

Verification and validation of digital twins and virtual testbeds

Ulrich Dahmen, Tobias Osterloh, Jürgen Roßmann

Institute for Man-Machine Interaction, RWTH Aachen University, Aachen, Germany

Article Info

Article history:

Received Sep 10, 2021

Revised Dec 25, 2021

Accepted Feb 1, 2022

Keywords:

Digital twin

Model

Simulation

Validation

Verification

Virtual testbed

ABSTRACT

Our society is surrounded by technical systems that increasingly influence and take over aspects of everyone's everyday life. As we build on the correct functioning, the need for appropriate methodologies that support the development of reliable products persists. However, the development and in particular the qualification become challenging due to their increasing inherent complexity. Virtual testbeds and digital twins are modern key concepts in this context and offer a wide range of opportunities for simulation-based verification and validation processes with virtual prototypes, especially in cases where real tests are hard to perform. However, the validity of test results derived from virtual experiments depends on the quality of the underlying simulation models. Of course, the models themselves must be verified before. But in the context of virtual testbeds and digital twins this becomes challenging since the corresponding models are usually complex themselves. Therefore, we presented an approach to establish structured verification and validation activities for digital twins and virtual testbeds, based on a modular configuration framework that supports the development of complex simulation models.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Ulrich Dahmen

Institute for Man-Machine Interaction, RWTH Aachen University

Ahornstraße 55, 52074 Aachen, Germany

Email: dahmen@mmi.rwth-aachen.de

1. INTRODUCTION

System failures are a crucial issue especially in technology driven enterprises and societies. Accepting a defect level of 0.1% for example means accepting 20,000 defective drugs and 300 failing pacemakers per year, 500 errors during surgeries per week, 18 aircraft crashes and 16,000 lost letters in the post office per day as well as 22,000 incorrectly booked checks per hour. Even if we omit failures that cost lives, malfunctioning systems still can cause great harm. Especially late detected errors and design flaws cause high costs, summarized by the popular "rule of ten" (costs for an undiscovered error increase by a factor of 10 from level to level). It is therefore essential to verify and validate systems and products, in order to ensure that "the thing is right" (verification) and that "it is the right thing" (validation). However, the verification and validation of complex systems is a complex task itself. For example, it is expected that the functional verification of advanced driver assistance systems (ADAS) requires statistically more than 6.62 billion driven kilometers, which in terms of time, costs and test coverage are difficult to achieve with real test drives alone [1]. Furthermore, they have to be repeated for every change in the system. The supplemental use of systematically generated simulated test drives offers the possibility of achieving sufficient test coverage within a reasonable time. The use of digital twins as virtual substitutes of physical systems in the development process is therefore increasingly coming

into focus, as it allows early verification and validation by the use of virtual prototypes and thus supports the functional validation of the resulting overall system. The great advantage of using digital twins is that instead of setting up complex differential equations as in traditional modeling and simulation, virtual scenarios can be composed of encapsulated objects (the digital twins) with direct reference to real objects. Nevertheless, it is important to remember that we are talking about simulation. Every simulation imitates real processes in order to gain insights into a part of reality or to predict behavior, and every simulation is based on a model, an abstracted (idealized) representation of reality. Only if the model's behavior correctly matches the relevant characteristics of the underlying system (within a certain tolerance), we may draw inferences about the system from experiments with the model. Unfortunately a comprehensive proof of correctness for a specific simulation model is (except for trivial models) not possible [2].

As already mentioned, simulation scenarios based on digital twins have the advantage that they can be created rather intuitively, since virtual objects can be connected into applications via the same interfaces as their real counterparts. The complexity arises in particular from the dynamics in the mutual interaction of the individual objects and their environment. On the other hand, this object-centric approach blurs the influences of the individual components over the entire simulation scenario, which makes it increasingly difficult to verify and validate the simulation output. Thus, the need for systematic methods regarding verification and validation of digital twin based simulation models persists. This paper introduces an approach to cut the verification problem into smaller problems, each with a lower complexity. It further presents different examples for the application of structured verification and validation methods. The following section of this paper provides the conceptual background and summarizes the key aspects of a modular configuration framework for object oriented simulation models, which is the basis for the structured verification and validation approach presented in the third section. Section four demonstrates the application of this approach with several examples, followed by a conclusion.

2. PROPOSED METHOD

Large system models that only accumulate vast amounts of engineering data without structures and processes to manage this information in a structured way are practically impossible to verify. For the verification and validation of complex simulation models based on digital twins, it is therefore necessary to establish structured processes concerning model generation and execution first. The approach presented in this paper extends the modular configuration framework from [3]. Basically, the framework is designed for a simulation architecture that decouples simulation algorithms (implementing general physical laws of a certain domain) and model descriptions of specific objects (providing specific parameter values). And it allows managing relations between different simulation domains as well as different and usually over-time-changing level of detail and integration. This way it formalizes and structures the modeling process to handle the complexity of cross-domain simulation models. Figure 1 illustrates the top-level view of the framework defining three main entities: a set of digital twins, a virtual testbed, and a set of simulation algorithms. The resulting simulation system is called scenario and is composed of digital twins, representing the individual objects involved. It is executed ("running the simulation") in a virtual testbed. The virtual testbed therefore provides a simulator, facilitating the integration of various simulation algorithms from different domains. The following sections will explain the individual elements and their interrelationships in more detail.

2.1. Digital twin

The term digital twin goes back to Grieves, who introduced it in 2003 in the context of product lifecycle management [4]. Seven years later, NASA adapted the concept for aerospace and used the term in the context of "ultra-realistic simulations" [5]. Subsequently, driven by progressing digitalization initiatives, the term became very popular. Most recent realizations arise in the context of smart production, internet of things (IoT) and industry 4.0. [6], [7] provide a good overview of current trends related to the digital twin. Currently, several focuses can be identified, which result from a broad spectrum of applications. In the context of cyber-physical systems, the digital twin is often understood as a virtual representation of a real product in the internet of things [8]. At the same time, the term is also used for detailed simulation models and virtual prototypes in the field of virtual commissioning [9], [10]. Furthermore, it is also common to use the digital twin as a virtual mirror of a real system [11], as it was initially introduced by Grieves. Unfortunately, the various ways in which the term is used lead to different and sometimes contradictory definitions, which means that the overall concept of the digital twin is quite blurred. Nevertheless, none of the views is wrong in principle; they

make sense in their respective contexts. However, the ambiguity of the term often leads to misunderstandings, since the different understandings also result in different requirements and features of a digital twin. To avoid this, we want to point out that this paper focuses on digital twins in the context of simulation models.

As indicated in Figure 1 the digital twins defined by the framework consist of two key components each, the model library and a collection of several digital twin configurations. The model library provides basic model elements for the system to be represented by the digital twin, condensing relevant engineering data from part specifications. It is useful to organize the library in several categories, usually having one category for software components (forming the data processing system) and another for the hardware (physical parts). The latter should be structured hierarchically and mirror the system's physical architecture (structural decomposition, see Figure 2(a) to apply formal model specification processes like proposed in [12]. Following this, the hardware category can provide model elements M on each hierarchical layer (system, assemblies, components) allowing the management of different level of integration. Each model element may further appear in different variants V , representing various level of detail of the element. Modeling different level of detail is necessary for the application of the digital twin during the whole life cycle because the appropriate level of detail depends on several circumstances. Additionally, several perspectives P can be managed for each model element. Different perspectives consider different simulation aspects and correspond to physical domains like mechanics, electronics or thermodynamics. This breakdown facilitates a flexible decoupling between different simulation domains and different levels of integration, and allows for an independent model refinement process.

The model elements in the model library are used to create a meaningful object model. Depending on which model elements are linked together on which levels and in which variants, different configurations are possible. The resulting model is called "digital twin configuration". Figure 2(b) illustrates the principle structure of such a configuration, using model elements from three different perspectives (P_1 - P_3). The model elements are connected to each other and to the digital twin's interface via intrinsic interconnections. The presented configuration mechanism allows to create and manage different models of the same system, appropriate for different purposes. This is a central point, because different models of the same system may be required as the purpose of investigation changes. Thus, in the context of modeling and simulation, the digital twin of a system can be regarded as a structural model administration unit. In contrast to a traditional simulation model, the digital twin manages a multitude of simulation models and combines them to a semantic unit.

2.2. Simulation algorithm

Analogously, the same structuring considerations hold for simulation algorithms as well. There is usually no single universal simulation algorithm applicable to arbitrary application areas covered by digital twins. Consequently, in order to enable a flexible, cross-application, and life cycle spanning usage of simulation technology, it is vital to define standardized interfaces for the integration of simulation algorithms within a virtual testbed. Thereby, it is possible to flexibly select the most suited simulation algorithm for versatile domains and designated applications or analyses. Thinking ahead, it is conceivable and reasonable to carry on this concept to realize a fine granular functional decomposition of simulation algorithms by disassembling the simulation algorithms into predefined core components (e.g. integrator and solver). Since the role of all core-components might vary, the overall functionality of the simulation algorithm is composed by the assembly of the respective core components, which then can be chosen regarding the problem-specific requirements [13]. This approach introduces a great amount of flexibility in the configuration process, significantly expanding the range of application of the framework, and thus is the base for a comprehensive view of digital twins.

Regardless of the simulation domain and the respective algorithmic realization, the overall objective of each simulation algorithm is the interpretation and evaluation of the domain specific model information provided by the digital twin (i.e. the static model properties). The simulation algorithms are responsible for the realization of the extrinsic and intrinsic interconnections and consequently enable the interactive analysis and prediction of the digital twins behavior.

2.3. Virtual testbed

The third and final entity in the modular configuration framework is the virtual testbed. The term virtual testbed originates from the analogy to a real testbed and describes a further development of the virtual test bench. While the virtual test bench investigates individual aspects in simulation, a virtual testbed aims to integrate the entire system in its operational environment into the virtual world. This makes it possible to support systems with simulation during their complete life cycle. First references of virtual testbeds originate

from various application domains, e.g. communication technology [14], medicine [15], aerospace engineering [16], [17], building services [18], as well as IoT [19]. However, early virtual testbeds were mostly application-specific, providing simulation functionality only for special requirements and issues. A harmonization of the term is finally found in [20], which transforms the virtual testbed into an application-independent concept. Central components are generic scheduling functionalities [21] and data management technologies for the data exchange between simulation algorithms.

A cross-domain virtual testbed like this is also part of the modular configuration framework. It provides the link between digital twins and simulation functionalities and thus brings the digital twins and their environment to life. For this purpose, it is necessary to create a suitable scenario. Therefore, the overall system of interest is broken down into the individual systems involved and relevant aspects to be considered are derived, based on the actual purpose of investigation (conceptual modeling). The next step is to allocate corresponding digital twins for the involved systems and to select an appropriate configuration for each. At this point, it may be necessary to initially create new digital twins or to extend already existing digital twins by creating a new configuration. Finally, the models get interconnected adopting extrinsic interconnections between their interfaces, as illustrated in Figure 2(c). Based on the existing perspectives in the defined scenario the required simulation algorithms are chosen, configured and loaded to the core engine of the virtual testbed. Conclusively, the desired simulation can be performed.

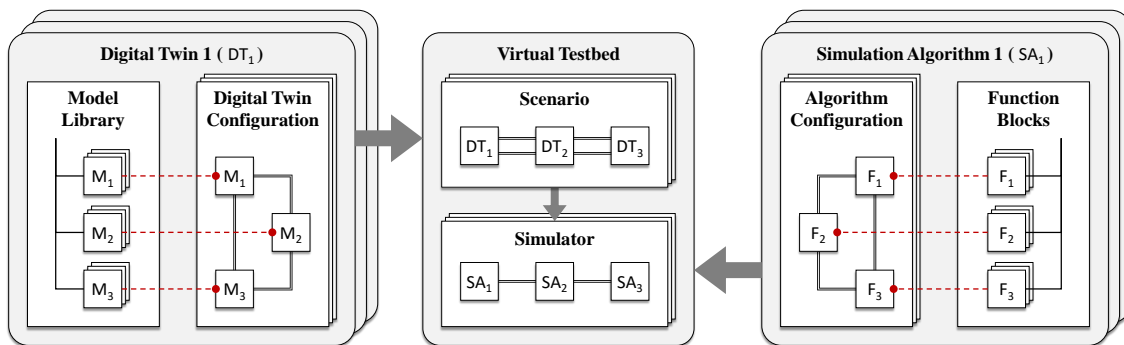


Figure 1. Concept of the modular configuration framework forming the basis for the structured verification and validation approach

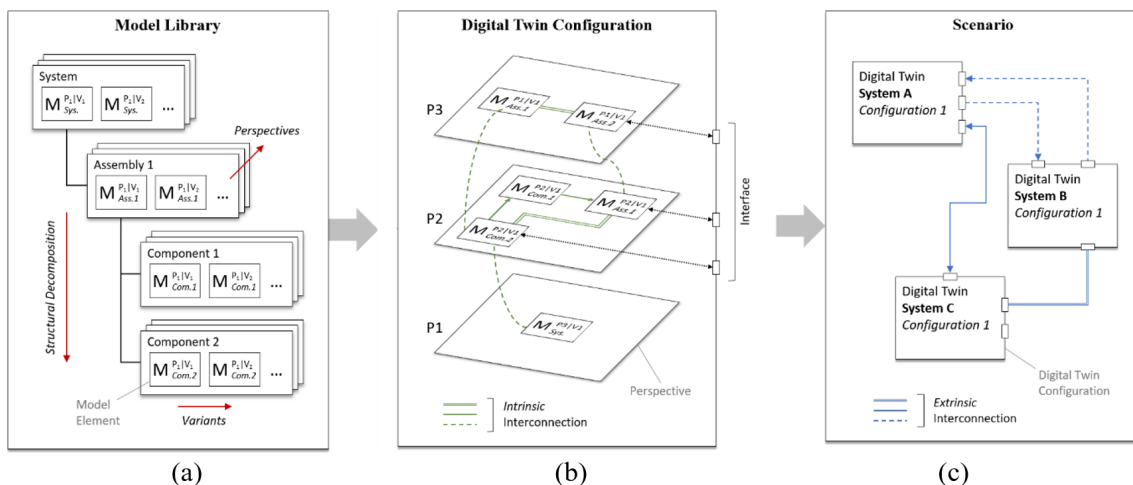


Figure 2. Configuration of a simulation scenario based on specific digital twin configurations, (a) detailed view on the structure of a digital twin's model library, (b) specific configuration of a digital twin, and (c) configuration of a specific simulation scenario composed of several digital twins.

2.4. Executive summary of the concept

A significant benefit of this framework is its flexibility and adaptability to different problems, which is an important prerequisite for the consistent use of modeling and simulation throughout the entire life cycle. On the one hand, there is a variety of digital twins, with each digital twin containing several model elements in different types and variants, from which digital twin configurations can be built. In turn, a wide variety of scenarios can be built from these configurations. On the other hand, there is a variety of simulation algorithms and each algorithm can consist of different modules that can be combined to form different configurations. Furthermore, a suitable virtual testbed can load simulation algorithms in different configurations into the simulator core and execute different scenarios.

3. RESEARCH METHOD

The increasing complexity of simulation models and interdependence of simulation functionalities blurs the verification and validation process and subsequently it remains an overwhelming task to trace back the influences of the various entities (digital twin, simulation algorithms and virtual testbed) to the validation result. Since every simulation bases on models (abstract and idealized representations of the real world) errors can occur at any point in a simulation study. Some errors are unavoidable (e.g. numerical errors), some are intentional (omission of physical phenomena to reduce the model complexity), but most are unintended. Especially since it is not possible to prove the correctness of a model (except for trivial models), it is necessary to gain confidence in the obtained simulation results by extensive verification and validation efforts. These activities must not be limited to the results of a particular simulation run, but instead must be considered during the development of the involved components. The modular architecture of the framework constitutes a base frame that allows to break down the complex verification and validation task into partial tasks that are easier to handle and thus allows to draw a reliable conclusion about the fidelity of digital twins. The principle areas that can be dealt with are largely independently of each other: i) Verification and validation of model elements; ii) Verification and validation of digital twin configurations; iii) Verification and validation of simulation algorithms; iv) Verification and validation of scenarios.

3.1. Verification and validation of model elements

The model elements in the model libraries are the elementary building blocks that are used to build a model. Their usually lower complexity significantly simplifies their verification and validation. During verification, the model code of the implemented model element is analyzed. This includes checking for syntactic correctness (is it a valid element?) as well as semantic correctness (does the implementation correspond to the concept model?). Since the model elements are written in a syntax appropriate to the used simulation tool, the concrete activities are tool-specific in detail.

During validation, on the other hand, the static model properties are reviewed. For this purpose, all physical characteristics (dimensions, weights and capacities) are compared with the values of the real component or the component to be developed. The corresponding reference values can be taken from data sheets or engineering design, depending on the application and project phase. The process can be automated if the development is combined with a model based systems engineering (MBSE) approach. In this case, the individual model elements are linked to the corresponding MBSE artifacts, which in turn reference product properties from the respective engineering domain. In case of a real existing component, values can also be validated by reference measurements, for example to confirm the position of the center of mass of a component calculated by a computer aided design (CAD) tool.

3.2. Verification and validation of digital twin configurations

Since successful verification and validation of all components does not necessarily result in a verified and validated configuration model, each digital twin configuration must be independently verified and validated. However, the verification results from the model elements can be used as a basis for this. The verification shall confirm that the model code correctly reflects the conceptual model. If a configuration was created with the help of the formal process, then the conceptual model is available as an extensible markup language (XML) specification describing a configuration table and a suitable interconnection table. In this case the verification can be automated by a static analysis of the model structure. If, on the other hand, the translation of the conceptual model into the model code is done automatically by appropriate tools, this step can be omitted.

However, this requires a successful verification and validation of the used model elements and confidence in the translator.

During validation it is examined whether the digital twin configuration represents the real system accurately enough. This especially concerns the critical evaluation of the selected level of detail and the selected level of integration. Both result from the configuration table and the interconnection table. Furthermore, reference experiments are possible. However, it must be taken into account that the closeness to reality of the algorithms used for simulation has a significant influence on the validation of the model. In principle, only algorithms that have already been verified and validated may be used here. Furthermore, the reference experiment must be designed in a way that the algorithm's validated range is not exceeded during execution. The similarity between simulated and real measured data of equivalent experiments can then be used to make a statement about the validity of the digital twin configuration. The specific metrics depend on the respective simulation domain.

3.3. Verification and validation of simulation algorithms

Simulation algorithms typically play a pivotal role as they describe basic physical effects that usually are independent from a specific application area. They also define which aspects of reality can be considered and thus modeled. Consequently, their verification and validation usually takes place at an independent or early stage. The verification of a simulation algorithm is supposed to show that the mathematical description of the considered physical effects is correctly converted into software code ("solving the equation right"). All methods from software engineering can be applied here to ensure code quality. This includes the use of suitable architecture principles (modularization and interface management), as well as measures to ensure their compliance (e.g. code reviews).

The validation of a simulation algorithm, on the other hand should ensure that all relevant physical effects are considered and that these effects represent reality adequately ("solving the right equation"). However, this alone turns out to be complex, since on the one hand the question for relevant effects cannot be generalized for all cases and on the other hand the question for a sufficiently accurate representation of reality depends on the level of detail of the mathematical formulation. This again depends on the application case. According to the use case, certain effects must be taken into account or can be neglected, or they must be considered in a very detailed or superficial way. For this reason, the framework provides different configurable variants of each algorithm. For a certain variant or configuration of an algorithm (which is often suitable for a whole class of similar problems) the question of validity can then be addressed. The standard method here is to perform suitable reference experiments. Therefore, representative cases in real and similar simulated test setups are considered and the results of the simulations are compared with the measured values. The choice of representative cases determines the scope of validity. Usually, entire parameter spaces are systematically traversed in order to generalize the results as far as possible and to be able to use the algorithm in various applications. However, a fundamental challenge here is that the level of detail of the used models influences or interferes with the examination of the closeness to reality of the algorithm. This must be taken into account when planning and evaluating the reference experiments, as well as the fact that the used models themselves must have already been verified and validated. Therefore, these experiments are usually performed under laboratory conditions with calibrated auxiliary objects.

3.4. Verification and validation of scenarios

The verification and validation of a scenario is typically the last step and also directly evaluates the simulation results obtained by the simulation. The verification ensures that the model has been implemented correctly and runs numerically stable. Typical measures for this are multiple calculations with different step sizes, the prior estimation of the expected results (this is qualitative and requires a lot of experience) as well as the calculation of control functions to check if, for example, conservation laws are kept during the whole runtime or if the system matrices are conditioned in a stable way.

The validation, on the other hand, checks whether the correct initial value problem has been solved. Therefore, especially the initial values of the scenario have to be examined. However, to perform reference experiments on a scenario level quickly becomes quite difficult. Usually a 1-to-1 comparison with real world ground truth data is no longer possible, since the scenario behavior are usually too chaotic to replicate exactly in reality or simulation. Furthermore, it is usually no longer possible to systematically run through parameter spaces because of the exploding number of variants and states. Consequently, it is useful to purposefully identify all relevant scenarios and test-cases beforehand, e.g. inspired by approaches as described in [22]. Based on

the identified scenarios the meaningful parameter combinations can be derived, narrowing the parameter space and concretizing the frame for the validation. This process can be performed iteratively, in order to purposefully converge to the region where a detailed validation is required (i.a. starting with a high-level sampling of the parameter space and refining the sampling in regions where the validation results need to be examined in detail). Since it is very unlikely to get a precise match between simulation and reality, it is recommendable to follow a slightly modified approach. Thereby, the comparison between simulation and reality is shifted to a higher level of abstraction, compensating the now missing spatial and temporal synchronization of simulation and reality. This can be achieved by deriving characteristic features of the scenario and defining metrics to calculate key performance indicators (KPIs). Then we can compare the KPIs calculated on simulation data with those KPIs calculated on real experiment data, see Figure 3. The virtual and real scenario do not need to fit exactly, they only must be equivalent, meaning provide the same characteristics. The captured simulation data itself may differ in detail, but if the same data processing algorithms, applied to both simulated and real data, calculate key performance indicators with a high congruence, it is a measure for a suitable modeling of the scenario's characteristics. Thus, the validation of the scenario is made by comparison on a semantic level. This approach can be called "weak validation" but turns out to be useful for functional verification and validation, since it allows to consider more realistic scenarios with bearable effort, which increases the practical relevance of the achieved results.

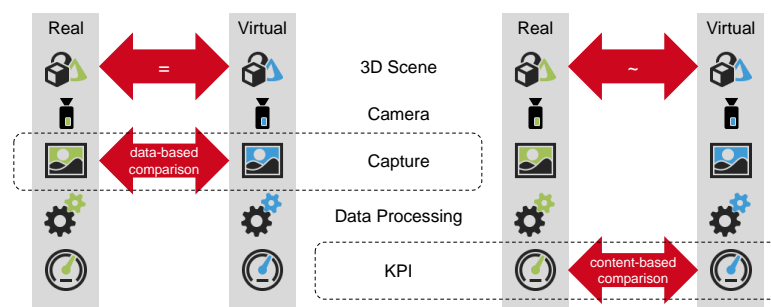


Figure 3. Scenario validation by using reference experiments and comparison on different levels

3.5. Executive summary of the approach

The presented concept for a step-by-step verification and validation has different advantages. One major benefit is the increasing reusability of individual verification and validation results leading to more reusable components and function blocks. A new application thus does not necessarily lead to a new validation of all components, which may reduce the effort for the simulation study significantly. Another aspect is the already mentioned reduction in complexity of each partial activity. In addition, one receives evaluable partial results, which makes the progress measurable. The following section will present some practical examples for verification and validation activities that comply with the concept.

4. RESULTS AND DISCUSSION

The presented modular configuration framework facilitates the decomposition of the verification and validation process in separate sub-problems and consequently provides the basis for tools to implement a tailored and well-suited verification and validation approach for a system of interest. The accumulation of all fused verification and validation concepts culminates in a structured and comprehensive end-to-end verification and validation of digital twins and virtual testbeds. Emerging from the basic frame of the presented structured verification and validation approach, the following examples present a generalized set of verification and validation measures that can be taken and also be transformed to arbitrary systems and domains.

Automated checks for plausibility and inconsistencies are presented for the verification and validation of model elements. The examination of digital twins is demonstrated by means of automated comparison of the configuration of the digital twin with formal specifications of the system. In addition, several tool-supported verification methods are demonstrated and domain specific analyses are performed. The verification and validation of simulation algorithms is pointed out by executing reference experiments and by the investigation of parameter influences to the simulation algorithms. The applications of the developed framework conclude

with the verification and validation of scenarios, where KPI-based analysis are applied and the linking of one scenario with different configurations of a simulation algorithm is exemplified.

4.1. Verification and validation of model elements

Even though verification and validation of single model elements may seem trivial and easy to realize, its importance should not yet be underestimated. Since the model elements are the atomic building blocks of digital twins and thus of scenarios, inconsistencies in single model elements might trickle through the different hierarchy levels of complex scenarios and result in erroneous simulation results. Consequently, in most cases it is practically hopeless to detect errors in single model elements by only validating complex scenarios.

4.1.1. Automated plausibility checks

Typically, the static model properties of model elements are set manually based on data e.g. from mechanical engineering. This process commonly is prone to error, since data has to be transferred between documents or software tools. Additionally, the modeling process and model abstractions often require to summarize some quantities. It is therefore useful to implement automated plausibility checks, to verify the static properties of the model elements. These checks may include: i) Numerical values have to be in certain parameter ranges, e.g. mass of body must be positive; ii) Some matrices must have certain properties, e.g. inertia tensor must be symmetric and positive definite; iii) Consistency of permissible limits, e.g. upper and lower limits of joint deflections in robot kinematics.

4.1.2. User interface support for the validation of static model properties

The automated plausibility checks might assist the verification of model elements, but will never completely be able to ensure the meaningfulness of the set of static properties. Therefore, a tool support might be useful, comprising all relevant static properties of model elements in a central user interface, see Figure 4. Thus, the manual validation of the model elements is eased. Prospectively, this might also be the initial point for additional adapted and tool specific automatized validation steps. Perhaps this might even serve as a bidirectional interface for the parameter exchange between model elements and different engineering tools.

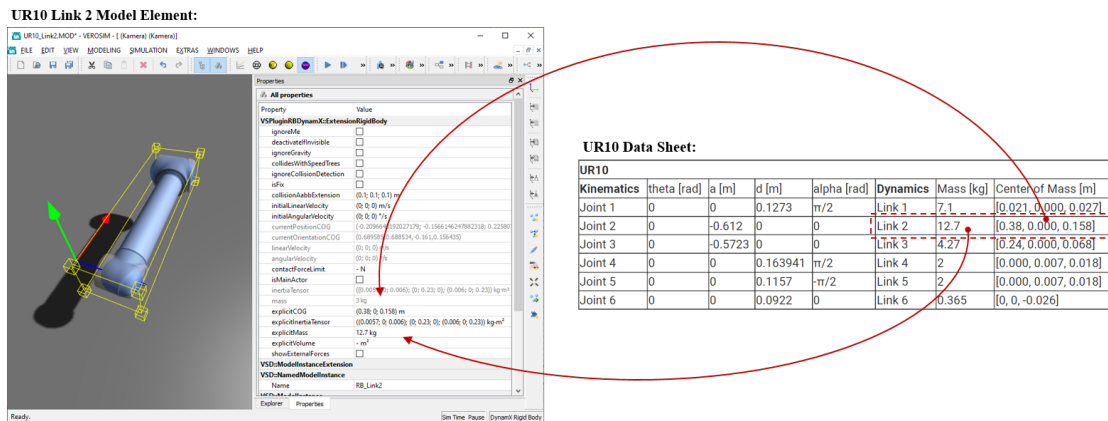


Figure 4. Validation of static model element properties

4.2. Verification and validation of digital twin configurations

The next logical step in the verification and validation process, seamlessly joining the verification and validation of the model elements, is the verification and validation of digital twin configurations. The consideration on the level of the individual configurations allows a more detailed statement on the validity of the digital twin, since different configurations can also have different scopes of validity.

4.2.1. Automated comparison to formal specification data

Especially in cases where models are supposed to be reusable as standard building blocks, it is useful to document the structure of the model by a formal specification. Possible adjustments to the respective application scenario usually only affect variable detail parameters, but not the basic structure. Before reusing the model again, it is recommended to ensure that the concrete model instance still complies with the specification

and is therefore implemented correctly. This can be done in preparation for a simulation run by means of a static model analysis. Starting point for this type of verification is a formal description of the model structure as it results from the configuration specification process in [12]. The process starts with a breakdown structure (from system to component level) of the physical system, which for example is created by systems engineers based on the system's physical architecture (if MBSE is applied). Although the process is basically defined tool-independent, a concrete implementation will generate result data in specific formats. For example, a first reference implementation creates a model configuration file in XML format that documents the specified model structure, the involved model elements and their interconnections. It enables the implementation of an automated check of the concrete model code for compliance with the formal specification. This check should be carried out after each change in the resulting configuration model code. Figure 5 shows an initial prototype that is capable to automatically check the configuration that is implemented for the simulation tool VEROSIM [23]. VEROSIM is a virtual testbed implementation that describes model components as a set of static object properties that are evaluated by simulation algorithm plug ins. Those algorithms themselves compose the corresponding differential algebraic equation system and perform the numerical integration on their own.

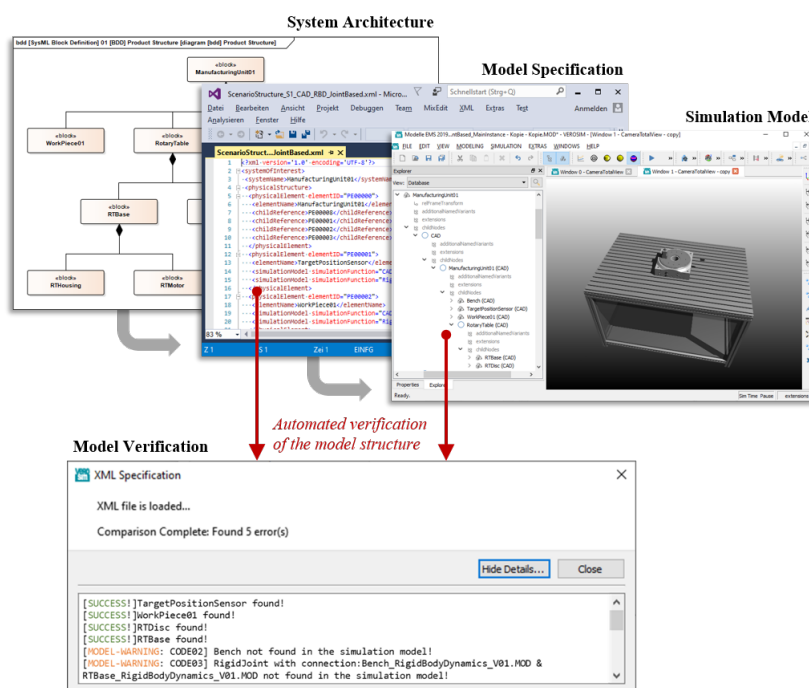


Figure 5. XML-based structural comparison of the simulation model with a formal model specification

4.2.2. Manual tool supported verification

Even models that appear rather simple on the outside often show a higher internal complexity (related to the respective implementation as data structure). As an example, we use the digital twin of an UR10 industrial robot, which is developed for a virtual testbed in an industrial environment. For path planning tasks, a kinematic model already exists. For the analysis of the robot dynamics and detection of collisions, the digital twin is extended by a configuration with a rigid body model. Figure 6 shows the 3D visualization of the rigid body replacement geometries (colored blue), which also serve as collision geometries. Using the 3D visualization, the basic position and shape of the individual rigid bodies can be easily checked, but the internal connection for example is hid. For this reason a verification method is implemented, which allows a static analysis of the rigid body structure. The analysis comprises several steps: First, all rigid bodies and their intrinsic interconnections (joints) inside the model are detected and listed with their properties, see right side of Figure 6. The list simplifies the manual verification of the physical properties of the individual rigid bodies (such as their masses and centers of gravity). The corresponding reference values can be taken from data sheets/specifications or reference measurements (validation of elements of the model libraries). In an additional step, the method supports the verification of the model structure by a graphical visualization of the resulting rigid body mesh with

its collider groups (the collider groups result from a joint property called "separateColliderGroup", that tells the simulation algorithm whether the rigid bodies involved in the joint can collide with each other). Looking at Figure 6 it is easy to identify the structure of the rigid body model as a kinematic chain. It is also visible that the individual links of the robot arm cannot collide with each other (because they belong to the same collision group, indicated by the same color), but they can collide with the environment. The configuration is obviously optimized for a special purpose of investigation (grouping into larger collision groups speeds up the calculation within the dynamics simulation). If the question should change in the future such that self-collisions of the robot are relevant, the configuration would have to be adapted accordingly. However, the necessary change is very small (a single property in the first joint) and it would hardly be noticed when the model is checked manually without tools. The presented verification method, on the other hand, makes it possible to quickly understand the actual model structure and check its suitability for the concrete application.

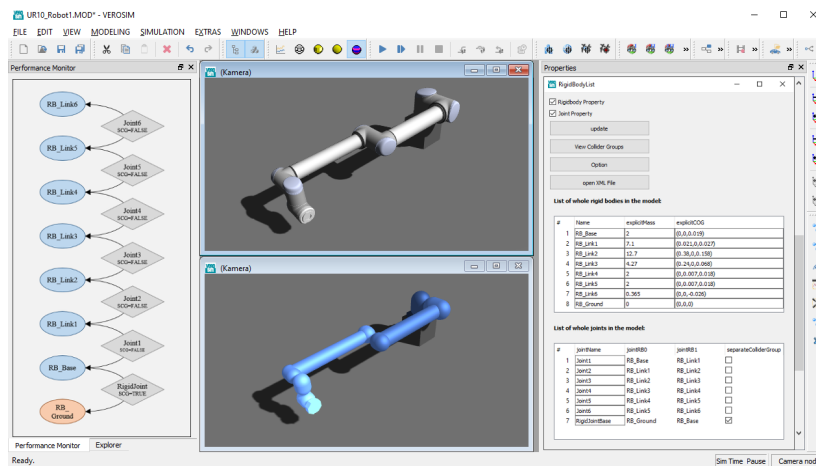


Figure 6. Several feature visualizations supporting manual verification of a digital twin configuration from an UR10 robotic arm

4.2.3. Automated verification based on metrics

Rigid body dynamics simulations in maximal coordinates are mostly based on the JMJT approach [24]. Following this approach, the system matrix \mathbf{A} comprises the system dynamics and consequently mirrors the dynamics behavior of the digital twin. In order to enable the simulation of interacting rigid bodies, contact and friction constraints are formulated based on linear complementarity constraints.

$$\overbrace{\mathbf{J} \cdot \mathbf{M}^{-1} \cdot \mathbf{J}^T}^{\mathbf{A}} \cdot \vec{\lambda} + \vec{b} = \vec{a}$$

$$\vec{\lambda} \geq \vec{0}, \vec{a} \geq \vec{0}, \vec{\lambda}^T \cdot \vec{a} = 0 \quad (1)$$

Since the system matrix \mathbf{A} solely depends on the configuration of the digital twin, the configuration of the digital twin can be analyzed by properties and metrics derived from the system matrix \mathbf{A} . One important metric of the system matrix is the condition number κ derived from a singular value decomposition (singular values $\sigma_{max}, \sigma_{min}$) of the matrix \mathbf{A} .

$$\kappa = \frac{\sigma_{max}}{\sigma_{min}} \quad (2)$$

The singular values (and thus the condition number) give a detailed insight into the major relevant influences to the overall system dynamics, since both metrics are significantly influenced by two factors: the system layout and configuration and the masses of the rigid bodies within the system. An automated analysis of the condition number therefore facilitates a static analysis of the configuration of a digital twin. In any case, a high condition number indicates the existence of system components that mainly influence the system dynamics, whereas other components could be neglected as well (e.g. uneven distribution of masses between

rigid bodies or a combination of very fine granular modeling and large structures in the same system). Thus, a static analysis of the condition number indicates possible inconsistencies in the configuration of a digital twin. As a result, the configuration of the digital twin should be adjusted by narrowing down the digital twin to only relevant components. This way, numerical instabilities of rigid body dynamics simulation algorithms that arise from ill-condition system matrices \mathbf{A} might be avoided as well.

4.3. Verification and validation of simulation algorithms

The accuracy of the simulation algorithms integrated to the virtual testbed is a fundamental prerequisite for the accurate simulation and analysis of digital twins. Thus, a detailed multilayered verification and validation of the applied simulation algorithms is vital. Solely in some cases, the validation might be performed analytically, but mostly it is rather difficult to pursue such analytical approaches. In this context, reference experiments are the holy grail for the validation of any simulation algorithm. Despite their complexity, reference experiments should be performed for the validation of each simulation algorithm that will be used in a virtual testbed.

4.3.1. Reference experiment for the validation of sensor simulation

Modern cars are equipped with an increasing amount of environmental sensors to enable a wide range of driver assistance systems and, in the future, autonomous driving. Among others, camera sensors are used for this purpose. However, the validation of such systems becomes more and more challenging. To reduce the number of real test drives, a virtual testbed with corresponding simulation algorithms for the simulation of optical sensors is used. In order to validate such driver assistance systems using simulated test drives, the corresponding simulation algorithms must first be validated. In this example, reference experiments under laboratory conditions are carried out in a first step. This approach compares real world experiments and virtual experiments directly on the sensor data layer. Since the laboratory allows to establish a very high congruence between real and virtual experiment setup (create nearly identical 3D scenes), the captured sensor data should provide a high congruence as well. Thus, the level of congruence can be used as a measure for the validity of the corresponding simulation algorithms (assuming that the model elements are sufficiently similar).

Figure 7 shows that the simulated and real capture are obviously very similar. A real world experiment as shown in Figure 7(a) and from an equivalent simulated experiment as shown in Figure 7(b). In order to also get quantitative measures, we apply different metrics in the following. The first metric should compare the image's sharpness. To calculate the sharpness of an image it is converted into grayscale and afterwards filtered with the Laplacian operator applying the same-named open source computer vision library (OpenCV) function [25]. The Laplacian function sums up the second x and y derivative of the image and can be used for edge detection, because in areas with edges the pixel values show a high variation of intensity. Since sharper images usually have more distinct edges, the Laplacian filtered image will show more light and dark areas. Consequently, the variance of the intensity values increases. Therefore, we calculate the variance of the filtered image as a measure for the sharpness of the original image (the greater the value, the greater the sharpness). In the given example, a value of 72.6 is calculated for the simulated image and a value of 76.5 for the real image. This is a very good match. However, the comparison of the sharpness of two images is only useful in combination with other comparison indicators, since two images with completely different contents can also have similar sharpness values.

Another metric is the analysis and comparison of the histograms of the two images (however, again it must be taken into account that images with different contents may still have similar histograms). Therefore, both images are first converted into the hue, saturation, value (HSV) color space. The reason for this is that unlike red, green, blue (RGB), HSV separates the luma (image intensity) from the chroma (color information). The luma strongly depends on the amount of light in the scene and varies with lighting changes and shadows. Chroma, on the other hand, correlates with the object's intrinsic properties and is usually more invariant to the lighting situation. Thus, we focus on the chroma by calculating HS-histograms (hue and saturation). The histograms in Figure 8 use 180 bins for hue values and 256 bins for saturation values and are normalized to show the relative frequencies. Both histograms are then compared to each other using the OpenCV function `compareHist` [26] with the "correlation" as comparison metric. As a result, we get a numeric parameter that expresses how well the histograms of the two images match with each other. For the metric correlation, higher values indicate a better match (the maximum value is 1.0). The value for the two histograms in Figure 8 is 0.676, revealing that in detail the colors show greater differences than visible at first sight. Figure 8(a) is

showing comparing hue values of real (blue) and virtual (orange) image and Figure 8(b) is showing comparing saturation values of real (blue) and virtual (orange) image.

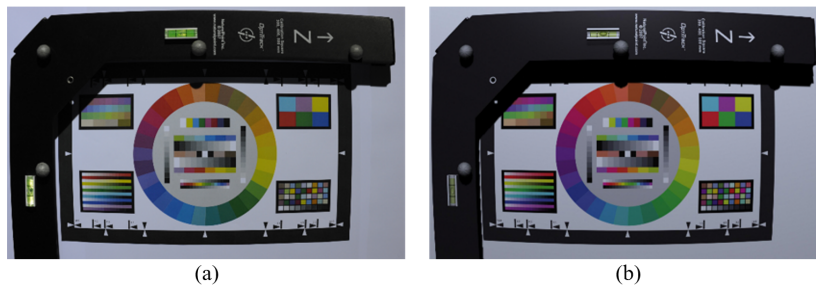


Figure 7. Comparison between captured image from (a) a real world experiment and (b) from an equivalent simulated experiment

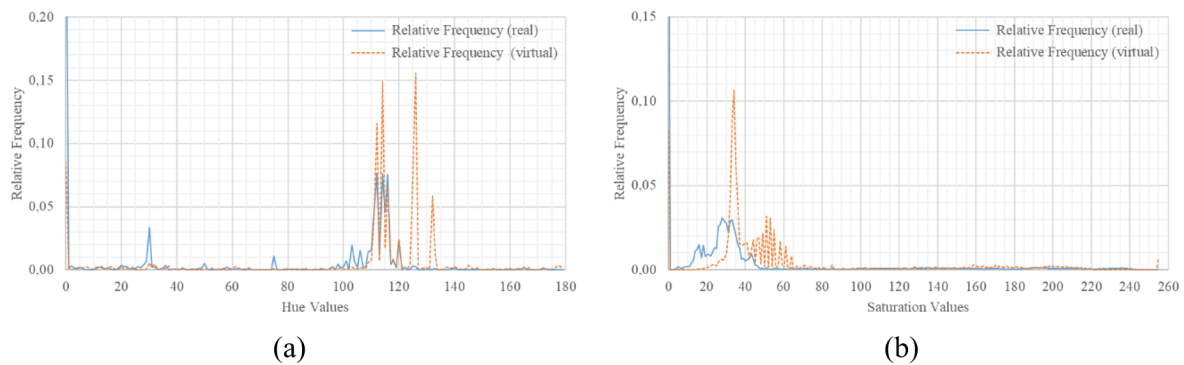


Figure 8. Comparison of histograms (a) comparing hue values of real (blue) and virtual (orange) image and (b) comparing saturation values of real (blue) and virtual (orange) image

Another measure for the similarity of two images is the structural similarity index measure (SSIM), which was designed to improve traditional methods such as peak signal-to-noise ratio (PSNR) and mean square error (MSE). The SSIM [27] ranges from zero ("not similar") to one ("similar") and is defined as (3):

$$SSIM(\mathbf{x}, \mathbf{y}) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \quad (3)$$

In practice, usually a single overall quality measure of the entire image is required, so we use the mean SSIM to evaluate the overall image similarity:

$$meanSSIM = \frac{1}{M} \sum_{j=1}^M SSIM(\mathbf{x}_j, \mathbf{y}_j) \quad (4)$$

In the actual example we calculate the meanSSIM separately for each color channel and get the following results: Red channel meanSSIM = 0.74, green channel meanSSIM = 0.75, and blue channel meanSSIM = 0.72. These values confirm the observations made in the histogram comparison.

Since we have two images that by principle show a certain degree of similarity, we apply a template matching approach as another metric. In general, template matching is a technique for finding areas inside an image that match (are similar) to a template image (patch). To identify the matching area the template image is moved pixel by pixel over the source image. At each location, a metric calculates a score to indicate how "good" or "bad" the match at that location is. The result is a matching matrix where the location with the highest value (respectively lowest value, depending on the used metric) marks the considered match. That in mind, we move

the two images over each other and determine matching scores using the OpenCV function `matchTemplate` [28] with the "normalized cross correlation" as metric. The extremum finally gives us a statement about a possible offset in the image section via the pixel coordinates of the matched rectangle as well as a value for the similarity via the corresponding score. The images in Figure 7 result in an offset of the match position for the upper left corner of exact zero pixel (in x and y direction), confirming the experimental setup. Furthermore, the matching score of 0.979 (best possible value is 1.0) indicates a very high congruence.

Finally, a 2D feature matching metric serves to find and compare features in both images. A feature is a characteristic image region that is very unique and can be easily recognized. Finding image features is called feature detection, usually followed by a feature description where the region around the feature is characterized so that it can be found in other images. Feature detection and description typically create a feature key point and a feature descriptor. The key point is characterized by the feature's 2D position, scale (proportional to the diameter of the neighborhood that needs to be taken into account), orientation and some other parameters. The descriptor contains the visual description of the patch and is used to compare and match key points representing the same object in different images. This is called feature matching. Here we use the oriented FAST and rotated BRIEF (ORB) detector to detect key points and compute descriptors. ORB is basically a fusion of the features from accelerated segment test (FAST) key point detector and binary robust independent elementary features (BRIEF) descriptor with modifications to enhance the performance. The feature detection in both comparison images is followed by a brute-force matcher to find similarities between those features. The matcher takes the descriptor of one feature in the first set and matches it with all features in the second set, returning the one with the smallest distance. All matches are then sorted according to their distances in ascending order, so that the best matches with a small distance come to front. Thus, the distances of the top matches serve as a measure for the similarity of the images. Figure 9 shows the top 160 feature matches and their corresponding positions for the given example. It is obvious that identical features were detected. To get some more details, we calculate the difference in the position of the associated key points for each feature match. The average deviation in x-direction is only -2.23 pixels with a variance of 9.63, and in y-direction only 0.80 pixels with a variance of 6.50. Considering that the images have a resolution of 1900 x 1200 pixels, the conformity of the features is very high.

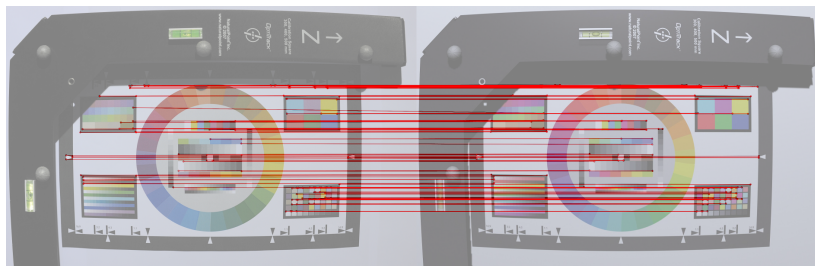


Figure 9. Semantic validation based on a feature comparison between real and simulated image

4.3.2. Reducing influences of the model on the validation of the simulation algorithm

The challenging part of a reference experiment is to extract the respective influences of the simulation algorithm and the simulation model to the superimposed result of the self-contained validation. The following approach seamlessly integrates into the proposed configuration frameworks and aims to reduce the influence of the simulation model to the validation process. Therefore, it is crucial to eliminate the uncertainties of the simulation model on the validation and thus concentrate solely on the simulation algorithm in order to investigate the decisive factor for the accuracy of the virtual testbed. The model uncertainties (the static properties of the digital twin) are described by a multivariate normal distribution, where the mean values μ and the covariance matrix Σ describe the statistical couplings of the model parameters.

$$D \sim N(\mu, \Sigma) \quad (5)$$

Based on this distribution, Monte Carlo Simulations for randomly generated sets of model parameters are performed. Each parameter set is generated with respect to the model uncertainties. The simulation output of a variety of simulation runs is compared to the reference data from the experiment. If the output of all simulation runs stays within a certain range around the measured reference data, it is safe to say that the applied

simulation model (in the range of the model uncertainties) does not have a major impact to the simulation output and the simulation algorithm can be considered validated regarding the given accuracy requirements. If no prior knowledge about the model uncertainties D is given, the parameter ranges can also be defined taking expert knowledge into account and a brute force exploration of the parameter space can be applied. If the parameter ranges are chosen properly, the qualitative results will be the same, but most certainly the computational efficiency will be much less.

We apply this validation approach to the dynamic simulation of a KUKA LWR-4 robotic arm. Therefore, the aforementioned rigid body dynamics simulation algorithm is applied. The model uncertainties are estimated based on multiple distinct parameter identifications [29]. In order to duplicate the real world reference experiments, the exact same trajectory was replicated in a virtual experiment and the reference torque was compared to the simulated torque. Figure 10 shows the result of 1000 validation runs. The measured reference torque is shown in red and the results of all validation runs is depicted by the blue highlighted area. As the shape of the reference data is reproduced by all simulation runs and the deviation is highly limited, the simulation model (within the given uncertainties) does not majorly influence the simulation output. Consequently, the simulation algorithm is the decisive factor for the accuracy of the simulation algorithm and is capable of reproducing and mirroring the real-world dynamics of robotic systems.

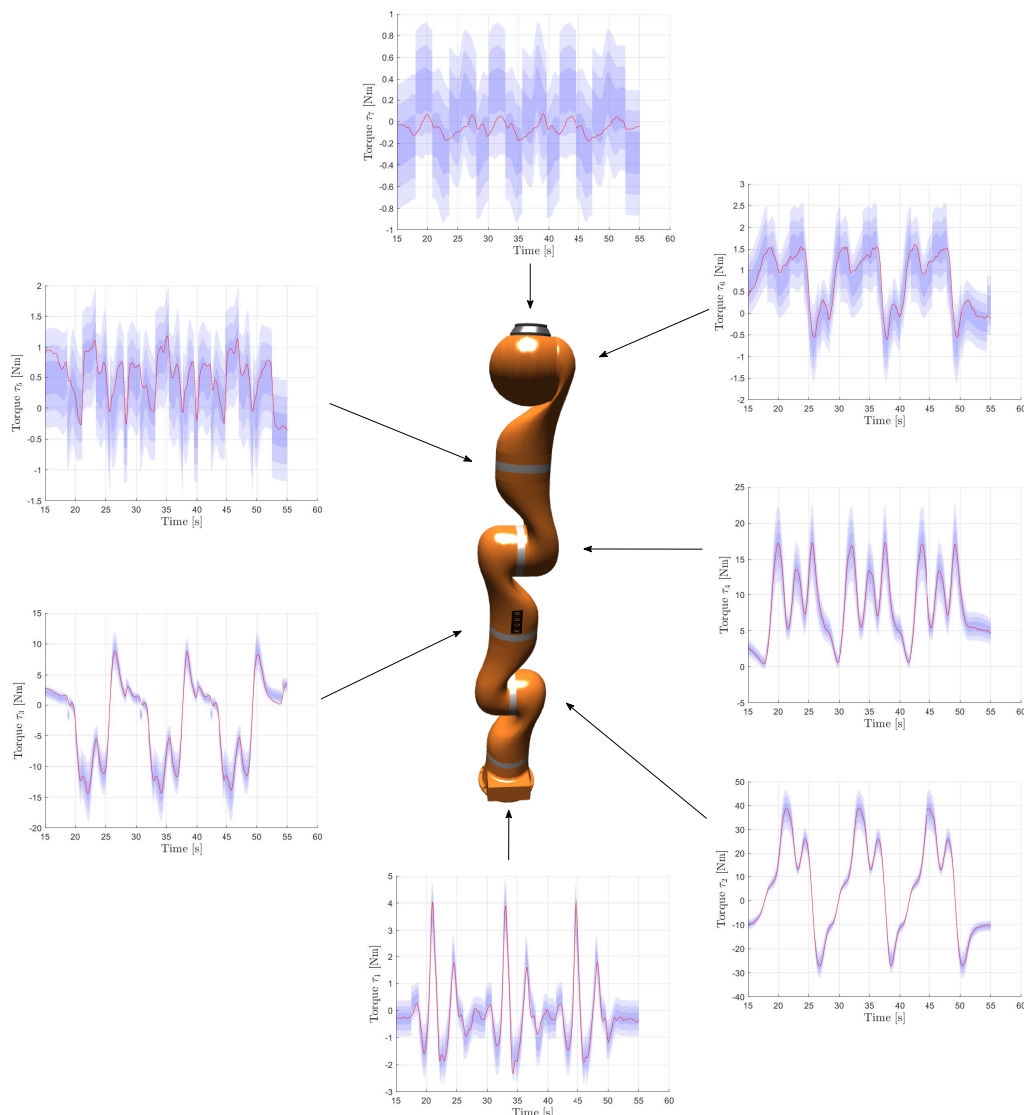


Figure 10. Validation of rigid body dynamic simulation for a KUKA LWR-4 digital twin

4.4. Verification and validation of scenarios

Based on the previous steps, the expense for the verification and validation of a scenario will be significantly reduced. Calling Figure 1 to mind we already verified and validated the digital twin with its model elements and configurations (left side) and the simulation algorithms (right side). Even though all these entities have been validated, it is not safe to say, that the complete scenario and its fusion with the simulation functionality is validated as well. Consequently, if the validation of the scenarios reveals some unexpected inaccuracies, the validation should iteratively be taken back to the previous steps and be performed in more details, until all entities (model elements, digital twins, simulation algorithms and scenario) can be considered validated.

4.4.1. Validation based of key performance indicators

For a scenario that is mainly based on rigid body dynamics, it is reasonable to validate the accuracy based on the fundamental conservation laws of physics (conservation of momentum and conservation of energy). The energy balance $E(t)$ can be determined directly based on the results of every simulation step. Taking into account all active components $E_{active}(t)$ (e.g. actuators) and all dissipative components $E_{dis}(t)$ (e.g. dampers or friction) within the system the energy balance should stay constant over time. Based on (6), the validation approach can be used to dynamically validate entire scenarios by determining the energy balance during the execution time of the simulation. Thereby, the simulation engineer is in the loop with the simulation and can directly evaluate the validity of the simulation.

$$E(t) = E_{stored}(t) - E_{active}(t) + E_{dissipative}(t) = const. \quad (6)$$

Applying this key performance indicator based validation approach, it is possible to generically validate complex scenarios and quantitatively compare scenarios, which at first glance are characterized by almost identical behavior. In order to demonstrate this validation approach, we investigated and validated the scenario of a robotic working cell equipped with the KUKA LWR-4. For the purpose of validation the robot performed randomly generated motions, schematically depicted in the middle of Figure 11. The diagrams to the left and right show the validation results of the overall scenario. The left diagram shows the validation, where all actors were realized based on external forces. The right diagram shows the validation, where all actors were realized based on constraints. Comparing both error plots, it is obvious that the chosen configuration of the scenario and simulation algorithms majorly influences the accuracy of the digital twin.

If the chosen validation approach directly emerges from the basic mathematical principles of physics, the validation approach can seamlessly be transferred to even more complex scenarios (e.g. robotic production lines). Additionally, using the conceptualized basic idea of applying the energy balance as key performance indicator of the scenario can smoothly be transferred to arbitrary scenarios covering other simulation domains like thermal simulation or simulation of electrical circuits.

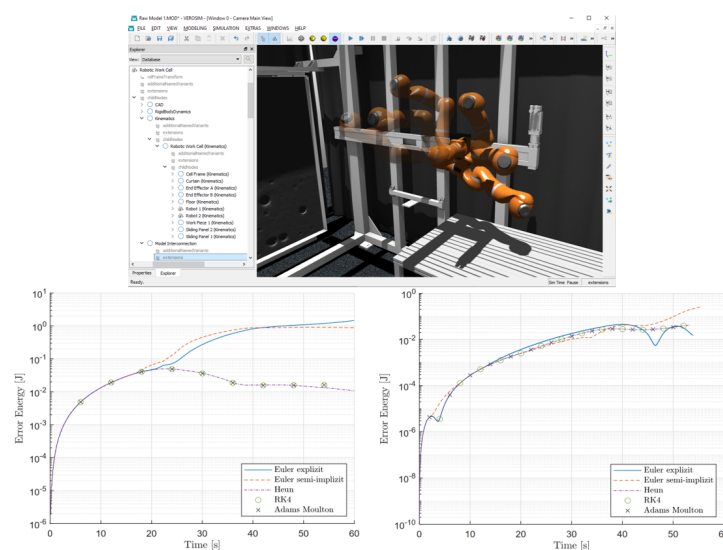


Figure 11. Validation of a robotic working cell scenario using the time-resolved energy balance

4.4.2. Validation of the choice of simulation algorithms

It is inevitable that the simulation algorithms loaded to the core engine of a virtual testbed must be chosen by experts. Nevertheless, even experienced engineers might make bad choices. Subsequently, it is crucial to validate the fusion of the scenario and the chosen simulation algorithms, i.e. make sure that the scenario can be simulated properly by the chosen simulation algorithms.

Based on the previous considerations on rigid body dynamics simulation, a condition number of the system matrix \mathbf{A} equal to infinity indicates singular systems (at least one singular value is equal to zero). This mostly results from redundant constraints within the system (e.g. dynamic simulation of mobile machinery). As all linear complementarity problem (LCP) solvers are based on an inversion of the system matrix \mathbf{A} , singular systems require adequate solving strategies (i.e. a special configuration of the simulation algorithm is needed): i) Regularization of the system matrix: By adding small additive values $C_{cfm} > 0$ to the main diagonal of the system matrix, it can be assured that a singular matrix becomes positive definite. Accordingly, the inverse dynamics (1) has a unique solution. Nevertheless, this comes by the cost of additional artificial compliance within the simulated system (see constraint force mixing principle) [30]; ii) Adaption of solver: As the system matrix \mathbf{A} is positive semi definite, the LCP can be transformed in an equivalent QP [31]. Consequently, a quadratic programming (QP) solver can be applied to calculate the inverse dynamics as well. As QP solvers are able to handle singular system matrices \mathbf{A} , the artificial compliance of the regularization is avoided $C_{cfm} = 0$.

Subsequently, the chosen simulation algorithm must be chosen appropriately for each scenario: Due to parallel kinematics, the simulation of mobile machinery most often results in singular system matrices. The following example demonstrates the influence of the two presented approaches on the simulation of a wheel loader. Figure 12 qualitatively contrasts the LCP solving approach with regularization of the system matrix as shown in Figure 12(a) and a QP solver without regularization, see Figure 12(b). As shown the simulation sequences in Figure 12, the artificial compliance of the LCP based regularization approach causes the shovel of the wheel loader to slowly drop over time, whereas the QP based simulation is able to hold up the shovel. Even though the simulation results at first glance look alike, a detailed quantitative analysis of the forces applied by the hydraulic cylinders show major differences, as illustrated in Figure 13.

The accuracy of the QP based simulation approach is traded for additional computational costs, thus the configuration of the simulation algorithm has to be performed wisely and must be matched to the respective requirements of the simulation scenario. Figure 13(a) is showing $C_{cfm} = 0:0$ and QP-Solver and Figure 13(b) is showing $C_{cfm} = 0:0001$ and LCP-Solver.

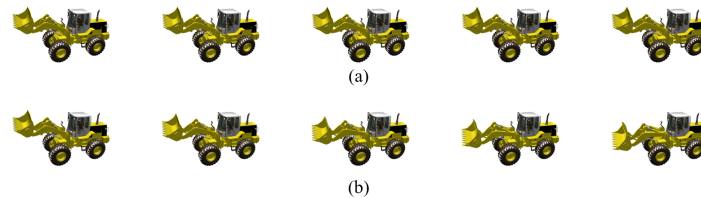


Figure 12. Qualitative comparison of inverse dynamics solving approaches (a) Simulation of a wheel loader with $C_{cfm} = 0.0$ and QP-Solver, (b) Simulation of a wheel loader with $C_{cfm} = 0.0001$ and LCP-Solver

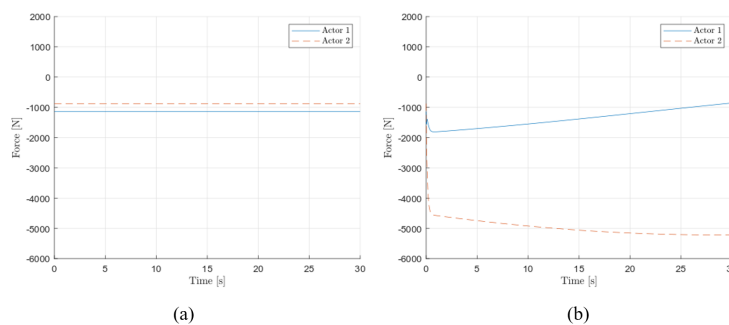


Figure 13. Quantitative comparison of inverse dynamics solving approaches, (a) $C_{cfm} = 0.0$ and QP-Solver, (b) $C_{cfm} = 0.0001$ and LCP-Solver

5. CONCLUSION

Various modern applications demonstrate that digital twins and virtual testbeds are key technologies for the development and operation of modern systems and complex system of systems for the sustainable and future-oriented usage of digital twins and virtual testbeds a well-suited and flexible approach for comprehensive verification and validation of the underlying structuring elements and technologies is required. Consequently, we built upon the previously presented modular configuration framework and presented a structured multilayered verification and validation approach, that takes on the persisting challenges in the verification and validation of technical systems. The fundamental approach is to disassemble the complex problem in smaller distinct and manageable subproblems. The unification of the results of these subproblems forms the basis for the comprehensive evaluation of the overall verification and validation result. Due to its modular breakdown, the modular configuration framework predefines the basic structure and possible strategies towards the methodical verification and validation of digital twins and virtual testbeds. The implemented approach was demonstrated in various cross-application examples and is the conceptual basis for the adoption to future applications.

ACKNOWLEDGEMENTS

This work is part of the project "ViTOS-II" and of the project "KImaDiZ", both supported by the German Aerospace Center (DLR) with funds of the German Federal Ministry of Economics and Technology (BMWi), support code 50 RA 1810 and support code 50 RA 1934.




REFERENCES

- [1] W. Wachenfeld and H. Winner, "The Release of Autonomous Vehicles," in *TAutonomous Driving*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 425-449.
- [2] W. L. Oberkampff and C. J. Roy, "Fundamental concepts and terminology," in *Verification and Validation in Scientific Computing*, Cambridge: Cambridge University Press, 2010, pp. 21-82.
- [3] U. Dahmen, T. Osterloh, and J. Rossmann, "Operational Modeling for Digital Twins Using the Formal Simulation Model Configuration Framework," *International journal of simulation: systems, science & technology*, vol. 20, no. 6, pp. 7.1 - 7.9, 2019, doi: 10.5013/IJSSST.a.20.06.07.
- [4] M. Grieves, "Digital twin: Manufacturing excellence through virtual factory replication," Whitepaper, 2014.
- [5] M. Shafto, M. Conroy, R. Doyle, E. Glaessgen, C. Kemp, J. LeMoigne, and L. Wang, "Draft modeling, simulation, information technology and processing roadmap," National Aeronautics and Space Administration, 2010. [Online]. Available: https://www.nasa.gov/pdf/501321main_TA11-MSITP-DRAFT-Nov2010-A1.pdf.
- [6] E. Negri, L. Fumagalli, and M. Macchi, "A review of the roles of digital twin in cps-based production systems," *Procedia Manufacturing*, vol. 11, pp. 939-948, 2017, doi: 10.1016/j.promfg.2017.07.198.
- [7] F. Tao, H. Zhang, A. Liu, and A. Y. C. Nee, "Digital twin in industry: State-of-the-art," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2405-2415, 2019, doi: 10.1109/TII.2018.2873186.
- [8] G. N. Schroeder, C. Steinmetz, C. E. Pereira, and D. B. Espindola, "Digital twin data modeling with automationml and a communication methodology for data exchange," *IFAC-PapersOnLine*, vol. 49, no. 30, pp. 12-17, 2016, doi: 10.1016/j.ifacol.2016.11.115.
- [9] T. Gabor, L. Belzner, M. Kiermeier, M. T. Beck, and A. Neitz, "A simulation-based architecture for smart cyber-physical systems," in *2016 IEEE International Conference on Autonomic Computing (ICAC)*, 2016, pp. 374-379, doi: 10.1109/ICAC.2016.29.
- [10] Federal Ministry for Economic Affairs and Energy. "Glossar industrie 4.0," 2020. [Online]. Available: <https://www.plattform-i40.de/PI40/Navigation/DE/Industrie40/Glossar/glossar.html>.
- [11] J. Ríos, J. Hernandez-Matias, M. Oliva, and F. Mas, "Product avatar as digital counterpart of a physical individual product: Literature review and implications in an aircraft," in *Advances in Transdisciplinary Engineering (ATDE)*, 2015, vol. 2, pp. 657-666.
- [12] U. Dahmen, T. Osterloh, and J. Rossmann, "Development of a modular configuration framework for digital twins in virtual testbeds," in *ASIM Workshop 2019 Simulation Technischer Systeme - Grundlagen und Methoden in Modellbildung und Simulation*, 2019, pp. 91-98.
- [13] T. Osterloh and J. Rossmann, "Versatile inverse dynamics framework for the cross application simulation of rigid body system," in *Proceedings of the 34th annual European Simulation and Modelling Conference 2020 (ESM'2020)*, 2020.
- [14] J. Panchal, O. Kelly, J. Lai, N. Mandayam, A. T. Ogielski, and R. Yates, "Wippet, a virtual testbed for parallel simulations of wireless networks," in *Proceedings. Twelfth Workshop on Parallel and Distributed Simulation PADS '98 (Cat. No.98TB100233)*, 1998, pp. 162-169, doi: 10.1109/PADS.1998.685282.
- [15] F. Tendick, M. Downes, T. Goktekin, M. C. Cavusoglu, D. Feygin, X. Wu, R. Eyal, M. Hegarty, and L. W. Way, "A virtual environment testbed for training laparoscopic surgical skills," *Presence*, vol. 9, no. 3, pp. 236-255, 2000, doi: 10.1162/105474600566772.
- [16] J. Bardina and T. Rajkumar, "Intelligent launch and range operations virtual test bed (ilro-vtb)," in *EProc. SPIE 5091, Enabling Technologies for Simulation Science VII*, 2003, pp. 141-148, doi: 10.1117/12.486335.
- [17] J. Grzywna, A. Jain, J. Plew, and M. Nechyba, "Rapid development of vision-based control for mavs through a virtual flight testbed," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, 2005, pp. 3696-3702, doi: 10.1109/ROBOT.2005.1570683.
- [18] M. Wetter and P. Haves, "A modular building controls virtual test bed for the integration of heterogeneous systems," in *Proceedings of the Third National Conference of IBPSA*, 2008, pp. 69-76.




- [19] J. Sendorek, T. Szydło, and R. Brzoza-Woch, "Software-defined virtual testbed for iot systems," *Wireless Communications and Mobile Computing*, vol. 2018, 2018, doi: 10.1155/2018/1068261.
- [20] M. Rast, *Cross-domain modeling and simulation as the basis for virtual testbeds (Domänenübergreifende Modellierung und Simulation als Grundlage für virtuelle Testbeds)*, Ph.D. Dissertation, Aachen: Apprimus, 2015.
- [21] M. Wetter, "Co-simulation of building energy and control systems with the building controls virtual test bed," *Journal of Building Performance Simulation*, vol. 4, no. 3, pp. 185-203, 2011, doi: 10.1080/19401493.2010.518631.
- [22] O. Kirovskii and V. Gorelov, "Driver assistance systems: analysis, tests and the safety case. iso 26262 and iso pas 21448," in *IOP Conference Series: Materials Science and Engineering*, vol. 534, no. 1, 2019, p. 012019, doi: 10.1088/1757-899X/534/1/012019.
- [23] J. Rossmann, M. Schluse, C. Schlette, and R. Waspe, "Control by 3d simulation—a new robotics approach to control design in automation," in *ICIRA 2012: Intelligent Robotics and Applications*, 2012, pp. 186-197.
- [24] D. Baraff, "Linear-time dynamics using lagrange multipliers," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, 1996, pp. 137-146, doi: 10.1145/237170.237226.
- [25] R. C. Gonzales and R. E. Woods, *Digital image processing*, 2nd ed. Pearson, 2002.
- [26] R. Brunelli, *Template Matching Techniques in Computer Vision: Theory and Practice*. Wiley Publishing, 2009.
- [27] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on image processing*, vol. 13, no. 4, pp. 600–612, Apr. 2004, doi: 10.1109/TIP.2003.819861.
- [28] OpenCV 2.4.13.7 - documentation, "OpenCV API Reference - imgproc. - Image Processing - matchTemplate," 2014. https://docs.opencv.org/2.4/modules/imgproc/doc/object_detection.html?#matchtemplate (accessed 2020-08-13).
- [29] Poommitol Chaicherdkiat, Tobias Osterloh, Chayakorn Netramai, Jürgen Roßmann, "Simulation-based Parameter Identification Framework for the Calibration of Rigid Body Simulation Models," in *2020 SICE International Symposium on Control Systems (SICE ISCS)*, 2020, pp. 12-19, doi: 10.23919/SICEISCS48470.2020.9083501.
- [30] "Open dynamics engine (ode) - manual," 2019. <http://ode.org/wiki/index.php?title=Manual> (accessed 2020-06-19).
- [31] R. W. Cottle, *Linear Complementarity Problem*. in *Encyclopedia of Optimization*, Boston, MA: Springer US, 2009, pp. 1873-1878.

BIOGRAPHIES OF AUTHORS






Ulrich Dahmen    received M.Eng. degree in electrical engineering with focus on information technology and environmental engineering at Hochschule Niederrhein University of Applied Sciences in 2016. Currently he is doing his PhD at the Institute for Man-Machine Interaction (MMI) at the RWTH Aachen University as a research assistant. His research focuses on simulation-based verification and validation of technical systems from the space, automotive and environmental sectors based on digital twins as virtual prototypes. He can be contacted at email: dahmen@mmi.rwth-aachen.de.



Tobias Osterloh    received M.Sc. degree in electrical engineering with focus on system technology and automation of RWTH Aachen University in 2015. Since 2016 he is research assistant with the Institute for Man-Machine interaction investigating novel methodologies for the dynamic simulation of digital twin. He coordinated the MMIs subprojects within the DLR-funded joint projects iBOSS-3 and KImaDiZ and was named group lead for aerospace projects in 2020. He can be contacted at email: osterloh@mmi.rwth-aachen.de.



Jürgen Roßmann    received the Dipl.-Ing. and Dr.-Ing. degrees in electrical engineering from the Technical University of Dortmund, Germany, in 1988 and 1993, respectively. He was the Group Head with the Institute of Robotics Research (IRF), Dortmund, Germany, and was appointed as a Visiting Professor with the University of Southern California, in 1998. He returned to IRF as the Department Head and in 2005, founded the Company EFR-Systems GmbH. Since 2006, he has been a Full Professor and the Director with the Institute for Man-Machine Interaction, RWTH Aachen University, Aachen, Germany. Also in 2006, he was appointed the Deputy Director for the DLRs Institute for Robotics and Mechatronics. Prof. Rossmann was the recipient of several national and international scientific awards. He is a member of the National Academy of Science and Engineering, Germany. He can be contacted at email: rossmann@mmi.rwth-aachen.de.