- The map is robust under realistic levels in simulation noise
- The best simulated image should visually look similar to the reference measured image
- Preferably the metric is bounded

5.2.1.3 Generalization of the AI model to real data:

This is an indirect method, as the effectiveness of the AI models and the DT models are combined. Yet This is the final verification for the purpose of the ASIMOV project.

6 Conclusion

This document described the V&V challenges that occurred and were discussed in the ASIMOV project, both in a conceptual and a practical, concrete way. We focused on topics that extend the existing, well understood practices in V&V of digital models. The topics include the different purposes of the DT in the steps and phases of the ASIMOV Solution. Based on those, we systematically explored functional and non-functional goals of the DT, for which we provided validation metrics. A subset of them has been applied in the use cases. Some of the metrics (e.g. those related to the fleet of DTs) will only become relevant later in the lifecycle of an ASIMOV Solution and could not be evaluated in the project yet.

7 Bibliography

[1] M. Grieves and J. Vickers, "Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems," 2017, pp. 85–113. doi: 10.1007/978-3-319-38756-7_4.
[2] M. Grieves, "Virtually Intelligent Product Systems: Digital and Physical Twins," 2019, pp. 175–200. doi: 10.2514/5.9781624105654.0175.0200.
[3] D. Wagg, K. Worden, R. Barthorpe, and P. Gardner, "Digital twins: State-of-the-art future directions for modelling and simulation in engineering dynamics applications," *ASCE - ASME J. Risk Uncertain. Eng. Syst. Part B Mech. Eng.*, Mar. 2020, Accessed: Jun. 02, 2021. [Online]. Available: https://eprints.whiterose.ac.uk/158771/

Version	Status	Date	Page
1.0	Public	2024-01-31	43/46

[4] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks, “Characterising the Digital Twin: A systematic literature review,” *CIRP J. Manuf. Sci. Technol.*, vol. 29, pp. 36–52, May 2020, doi: 10.1016/j.cirpj.2020.02.002.
[5] R. Doornbos, “ASIMOV Reference Architecture”.
[6] N. Braun, “ASIMOV Application to Systems - Methods and Techniques.” ASIMOV, Dec. 01, 2022.
[7] “ISO 9000:2015(en), Quality management systems — Fundamentals and vocabulary.” Accessed: Nov. 24, 2023. [Online]. Available: https://www.iso.org/obp/ui/#iso:std:iso:9000:ed-4:v1:en
[8] O. Balci, “Verification, validation, and accreditation,” in *1998 Winter Simulation Conference. Proceedings (Cat. No.98CH36274)*, Dec. 1998, pp. 41–48 vol.1. doi: 10.1109/WSC.1998.744897.
[9] “INCOSE SEBoK 2.8.”
[10] R. G. Sargent and O. Balci, “History of verification and validation of simulation models,” in *2017 Winter Simulation Conference (WSC)*, Las Vegas, NV: IEEE, Dec. 2017, pp. 292–307. doi: 10.1109/WSC.2017.8247794.
[11] P. Durst, C. Bethel, and D. Anderson, “A historical review of the development of verification and validation theories for simulation models,” *Int. J. Model. Simul. Sci. Comput.*, vol. 08, Jan. 2017, doi: 10.1142/S1793962317300011.
[12] O. Balci, “Golden Rules of Verification, Validation, Testing, and Certification of Modeling and Simulation Applications,” p. 7, 2010.
[13] E. Puiutta and E. M. S. P. Veith, “Explainable Reinforcement Learning: A Survey,” in *Machine Learning and Knowledge Extraction*, A. Holzinger, P. Kieseberg, A. M. Tjoa, and E. Weippl, Eds., in Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 77–95. doi: 10.1007/978-3-030-57321-8_5.
[14] R. Sargent, “Verification and validation of simulation models,” presented at the Engineering Management Review, IEEE, Jan. 2011, pp. 166–183. doi: 10.1109/WSC.2010.5679166.
[15] J. C. Hsu and M. Butterfield, “Modeling Emergent Behavior for Systems-of-Systems,” *INCOSE Int. Symp.*, vol. 17, no. 1, pp. 1811–1821, 2007, doi: 10.1002/j.2334-5837.2007.tb02985.x.
[16] J. Axelsson, “What Systems Engineers Should Know About Emergence,” *INCOSE Int. Symp.*, vol. 32, no. 1, pp. 1070–1084, 2022, doi: 10.1002/iis2.12982.
[17] S. Moritz, “ASIMOV D2.1 Parameter Identification Process.” Sep. 12, 2022.
[18] T. Lodewyckx, F. Tuerlinckx, P. Kuppens, N. B. Allen, and L. Sheeber, “A hierarchical state space approach to affective dynamics,” *J. Math. Psychol.*, vol. 55, no. 1, pp. 68–83, Feb. 2011, doi: 10.1016/j.jmp.2010.08.004.
[19] S. Brooks, *Handbook of Markov Chain Monte Carlo*. New York: Chapman and Hall/CRC, 2011. doi: 10.1201/b10905.
[20] M. Stone, “Cross-Validatory Choice and Assessment of Statistical Predictions,” *J. R. Stat. Soc. Ser. B Methodol.*, vol. 36, no. 2, pp. 111–133, Jan. 1974, doi: 10.1111/j.2517-6161.1974.tb00994.x.
[21] Y. Barlas, “Formal aspects of model validity and validation in system dynamics,” *Syst. Dyn. Rev.*, vol. 12, no. 3, pp. 183–210, 1996, doi: 10.1002/(SICI)1099-1727(199623)12:3<183::AID-SDR103>3.0.CO;2-4.
[22] Y. Barlas and S. Carpenter, “Philosophical roots of model validation: Two paradigms,” *Syst. Dyn. Rev.*, vol. 6, no. 2, pp. 148–166, 1990, doi: 10.1002/sdr.4260060203.
[23] J. W. Forrester, *Tests for Building Confidence in System Dynamics Models*. System Dynamics Group, Sloan School of Management, Massachusetts Institute of Technology, 1978.
[24] F. J. Anscombe, “Graphs in Statistical Analysis,” *Am. Stat.*, vol. 27, no. 1, pp. 17–21, 1973, doi: 10.2307/2682899.
[25] W. L. Oberkampf and M. F. Barone, “Measures of agreement between computation and experiment: Validation metrics,” *J. Comput. Phys.*, vol. 217, no. 1, pp. 5–36, Jan. 2006, doi: 10.1016/j.jcp.2006.03.037.
[26] “Anscombe’s quartet,” *Wikipedia*. Nov. 20, 2023. Accessed: Nov. 28, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Anscombe%27s_quartet&oldid=1186017553
[27] J. P. C. Kleijnen, “Verification and validation of simulation models,” *Eur. J. Oper. Res.*, vol. 82, no. 1, pp. 145–162, Apr. 1995, doi: 10.1016/0377-2217(94)00016-6.
[28] J. W. Bikker, B. Schriever, and M. Tijink, “Empirical models, understanding and validation,” 2023. [Online]. Available: www.cqm.nl/asimov/Empirical_models_understanding_and_validation
[29] R. G. Sargent, “Verification and validation of simulation models,” *J Simul.*, vol. 7, no. 1, pp. 12–24, 2008, doi: 10.1057/jos.2012.20.
[30] J. van Hulst, “ASIMOV Methods and Tools for Training AI with Digital Twin.”

[31] N. Braun, “ASIMOV Proof of Concept Demonstration and Evaluation of Unmanned Utility Vehicle.” Nov. 25, 2022.

[32] J. Langner, J. Bach, L. Ries, S. Otten, M. Holzapfel, and E. Sax, “Estimating the Uniqueness of Test Scenarios derived from Recorded Real-World-Driving-Data using Autoencoders,” *2018 IEEE Intell. Veh. Symp. IV*, pp. 1860–1866, Jun. 2018, doi: 10.1109/IVS.2018.8500464.

[33] L. Westhofen *et al.*, *Criticality Metrics for Automated Driving: A Review and Suitability Analysis of the State of the Art*. 2021.

[34] M. Aiguier, P. L. Gall, and M. Mabrouki, “A Formal Definition of Complex Software,” in *2008 The Third International Conference on Software Engineering Advances*, Sliema, Malta: IEEE, Oct. 2008, pp. 415–420. doi: 10.1109/ICSEA.2008.59.

[35] E. Roth *et al.*, “Analysis and Validation of Perception Sensor Models in an Integrated Vehicle and Environment Simulation,” *undefined*, 2011, Accessed: Jul. 13, 2022. [Online]. Available: https://www.semanticscholar.org/paper/Analysis-and-Validation-of-Perception-Sensor-Models-Roth-Dirndorfer/78d30e01397e735590b765aedb9fa916f6f1de12

[36] R. Doornbos, “ASIMOV D2.3 Architecture Of Digital Twins.” Jun. 28, 2023.

[37] Z. F. Magosi, H. Li, P. Rosenberger, L. Wan, and A. Eichberger, “A Survey on Modelling of Automotive Radar Sensors for Virtual Test and Validation of Automated Driving,” *Sensors*, vol. 22, no. 15, Art. no. 15, Jan. 2022, doi: 10.3390/s22155693.

[38] A. Schaermann, A. Rauch, N. Hirsenkorn, T. Hanke, R. Rasshofer, and E. Biebl, “Validation of vehicle environment sensor models,” in *2017 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2017, pp. 405–411. doi: 10.1109/IVS.2017.7995752.

[39] W. L. Oberkampf and T. G. Trucano, “Verification and validation benchmarks,” *Nucl. Eng. Des.*, vol. 238, no. 3, pp. 716–743, 2008.

[40] P. van Straten, “Validation of lidar simulation - Partnering with XenomatiX - Simcenter.” Accessed: Nov. 27, 2023. [Online]. Available: https://blogs.sw.siemens.com/simcenter/validation-of-lidar-simulation/

[41] “Sensors reference - CARLA Simulator.” Accessed: Sep. 14, 2023. [Online]. Available: https://carla.readthedocs.io/en/latest/ref_sensors/#lidar-sensor

[42] T. Hanke *et al.*, “Generation and validation of virtual point cloud data for automated driving systems,” in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, Oct. 2017, pp. 1–6. doi: 10.1109/ITSC.2017.8317864.

[43] P. van Straten, “Lidar systems simulation and validation - Simcenter.” Accessed: Nov. 27, 2023. [Online]. Available: https://blogs.sw.siemens.com/simcenter/lidar-systems-simulation-and-validation/

[44] H. Vanrompay, “ASIMOV Proof of Concept Demonstration and Evaluation electron microscopy.” Nov. 26, 2022.

[45] E. J. Kirkland, *Advanced Computing in Electron Microscopy*. Boston, MA: Springer US, 2010. doi: 10.1007/978-1-4419-6533-2.

[46] I. Lazić and E. G. T. Bosch, “Analytical Review of Direct Stem Imaging Techniques for Thin Samples,” in *Advances in Imaging and Electron Physics*, vol. 199, P. W. Hawkes, Ed., Elsevier, 2017, pp. 75–184. doi: 10.1016/bs.aiep.2017.01.006.

[47] U. Dahmen, T. Osterloh, and J. Roßmann, “Verification and validation of digital twins and virtual testbeds,” *Int. J. Adv. Appl. Sci.*, vol. 11, no. 1, p. 47, Mar. 2022, doi: 10.11591/ijaas.v11.i1.pp47-64.

[48] D. Reeb, K. Patel, K. Barsim, M. Schiegg, and S. Gerwinn, “Validation of Composite Systems by Discrepancy Propagation.” arXiv, Oct. 21, 2022. Accessed: Nov. 03, 2022. [Online]. Available: http://arxiv.org/abs/2210.12061

[49] J. García and F. Fernández, “A comprehensive survey on safe reinforcement learning,” *J. Mach. Learn. Res.*, vol. 16, no. 1, pp. 1437–1480, Jan. 2015.

[50] M. G. Kapteyn, J. V. R. Pretorius, and K. E. Willcox, “A probabilistic graphical model foundation for enabling predictive digital twins at scale,” *Nat. Comput. Sci.*, vol. 1, no. 5, pp. 337–347, Jan. 2021, doi: 10.1038/s43588-021-00069-0.

[51] J. Degrave *et al.*, “Magnetic control of tokamak plasmas through deep reinforcement learning,” *Nature*, vol. 602, no. 7897, pp. 414–419, Jan. 2022, doi: 10.1038/s41586-021-04301-9.

[52] C. Kober, V. Adomat, M. Ahanpanjeh, M. Fette, and J. P. Wulfsberg, “Digital Twin Fidelity Requirements Model for Manufacturing,” 2022, doi: 10.15488/12145.

[53] Y. Yu, “Towards Sample Efficient Reinforcement Learning,” *Proc. Twenty-Seventh Int. Jt. Conf. Artif. Intell.*, 2018.

[54] T. Bergs, S. Gierlings, T. Auerbach, A. Klink, D. Schraknepper, and T. Augspurger, “The Concept of Digital Twin and Digital Shadow in Manufacturing,” *Procedia CIRP*, vol. 101, pp. 81–84, Jan. 2021, doi: 10.1016/j.procir.2021.02.010.
[55] W. L. Oberkampf and S. Ferson, “Model Validation Under Both Aleatory and Epistemic Uncertainty.,” Sandia National Lab. (SNL-NM), Albuquerque, NM (United States), SAND2007-7163C, Nov. 2007. Accessed: Sep. 25, 2023. [Online]. Available: https://www.osti.gov/biblio/1146749
[56] S. Boschert and R. Rosen, “Digital Twin—The Simulation Aspect,” in *Mechatronic Futures*, P. Hehenberger and D. Bradley, Eds., Cham: Springer International Publishing, 2016, pp. 59–74. doi: 10.1007/978-3-319-32156-1_5.
[57] D. Silver *et al.*, “Mastering the game of Go without human knowledge,” *Nature*, vol. 550, no. 7676, Art. no. 7676, Oct. 2017, doi: 10.1038/nature24270.
[58] C. Metz, “In Two Moves, AlphaGo and Lee Sedol Redefined the Future,” *Wired*. Accessed: Sep. 26, 2023. [Online]. Available: https://www.wired.com/2016/03/two-moves-alphago-lee-sedol-redefined-future/
[59] J. Farebrother, M. C. Machado, and M. Bowling, “Generalization and Regularization in DQN,” arXiv, arXiv:1810.00123, Jan. 2020. doi: 10.48550/arXiv.1810.00123.
[60] L. Wright and S. Davidson, “How to tell the difference between a model and a digital twin,” *Adv. Model. Simul. Eng. Sci.*, vol. 7, no. 1, p. 13, Dec. 2020, doi: 10.1186/s40323-020-00147-4.
[61] A. R. Lupini, P. Wang, P. D. Nellist, A. I. Kirkland, and S. J. Pennycook, “Aberration measurement using the Ronchigram contrast transfer function,” *Ultramicroscopy*, vol. 110, no. 7, pp. 891–898, Jun. 2010, doi: 10.1016/j.ultramic.2010.04.006.