Experimentable Digital Twins—Streamlining Simulation-Based Systems Engineering for Industry 4.0

Michael Schluse, Marc Priggemeyer, Linus Atorf, and Juergen Rossmann

***Abstract*—Digital twins represent real objects or subjects with their data, functions, and communication capabilities in the digital world. As nodes within the internet of things, they enable networking and thus the automation of complex value-added chains. The application of simulation techniques brings digital twins to life and makes them experimentable; digital twins become experimentable digital twins (EDTs). Initially, these EDTs communicate with each other purely in the virtual world. The resulting networks of interacting EDTs model different application scenarios and are simulated in virtual testbeds, providing new foundations for comprehensive simulation-based systems engineering. Its focus is on EDTs, which become more detailed with every single application. Thus, complete digital representations of the respective real assets and their behaviors are created successively. The networking of EDTs with real assets leads to hybrid application scenarios in which EDTs are used in combination with real hardware, thus realizing complex control algorithms, innovative user interfaces, or mental models for intelligent systems.**

***Index Terms*—eRobotics, experimentable digital twin (EDT), intelligent systems, simulation-based systems engineering, simulation-based X, virtual testbed (VTB).**

I. Introduction

IN THE Industry 4.0 era, physical and virtual worlds are growing together. Cyber physical systems realize smart systems being connected via the internet of things and services. This results in new requirements and application areas for simulation technology.

At first, smart systems today are "systems of systems." Their components are smart systems on their own, the resulting overall functionality is the result of various systems working together in a proper way. When developing such systems, only *simulations on system level*, which are still detailed on component level, are able to gain the insights necessary to, e.g., analyze, optimize, verify, and validate those systems. At the same time, those *simulations have to incorporate various disciplines and engineering aspects*. Fig. 1 illustrates this using a "ReconCell," a reconfigurable assembly work cell which will be used as a use case to introduce the methodology presented later in this paper. To be able to analyze such a ReconCell in a detailed way, simulations have to incorporate, e.g., kinematics, rigid body dynamics, sensors, user interfaces, controllers, environment, communication, human workers, as well as superordinated manufacturing execution systems (MES).

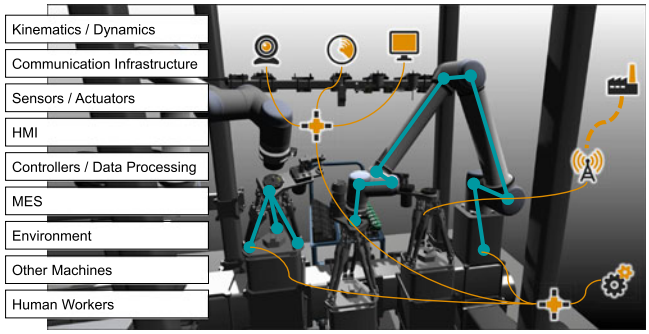

Fig. 1. Example for a "system of systems" in the field of production technology, the ReconCell [1]. Various aspects have to be investigated and integrated in comprehensive simulations.

Second, smart systems need simulation technology to implement their functionality. The realization of the various Self-X (description, networking, commissioning, diagnosis, optimization, ...) concepts require detailed digital models of the corresponding real systems which must be able to be simulated, and therefore, to become experimentable in various ways. In Industry 4.0, simulations are not only an engineering tool for development. They are now used inside the physical systems to realize intelligent systems, intuitive user interfaces, training simulators, etc. For this, *appropriate structures and processes to consistently use simulations in varying application scenarios throughout the life-cycle of those systems* (see Fig. 2) based on a single set of simulation models (which become more and more detailed over time) are needed.

Third, the development of smart systems changes dramatically. Systems become more complex and must be developed by interdisciplinary and distributed development teams in shorter

Manuscript received June 4, 2017; revised September 4, 2017, January 25, 2018, and January 31, 2018; accepted February 1, 2018. Date of publication February 12, 2018; date of current version April 3, 2018. The ReconCell project was supported by the European Union's Horizon 2020 Research and Innovation Program under Grant 680431. Paper no. TII-17-1204. *(Corresponding author: Michael Schluse.)*

The authors are with the Institute for Man-Machine Interaction, RWTH Aachen University, Aachen 52062, Germany (e-mail: schluse@mmi.rwth-aachen.de; priggemeyer@mmi.rwth-aachen.de; atorf@mmi.rwth-aachen.de; rossmann@mmi.rwth-aachen.de).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2018.2804917

1551-3203 © 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

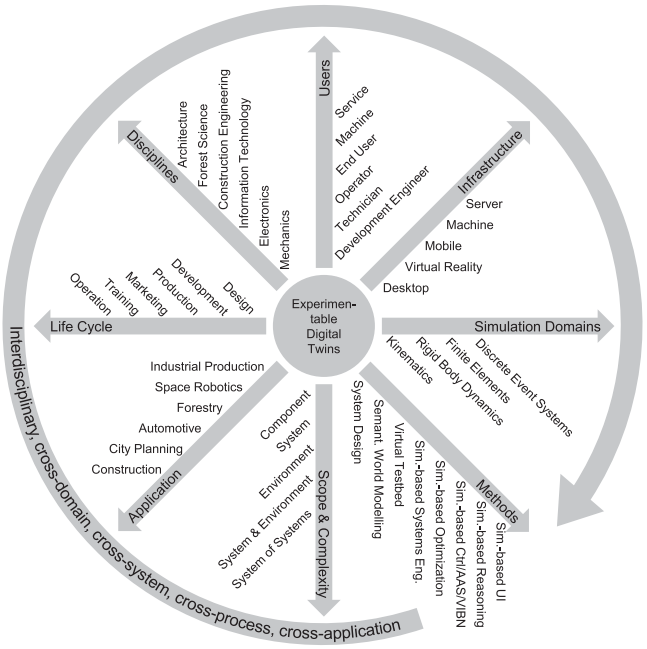

Fig. 2. In Industry 4.0, simulations are not only used for engineering but have to address various different aspects.

time frames. Development processes have to keep pace with this. Simulations provide an *experimentable knowledge base*, allowing to introduce new agile development processes based on new communication, documentation, and test infrastructures covering the entire life-cycle.

Addressing these (and other) requirements, this paper proposes a *new structuring element for simulation-based engineering processes called "experimentable digital twin" (EDT)* as well as a *new approach to simulation technology bringing those EDTs to life called "virtual testbeds (VTB)*" used in diverse *application scenarios called "simulation-based X*," all covered by an integrated development process called *simulation-based systems engineering* (see Fig. 16). EDTs provide a one-to-one replica of a real system incorporating all components and aspects relevant to use simulations for engineering purposes but also inside the real system during real-world operations. The concept of EDTs is based on the notion of digital twins, one of the key Industry 4.0-concepts (see Section III-A). VTBs provide comprehensive simulations of interacting EDTs in their operational environment in various application scenarios (see Section III-C). A tight integration of virtual and real worlds allows for a seamless transition between virtual and physical systems and to seamlessly integrate simulations into physical systems (see Section III-D). This greatly enhances the state of the art in simulation technology which is characterized by various approaches used for the detailed analysis on component level but limited analysis capabilities on system level, almost no re-use of models and simulations, no direct transition, and especially no integration between virtual models, simulations, and the physical system, and only a very loose integration of simulations into engineering processes. Coming back to Fig. 2, EDTs are now the common basis for the use of simulation technology throughout the life-cycle, for different disciplines, users, infrastructures, simulation domains, engineering methods, model scopes, and application areas. Furthermore, one can now easily combine different elements of the different dimensions and easily move applications from one element to another (e.g., from virtual reality applications to real-time simulations inside the machine during real-world operation).

The rest of this paper is organized as follows. Section II summarizes the state of the art concerning simulation technology for industrial production. Section III introduces the concept of EDTs and illustrates how EDTs are modeled, simulated, and integrated with real-world systems. These EDTs are an enabling technology for simulation-based X approaches. Section IV shows how EDTs are used throughout the life-cycle focusing simulation-based optimization (SbO) and control. This contribution ends with a conclusion in Section V.

II. State of the Art

Taking a look at the state of the art of simulation technology reveals various approaches. Discrete event simulation systems [7], equation-based/signal oriented approaches like MATLAB/Simulink [8], the declarative Modelica modeling language [9] and its tools, FEM-based simulation tools (e.g., [10]), game engines (e.g., [11]), or generic mechatronic systems (e.g., [12]) are probably the most well-known ones. In the field of robotics, some known tools are the various simulation tools provided by robot manufacturers like KUKA.Sim (www.kuka-robotics.com), V-REP [13], and GAZEBO [14] (both specialize on sensors and image-based assembly systems), or other commercial products like Emulate3D (www.demo3d.com), ISG-virtuos (www.isg-stuttgart.de), machineering (www.machineering.de), and the Siemens PLM product family (www.plm.automation.siemens.com).

To simulate complex systems including system dynamics, sensors, data processing, etc., different simulation systems must be combined. Due to missing interoperability of these systems and their underlying models, the realization of such system level simulations incorporating all individual simulations on component level today becomes almost impossible. Concepts like the digital factory aim at simulating entire production plants but are restricted to this very specific application area and often to simulation algorithms provided by one of the tools mentioned above.

Functional mockup units (FMUs) [6] are an important step improving the interoperability of different simulation systems. Using FMUs is perfect if different subsystems, which are connected by a limited number of inputs and outputs exchanging a limited amount of information, are coupled. Otherwise the resulting network of connected FMUs becomes too complex and often almost unmanageable or too much time is needed for information exchange. In addition to this, their interaction must be fully described using their inputs, outputs, and state variables, which renders, e.g., "system wide" rigid body or sensor simulations with interacting rigid bodies of different subsystems impossible because here additional simulation algorithms

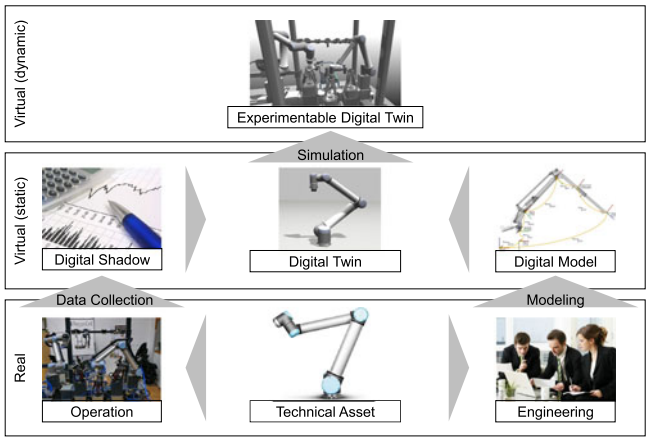

Fig. 3. Correlation of the Industry 4.0 terms "technical asset," "digital shadow," "digital twin," and "EDT" (based on [18]).

working on top of the component models need detailed internal information of the distinct models of the different subsystems.

To sum up, a lot of tools and interfaces are available solving various problems, but simulating complex systems in their entirety still is a big challenge. This works to a certain extend in distinct application fields like production technology or the automotive sector, but a practicable overall approach is still missing. What is needed is a methodology where the models can still be developed on component level, are then combined into simulations on system level, and can then be used in various applications. For this, a new methodology is necessary to model (from a behavioral and structural point of view) one-to-one replicas of real-world artefacts (EDTs), to set up networks of interacting EDTs replicating real-world scenarios, and to simulate these scenarios in a simulation infrastructure organizing all the "subsimulators" necessary for this—and all this in an application-neutral form and covered by an overall integrated systems engineering approach.

III. Concept of EDTs

The notion "digital twin" is inspired by the developments known under terms like "Industry 4.0" or "industrial internet" (see Fig. 3). In general, a digital twin is a one-to-one virtual replica of a "technical asset" (e.g., machine, component, and part of the environment). A digital twin contains models of its data (geometry, structure, ...), its functionality (data processing, behavior, ...), and its communication interfaces. It integrates all knowledge resulting from modeling activities in engineering (digital model) and from working data captured during real-world operation (digital shadow). Simulators are used to make the digital twin experimentable [15], leading to the notion "**EDT,**" first introduced in [3].

A. Basic Idea

EDTs combine the ideas of digital twins with model-based systems engineering (MBSE) and simulation technology (see Fig. 4) to provide a new structuring element for simulation-based systems engineering processes for a variety of different

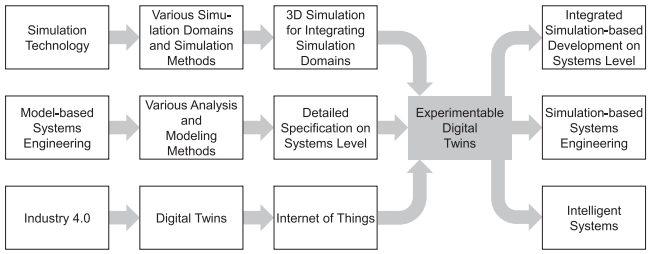

Fig. 4. Combination of simulation technology, MBSE, and Industry 4.0-concepts leads to the concept of "EDTs" and various benefits for the different domains.

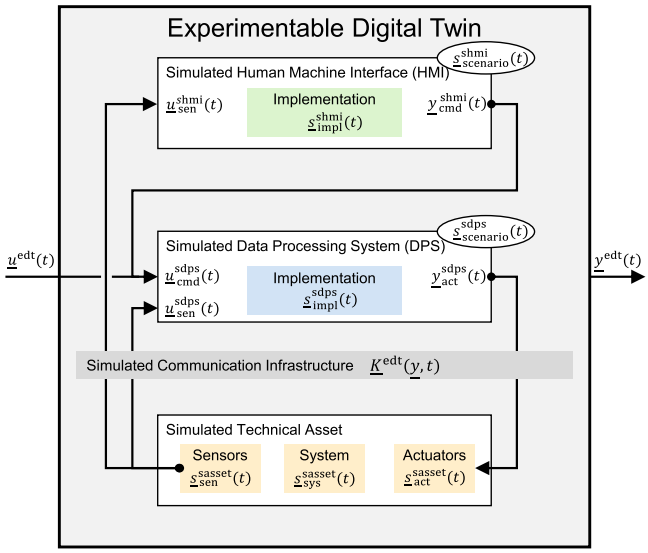

Fig. 5. Basic structure of EDTs.

applications from development over optimization, verification, user interfaces, and training, up to the realization of intelligent systems, to just name a few examples. For this, simulation technology delivers all the necessary methods to simulate nearly any aspect of complex systems, while new concepts of three-dimensional (3-D) simulation technology [5] allow to integrate various simulation domains. Industry 4.0 provides new concepts to connect different parts of complex systems, MBSE is used to systematically explore the requirements, structure, and behavior of those systems. The resulting networks of interconnected and interacting *EDTs integrate and connect different model components regardless of the concrete simulation algorithms used*. For this, their inherent communication capabilities are used for port-based communications, specialized integration concepts are used for simulation algorithm interaction. This way, EDTs enable an integrated simulation-based development on system level as well as the realization of intelligent systems.

A systematic view on EDTs is given in Fig. 5: An EDT combines a simulated data processing system (DPS) with a simulated technical asset and a simulated human-machine interface (HMI). The simulated technical asset comprises the simulated system represented by its internal simulation state vector $\underline{s}_{\mathrm{sys}}^{\mathrm{sasset}}(t)$ (e.g., robot), its simulated sensors $\underline{s}_{\mathrm{sen}}^{\mathrm{sasset}}(t)$ (e.g., camera), and actuators $\underline{s}_{\mathrm{act}}^{\mathrm{sasset}}(t)$ (e.g., motors). Each state vector

$\underline{s}(t) = [\underline{x}(t), \underline{a}, \underline{A}]$ consists of the current state of corresponding component, its parameters, and its algorithms. The simulated technical asset has to adequately mimic the corresponding real world $\underline{x}_{\text{sen}}^{\text{rasset}}(t)$, $\underline{x}_{\text{sys}}^{\text{rasset}}(t)$, and $\underline{x}_{\text{act}}^{\text{rasset}}(t)$.

The DPS usually processes sensor data and/or commands a system, e.g., mapping or motion planning algorithms. It is represented by its internal state $\underline{s}_{\text{impl}}^{\text{sdps}}(t)$, sensor input $\underline{s}_{\text{sen}}^{\text{sdps}}(t)$, actuator output $\underline{s}_{\text{act}}^{\text{sdps}}(t)$, and its perceived environment $\underline{s}_{\text{scenario}}^{\text{sdps}}(t)$ (e.g., generated maps). The same holds for the HMI. The EDT has inputs $\underline{u}^{\text{edt}}(t)$ and outputs $\underline{y}^{\text{edt}}(t)$, e.g., to exchange production commands and results from and to superordinated MES or to communicate with external tools. All components of the EDT as well as the EDT itself communicate using a simulated communication infrastructure $\underline{K}^{\text{edt}}(\underline{y}(t), t)$ which resembles the real communication infrastructure of its real-world counterpart.

The result may be fairly complex models leading to a combined simulation state vector $\underline{s} := \left[\underline{s}^{\text{sasset}}, \underline{s}^{\text{sdps}}, \underline{s}^{\text{shmi}}\right]$. The simulation $\underline{s}(t) = \Gamma\left(\underline{s}_0, \underline{u}^{\text{edt}}(t), t\right)$ starts with an initial state $\underline{s}_0 := \underline{s}(0)$, the output is calculated via $\underline{y}^{\text{edt}}(t) = \Phi\left(\underline{s}(t), \underline{u}(t), t\right)$, and combines the simulation state $\underline{s}$ of all components of the EDT as well as its input $\underline{u}$. Γ contains all algorithms necessary to carry out the simulation. These algorithms are provided by the different EDT components (e.g., the DPS) or by the overall simulator itself. In the latter case, they rely on and combine the models provided by the EDT components (e.g., rigid body dynamics or sensor simulation).

To simulate the ReconCell, we add EDTs of, e.g., the robot, the grippers, the work pieces, and the work cell environment to a *scenario model*. This can be done independently from the concrete modeling of each EDT. Only the semantics and the inputs and outputs must be defined before. In our example, the simulated technical asset of robot and gripper is modeled using rigid body dynamics techniques, the sensors use specialized sensor simulation approaches, and the DPSs are implemented using MATLAB/Simulink. Having done so, the set of EDTs forms the virtual ReconCell scenario, which is ready to be simulated (see Fig. 6). [1] The simulation state vector of the scenario $\underline{s}^{\text{scenario}}(t) = \left[\underline{s}^{\text{edt},1}(t), \underline{s}^{\text{edt},2}(t), \ldots\right]$ combines the states of all participating EDTs, so that the resulting simulation task is given via $\underline{s}^{\text{scenario}}(t) = \Gamma\left(\underline{s}_0^{\text{scenario}}, t\right)$.

B. Modeling EDTs

EDTs can be of varying complexity, i.e., they do not always contain all components as shown in Figs. 5 and 6. The workpiece EDT, e.g., only consists of the simulated technical asset $\underline{s}_{\text{sys}}^{\text{sasset}}(t)$, while only two other EDTs have a HMI (the console and the UR10 robot). But how are they modeled? Here MBSE comes into play. It provides a systematic approach to analyze and model even complex systems. The result of the MBSE process are, e.g., SysML diagrams, which are the starting point for the modeling activities from the simulation point of view. They deliver the structure, the connections between the distinct "blocks," and the behavior of some of the blocks. In addition to this, requirements are defined that the developed system has to fulfill, which can in turn be tested using different test cases.

[1] Please note that there are no connections between some of the EDTs. The connections between the robot's actuators and sensors are part of the EDT of the robot and are not visible from this level of detail. Other interactions do not need to be modeled explicitely. This includes the mutual interdependency between the movements of workpiece and robot or the influence of robot movements on camera images.

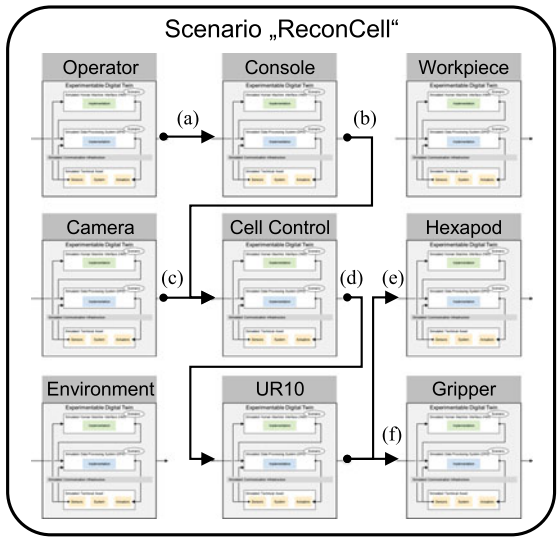

Fig. 6. Multiple interconnected and interacting EDTs make up an application scenario. Signal contents: a) operator interaction, b) op. commands, c) camera images, d) movement & gripper commands, e) fix/release, and f) open/close.

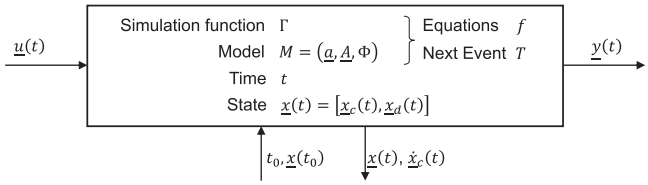

Fig. 7. Structure of the single components (blocks) of EDTs (based on [6]).

Fig. 8 (left) depicts the block definition diagram of the ReconCell (the diagrams use the SysML notation and are simplified to concentrate on the basic concepts), consisting of e.g., a robot, its actuators (motors), sensors (camera), and its robot controller (RC). It also contains the workpiece, a console, and further blocks representing the environment. The next step is the definition of ports and data flow between different blocks. The right part of Fig. 8 illustrates a simple example concerning the data acquisition: The cell control commands the robot movement via the RC (which in turn controls the motors) and retrieves image data from the camera. So far, this is a standard MBSE process which continues with the definition of behaviors, constraints, or parameters.

To enable simulation, these blocks contain models $M^{\text{edt}} = (\underline{a}, \underline{A}, \Phi)$ providing parameters $\underline{a}$ used by superordinated simulation algorithms Γ (like rigid body dynamics, sensor simulation) or the component algorithms $\underline{A}$ themselves (e.g., motor behavior modeled using MATLAB). All those algorithms must follow the structure depicted in Fig. 7. From our experience,

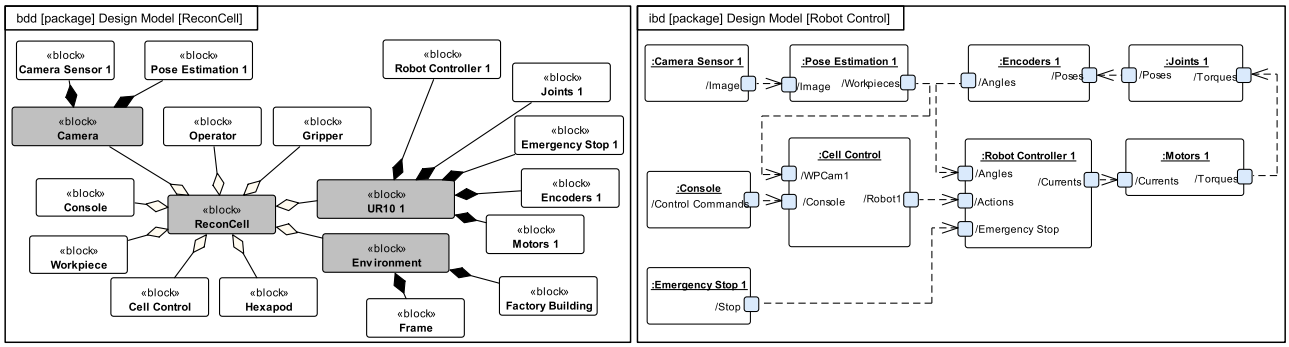

Fig. 8. Two (simplified) MBSE diagrams for the ReconCell use case.

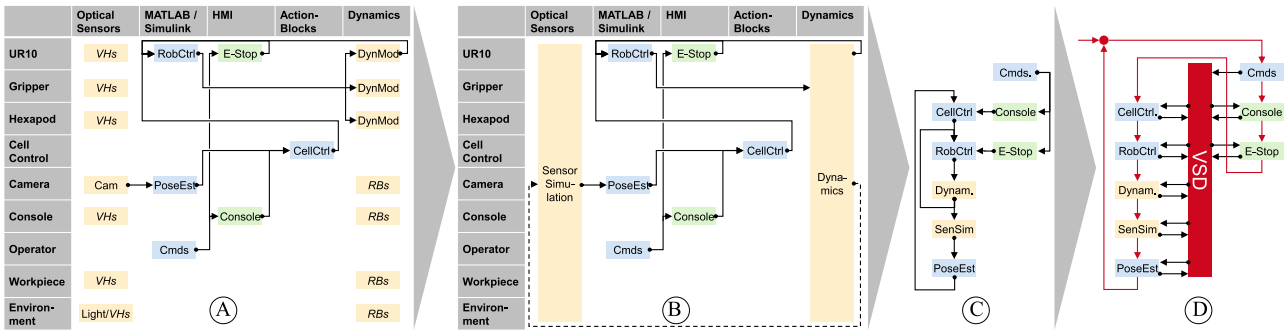

Fig. 9. Automatic model analysis process. a) EDT components and simulation algorithms with their explicitely modeled data flows and interactions. Columns are simulation functions, rows are EDTs, VHs = visible hulls, and RBs = rigid bodies. b) Automatic grouping by simulation functions, e.g., of sensor simulation and dynamics. c) Resulting simulation network, executed by the scheduler. d) Simulation results are routed via the VSD.

all major simulation algorithms necessary, such as for industrial production or space robotics, fit in this scheme.[2]

C. Simulating Interacting Experimentable Twins

But how can we simulate such a network of EDTs to get $\underline{s}^{\text{scenario}}(t)$? The VTB approach delivers the necessary building blocks. VTBs are the run-time environment for EDTs and allow the entire network of interacting EDTs to be simulated within one single integrated framework—the VTB. The basis for scenario simulation in VTBs is an *automatic model analysis process.* Fig. 9 can be directly and fully automatically derived from Fig. 6. Fig. 9(a) shows the various components of all EDTs with their explicitely modeled data flows and interactions. For more complex systems, the shown matrix can be significantly larger. Each block represents an element from the EDT formalism, i.e., technical asset (yellow), DPS (blue), or HMI (green), as described in Section III-A and shown in Fig. 5. Columns represent simulation domains, while rows are EDTs. All shown components have a real counterpart in the physical world and a virtual representation as part of EDTs inside the VTB.[3]

In this example, we have three types of algorithms, rigid body dynamics, sensor simulation, and data processing algorithms. When loading the full scenario model, the simulator assigns all rigid body dynamics models to the rigid body simulation so that they can interact with each other [Fig. 9(b)]. In addition to this, both rigid body dynamics and sensor simulation work on the same geometric model; hence the simulated sensor data reflects the movement of the EDTs.[4] After having grouped components belonging together, the resulting network is analyzed by the scheduler of the simulation system [see Fig. 9(c)]

[2]It is important to mention that the main focus of this paper is neither to propose new simulation algorithms nor to develop new co-simulation interfaces. We are using the developments available so far but add a new methodology on top of them. To model the single components (of the EDTs), we use standard simulation technology as provided by MATLAB, ANSYS, OpticStudio, etc., use standard interfaces like FMI if available or proprietary interfaces (like the MATLAB-C-Interface or the DDE interface of OpticStudio) if necessary and combine this with our own simulation algorithms, e.g., in the field of rigid body dynamics and sensor simulation.

[3]A detailed explanation of Fig. 9(a) follows: The human operator's commands are simulated by a MATLAB script; they are routed via the cell's operator console. A camera is the only active optical sensor in this (simplified) example. The other EDTs contribute to the camera images only by their visible hull geometry (VHs) and by environmental light sources. Assembly processes for this ReconCell are programmed via ActionBlocks (a visual programming framework, see [20]) and executed by the UR10 RC, which is simulated in MATLAB/Simulink. Finally, active cell components (UR10 robot, gripper, configurable hexapod mount) move according to their dynamic models. The EDT of the workpieces do not move actively on their own (they can only be grabbed or placed), hence they have no excplicit data flow connection modeled.

[4]Coming back to FMUs, they are perfect for the columns "MATLAB/Simulink, HMI, ActionBlocks," which is the typical usage scenario for FMUs. From the overall system perspective, the subsystems are "black boxes," their interaction is defined via the inputs and outputs. FMUs cannot be used for the columns "dynamics, optical sensors," because here the models of the subsystems themselves (the "bodies" used for sensor simulation and rigid body behavior) are combined into a new simulation component.

and simulated.[5] To guarantee a consistent and performant data transfer between the different simulation algorithms, the simulation results are routed via a versatile simulation database (VSD, [16]) of the simulation system [see Fig. 9(d)] . The VSD is a real-time database which is able to store any data structure which can be modeled by UML class diagrams.

To enable the realization of comprehensive and close-to-reality EDTs, the availability of a simulator offering the performance characteristics necessary to set up such VTBs is crucial for the realization of EDTs (see [5]). Therefore, we developed a new architecture for simulation systems (containing, e.g., the VSD concept) leading to a reference implementation called *VEROSIM* [16], which is the basis of all examples shown in this paper. This framework itself is purely abstract, so that it can act as the basis for the *implementation or integration* of virtually any type of simulation algorithm or tool.

D. EDTs in Hybrid Scenarios

Connecting (parts of) EDTs simulated in VTBs with real assets leads to hybrid scenarios in which simulation technology in the form of EDTs is used in direct connection with the real hardware. Easily replacing real/virtual components by their virtual/real counterparts is possible because an EDT replicates its real counterpart not only with regard to its input/output behavior, but also with regard to its internal structure.

Hybrid scenarios allow on one hand for a seamless bidirectional transition between virtual and the corresponding real systems to realize complex control algorithms (simulation-based control (SbC), e.g., for the realization of RCs (mobile robots and robot manipulators) knowing and interacting with their environment (see Fig. 13) or the realization of intelligent sensors with integrated sensor data processing) or innovative user interfaces (simulation-based UI, e.g., for the realization of driver assistance systems). Here, the approach follows the goals of rapid control prototyping [17] and virtual commissioning [4] and provides an implementation for this. The benefit now is, that all the important parts of the simulation infrastructure are now available not only for development but also on the target system, so that we can use this infrastructure for algorithm implementation and we can run the original simulation model on the target system without code conversion, which eases debugging and communication.

On the other hand, this allows to use simulations to make systems situation aware and "intelligent" (mental models or simulation-based reasoning, [2]). Here, EDTs store all the information of the environment and VTBs simulate the behavior of the system and its environment in an experimentable model, and this way enables the system to test action alternatives and to assess the behavior of other surrounding "agents."

[5]The resulting simulations run in real-time or may be faster or slower than real time, depending on the complexity and amount of the components and models involved. To "scale" this, EDTs may provide different variants of models like very detailed or simplified model which can be used in different types of simulations (e.g., detailed analysis versus interactive training simulators). It is up to the user to select the simulation algorithms (and this way the models) to use; the simulator then combines the necessary parts.

Fig. 13 illustrates the basic idea. The DPS of the EDT of the robot is connected to the real robot. The connection between the digital and real twins takes place via their respective communication infrastructures involved (the depicted connections of the inputs and outputs are thus realized technically via the illustrated dotted lines). EDT components can thus take over tasks in real systems fully transparently. This applies to the DPS, as well as to the user interface, as well as to the combination of these systems.

IV. EDTs as Enabler for Simulation-Based X

(Possibly hybrid) networks of interacting EDTs simulated in VTBs are the basis for diverse "simulation-based X" applications, including new methods of simulation-based systems engineering as well as SbO, reasoning, verification & validation, control, and HMI [5].

A. Simulation-Based Systems Engineering

MBSE is an emerging technology enabling system engineers to systematically exploit, model, communicate, and verify the functionalities of a new system in a formal way. Having done so, the simulation of these models is the natural next step but requires the integration of MBSE with detailed simulation models which is a major show-stopper today. Simply modeling the behavior of each SysML block (see Fig. 7) inside these blocks and model the interactions by connecting these blocks is not feasible and practically impossible, because the resulting diagrams become far too complex and unmanageable.[6] EDTs close this gap as illustrated above.

B. Simulation-Based Optimization

By harnessing the power and flexibility of EDTs and VTBs, SbO tasks can be executed that are almost impossible to carry out without EDTs. In the following example, a ReconCell contains two UR10 robots which are used to assemble automotive headlight housing. Due to quality assurance requirements, good visibility of the workpiece throughout the assembly process must be ensured. The resulting engineering problem *where to place the camera to maximize visibility* is now solved using SbO with EDTs.

For this, all workpieces are marked with a distinct unique color (in this case blue) so they can easily be identified during image processing. This is done by changing the workpiece EDT's material color, i.e., by modifying the according a_i of $\underline{s}^{\text{sys}}_{\text{sasset}}(0)$. Fig. 10 shows simulated camera images and illustrates how the workpiece is sometimes temporarily occluded by robot arms or cell struts. Now let $B \in \mathbb{N}$ be the total number of blue pixels observed during a full assembly task, which we want to maximize using the objective function $f(p_i) = B$

$$p_i^* = \underset{p_i \in \mathbb{P}}{\arg\max}\ f(p_i) \quad \text{for } i = 1, \ldots, N \tag{1}$$

[6]Imagine, for example, all the connections necessary to model the interactions between the workpieces and their environment including the robots, their grippers, etc.

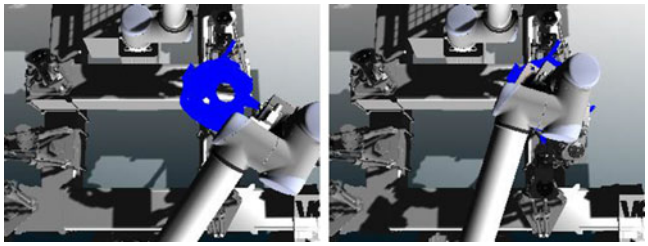

Fig. 10. Simulated camera views from within the robot work cell during production. The workpiece is marked blue. The right image was taken shortly after the left one.

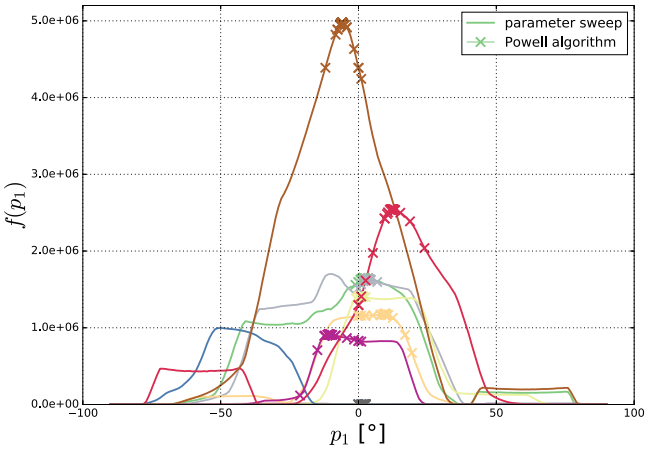

Fig. 11. Optimization results for (1). Lines show parameter sweeps $f(p_i)\forall i$. Crosses indicate evaluations by the Powell optimizer [23].

Here, the parameters $p_i \in \mathbb{P}$ with $\mathbb{P} = [-90°, 90°]$ are possible camera pan angles for N different mount points in the cell. Hence, we solved N instances of (1) using different optimization algorithms and chose the overall winner as best mount option.

Fig. 11 depicts results of the optimization runs using the Powell algorithm [23], which worked best compared to several others tested. With $N = 10$, the optimizations required about 500 full simulation runs. One assembly task takes about 3 min to complete, which can be simulated several times faster than real-time on a modern workstation with a dedicated 3-D graphics processing unit (GPU). In our configuration, the complete optimization took less than 4 h. Step time for dynamic simulation was $dt = 10$ ms, while optical sensor simulation was executed with $dt = 100$ ms. The result yields the most appropriate cell configuration for a given task fully automatically. The results can be directly transferred to the physical cell by mounting the camera accordingly.

C. Simulation-Based Control

At several points in the engineering process of a Cyber physical system, engineers need to make the transition from the VTB to the real world and back to the virtual world. EDTs complement this by, e.g., providing a methodology to seamlessly transfer controllers or user interfaces of EDTs to their counterpart, the real technical assets, named SbC. In terms of the ReconCell scenario, SbC is used to realize smart controllers for the UR10 robots used here (see Fig. 12).

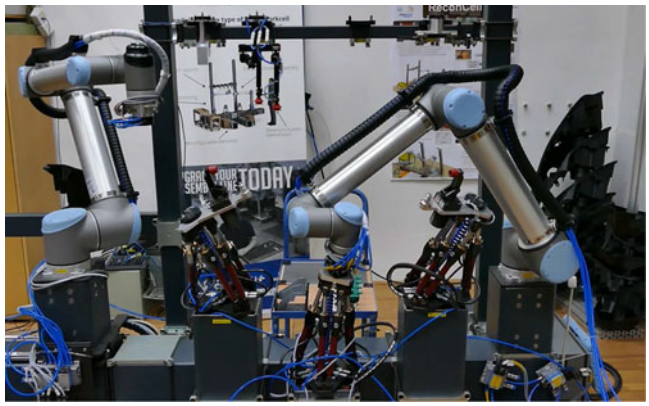

Fig. 12. Real setup of the virtual ReconCell shown in Fig. 1. All controllers are implemented via SbC.

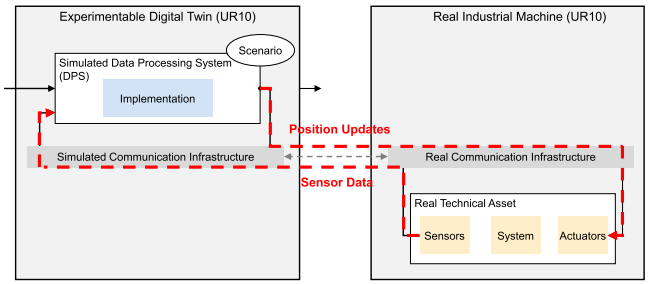

Fig. 13. Networking EDTs and their physical counterparts in SbC scenarios, here shown for a UR10 robot in the ReconCell.

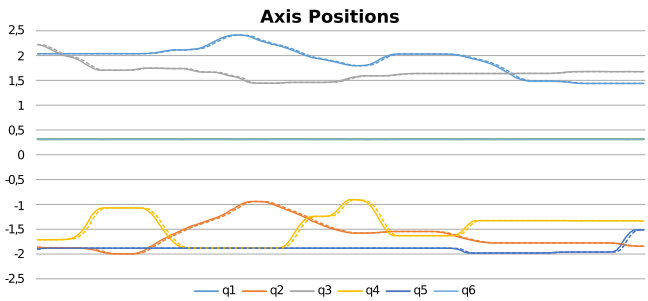

Fig. 14. Actual sampled UR10 positions during a reconfiguration process in ReconCell (solid: commanded, dashed: measured).

Fig. 15. Examples for application areas of EDTs beyond industrial production covered so far (f.l.t.r.): Automotive, space robotics, forestry, and intralogistics.

Fig. 13 shows the idea of SbC, the use of hybrid scenarios, as introduced in Section III-D, to realize DPS for real technical assets, here used for the UR10 robots. From an EDT point of view (see Fig. 13), the robots consist of torque sensors and position encoders (sensors) as well as servo motors (actuators)

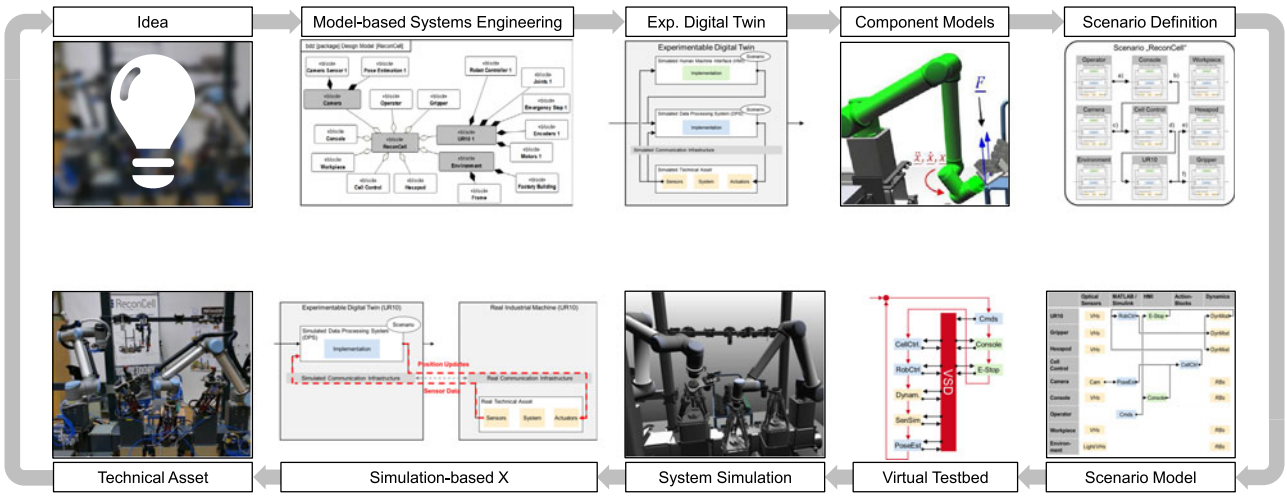

Fig. 16. Life-cycle spanning simulation-based systems engineering using EDTs.

for each joint and a system model describing the robots dynamic behavior (system). In addition to this, the robots have an HMI as well as a DPS. This EDT can be operated by input values for control mechanisms implemented in the DPS for a specific scenario.

The simulated DPS holds a "scenario," which in this case is a recursive EDT of the UR10 working in its environment (see Fig. 6). External sensor data is used to find the workpieces to be assembled, to calculate their exact positions and orientations, and to integrate them into this scenario model. It therefore holds the system state determined based on measurements from the physical system. From a system-theoretical point of view, this internal scenario builds upon (e.g.) observers [22] to get a deep insight into the system behavior. In the ReconCell context, it is used for sensor fusion, path planning, and automatic collision avoidance—all realized using EDTs and a "real-time VTB." The scenario model provides all necessary data for this and allows to test the calculated plans before and during execution.

When this simulated DPS is used to control the real robot, it is directly connected to the real machine by simply reconnecting the same (virtual) ports of the DPS previously connected to the (virtual) ports of the virtual machine to the (real) ports of the real machine; the simulated controls now control the robot movements (see Fig. 13). An important precondition for this—besides a real-time simulation of the DPS—are (from a structural point of view) identical interfaces between (virtual) DPS and (virtual/real) machine as well as a real-time data exchange accomplished by an appropriate interface between simulation and the physical system. In general, both the real machine as well as the EDT, utilize corresponding communication infrastructures K to pass sensor data and control signals for actuators between the technical asset and the DPS. For SbC, those communication infrastructures are connected via a real-time interface to exchange data between the real technical asset and the simulated DPS (see Fig. 13, red dashed line). As a result, the EDT on the left can now access the sensor data provided by the real machine to generate and feed back control signals for the actuators in the real technical asset.

Fig. 14 gives an example for the output of this simulated DPS moving the real robot. The DPS runs inside a real-time VTB using the real-time operating system QNX and accepts set points as inputs, position updates are sent to the robot to be executed by the actuators (solid). The graph also shows the axis positions of the real robot sampled during a reconfiguration process of the robotic work cell (dashed). The offset between commanded and measured positions originate from the 4 ms timing constraint for the robot control. Simulation provides all the necessary information for the control scheme (e.g., current state of passive fixtures) that would not be available to the real machine if it was operated separately. In this example, the reconfiguration by itself becomes a complex task if viewed from a conventional RC's perspective. Such a controller requires a large number of poses which have to be taught manually. In addition to this, it has to manage all possible work cell configurations for the manipulation of a variety of work pieces. Due to the application of SbC utilizing the state information provided by the EDT, collision free trajectories that are based on a minimal set of reference poses, like the one shown in Fig. 14, are generated fully automated in the virtual environment.

Control schemes can become arbitrarily complex considering the ability of the presented concept to interconnect a variety of EDTs modeling large application scenarios. Therefore, SbC is the basis for the realization of simulation-based reasoning [2]. The scenario model inside the DPS serves as a *Mental Model* integrating all knowledge of the controlled system itself as well as its environment allowing to implement situation awareness concepts (perception of environmental elements and comprehension of their meaning, managing the resulting environment model, projection of the effects of changes using simulation, . . .).

V. Conclusion

This paper introduces the concept of EDTs as a new structuring element for simulation-based systems engineering processes and their interdisciplinary and cross-domain simulation

in VTBs. This enables comprehensive simulations on system level, [7] seamlessly connects virtual and real worlds in hybrid scenarios,[8] introduces new structures and processes to consistently use simulations in varying application scenarios throughout the life-cycle,[9] and thus streamlines the engineering process in the era of Industry 4.0 (see Fig. 16).

The models of the EDTs extend the MBSE system model and are directly linked to, e.g., SysML blocks, requirements, and testcases [19]. This way, EDTs greatly contribute to the overall vision of MBSE concerning the formalized application of modeling to support the entire life cycle, integrating modeling and simulation into a consistent process for small components as well as for large systems of systems as planned by [21] for 2025. But still missing is an approach allowing for a seemless and practical bidirectional connection between MBSE and simulation. EDTs are the missing piece of the puzzle to do this. Using EDTs, the power of today's simulation technology can be directly used during the SE process.

Following the EDT concept, new scenarios can be configured by selecting and configuring the participating EDTs and (if necessary) adding the new EDTs, only.[10] It is up to the VTB infrastructure to derive the experimentable scenario.

The concepts of EDTs and VTBs have proven their feasibility when realizing a large variety of different applications. The initial application spectrum in space, industrial production, and the environment has recently expanded, e.g., into the area of construction and facility management as well as automotive (see Fig. 15). All these applications clearly show that the approach presented in this paper is feasible and promising.

But what are the consequences for the development of components, systems, or systems of systems? The answer is threefold. First of all, *use EDTs for the development of future systems*. Providing EDTs and using simulation (e.g., during the MBSE process, for development, characterization, verification, validation, and optimization) should become as natural as developing the hardware and/or software itself. This holds great promises concerning efficient development, quality, performance, reuse, and technology transfer for new components and significantly raises the modeling and simulation power of simulation technology. The continuous and integrated use of simulation technology in a simulation-based systems engineering process leads to cost-efficient development processes, better designs, and more reliable systems.

Second, *bring simulation technology to the real system*. EDTs as well as VTBs provide key technologies for the development of intelligent systems. They ease the development of complex algorithms and their transfer to the real system. This allows to realize "safe systems"—i.e., systems which supervise themselves using simulation—and mental models, a key component for intelligent systems.[11]

Third, *continue with the development of the overall approach*. Today, we can provide a reference implementation which is ready to be used in a variety of application areas. But there is still a lot of work to do, e.g., concerning the development of simulation algorithms (such as flexible or soft bodies), for the integration of different simulation algorithms, to integrate VTBs in development infrastructures, and so on.

References

[1] A reconfigurable robot work cell for fast set-up of automated assembly processes in SMEs, 2017. [Online]. Available: www.reconcell.eu

[2] J. Rossmann, E. Kaigom, L. Atorf, M. Rast, G. Grinshpun, and C. Schlette, "Mental models for intelligent systems: eRobotics enables new approaches to simulation-based AI," *KI-Künstl. Intell.*, vol. 28, no. 2, pp. 101–110, 2014.

[3] M. Schluse, L. Atorf, and J. Rossmann, "Experimentable digital twins for model-based systems engineering and simulation-based development," in *Proc. Annu. IEEE Int. Syst. Conf.*, Montreal, QC, Canada, 2017, pp. 628–635.

[4] *Virtual Commissioning*, VDI/VDE-Standard 3693, 2016. [Online]. Available: www.vdi.de/3693

[5] S. Kadry and A. El Hami, Eds., *E-Systems for the 21st Century: Concept, Developments, and Applications*. Waretown, NJ, USA: Apple Academic Press, 2015.

[6] Functional mockup interface, 2014. [Online]. Available: www.fmi-standard.org

[7] J. Banks, *Discrete-Event System Simulation*. Englewood Cliffs, NJ, USA: Prentice-Hall, 2010.

[8] MATLAB/Simulink, 2017. [Online]. Available: www.mathworks.de/products/simulink/

[9] P. Fritzson, *Principles of Object-Oriented Modeling and Simulation With Modelica 2.1*. Hoboken, NJ, USA: Wiley, 2003.

[10] COMSOL, 2017. [Online]. Available: www.comsol.com

[11] The Unreal game engine, 2017. [Online]. Available: www.unrealengine.com

[12] Simmechanics, 2017. [Online]. Available: www.mathworks.de/products/simmechanics/

[7]Today, setting up such scenarios at a practically relevant scale leads to a lot of work necessary for each individual scenario. The model analysis process deriving simulation models from the network of EDTs by analyzing the models of the individual EDTs, their overlaps and their interactions, and deriving an experimentable overall model from the scenario to be simulated addresses this problem.

[8]The concept of rapid control prototyping is known for more than 10 years, tools like MATLAB/Simulink provide a consistent toolset for this. Virtual commissioning approaches are used especially in the production sector and are supported by tools like ISG-virtuos. But today, realizing x-in-the-loop scenarios requires to set up new models for each individual application. EDTs are a one-to-one replica of a real world asset—also from a structural/architectural/interface point of view. This allows to replace each EDT and each EDT component by its real counterpart at any time during the development process by changing the corresponding connections. No code conversion is necessary.

[9]Today, we have everything necessary to solve individual problems but no approach covering the entire scenario to be simulated and no process leading to this scenario. This article presents a new holistic approach integrating the various existing approaches used so far (MBSE, simulation algorithms and tools, co-simulation approaches, x-in-the-loop, VR, . . .). Up to our knowledge, there is no approach available which integrates MBSE, various simulation approaches, Industry 4.0, and x-in-the-loop concepts to realize the various simulation-based applications in all these disciplines and domains.

[10]Each simulation tool today offers model libraries to simplify and speed up the process to set up a new model. But these libraries are tool specific. Necessary is an approach allowing this on system level incorporating different simulation approaches.

[11]Coupled experiments with real components are relevant for development as well as for production stages during the life-cycle. During development, SbC allows for prototyping data processing algorithms and carrying out virtual commissioning. During production, SbC allows to implement controllers which have to cope with detailed knowledge about the asset itself and/or its environment as basis for calculations or simulations. This is the starting point, e.g., for the realization of intelligent systems which need to evaluate decision alternatives or asset administration shells exposing highly detailed knowledge (including its behavior) about the asset to the digital world. The major benefits of SbC are the quick and seamless transition between virtual and hybrid scenarios without the need to change the models involved (only redirect some connections), the full access to the original controller implementation during real-world operation and the possibility to use the simulator infrastructure to implement controllers which have to cope with 3-D environment as basis for calculations or simulations.

[13] Coppelia Robotics Software, V-rep, 2017. [Online]. Available: www.coppeliarobotics.com
[14] Open Source Robotics Foundation, Gazebosim 2017. [Online]. Available: gazebosim.org
[15] *Simulation of Systems in Materials Handling, Logistics and Production–Fundamentals*, VDI-Standard 3633 Part 1, 2016. [Online]. Available: www.vdi.eu/3633
[16] J. Rossmann, M. Schluse, C. Schlette, and R. Waspe, "A new approach to 3D simulation technology as enabling technology for eRobotics," in *Proc. 1st Int. Simul. Tools Conf. Expo.*, Brussels, Belgium, 2013, pp. 39–46.
[17] D. Abel and A. Bollig, *Rapid Control Prototyping: Methoden und Anwendungen*. New York, NY, USA: Springer, 2006.
[18] WGP-Standpunkt Industrie 4.0, Wissenschaftliche Gesellschaft für Produktionstechnik WGP e.V, 2016.
[19] J. Rossmann *et al.*, "Integrated model-based system specification and simulation: Case study on sensor design for extraterrestrial applications," in *Proc WInTeSys 2017*, Paderborn, Germany, 2017, pp. 281–294.
[20] C. Schlette, D. Losch, and J. Rossman, "A visual programming framework for complex robotic systems in micro-optical assembly," in *Proc. 41st Int. Symp. Robot.*, 2014, pp. 750–755.
[21] INCOSE: Systems Engineering Vision 2025, 2014.
[22] D. G. Lueneberger, "Observing the state of a linear system," *IEEE Trans. Mil. Electron.*, vol. 8, no. 2, pp. 74–80, Apr. 1964.
[23] M. J. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *Comput. J.*, vol. 7, no. 2, pp. 155–162, 1964.

Michael Schluse received the Dipl.-Ing. and Ph.D. degrees in electrical engineering from the Technical University of Dortmund, Germany, in 1996 and 2002, respectively.

Between 1996 and 2005, he was a Researcher and a Team Leader for "System Technology" with the Institute of Robotics Research, Dortmund. Between 2005 and 2006, he was the Department Head with the EFR-Systems GmbH and was responsible for the development of modern 3-D simulation technologies and versatile virtual reality systems. Since 2006, he has been the Chief Engineer with the Institute for Man-Machine Interaction, RWTH Aachen University, Aachen, Germany.

Marc Priggemeyer received the Dipl.-Ing. degree in computer engineering from the RWTH University Aachen, Aachen, Germany, in 2013.

Since 2013, he has been a Researcher with the Institute for Man-Machine Interaction, RWTH University Aachen, Aachen, Germany. His research interests include the area of hard real-time computations and communication infrastructures for simulation-based control applications in cyber-physical, embedded, and robotic systems.

Linus Atorf received the Diploma degree in physics from RWTH Aachen University, Aachen, Germany, in 2011, where he is currently working toward the Ph.D. degree in electrical engineering from the Institute for Man-Machine Interaction.

His research interests include virtual testbeds, simulation-based optimization, multibody 3-D simulation, intelligent robotic systems, computer vision, and virtual reality.

Juergen Rossmann received the Dipl. Ing. and Ph.D. 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.