SEAMLESS INTEGRATION OF DEVICE AND FIELD DATA INTO THE SYSTEM SIMULATION OF A HYDRAULIC SERVO-PRESS USING AAS

Raphael Alt[1]*, Malte Becker[2], Roland Bublitz[4], Sebastian Heppner[3], Heiko Baum[1], Tobias Kleinert[3], Katharina Schmitz[1]

[1] *Fluidon GmbH, Jülicher Str. 338A, 52070 Aachen, Germany*
[2] *Institute for Fluid Power Drives and Systems (ifas), RWTH Aachen University, Campus-Boulevard 30, 52074 Aachen, Germany*
[3] *Chair of Information and Automation Systems for Process and Material Technology (IAT), RWTH Aachen University, Turmstr. 46, 52072 Aachen, Germany*
[4] *Parker Hannifin Manufacturing Germany GmbH & Co. KG, Gutenbergstrasse 38, 41564 Kaarst, Germany*

* Corresponding author: E-mail address: raphael.alt@rwth-aachen.de

ABSTRACT

This contribution presents a framework for improving the simulation-based development process relying on seamless data and parameter exchange between general types of components, physical components of a specific system and the respective simulation model. The proposed solution relies on the concept of the Asset Administration Shell (AAS) to leverage the availability and interoperability of the heterogenous assets and to access their proprietary properties. Therefore, three AAS-based solutions are introduced to integrate different asset kinds. They represent component type data and simulation models provided by the component supplier, simulation models of components instantiated in a local simulation environment, and real components in operation with different communication interfaces. The solutions are implemented in a framework and demonstrated successfully through different simulation-based engineering use-cases using a servo-hydraulic press as a reference system.

Keywords: Industrie 4.0, Asset Administration Shell, Digital Twin, Simulation-based Development Process, OPC UA, NFC

1. INTRODUCTION

Model-Based Systems Engineering (MBSE) is playing an increasing role in the development of complex machines and products, with ever more demanding requirements. System simulations can add significant value at all phases of a machine's product life-cycle, from concept validation and component dimensioning, function and logic control development, virtual commissioning, process optimization, condition monitoring, and remote maintenance. However, their effectiveness as a basis for decision-making relies on a sufficiently accurate representation of the system state in the simulation, which depends on the level of detail of the simulation models and the congruence between actual system and component characteristics and model parameters assumed for simulation. Since simulation models, which are usually created during the engineering phase in the beginning of a product life-cycle, typically lack a tight coupling to the actual system and its respective engineering artifacts, there is a growing discrepancy along the life-cycle. The current process of aligning relevant data and parameters of the simulation models with engineering artifacts, technical documents, and the real system in operation is conducted manually. This process requires coordination across multiple

stakeholders, is labor-intensive and prone to errors, and is therefore rarely performed. The poor linkage is caused by the high heterogeneity, non-interoperability, and lack of tool-independent Application Programming Interfaces (API) to access relevant data and information to both, components in the field and in the simulation.

Therefore, the aim of this publication is to enable interoperable and cross-manufacturer automated bi-directional exchange of simulation-relevant parameters between machine components and their associated simulation models according to the principle of digital twins. This eliminates the need for manual parameterization steps, allowing individual development steps in the development process to be flexible, error-free, and time-efficient.

After showing the specific challenges and requirements that emerge from the simulation in this context in section 3, a solution that strongly relies on the Asset Administration Shell (AAS) is presented in section 4. In this approach, the AAS integrates three different asset kinds for facilitating the standardized exchange of asset information in an open and interoperable data synchronization layer - components or models – across a value creation network over multiple phases of the product life-cycle.

After introducing relevant features of the AAS, the solution approaches for the integration of three different asset kinds are presented in section 5. They consider:

- component type data and simulation models provided by the component supplier,
- simulation models of components instantiated in a local simulation environment,
- real components in operation with different communication interfaces.

The feasibility of the implemented framework with respect to the three asset kinds is demonstrated and validated in section 6 through the realization of engineering use-cases of the drivetrain of a hydraulic servo-press provided by Parker Hannifin.

2. DIGITAL TWINS IN MODEL BASED SYSTEMS ENGINEERING

Shorter product life-cycles and increasingly sophisticated product functionalities require the development of methodologies that effectively manage the complexity of multi-disciplinary engineering. At the same time, the flexibility and scalability of the established design methods shall be maintained. MBSE serves as an established approach in this context, for continuous description and analysis of the system to be developed, by utilizing simulation models, from the early phase of conception through the entire product life-cycle. [1]

Within MBSE, simulation models play an important role in analyzing, testing, and optimizing systems in a controlled virtual environment, even before the creation of physical prototypes. This approach not only reduces the costs and time associated with the development of physical models but also enhances the precision of system behavior predictions under varying conditions and life-cycle stages [2]. To serve as a reliable foundation for decision-making in design and optimization processes, the simulation model must accurately reflect the system behavior in different operating points. Conversely, any optimization performed on the simulation model should be reflected in the regarded system. Therefore, a bidirectional data flow for the continuous updating and synchronization of both the model and the current system state is crucial. For machine design and system optimization, simulation models are often represented by lumped parameter 0D or 1D system simulations. These system simulations incorporate the individual behavior of all components which are part of the machine. For these kinds of simulations, the required data flow between the physical machine and the simulation model normally includes the exchange of model parameters as well as input and output signals.

Simulation models can be utilized for different purposes over the entire product life-cycle of a machine. To ensure alignment between the simulation model and real system, occasional synchronization of parameters is essential, especially when changes have been made to either the machine or the simulation model. **Figure 1** illustrates the application scenarios for simulation models as well as the required points for parameter updating.

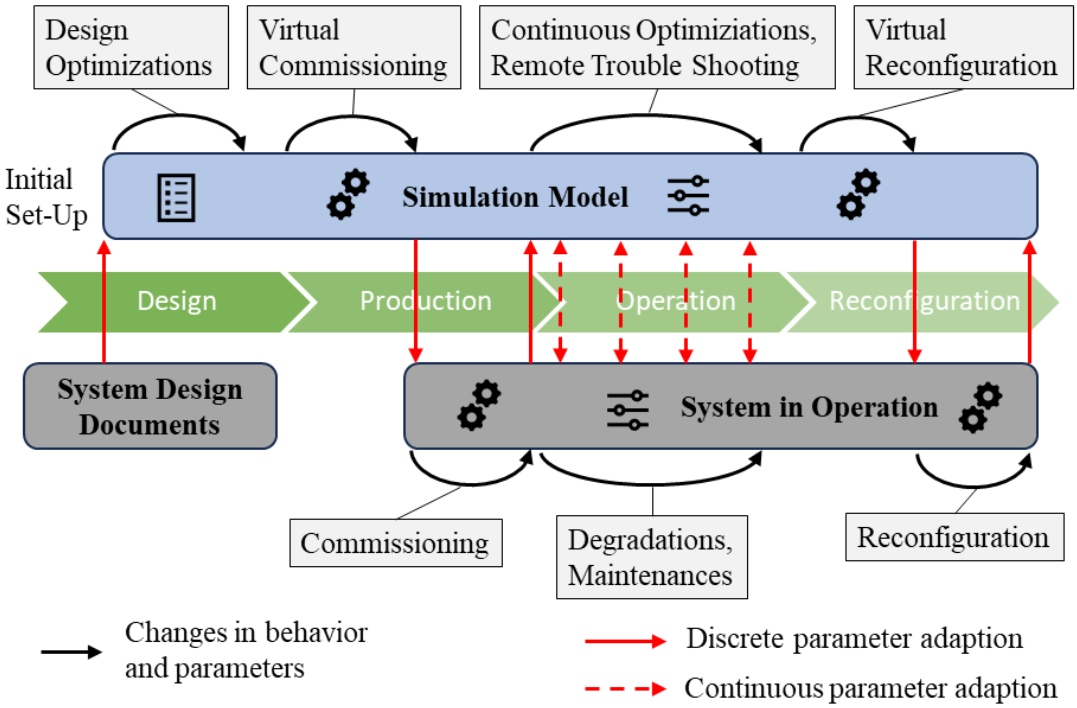

Figure 1: Changes of Machine and Simulation Model Twin across the product life-cycle

Initially, the system simulation model is set-up with a simulation environment, by linking simulation models of individual components. After general concept validation, during the dimensioning process, the simulation component parameters are adjusted to match the sizes and characteristics of the components available on the market. Afterwards, the simulation model is connected to a programmable logic controller, for control and controller design as well as virtual commissioning. The optimal parameters determined are subsequently transferred to the machine and further adjusted during the real commissioning process. During the operational phase, the simulation model is used continuously for monitoring and optimizations, e.g., to improve the energy efficiency. Concurrently, the physical system experiences continuous changes of its characteristics due to degradation, such as wear or contamination. Prior to reconfiguration of a manufacturing process, the intended changes are first prepared and tested using the simulation model, and only then implemented to the physical system, similar to the processes described during commissioning.

The frequent modifications in both, the machine and its model, throughout the product life-cycle underline the need for continuous mutual synchronization. Manual synchronization, as suggested in **figure 2** by the concept of Digital Model, represents the current state-of-the-art in the industry. Manual synchronization and adaption are labor-intensive and prone to errors, leading to outdated models that diminish in quality as a foundation for decision-making. Therefore, widespread adoption of simulation models in industry remains limited despite the many advantages they offer.

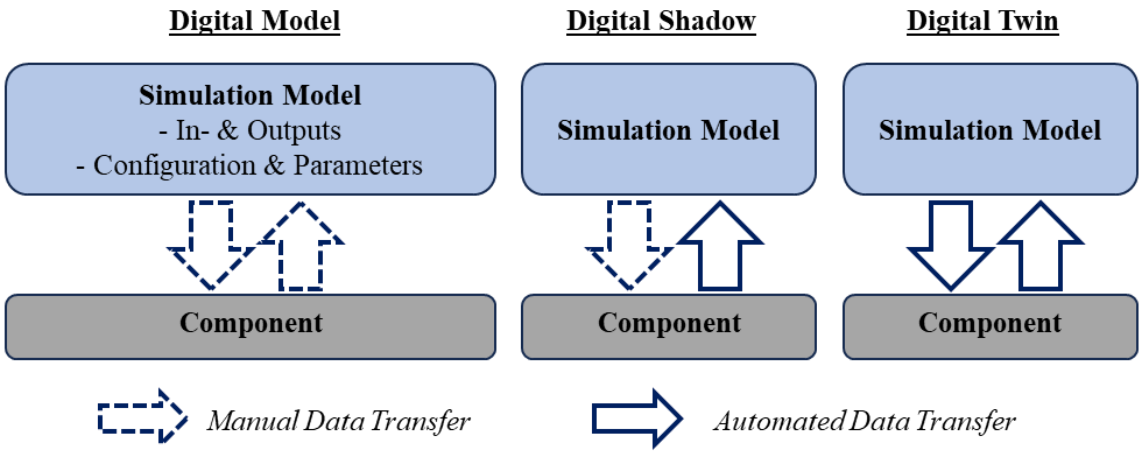

Figure 2: Digital Model, Shadow, Twin, after [3]

Given these challenges, there is a compelling need for a tightly coupled relationship between the simulation model and the system at design and in operation phases, facilitated by automated bidirectional data flow. This becomes even more important when the simulation model is integrated into real-time control loops with the corresponding real system. A digital object, which possesses this level of automated coupling with the real object, is referred to as a Digital Twin [3], see figure 2. Therefore, the implementation of Digital Twins serves to overcome these limitations to fully realize the potential of MBSE and enable efficient, effective, and economical application of digital engineering.

3. CHALLENGES AND REQUIREMENTS FOR USING SIMULATION-BASED DIGITAL TWINS IN ENGINEERING

The evolution from traditional engineering approaches to MBSE has introduced both opportunities and challenges in the engineering landscape. This transition necessitates a reevaluation of the requirements and challenges associated with engineering using Digital Twins.

MBSE offers a holistic approach to engineering, wherein all relevant requirements and aspects are described across various domains and are consistently interlinked. This allows for transparent visibility of the impact of design changes across all modeled systems and processes, enabling conflict identification and resolution. The approach also facilitates iterative design changes throughout the product life-cycle, incorporating feedback loops. Moreover, MBSE allows for the decomposition and composition of the overall system into subsystems, thereby enabling independent and parallel development as well as reusability.

This leads to the following implications and consequences during product development:

- The holistic view of MBSE requires the representation of systems from various perspectives and domains, leading to the involvement of multiple stakeholders and the utilization of different engineering tools.
- The approach considers system composition and decomposition requiring modular subsystems and components with defined system boundaries and interfaces that can separately be developed and then combined into an overall system.
- The involvement of multiple stakeholders and engineering tools results in heterogeneous and distributed data sources with proprietary data formats. For seamless integration of all partial solutions across various engineering client applications, the shared data must increase in availability, consistency, and interoperability with unambiguously defined semantics.

Given these implications, the prerequisites for digital engineering include high availability, continuity, and consistency of all relevant and interconnected data across organizational boundaries, life-cycle phases, and tools. Only on this foundation, development processes can holistically be analyzed, optimized, and ultimately automated.

From the described implications of MBSE, the requirements for simulation models following the Digital Twin approach can be derived as follows:

- Simulation models of components should accurately represent the behavior of the real components, requiring suitable modeling of the physics and accurate parameterization.
- To ensure reusability and scalability of simulation models, a modular and encapsulated design with clearly defined interfaces is required for integration. To obtain flexibility and the ability to adapt system variations, relevant parameters of the simulation model should be accessible from external.
- To facilitate integration into various applications and tools, simulation models should be interoperable with unambiguously defined properties.
- To ensure that multiple stakeholders and engineering applications obtain an equal and most recent system state, the simulation instance should be globally accessible and reflect a unique data source, adhering to the principle of single-source-of-truth.

To fully leverage the capabilities of Digital Twins, tools and applications are needed that enable seamless linking of Digital Twins with their real-world counterparts and integration into various engineering scenarios. This, in turn, imposes similar requirements concerning data availability, interoperability, unambiguity, and consistency for the component's parameters at design stage or of the physical component in operation.

4. SOLUTION APPROACH FOR SEAMLESS SIMULATION-BASED ENGINEERING

The proposed solution aims to support efficient and effective collaboration in engineering across multiple stakeholders within a value creation network. The approach addresses key aspects such as:

- openness and decentralization on both sides, data providers and consumers
- modularity and reusability through encapsulation and interoperability
- high data availability, continuity, and consistency.

The presented approach for seamless simulation-based engineering comprises two parts.

The first part of the solution focuses on creating tool-independent and modular simulation models that represent realistic machine behavior. This is achieved through encapsulated Functional Mock-up Unit (FMU) Co-Simulations. A FMU encapsulates a proprietary simulation model and provides a standardized interface that allows for the integration of simulations into various simulation environments [4]. Thereby the concept supports modularity and interoperability for heterogeneous components, which are provided by different suppliers and might also be created by different simulation tools [5].

The second part of the solution focuses on establishing a globally available data layer and providing interoperable access to relevant aspects of assets under consideration. For this, the standardized model of the Asset Administration Shell (AAS) is used. Improvements in asset data access are the foundation for seamless data exchange between components, their respective documentation and simulation models according to the principle of Digital Twin [6].

4.1 Functional Mock-up Unit Co-Simulation

The Functional Mock-up Interface (FMI) open standard maintained by the Modelica Association aims to simplify the usage and exchange of simulation models and is supported by more than 180

simulations tools [7]. The Functional Mock-up Unit (FMU) for Co-Simulation is an implementation of the FMI and enables the encapsulation of simulation models, including a suitable solver. FMU Co-Simulation provides widely applied technology suitable to enhance the versatile integration of simulation models in the MBSE context, also recommended by the prostep ivip SmartSE project [8]. It provides the following advantages in the presented solution approach:

- The FMI standard allows stakeholders to create simulation models using their preferred tools, while enabling the exchange and integration of these models into systems simulation. Provided the required licenses are available, FMUs remain tool-independent regarding their deployment. This enhances the portability and reusability of models and effectively reduces vendor dependency.
- FMUs enable the encapsulation of components, ranging from standard simulation models, such as hydraulic valves from standard component libraries, to individually developed models incorporating intelligent functions or logic control behavior, such as a motion controller. This enables precisely reflecting the component behavior designed by the component supplier and enables, as black-box entities, to share their specialized knowledge securely.
- FMU Co-Simulations can be integrated into various simulation environments and heterogenous system simulations with minimal configuration effort with respect to model and solver parameterization. This characteristic facilitates a plug-and-play methodology, simplifying the simulation process and boosting its effectiveness and user-friendliness.

Figure 3 shows relevant aspects for the presented approach of FMU simulation model integration and interaction. An FMU for Co-Simulation consists of one zip-file with the extension ".fmu" containing an XML-file with the definition of all variables and parameters that are exposed to the simulation environment as well as a set of C-functions and dynamic libraries provided in either source or binary form, used to initialize with defined parameters and run the simulation [9]. The integration, initialization, and orchestration during system simulation are carried out by the simulation tool. During the integration of each FMU, a Simulation Component is created as a tool-specific wrapper class to bind the FMU to the simulation tool. Upon instantiation, each Simulation Component can uniquely be addressed by a tool-specific identifier (ID). Furthermore, parameters described in the modelDescription.xml can be accessed by the simulation tool and, if provided, via a tool-specific API.

Notably, to introduce a clear distinction, a simulation model file is further defined as **Simulation Model**, whereas in contrast a **Simulation Component** defines the Simulation Model instantiated in a runtime environment. For more details on the coupled Co-Simulation in this context, see [5].

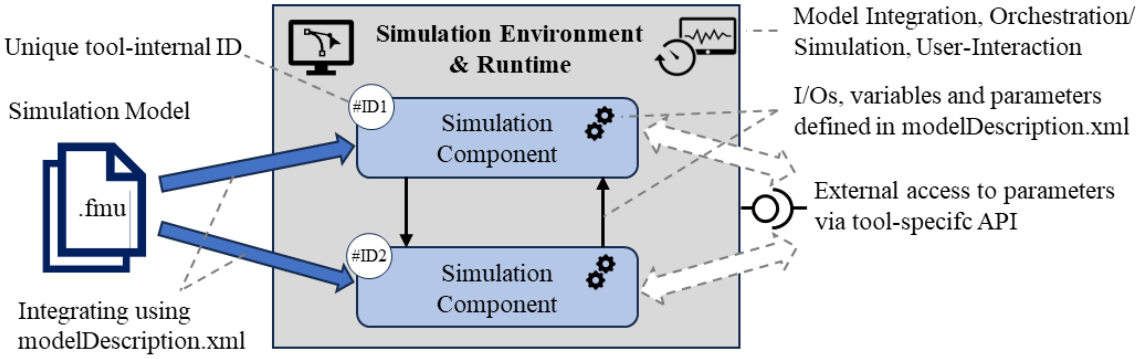

Figure 3: Instantiated Simulation Model and access to its parameters

4.2 Asset Administration Shell

To support MBSE across heterogeneous assets and engineering tools involving multiple stakeholders throughout the entire life-cycle, an open, decentralized and therefore integrative approach for exchanging of information is required. An Asset is any element that represents a perceived or actual value for an organization [6]. In the context of this publication, physical components are considered assets, e.g. a hydraulic cylinder, and its related engineering documents, data, and artifacts, such as simulation models. The presented architecture is based on the concept of the Asset Administration Shell (AAS). The AAS has a technology-neutral meta-model for implementing Digital Twins, as specified by the Industrial Digital Twin Association [6]. One relevant objective of the AAS focuses on enhancing interoperability and availability of properties and services of any asset within value creation networks. Aspects of the AAS relevant to the solution described in this publication are depicted in **figure 4** and briefly described below. The full specification can be obtained from [6, 10, 11].

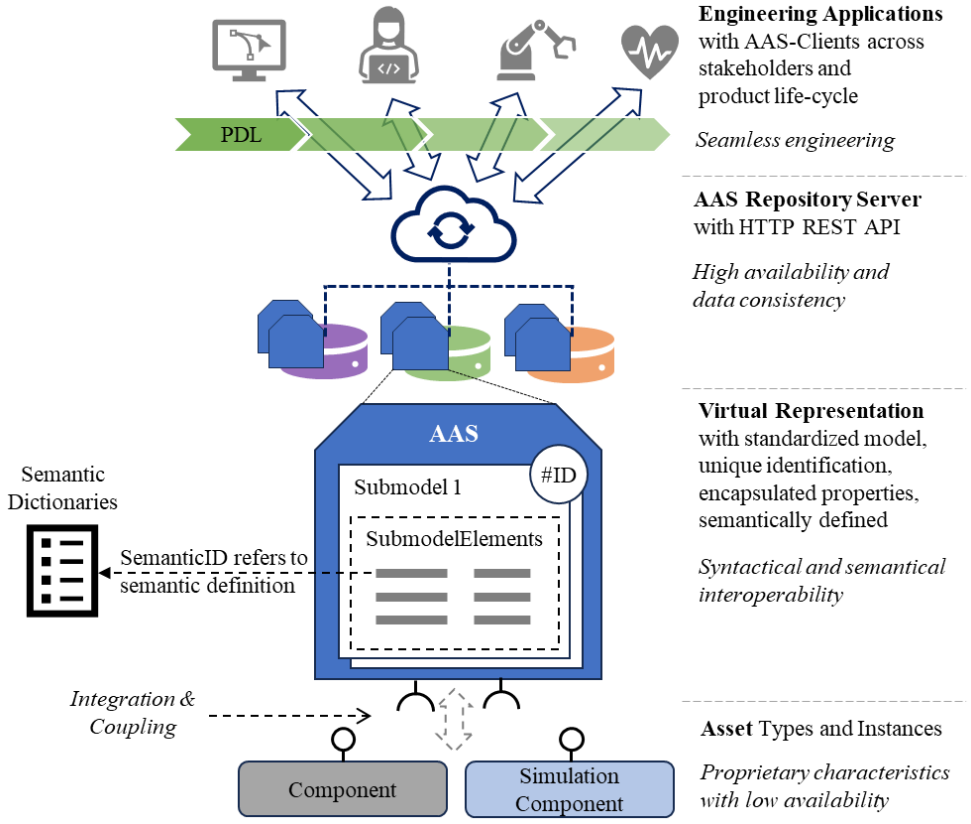

Figure 4: AAS and AAS Repositories for leveraging availability and interoperability of proprietary asset characteristics

An AAS is a virtual representation of an asset, while both are globally uniquely identifiable and referenced to each other. Assets can be of two kinds, Type and Instance. The Asset Type is a generalized classification that encapsulates common attributes and data structures of many Asset Instances, such as general component documentation or simulation model files. The Asset Instance, on the other hand, refers to a unique or individually existing element, such as a manufactured component or an instantiated simulation model. Typically, the AAS of an Asset Instance (AAS Instance) is derived from the AAS of an Asset Type (AAS Type) at a particular point in the product life-cycle to reflect the information relevant to the individual Asset under consideration. To illustrate this difference, an Asset Type could describe a component as found in a catalogue, whereas an Asset Instance would be a concrete component of that type to be shipped to a customer. [6, 12]

In an AAS, the characteristics of an Asset are modeled as SubmodelElements, encapsulating, for example, parameters or files. SubmodelElements are thematically grouped into SubmodelElementCollections and Submodels. Submodels can subsequently be standardized as templates to accommodate a set of relevant properties required for specific use-cases, such as the "Provision of Simulation Models" [13]. Both Submodels and SubmodelElements can be annotated with SemanticIDs. These unique identifiers that refer to semantic descriptions in semantic dictionaries, like eClass [14] or IEC CDD [15], describing the semantics of the content of the annotated element. This allows, in theory, the provision of the Asset's proprietary characteristics not only syntactically but also semantically interoperable.

To further enhance the availability across the entire value chain, including the product life-cycle and value networks, AASs are made available in AAS Repositories hosted on web servers. Using the specified HTTP REST API [10], and assuming the necessary access rights are granted, AASs can be accessed globally via the Internet by multiple AAS clients as part of engineering applications. This concept realizes an open and integrative approach that allows decentralized creation, provisioning, querying, and modification of AASs, to have further access to the respective asset's properties. For more details on implementing the AAS Data Layer through the AAS Server-Client Architecture, readers are referred to [12].

5. AAS-BASED ARCHITECTURE FOR SEAMLESS INTEGRATION OF SIMULATION MODELS IN MBSE

The seamless integration of simulation models into the engineering process and their close coupling with real-world assets according to the Digital Twin approach requires the implementation of three critical interfaces. These interfaces, shown in **figure 5**, are realized through the concept of the AAS and described in the following:

1. Interface for the selecting component types and retrieving of component data and engineering files, such as simulation models provided by the component suppliers.
2. Interface for bi-directional interaction with instantiated simulation models for a specific system.
3. Interface for bi-directional interaction with engineering artifacts and data created for a specific system and with actual components during operation.

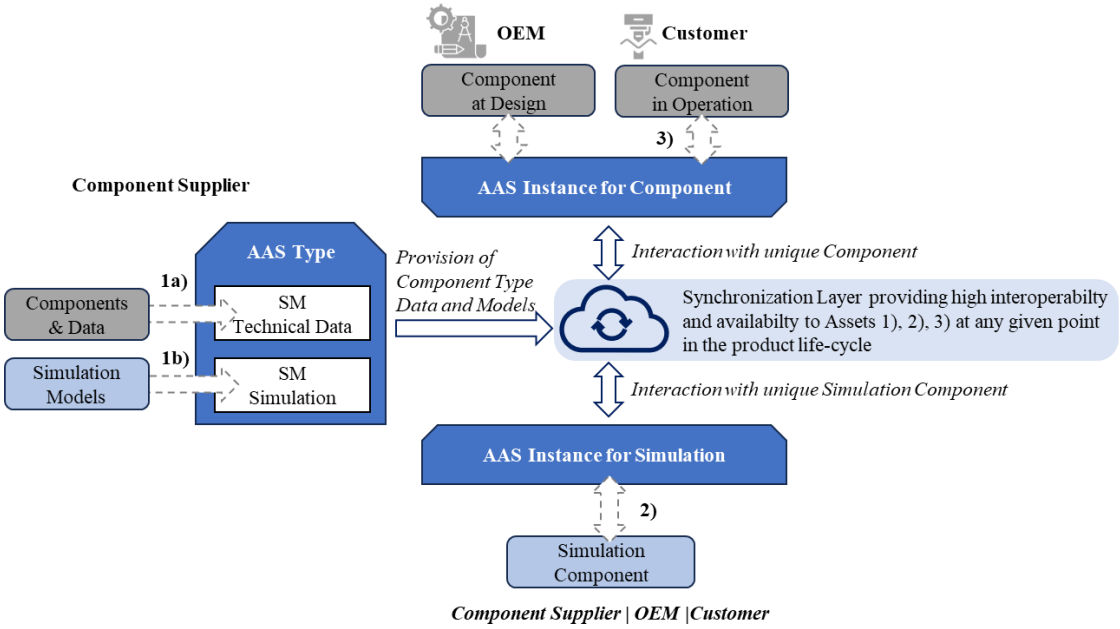

Figure 5: AAS-based Digital Twin synchronization interfaces for MBSE

1) AAS Type for providing component data and simulation model files

Component manufacturers hold valuable data and engineering files for their components, which can significantly support Original Equipment Manufacturers (OEM) in their overall system engineering processes. To facilitate interoperable data sharing for the targeted engineering discipline, the component manufacturer provides its proprietary component data, originating from its database, via AAS Types encapsulated in semantically annotated properties. The manufacturer is responsible for the initial linking of its proprietary data and the properties within the AASs. It is reasonable that due to a significant quantity and diverse range of component configurations, not all AAS Types are pre-generated and stored. Instead, they are generated on-access when required and subsequently made available via the defined AAS REST API from the server.

In the context of this paper, two submodels are particularly relevant and described briefly:

a) The Submodel Generic Frame for Technical Data for Industrial Equipment in Manufacturing [16] (SM Technical Data) specifies both the class of the component and its relevant technical parameters. This information can be used for searching components based on defined specifications, automated sizing processes, or for parameterizing simulation models.

b) The Submodel Provision of Simulation Models [17] (SM Simulation) provides essential information for selecting the provided Simulation Models appropriate to the engineering use-cases considered. Therefore, attributes for defining the considered engineering domain and the simulation purpose, such as sizing, virtual commissioning, or condition monitoring, are clearly defined. Moreover, the Submodel specifies properties for integrating the simulation model file into a simulation environment, such as the file format, the operating system requirements, and recommended solver settings. Within the scope of this publication, encapsulated simulation models are provided as tool-independent FMU Co-Simulation for lumped parameter system simulation (refer to chapter 4). More details on the implementation can be found in [12].

2) AAS Instance for providing access to the Simulation Component

To realize interoperable accessibility and high availability to simulation model files instantiated in local simulation tool environments – the Simulation Components – an AAS Instance is created for each of them to encapsulate relevant parameters. The AAS Instances are then made accessible in an AAS Repository via AAS HTTP REST API.

Figure 6 depicts the proposed modeling concept to provide relevant information for allowing simulation tools to link the Simulation Component and encapsulate its parameters in the AAS. It currently represents a non-standardized extension of the existing SM Simulation specification.

A link between the AAS Instance and the Simulation Component is established by referring to its unique ID in the Reference *LinkedSimulationComponent*. The ID is generated and linked during the instantiation of the Simulation Model. The *Parameter* SECs provide information to the Simulation Component parameters, intended to access from external clients. The Property ID defines the specific parameter identifier. In case of FMUs, the parameter identifiers are defined in the modelDescription.xml. The Reference SemanticReference, refers to a semantic definition via *SemanticID* to ensure an unambiguous definition of the parameter, aiding in identifying and matching corresponding properties in other submodels. Finally, the Property *Value* reflects the current parameter value of the Simulation Component.

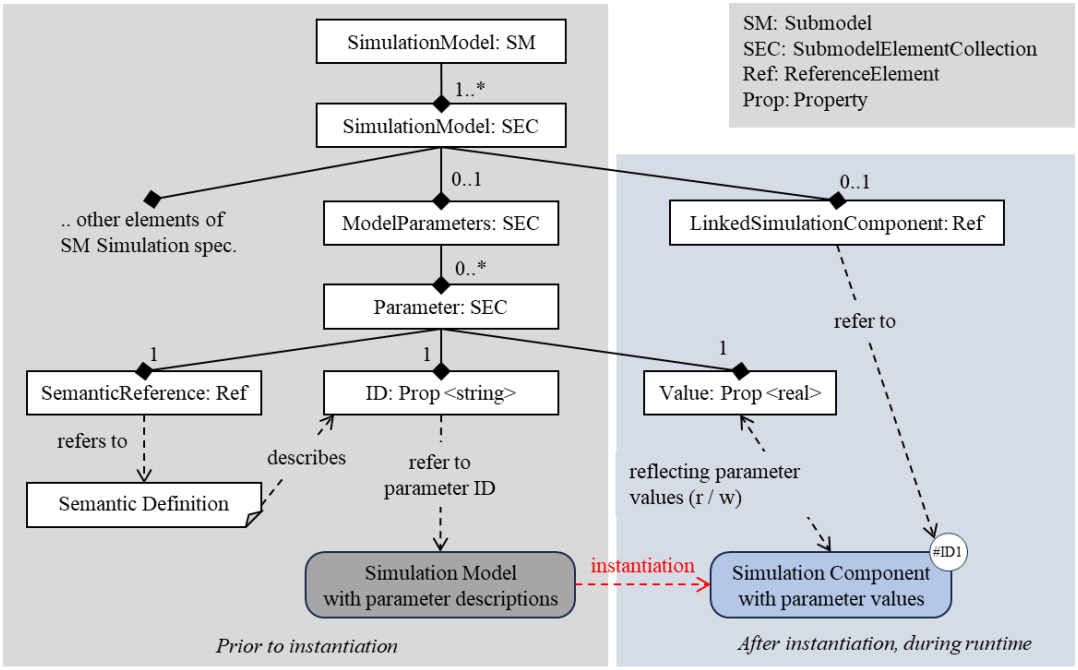

Figure 6: Modelling the coupling between AAS Instance and Simulation Component

Both the ID and the corresponding *SemanticReference* of the Simulation Model parameters are initially modeled in the SM Simulation of the AAS Type by the provider of the Simulation Model. The simulation tool realizes the instantiation of the Simulation Model and provides the corresponding AAS Instance with read and write access to the Simulation Component parameters via the AAS. In this way, proprietary Simulation Component parameters become globally and interoperable accessible with defined semantics for numerous external engineering client applications.

Although the realization described in this publication is implemented for a specific simulation tool using FMU simulation models, the concept is not limited to those and can similarly be applied to others.

3) AAS Instance for providing access to the component data

This interface provides interoperable access to data and parameters of real machine components and systems starting from the initial commissioning, during operation, including reconfigurations. Analogous to the AAS Interface of the Simulation Component, a dedicated AAS Instance is linked to its respective component in order to encapsulate component data and configuration parameters. If no AAS Instance already exists from the previous engineering phase that can be linked to the component instance, a new AAS Instance is created from the AAS Type and made available in an AAS Repository.

As machines consist of many heterogeneous components, different interfaces are used to communicate with the components themselves. **Figure 7** illustrates the coupling between the AAS and components in operation based on different communication interfaces. The controller of the drive system considered in this publication uses Ethernet-based OPC UA communication, which is widely used in the manufacturing industry. All relevant parameters of the controller can be accessed through the OPC UA interface of the device. To link the OPC UA device variables to the AAS, an OPC UA-AAS-gateway is employed. The mapping between proprietary device parameters of the OPC UA namespace and the properties in the AAS Submodels is provided by the component manufacturer. Provided that OPC UA access rights are available for operation, the controller's OPC UA and AAS Instance endpoints are the only specifications required for the linkage configuration. Should parameters require cyclic transfer, the desired timing must be specified.

Other bus systems, e.g., EtherCAT, EtherNet/IP, ProfiNet,and IO-Link can be connected similarly via specific gateways.

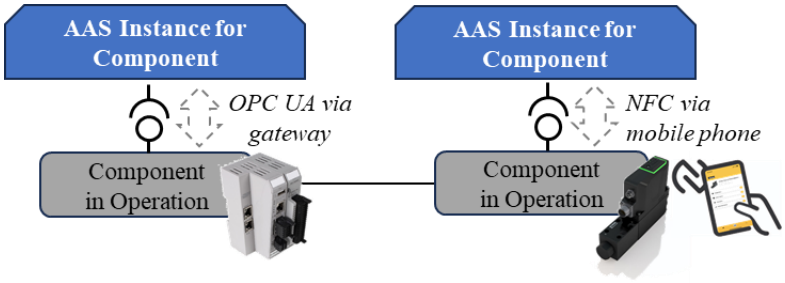

Figure 7: AAS parameter interfaces of the components in operation,
left: continuous transfer via OPA UA and gateway,
right: intermitted transfer via NFC and mobile app

However, not every parameterizable component is permanently accessible via a digital communication interface. Often, only a temporary wired connection to a PC is used for parameterization, for example, during commissioning. Therefore, wireless alternatives such as Bluetooth and NFC are also being used more frequently to enable and facilitate data transfer at certain points.

The valve considered in the use-cases of this publication provides an NFC interface. It can be used via a mobile device to extract all user parameters and additional operating and diagnostic data. The data obtained is accessible via an app on the mobile device and then synchronized to the AAS Instance in the AAS Repository. This temporary connection via mobile phone and NFC bypasses access to machine-critical real-time communication.

In the presented application, only data for device identification (asset ID and article number), the device parameters and condition monitoring information for future analysis of the system are exchanged. Generally, this approach enables not only the adjustment of control parameters, but also the full configuration of the controller, encompassing the connected valves and sensors. Direct manual entry on the machine is henceforth unnecessary.

Like the AAS Instance of the Simulation Component, proprietary and only locally available parameters of the physical component and its configuration become globally and interoperable accessible with defined semantics for numerous external engineering applications.

6. PROOF-OF-CONCEPT AND USE-CASES VALIDATION

The presented concepts are implemented as partial solutions and combined in a framework using the Eclipse BaSyx Python SDK [18]. In the following, common scenarios from different phases of the engineering process are demonstrated using a servo-hydraulic press. The framework's functionality is briefly explained, and benefits are illustrated in different scenarios.

The servo-hydraulic press consists of a drive-controlled pump, four parallel positioning axes, a drawing cushion axis, and a press cylinder. In addition, the press is controlled by a PAC 120/PACHC motion controller [19, 20]. For the proof-of-concept, only one positioning axis of the servo-hydraulic press is considered and set up in the simulation environment DSHplus [21]. The system simulation model is built as FMU Co-Simulation, in which relevant system components are modeled as individual FMUs and then connected at their hydraulic volume node signal interfaces.

6.1 Sizing and function validation

During sizing and function validation, the selection of the components and the initial parameterization in the system simulation is usually done manually by the engineer searching suitable components and their respective parameters from different data sources, e.g., websites and data sheets. This process is error-prone and requires a significant amount of time. To support the selection of suitable components, the framework allows searching among AAS Types in AAS Repositories offered by component manufacturers that match the desired requirements. From the selected AASs, the framework supports the initial creation of a Simulation Component by instantiating a provided Simulation Model as well as transferring parameters to an existing Simulation Component. This leads to a faster and error-free set up of a system simulation model, that closely matches the real properties of the system components. If necessary, the user can carry out simulative investigations and repeat the process with other component parameters. More details on the implementation are provided in [5, 12]. In this scenario, the interfaces 1. and 2. described in chapter 5 are used.

Figure 8 shows the general features of the framework for the component selection and model parameterization. In the regarded use-case, the engineer is looking for a valve that enables the hydraulic press to meet the requirements and full functionality. The system simulation structure (figure 8-1) is derived from the hydraulic scheme. However, in the beginning, the components in the simulation do not include a Simulation Model (figure 8-2a). From basic calculations, the valve dimensions are estimated by the engineer to be in a range between 10 to 40 bar nominal pressure difference and with a nominal flow rate between 50 and 100 l/min. In order to obtain a selection of suitable components, the framework offers the possibility to query an AAS Repository of the component manufacturer using search criteria defined by the user. Search criteria can be added in the query definition section (figure 8-3) via the specification of the semanticID of the regarded property with respective value constraints (min, max, equal). For better usability, separated drop-down lists and checkboxes are provided, such as for defining a specific component type or for targeting AASs that also provide a Simulation Model. However, in both cases, the query criteria are derived similarly and added to the others. Notably, as described in chapter 5, the component type is specified in the SM Technical Data of the AASs and defined via eClass IRDI. The availability of an FMU simulation model can similarly be determined from the SM Simulation.

The AAS Repository section (figure 8-4) lists AASs that comply with the combined query criteria (figure8-4a), and the engineer can select from these for further model initialization via “Initialize Simulation Component”. As described in chapter 5-2, if the selected AAS provides a specific FMU valve Simulation Model, it is integrated and instantiated by the tool to become a Simulation Component. Otherwise, an AAS with a default valve Simulation Model from the simulation tool provider AAS Repository is considered in the same way to obtain a Simulation Component (figure 8-2b). During the integration process, the respective AAS Instance is generated and coupled with the Simulation Component via their unique IDs (figure 8-4b). This AAS Instance serves as the representation of the Simulation Component in the synchronization layer for providing interoperable access to external applications. Based on the semantic annotation of the simulation parameters in the AAS Instance, the Simulation Component can further be re-parameterized via “Sync Parameters from AAS” by selecting other AASs with or without SM Simulation. This process can arbitrarily be repeated until the desired system behavior is archived. In the presented example, the AAS with a default valve Simulation Model was initialized with default parameters of 45 l/min @ 35 bars, then re-parameterized with the AAS of a valve with 50 l/min @ 35 bars. Since the semanticID of the parameter property includes value unit definition, automated unit conversion can also be included during the parameter transfer in both directions. Here, l/min and bar were converted to m^3/s and Pa, as required by the Simulation Component in the simulation tool.

This approach allows both to integrate specific and encapsulated component specific Simulation Models, as well as for considering AASs in the simulation that only contain technical parameters without any Simulation Model. This leaves great flexibility and lowers the entrance barrier of component manufacturers that can only provide very basic AASs of their components. In any case, during the creation of the Simulation Component, the corresponding AAS Instance is generated and linked, providing an interoperable interface for bi-directional parameter transfer.

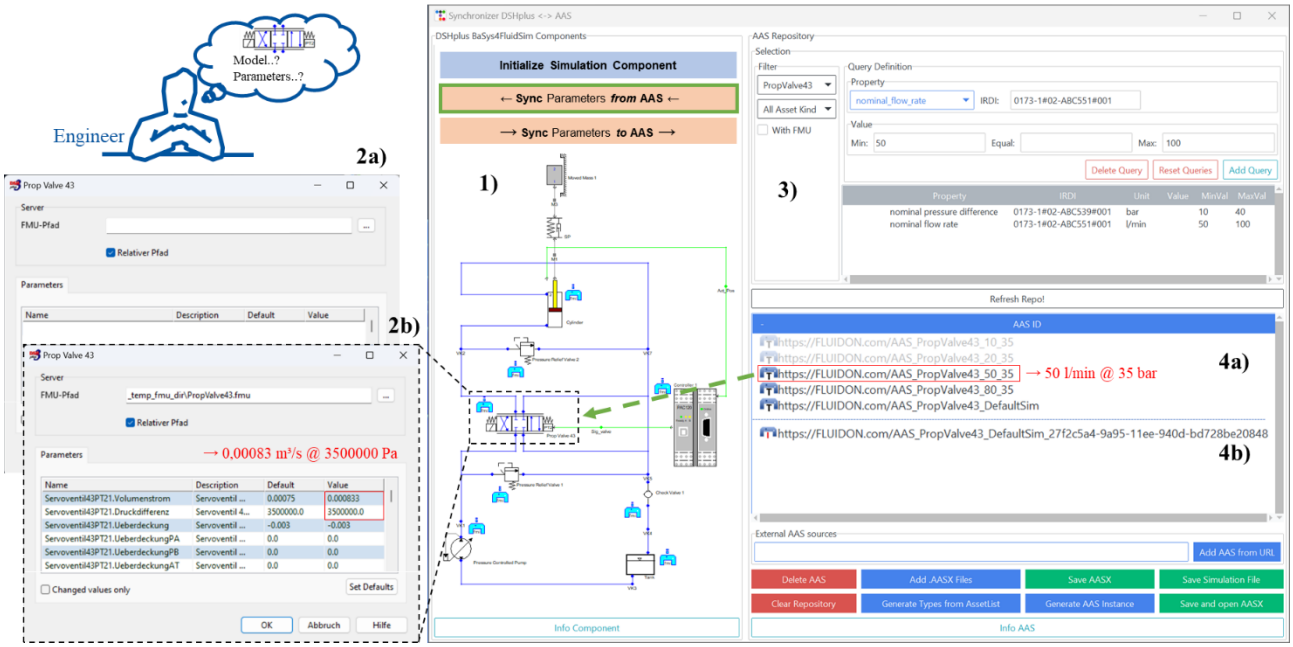

Figure 8: Implemented framework during selection and parameterization

6.2 Commissioning and operation phases

In the virtual commissioning scenario, the commissioning steps are carried out virtually in the simulation environment before the actual assembly of the physical machine. This enables potential errors to be detected and eliminated at an early stage, as well as improves and speeds up the commissioning process of the real machine. Typical steps include, for example, determining control parameters, estimating capacitive lines between components, and analyzing the interaction of subsystems with and without the control hardware. These tests and optimizations can be performed using simulation, for example, via Software-in-the-Loop and Hardware-in-the-Loop. Therefore, it is crucial that the system simulation behaves closely to the real system, for which a solution was presented in the previous chapter. When the machine is set up and put into operation during commissioning, all the parameters that were optimized in the simulation should be transferred to the components of the physical machine. The interfaces 2. and 3. described in chapter 5 are used in the following scenarios.

In the example shown in **figure 9**, the user optimized the parameters in the Simulation Components of the controller and the valve during virtual commissioning (figure 9-1). After the user has selected the Simulation Component and the targeted AAS Instance of the real PAC120 controller by pressing “Sync Parameters to AAS”, the parameters are transferred automatically in three combined steps. First, simulation parameters are updated in the SM Simulation of the AAS Instance of the respective Simulation Component (AAS Instance is not shown in the figure). The parameter values are then transferred, using the SemanticReferences of the Parameters, to the semantically equivalent properties found in various Submodels of the AAS Instance of the PAC120 controller (figure 9-2), which the Parker AAS Repository provides. From there, the parameters are updated in the PAC120 controller via the OPC UA-AAS-gateway (figure 9-3). If required, the operator can now access and further

optimize the parameters via HMI. Similarly, the valve specific settings and parameters are transferred from the simulation to the real component, with the exception that instead of OPC UA, the mobile phone application with NFC interface is used by the operator.

As commissioning is often iterative, parameters that were changed during commissioning should similarly be transferred back from the real component into the simulation environment. According to the principle of Digital Twin, the mechanism presented for transferring component parameters via AAS interfaces is bi-directional, allowing both the component and the simulation model values to be updated automatically when changes are made to either. However, when new parameter values are updated to the physical components, it must be ensured that the machine is in a safe state. Therefore, exchange is initiated manually by the operator in the HMI.

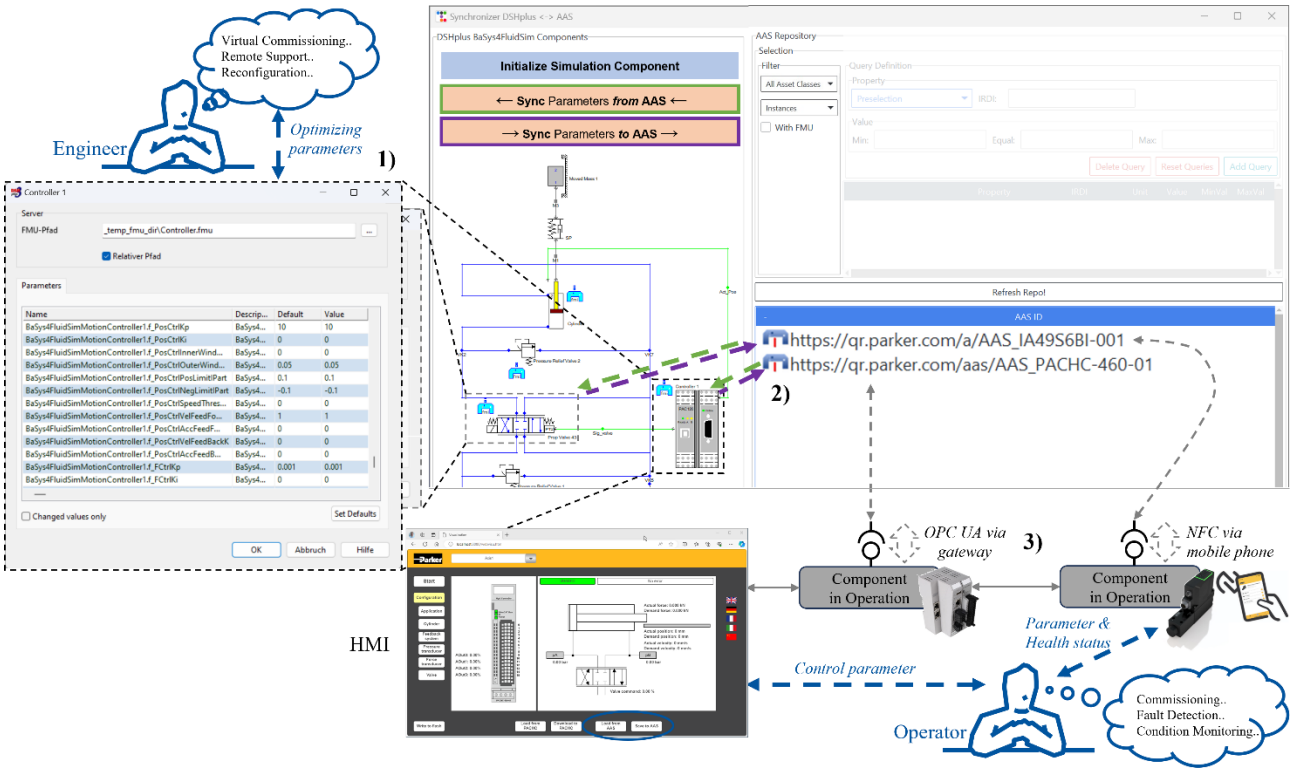

Figure 9: Bi-directional parameter transfer to the field devices for operational phases

In order to always have a profound simulative decision-making basis, and efficiently propagate found optimizations back to the machine, the presented mechanism for bi-directional parameter synchronization between simulation and field is not only important during commissioning, but also crucial to other use-cases among the entire operational phase.

In the case of unexpected irregularities or machine degradation during operation, solutions can be found more effectively and efficiently enabled by the transparency of simulative investigations. For example, when friction in the cylinder of the servo-press changes, the controller parameters need to be re-parameterized to compensate for the changes. Similarly, this applies to scheduled changes and feasibility tests, for example, to answer whether the servo-press used in operation can perform a modified pressing task. In all cases, the solution also improves the possibility of simulation-based remote support by another engineer or even the OEM, provided access rights are granted.

7. CONCLUSION AND OUTLOOK

This publication presents a framework that enables seamless simulation-based engineering across the entire life-cycle and with different stakeholders. Therefore, relevant requirements emerge in Model-Based System Engineering to better handle complex, modular, and multi-domain product development. The identified requirements aim to improve the availability, consistency, and interoperability of the assets involved, aiming to efficiently and seamlessly link all the steps and tools involved in engineering.

For this purpose, an open, decentralized approach based on an interoperable layer for data exchange is established in the framework by considering three relevant interfaces. The interfaces are realized based on the Asset Administration Shell (AAS) concept. The AAS accepts heterogeneous assets and provides interoperable access and better availability to component type data and simulation models of the suppliers, to simulation models instantiated in a simulation environment, and to the components operated in the field with various communication interfaces. The resulting data layer is the basis for enabling simulation-based engineering according to the Digital Twin concept, with automated bi-directional data exchange for the synchronization of system states between simulation models and real components, which is required in numerous use-cases with different engineering client applications across engineering and operation phases.

The concept was successfully implemented in a framework in Python. Thereby, different kinds of components were integrated through the AAS. Amongst these are the component database from Parker, the instantiated simulation models in a simulation environment, and two field devices from Parker using OPC UA and NFC. The basic feasibility was demonstrated by realizing different use-case scenarios on a servo-hydraulic reference system from Parker and a system simulation in DHSplus. The framework supports use-cases such as the initial dimensioning and function validation of the components in the machine, the commissioning process and further optimization measures during operation. It is concluded that the proposed and implemented AAS-based solution simplifies and optimizes the simulation model-based product development for OEMs, enables more reliable simulation-based decision basis in operation for the operator, and offers new business models for component manufacturers by providing easily accessible information.

For widespread use in practice, it must be clarified to what extent and which parameters may be written into an operational machine from the outside. Therefore, role concepts must be developed to solve these safety-relevant issues.

Besides, the concept presented is currently based on an existing simulation structure. In the future, it should be possible to derive and generate the hydraulic simulation structure automatically from the existing machine topology. Furthermore, the approach presented here mainly considers static parameters, whereas access to dynamically changing I/O signals via the AAS would also be relevant for some use cases. Based on this, stand-alone Simulation Components with an independent runtime environment and communication interface can also be realized for more flexibility and tool independence.

8. ACKNOWLEDGMENTS

The content of this paper is created during the research project BaSys4FluidSim under the reference 01IS21071A. The project is funded by the Federal Ministry of Education and Research (BMBF) and coordinated by the German Aerospace Center (DLR). The responsibility for the content of this paper lies with the authors. The authors would like to thank all project participants for their support.

REFERENCES

[1] Gausemeier, J., Dumitrescu, R., Echterfeld, J., Pfänder, T., Steffen, D., Thielemann, F „Innovationen für die Märkte von Morgen. Strategische Planung von Produkten“, Dienstleistungen und Geschäftsmodellen. Carl Hanser Verlag, München, 2019.

[2] K. Henderson and A. Salado “Value and benefits of Model-Based Systems Engineering (MBSE): Evidence from the literature,” Systems Engineering, vol. 24, no. 1, 2021, doi: 10.1002/sys.21566

[3] Fuller, Aidan et al. “Digital Twin: Enabling Technologies, Challenges and Open Research.” IEEE Access 8 (2019): 108952-108971

[4] J., Andreas, et al. “The Functional Mock-up Interface 3.0 - New Features Enabling New Applications.” Proceedings of the 14th International Modelica Conference September 20-24, 2021, Linköping, Sweden (2021), doi: 10.3384/ecp2118117.

[5] M. Becker, S. Heppner, R. Alt, T. Kleinert, und K. Schmitz „IMPROVING FLUID POWER SYSTEM SIMULATION THROUGH AN AAS-BASED SIMULATION FRAMEWORK”, in Proceedings of the ASME/BATH 2023 Symposium on Fluid Power and Motion Control FPMC2023, October 16-18, 2023, Sarasota, Florida, USA

[6] IDTA „Asset Administration Shell Specification - Part 1: Metamodel Schema“, March 2023.

[7] Modelica Association “Functional Mock-up Interface”, https://fmi-standard.org/, visited: 23.10.2023

[8] prostep ivip “Smart Systems Engineering – Collaborative Simulation-Based Engineering Version 3.0, Part E – FMI Best-Practice Guide”, https://www.prostep.org/, visited: 01.12.2023

[9] T. Blochwitz, et al. “Functional Mockup Interface 2.0: The Standard for Tool independent Exchange of Simulation Models”, Proceedings of the 9th International Modelica Conference, September 3-5, 2012, Munich, Germany, DOI 10.3384/ecp12076173

[10] IDTA „Asset Administration Shell Specification - Part 2: Application Programming Interfaces“, April 2023.

[11] IDTA „AAS Specifications“, https://industrialdigitaltwin.org/content-hub/aasspecifications, visited 01.10.2023

[12] S. Heppner, T. Miny, T. Kleinert, M. Becker, K. Schmitz, und R. Alt „Asset Administration Shells as Data Layer for Enabling Automated Simulation-based Engineering “, in 2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA), Sep. 2023, doi: 10.1109/ETFA54631.2023.10275474.

[14] eClass e.V. “eClass standard”, https://eclass.eu/, visited: 23.04.2023

[15] International Electrotechnical Commission “IEC 61360-4 - IEC/SC 3D - Common Data Dictionary”, https://cdd.iec.ch/cdd/iec61360/iec61360.nsf/TreeFrameset?OpenFrameSet, visited: 24.06.2023

[16] IDTA “IDTA 02003-1-2 Generic Frame for Technical Data for Industrial Equipment in Manufacturing”, August 2022

[17] IDTA “IDTA 02005-1-0 Provision of Simulation Models”, Specification, December 2022

[18] Chair of Information and Automation Systems for Process and Material Technology “basyx-python-sdk”, https://github.com/rwth-iat/basyx-python-sdk, visited 15.12.2023

[19] Parker Automation Controller Series PAC120 Operation Manual, Bulletin MSG11-5715-719/UK, 2021-12-06

[20] Electrohydraulic Control Module Series PACHC Operation Manual Bulletin MSG11-5715-720/UK, 23-02-17

[21] Fluidon GmbH “Simulation, Analyse und Optimierung fluidtechnischer Systeme“, https://www.fluidon.com/dshplus, visited: 12.12.2023