Development of Modular Spacecraft with Experimentable Digital Twins

Workshop on Simulation and EGSE for Space Programmes (SESP)
26 - 28 March 2019

ESA-ESTEC, Noordwijk, The Netherlands

Michael Schluse[(1)], Tobias Osterloh[(1)], Jürgen Roßmann[(1)]

[(1)]Institute for Man-Machine Interaction, RWTH Aachen University
Ahornstraße 55, 52074 Aachen, Germany
Email: {schluse,osterloh,rossmann}@mmi.rwth-aachen.de

Abstract – Digital Twins (DT) currently get a lot of attention from various application areas. The DT notion originates from Product Lifecycle Management and is closely related to the Industry 4.0 context. Additionally, DT are frequently regarded from a simulation technology point of view. Therefore, an extended definition of the DT term and a concept for the interplay of simulation technology and DT is needed. This paper examines the combination of DT and simulation technology and focuses on the newly developed Experimentable Digital Twin (EDT) methodology. The key concept of this methodology is to link various EDT to a network of interacting EDT. Virtual Testbeds (VTB) bridge the gap between the EDT networks and simulation technology by applying various simulation algorithms. The feasibility and potential of this approach is demonstrated within various applications from the context of modular spacecraft.

Keywords Digital Twin, Experimentable Digital Twin, Virtual Testbed, Simulation Platform, Simulation Technology

INTRODUCTION

Digital Twins (DT) are revolutionizing our view of systems. They are result and basis of digitization at the same time, because they are created by transforming physical assets into digital representations, and, on this basis, enable the digitization of processes, products and business models. In fact, so far, various definitions of the DT term can be found. It may be e.g. the virtual part of a Cyber-Physical System (CPS), an ultra-realistic system model, or a one-to-one replica of a CPS. In general, a DT describes its physical asset, predicts/controls its behaviour, and integrates it into surrounding systems/processes. DT are based on Digital Models developed during the engineering process and are continuously updated during operation (see Fig. 1). From the information technology point of view, the DT communicates with its asset during operation via internal and with its environment via external communication interfaces. In this way, the DT provides both a semantic entity and a structuring element for development and digitization of assets and workflows. In operation, the DT is the mediator both between assets and especially between humans and asset [11].

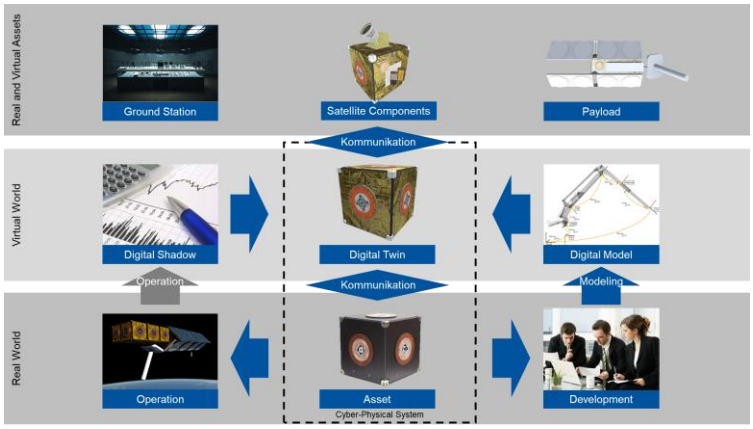

Fig. 1. The basic idea of the Digital Twin (based on [2])

From the perspective of simulation technology, the prediction of its behaviour is paramount. By applying simulation technology the DT becomes experimentable, becomes an **Experimentable Digital Twin** (EDT). EDTs integrate the different aspects on DTs mentioned above used for e.g. modelling, virtual AIT and simulation on system level. EDTs hierarchically model the artefacts making up the segments of a space system on different granularity levels by following the physical architecture of the space system and by considering the physical interfaces of each artefact, see also [3]. The modelling process results in an **EDT Scenario**, a network of EDTs (including all space systems involved as well as their environment) interacting with each other on an explicit (explicitly modelled interconnections via e.g. mechanical, thermal, electrical and data interfaces) or implicit level (e.g. collisions, targets of optical sensors) [3]. When staying on the EDT modelling level, the engineer can concentrate on the general architectural and functional aspects of the system.

It is the task of so-called **Virtual Testbeds** (VTB) to execute EDT Scenarios, i.e. to bring EDT to life by applying e.g. simulation technology. Therefore, each EDT contains all the information necessary for different simulation algorithms like rigid/soft body dynamics, contact dynamics, sensor/actuator dynamics, orbital dynamics, solar radiance, data processing algorithms, different types of networks etc. To prepare the simulation, a VTB analyses the given network of EDTs, schedules the execution of the different simulation algorithms/simulators and executes the resulting experimentable model. (If possible) interactive simulations, extensive simulation logs and various tools for online/offline analysis give a deep insight into the system behaviour. To realize VTBs, a reference architecture has been developed consisting of a VTB Kernel (for real-time model management and communication), VTB Services (e.g. simulation algorithms and data interfaces), and VTB Run-time Environments to integrate VTB in execution environments like Desktop, VR/AR, Cave, real-time system, App, cloud service or co-simulation setups.

Adding EDT-based methods and an EDT-based development workflow to the general EDT concept results in the **EDT methodology** providing a consistent set of methods, tools and workflows used throughout the life-cycle of systems. In this paper, the EDT methodology is applied to the development, virtual AIT and validation of modular spacecraft based on the **iBOSS concept** [4, 5]. The components of modular iBOSS spacecraft (so called iSATs) are integrated into standardized building blocks (iBLOCKs, intelligent Building Blocks) that are interconnected via standardized interfaces (iSSIs, intelligent Space System Interfaces, i.e. modular 4-in-1 interfaces (mechanical, electrical, thermal and data)). The major goals of the iBOSS concept are to ease and speed-up the development of spacecraft (from a technological and development process point-of-view), while at the same time reducing the costs of the spacecraft and increasing the flexibility of the overall development process (e.g. late integration). On the other hand, this modular approach increases the complexity of the spacecraft itself as well as the complexity of the development process. This challenge is targeted by the **iBOSS development approach**, where EDTs act as the central structuring element of the development process.

The rest of the paper is organized as follows. The next chapter summarizes the state of the art in the field of simulation technology for spacecraft. The following three chapters introduce the EDT concept ending up with different perspectives on the same EDT, VTBs acting as run-time environments for EDTs and EDT-based development methods and workflows for the development of modular spacecraft. At the end, this paper illustrates the feasibility of the presented approach investigating different aspects developed and analysed using the EDT methodology.

STATE-OF-THE-ART

Capturing the state-of-the art of simulation technology is of multidimensional nature, as it has to be differed between generic simulation approaches, domain specific simulation tools and algorithms, complete simulation frameworks/platforms and finally standards that are used and applied within these frameworks. **Simulation approaches** can be classified in equation-based simulation, signal-oriented simulations and discrete-event simulations. Equation-based simulation explicitly model the differential algebraic equations (DAE) and solve them based on numerical methods, like in Modelica [6] or Matlab [7]. Typically, these approaches are the foundation of most simulation algorithms. Signal-oriented simulations are defined by block diagrams describing the signal flow between the blocks, like Simulink [7]. Discrete-event simulations (DES) constitute models of discrete processes and are based on discrete events instead of numerical methods [8]. Based on these approaches, almost every engineering domain has its own **simulation tools**. Some of the most well-known simulation domains are structural dynamics, commonly covered by FEA simulations (e.g. [9]), rigid body dynamics covered by simulation tools like MSC Adams [10], SimMechanics [12] or modern game engines [11]. Simulations in the field of robotics either are supplied directly by the manufactures themselves, or are based on systems like the ROS-based simulation system GAZEBO [13] or V-REP [14].

Taking a closer look at currently used **simulation platforms in space** (e.g. BASILES [15], K2 [16], SimSat [17], EuroSim [18]) reveals that all simulators are based on the same principles. First of all a model has to be defined and developed. This includes the code generation for the model. Based on the model, the simulator itself is defined by selecting the required models and libraries. Based on this, the simulator is compiled, i.e. an executable program is generated. Finally, the simulator is ready to use, and simulations can be executed. As a consequence, for each variant or scenario to be investigated a specific simulator has to be compiled. Typically, these simulation platforms apply certain **simulation standards**, e.g. to achieve model compatibility between simulation platforms or different applications. The most relevant standard for space systems is SMP2 (Simulation Model Portability v1.2) [19]. The FMI standard (Functional Mock-up Interface) [20] is an additional important step towards simulation platform interoperability. Especially, when it comes to combining different subsystems with limited numbers of defined inputs and outputs the FMI standard is quite promising.

In summary, a lot of domain specific tools and interfaces are available to solve various problems. Nevertheless, simulating complex systems like modular satellites in their entirety is still a big challenge using the available simulation platforms. Here, simulation models are grouped according to the simulation domains and are statically coupled to a simulator, see Fig. 7 left. This is a time-consuming process, highly suited to classical monolithic satellite systems. The result is a lack of flexibility in the configuration process of the system to be simulated. In order to investigate a variety of modular satellite layouts efficiently, a new approach is indispensable.

THE EXPERIMENTABLE DIGITAL TWIN CONCEPT

The EDT methodology aims for closing this gap. Based on the term „Asset" this section gives a definition of the term "Experimentable Digital Twin" as a concretization of the generic "Digital Twin" concept. Operational scenarios are specified by networks of interacting EDTs in so-called "EDT Scenarios".

Assets of Modular Spacecraft

An asset is an "entity that has a perceived or actual value for an organization and is owned or individually managed by the organization" [21]. Material and immaterial assets are distinguished. Material assets such as products and production facilities (or parts of them) as well as human workers are part of the physical world. Immaterial assets such as data, data models and simulation models are part of the virtual world. Concrete examples of assets with respect to modular iBOSS-based spacecraft are therefore iBLOCKs, iSSIs and iSATs as well as components integrated into or attached to iBLOCKs like antennas or solar panels (see Fig. 2). From the point of view of the DT, the asset can also be stated as "Real Twin".

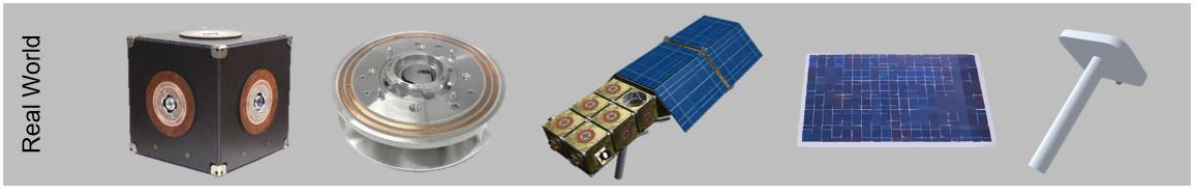

Fig. 2. Some assets with respect to the iBOSS concept (f.l.t.r.): iBLOCK, iSSI, iSAT, solar panel, antenna

From Digital Twins to Experimentable Digital Twins

Colloquially formulated a DT is the virtual counterpart to a real asset. However, the meaning of the word combination "digital twin" is not clearly defined. Generally, a twin is the same thing twice, at least with regard to the characteristics considered in the respective situation. The same applies to the DT, too. An entity is considered twice, namely an asset once in the real world and once in the virtual world. The DT denotes the digital half of the twins (or of this pair) and thus provides a vivid term for the virtual artefacts that are created during the digitization of real assets.

The use of the DT term in relation to technical systems goes back to Grieves, who mentions this term in 2003 in the context of "Product Lifecycle Management" [22]. In 2010, NASA revisits the concept for aerospace and hereby refers to an "ultra-realistic simulation" [23]. Subsequently, the term was investigated from many perspectives, e.g. from the perspective of simulation, CPS or smart production. In 2018, Gartner classifies the DT as part of the digitized ecosystem being one of the five major technological trends of the next decade and predicts the technology to reach the "productivity plateau" in 5-10 years [24]. The lack of a unique definition of this term becomes evident when looking at [21], which calls the concept of DT on the one hand "virtual digital representation of physical assets" and on the other side "simulation model" (see right side of Fig. 3). In the following, we will use the first DT definition also referred to as Asset Administration Shell in the Industry 4.0 context or simplified as Data Processing System (DPS) (see Fig. 4).

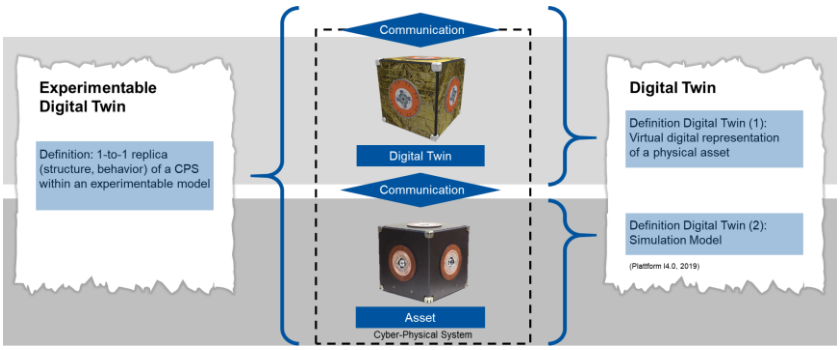

Fig. 3. The Definition of the Digital Twin (excerpt of Fig. 1)

Five components are currently considered as the "superset" of a DT. Originally, the term DT was solely used to refer to the **Digital Model** (geometry, structure, ...) of a physical asset that (often through simulation) allows the **prediction of its behaviour**. Within a short time, additional tasks were assigned to the DT. It stores the **operating data** generated during the operation of physical assets (sometimes called "Digital Shadow") and describes the **functions and services** and **communication capabilities** (interfaces to others, ...) provided by the asset. Most current realizations of DT arise from the Industry 4.0 context and focus on one of those components like using the DT as node in the Internet of Things or as a simulation model (see above). Compared to this, an **Experimentable Digital Twin** considers the CPS in its entirety, including its internal structure, all physical interactions as well as internal and external communication interfaces

(see also Fig. 3). An EDT joins all the individual components of a DT mentioned above. In combination with the definition of the experimental model [25], this leads to the following definition of the EDT: *An Experimentable Digital Twin (EDT) is a virtual digital 1-to-1 replica of a CPS including its structures, models and data, interfaces and communication capabilities as well as its dynamic processes in an experimentable model.*

Interaction of Assets and Experimentable Digital Twins

Looking at the interaction between assets and DT with each other and with their environment, three interaction types can be distinguished (see Fig. 3). Physical assets interact with each other on the **physical level**. An example for this is the interaction between a robot and the object to be handled. This interaction leads to energy and material flow according to different disciplines such as mechanical and electrical engineering, hydraulics or thermodynamics. As part of **internal communication**, the DT exchanges data and commands with its asset via individual, device or software-specific interfaces. In the field of **external communication**, DT communicate with each other via suitable communication protocols. The interactions of EDTs (consisting of both, asset and DT) on these thee levels are described in an abstract form by interaction points, so-called **ports**, which in their entirety form the **interface** of an EDT. Concrete applications are modeled by connecting EDTs. The result are networks of interacting EDTs, so called **EDT Scenarios** (see Fig. 4).

Basic Structure of Experimentable Digital Twins and EDT Scenarios

A systematic view on EDTs is given in Fig. 4 (left): An EDT combines **EDT Components**, i.e. a simulated data processing system (DPS, also referred to as DT, see Fig. 3) with a simulated asset consisting of sensors, actuators and the internal process itself (the simulated human-machine interface [26] is left out here). The current state of the EDT is represented by its state vector $\underline{s}_{edt}(t) = [\underline{s}_{asset}(t), \underline{s}_{dps}(t)]$. The DPS usually processes sensor data and/or commands the asset, e.g. mapping or motion planning algorithms. It may hold its own EDT Scenario to e.g. manage generated maps or environment models for motion planning. Each EDT (and in turn each EDT Component) has ports, i.e. inputs $\underline{u}_{edt}(t)$ and outputs $\underline{y}_{edt}(t)$ to interact on the three different levels (physical, internal/external communication) with the EDT/EDT Components in their environment. All EDT Components as well as the EDT itself communicate using a simulated communication infrastructure $\underline{u}(t) = K_{edt}(\underline{y}(t), t)$, which resembles the real communication infrastructure of its real world counterpart.

The EDT state and the output depend on the EDT input and is calculated by appropriate simulation and data processing algorithms Γ_i, Φ_i (i represents a simulation algorithm). The simulation $\underline{s}_{edt}(t) = \Gamma(\underline{s}_{edt,0}, \underline{u}_{edt}(t), t)$ starts with an initial state $\underline{s}_{edt,0} := \underline{s}_{edt}(0)$, the output is calculated via $\underline{y}_{edt}(t) = \Phi(\underline{s}_{edt}(t), \underline{u}_{edt}(t), t)$ and combines the simulation state $\underline{s}_{edt}(t)$ of all components of the EDT as well as its input $\underline{u}_{edt}(t)$. Γ and Φ combine the individual Γ_i and Φ_i, and contain all algorithms necessary to carry out the coupled simulation. These algorithms are provided by the EDT Components (e.g. the DPS) or by the 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).

The right side of Fig. 4 illustrates a simplified **EDT Scenario** made up by interconnecting EDTs. Each EDT of the modular satellite represents one iBLOCK or the payload antenna, each EDT of the servicer satellite represents one satellite component, the EDTs of the space environment represent the sun and the earth. The network of connected EDTs hierarchically form the EDT of the iSAT, the servicer satellite, and the space environment respectively. Only "man-made" **explicit interactions** like mechanical, electrical, thermodynamic and data connections have to be modelled explicitly, which keeps the EDT Scenario understandable and manageable [3]. "Natural" **implicit interactions** like the sun illuminating the satellite or the physical interaction between manipulator and modular satellite can be omitted. It is the task of the simulation algorithms to detect and process those interactions automatically [3].

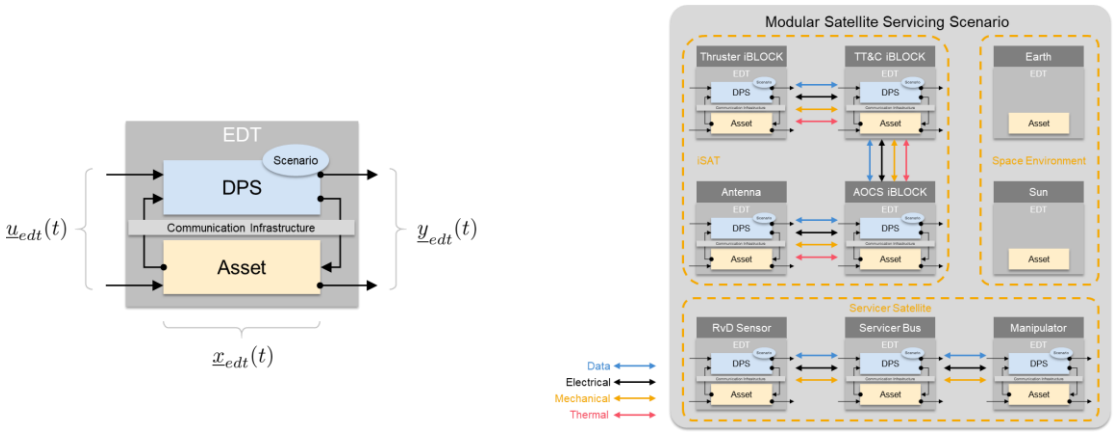

Fig. 4. Simplified basic structure of an EDT, example EDT Scenario

Different Perspectives on the same EDT and EDT Scenarios

In all development steps, EDTs, EDT Scenarios and their execution in VTBs (see next chapter) are always considered from all relevant **perspectives**, which may be an engineering discipline or a simulation domain. This way, all users – from development engineers over project managers to customers and end-users – are able to obtain the necessary overall view of the entire system under consideration of the development results of all the perspectives involved. This is the only way to answer questions such as "How does the energy consumption of the servicer satellite change when moving the robot based on the RvD sensor data taking into account orbital mechanics and solar energy production?". Today, specific simulation models are modelled manually for each perspective and analyzed independently from each other. Using interconnected networks of EDTs in specific EDT Scenarios in turn leads to experimentable and coupled models. Fig. 5 shows examples of individual perspectives that are united by the EDT Scenarios.

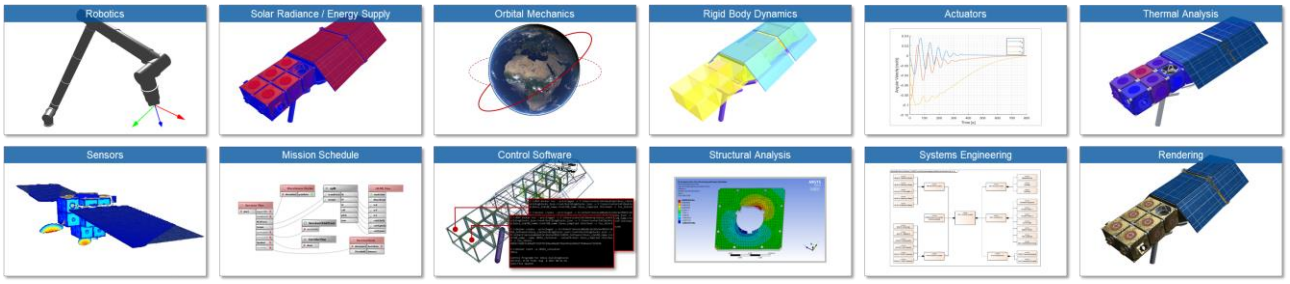

Fig. 5. Relevant perspectives on EDTs and EDT Scenarios when developing modular satellites

VIRTUAL TESTBEDS: RUN-TIME ENVIRONMENTS FOR EXPERIMENTABLE DIGITAL TWINS

It's the task of so called Virtual Testbeds (VTBs) to bring EDTs to life by carrying out the simulation and data processing algorithms involved – and interfacing real hardware in hybrid (partly virtual, partly real) scenarios.

Basic Structure of Virtual Testbeds and the Simulation Database

On its left side, Fig. 6 illustrates the basic structure of VTBs consisting of a **VTB Kernel**, a Micro-Kernel providing real-time data management and VTB internal communication means (used for communication inside the Kernel, between Kernel and Services, and inside/between Services). The Virtual Testbed Active Simulation Database (VSD) provides data management services, which manage EDT Scenarios consisting of EDTs and EDT Components as well as their interactions (see EDT namespace on the right side of Fig. 6). In addition to that, the VSD holds all the necessary simulation models and data processing algorithms serving as the basis for various simulation algorithms (e.g. kinematics, rigid body and orbit dynamics, thermodynamics or sensors). Simulation algorithm specific models are derived via functional mappings between generic representations (models, ports, interactions) and the individual simulation models/algorithms. For the management of those models, the VSD can be extended by appropriate data models (the VSD is able to store any data structure that can be modeled by UML class diagrams, because it is based on the same meta model).

The VTB Kernel is the basis for the implementation/integration of various simulation and data processing algorithms as well as communication and model exchange interfaces, all of them provided as Micro-Services, called **VTB Services**. By selecting the necessary VTB Services, different VTB configurations can be set up (e.g. for high-precision simulation, for real-time simulation or for real-time control of physical assets), which can be combined with different **VTB Run-time-Environments** providing interfaces to different hardware setups (e.g. on desktop computers, as cloud services, for multi-screen visualization or on mobile devices).

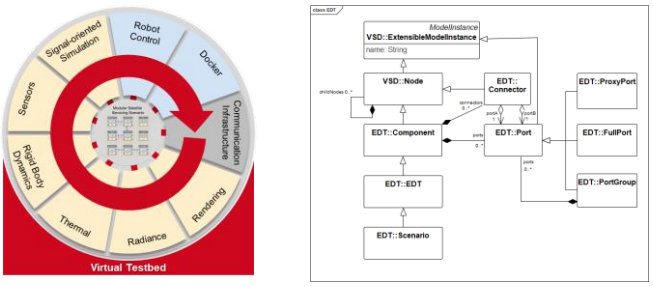

Fig. 6: Simplified structure of VTB (left), simplified data model of the VSD (right)

VTB Kernel, VTB Services and VTB Run-time Environments are fully independent from the EDT Scenario to be simulated. Compared to classical approaches, the EDT methodology sets up networks of interacting EDTs replicating real-world scenarios, and simulates these scenarios in a VTB automatically organizing all the "sub-simulators" necessary for this. The key of this new methodology is a systematic decoupling of EDTs and simulation algorithms, see Fig. 7, right.

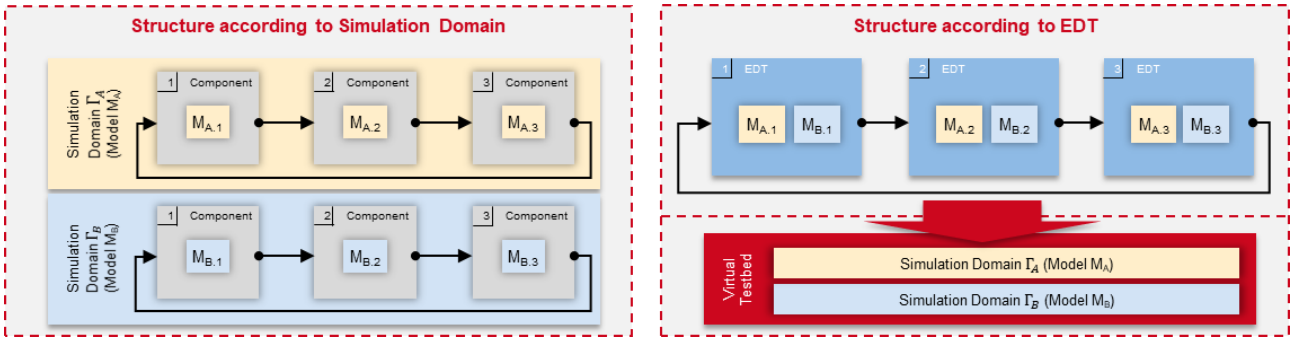

Fig. 7. Comparison of state-of-the-art simulation platforms and the EDT methodology

Simulation and Data Processing Algorithms

The basis for the execution (resp. simulation) of EDT Scenarios is a time-discrete scheduler which is able to carried out coupled simulations of continuous (1, index "c") and time-discrete (2, index "d") simulation algorithms:

$$\begin{aligned} \dot{\underline{s}}_c(t) &= f_c\left(\underline{s}_c(t), \underline{u}(t), t\right) \\ \underline{0} &= h_c\left(\underline{s}_c(t), \underline{u}(t), t\right) \\ \underline{y}(t) &= g\left(\underline{s}_c(t), \underline{u}(t), t\right) \\ t_{R,next}(t) &= T\left(\underline{s}_c(t), \underline{u}(t), t\right) \end{aligned} \quad (1)$$

$$\begin{aligned} \underline{s}_d(t^*) &= f_d\left(\underline{s}_d(t), \underline{u}(t), t\right) \\ \underline{y}(t) &= g\left(\underline{s}_d(t), \underline{u}(t), t\right) \\ t_{R,next}(t) &= T\left(\underline{s}_d(t), \underline{u}(t), t\right) \end{aligned} \quad (2)$$

t^* denotes the next event time in super-dense time representation [27]. The integration of simulation and data processing algorithms is inspired by the FMI standard [20]. VTB Services offer so called **simulation entities** (see Fig. 8) which provide the transition functions f_c and f_d, secondary conditions h_c and output functions g as well as T to calculate the time of the next event or the following time step. Those simulation entities can be generic in the form that they model some generic theory (like kinematics, rigid body dynamics, sensor dynamics) or EDT Component specific in the form that they model some component specific data processing (like sensor data processing, robot control) or simulation (some component specific behaviour) algorithm. For each simulation algorithm Γ_i, Φ_i a set of functions $f_{c,i}$, $f_{d,i}$, $h_{c,i}$, g_i and T_i has to be provided by the corresponding VSD Service. The VSD Service itself combines model information M and the actual implementation of the simulation algorithm. In the following, the mapping of simulation algorithms to corresponding simulation entities is illustrated.

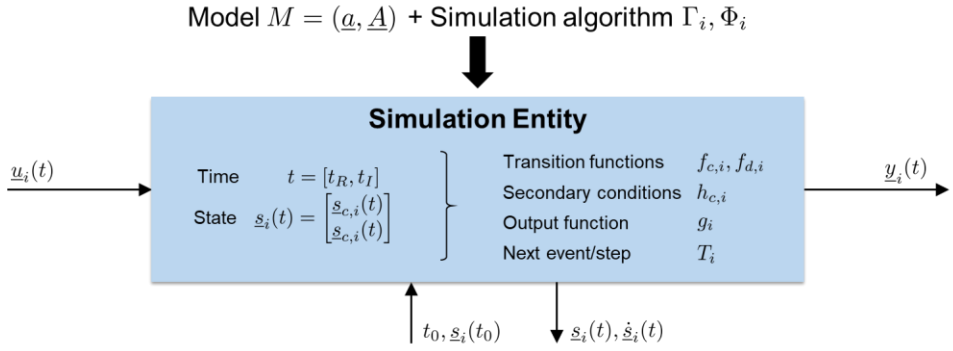

Fig. 8. Basic structure of a simulation entity

Rigid Body Dynamics: Assuming a maximal coordinate approach towards rigid body dynamics simulation [28], the state of rigid body dynamics system $\underline{s}_{RBD}$ is given by the positions, orientations and velocities of all rigid bodies. The transition function f_{RBD} of rigid body dynamics simulations is based on the principle of momentum conservation. The input function $\underline{u}_{RBD}(t)$ is given by actuator forces/torques $\underline{f}_{act}$ and disturbance forces/torques $\underline{f}_{dis}$:

$$f_{c,RBD}\left(\underline{s}_{c,RBD}(t), \underline{u}_{RBD}(t), t\right) = \begin{pmatrix} \underline{M}^{-1} \cdot \left[\underline{f}_{ext} + \underline{f}_{act} + \underline{f}_{dis} + \underline{J}^T \cdot \underline{\lambda}\right] \\ \underline{v} \end{pmatrix} \quad (3)$$

The secondary conditions $h_{c,RBD}$ are given by constraints between rigid bodies, enforced by the constraint forces $\underline{\lambda}$. The constraint forces are calculated according to the $JM^{-1}J^T$-approach [28]:

$$h_{c,RBD}\left(\underline{s}_{c,RBD}(t), \underline{u}_{RBD}(t), t\right) = \underline{J} \cdot \underline{M}^{-1}\underline{J}^T \cdot \underline{\lambda} + \underline{J} \cdot \underline{M}^{-1} \cdot \left(\underline{f}_{ext} + \underline{f}_{act} + \underline{f}_{dis}\right) + \underline{c} = \underline{0} \quad (4)$$

This mathematical structure forms the base for the execution of arbitrary EDTs of a modular satellite seen from the perspective of rigid body dynamics. It is important to notice, that the EDT of a modular satellite system is composed from the EDTs of each of the components or sub-systems. The interconnections between the sub-systems are based on the standardized iSSI. Fig. 9 shows the basic structure of an EDT of a modular satellite.

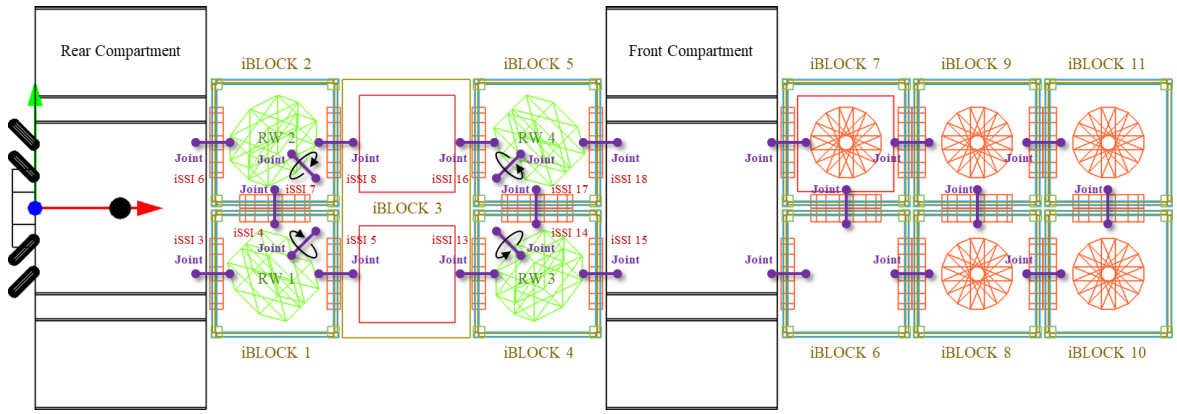

Fig. 9. EDT of a modular satellite, seen from the perspective of rigid body dynamics

Orbit Dynamics: In order to incorporate orbital mechanics and orbital perturbations to the satellite dynamics, additional simulation entities are used to describe and calculate orbital forces and disturbances like gravitation, atmospheric drag, gravity gradient or solar pressure [29]. Based on the current state of the each EDT (i.e. positions and velocities as input for the simulation entities), the simulation entities calculate the external forces or disturbance forces (i.e. forces and torques as output of the simulation entities), that are used as input for the rigid body dynamics simulation. Other simulation entities provide complete simulation algorithms/models for the magnetic field of the earth. The output of these simulation entities is used to provide EDTs for subsystems like magnetic torquers or magnetic field sensors.

Thermodynamics: The thermal perspective of the EDT Scenario is realized via FMU-based simulation entities. The FMUs are connected according to the physical rate of heat flow. On EDT level, a thermal connector linking two thermal ports models this heat flow. This results in a thermal network of iBLOCKs. The inputs and outputs of the simulation entities are the temperatures and heat flow rates. A simplified version of the simulation entity for the thermal simulation, can be described by the partial differential equation $f_{c,thermal}$.

$$f_{c,thermal}\left(\underline{s}_{c,thermal}(t), \underline{u}_{thermal}(t), t\right) = \frac{\partial Q}{\partial t} = -k \cdot A \cdot \frac{\partial T}{\partial x} \tag{5}$$

Additional simulation entities determine the direct, albedo and infrared radiation to the satellite surface accompany the FMU simulation entities and thereby enable to execute the thermal perspective of the satellite. Once again, the EDTs of each individual satellite component aggregate the surface of the satellite. This way, effects like self-shadowing are inherently reflected. As the simulation of radiance does not require dynamic inputs, it just evaluates the output function $g_{radiance}$ by determining the lit surfaces $A_{solar}, A_{albedo}, A_{infrared}$ (which are dependent on the state $\underline{s}_c(t)$ of the EDTs).

$$g_{radiance}\left(\underline{s}_c(t), \underline{u}(t), t\right) = S \cdot A_{solar}\left(\underline{s}_c(t)\right) + A \cdot A_{albedo}\left(\underline{s}_c(t)\right) + I \cdot A_{infrared}\left(\underline{s}_c(t)\right) \tag{6}$$

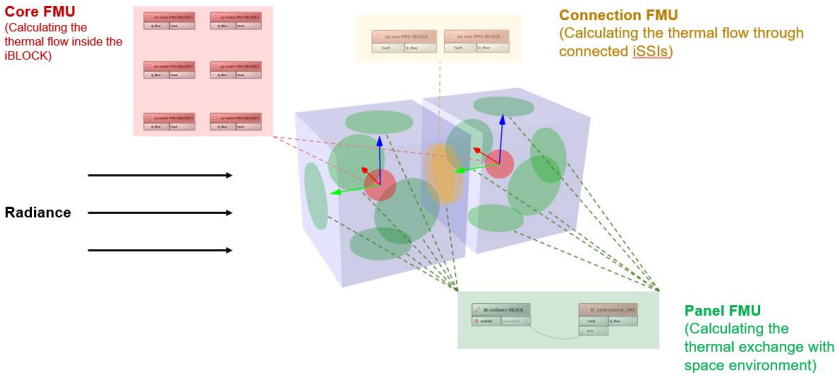

Fig. 10. Schematic representation of the thermal perspective of an EDT of two modular building blocks

Optical Sensors: The simulation of optical sensors like cameras or laser scanners is provided by another important set of simulation entities. Input to these types of simulation algorithms is a subset of the complete state of all EDTs (i.e. poses and material properties of all assets), as well as the pose of the sensor. Based on this state, the sensor simulation entities evaluate all surrounding EDTs and combine them with the EDT of the sensor itself. Exemplarily, the EDT of a camera covers information and simulation models for the optics, discretization process and resolution and generates a simulated picture (see Fig. 11), whereas the EDT of a laser scanner is described e.g. by scan patterns, opening angles and frequencies [32]. Assuming no sensor dynamics, the sensor simulation entity can be interpreted as a static evaluation of the state of all EDTs $\underline{s}_c(t)$ (in more detail, the poses and geometries of all assets are evaluated). Therefore, only the output function $g_{camera}\left(\underline{s}_c(t), \underline{u}(t), t\right)$ needs to be evaluated. The output function transforms all information from the state $\underline{s}_c(t)$, to the sensor frame, e.g. by means of extrinsic and intrinsic camera parameters

$$g_{camera}\left(\underline{s}_c(t), \underline{u}(t), t\right) = g_{camera}\left(\underline{s}_c(t)\right) = T_{intrinsic} \cdot T_{extrinsic} \cdot \underline{s}_c(t) \tag{7}$$

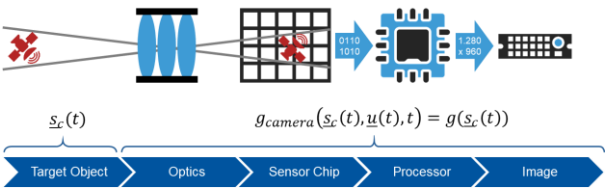

Fig. 11. Functionality of the simulation entity for optical sensors

Data Processing Algorithms: The modularity of the electronics and the operational software is a key concept in the iBOSS concept. EDTs of the operational software executed on each On-board Computer (OBC) accurately describe the distributed computation infrastructure. The network of EDTs intrinsically models the real network topology of the iSAT. Thereby, it is possible to investigate routing protocols or network failures effortlessly. To offer a platform independent and flexible execution of the operational software, each OBC is represented by a Docker container [30], which is part of the EDT model. Thereby, a strict logical separation between the software running on each OBC is achieved and additional functionalities like processor and memory load analysis provided by the Docker framework can be used.

The Model Analysis Process to Generate the Executable Model

The EDT Scenario (see e.g. Fig. 4, right) describes the explicit interactions of the EDTs. The EDTs themselves hold all models necessary to simulate the EDT Scenario incorporating selected perspectives (see Fig. 5). The remaining task is to find a transition between the formal specification of the EDT Scenario (see e.g. Fig. 4, right) and a formal specification of an executable model. Fig. 12 illustrates the overall workflow. The important steps here are to assign EDT Components to simulation entities according to the selected perspectives, to merge strongly coupled simulation entities into **simulation tasks** and to timely order the tasks. The result is a **scheduling plan** to be executed by the scheduler.

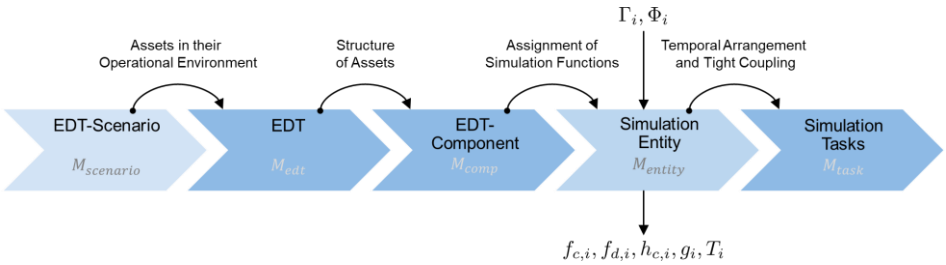

Fig. 12. The process from an EDT Scenario to an executable model consisting of simulation tasks

Fig. 13 visualizes this process for the EDT Scenario above. The tables on the left can be derived from the EDT Scenario fully automatically. Part 1 shows the various components of all EDTs with their explicitly modeled interactions (for more complex systems, the matrix can be significantly larger). Each block on the table represents an EDT Component, i.e. asset (yellow) or DPS (blue). Columns represent perspectives (here simulation algorithms and DPS programs), while rows are EDTs. This example comprises six perspectives, rigid body dynamics, sensor, thermal, radiance, signal-oriented simulation (used to model the DPS of the servicer) and Docker containers (providing implementations of the iSAT DPSs).

When loading the EDT Scenario, the VTB assigns all EDT Components to their perspectives represented by simulation entities provided by VTB Services. The network of EDT Components shown in table (1) is converted into a network of simulation entities as shown in table (2). Possible results are simulation entities bringing together multiple EDT Components or simulation entities responsible for exactly one EDT Component (right table). Here, the simulation entity responsible for rigid body dynamics (the same holds for sensor and radiance simulation) brings together all EDT Components with rigid bodies so that they can interact with each other. At the same time, each simulation entity specifies, which parts of the VSD model act as inputs, outputs and states (to guarantee a consistent and performant data transfer between the different simulation algorithms, the simulation results are routed via the VSD). This way, the VTB detects, that rigid body dynamics, sensor and radiance simulation work on the same geometric model; hence the simulated sensor and radiance data has to reflect the movement of the EDTs. This introduces new interdependencies shown as green arrows. In addition to this, simulation entities are identified, which have to be grouped into one simulation task. Here, all FMUs simulating the thermal behavior have to be solved by one integrator, all Docker containers are managed by one Docker Daemon (both illustrated by dotted gray lines).

After grouping the EDT Components, the resulting network of continuous simulation tasks is analyzed by the VTB Scheduler, leading to the scheduling plan on the right (this simplified example assumes, that all simulation tasks use the same time step). In this step the edges of the network of EDT Components shown in table (2) representing information flow are converted into edges representing the execution order. When analyzing the resulting graph, two cyclic dependencies are detected that have to be resolved by the user. In this case, two edges (the dotted blue ones) are removed. The consequence is, that the AOCS and manipulator sensor data are processed by the AOCS and manipulator DPS with a delay of one time step.

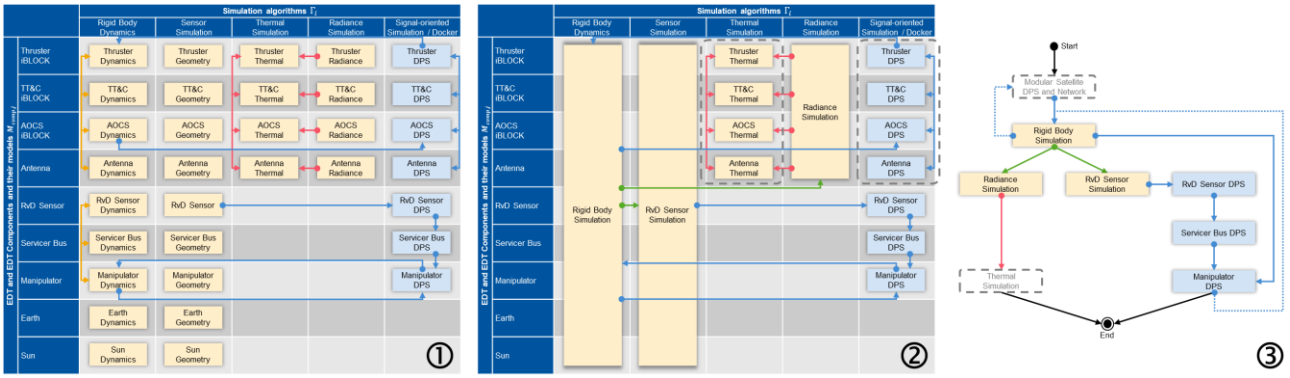

Fig. 13. The model analysis process for the EDT Scenario depicted in Fig. 4

Coupled Simulation of the EDT Scenario

The result of the model analysis process is the scheduling plan consisting of timely ordered simulations tasks. Each simulation tasks consists of one or more simulation entities. They represent simulation or data processing algorithms and are implemented directly inside VTB Services or are provided by external simulators attached to the VTB. The **coupled simulation on the level of simulation tasks** realizes a "loose coupling" of those tasks, i.e. all tasks bring their own integrator. The coupling takes place by the propagation of outputs and the successive use of the same state variables by different tasks (e.g. kinematics, rigid body dynamics, sensor simulation, …). The **coupled simulation on the level of simulation entities** allows for a strong coupling of different simulation algorithms sharing only one common integrator (e.g. integration of rigid body and bulk solid simulation).

USING EXPERIMENTABLE DIGITAL TWINS TO DEVELOP MODULAR SPACECRAFT

The EDT concept and its realization by VTB is the basis for a streamlined workflow to develop modular iBOSS-based spacecraft. This workflow integrates the classical Model-based Systems Engineering (MBSE) process with state-of-the-art simulation approaches into a novel Simulation-based Systems Engineering process based on concepts from Model-driven Engineering (MDE) and Industry 4.0. Fig. 14 illustrates the overall development process. The starting point of the process is a detailed specification of an iBOSS-based satellite systems in SysML providing a formal specification of the idea including customer requirements and possible realizations (components, structure, interfaces, …) [31]. In a second step, these components are specified via EDTs including detailed models for the perspectives considered in the development as shown in Fig. 5. New EDTs are modeled based on their specification as a template and are detailed by model parts originating from different simulation domains. Components of particular interest, such as the sides of the individual iBLOCKs, are detailed in FEM models for structural analysis. Thermal models of the modules and interfaces describe the thermal behavior of the assets. Models particularly derived from CAD models for rigid bodies, sensors and actuators as well as the radiation input are integrated. The integration of the real control software, a complete robot control and the mission sequence control in the EDTs, enable the simulation of entire missions. Through the connection of the EDT ports their physical interactions in the form of suitable passive or active connections as well as their internal and external communication are modeled. All EDTs can be realized completely independent. Only this networking brings them together and is the starting point for the transformation of the EDT Scenario into an executable model.

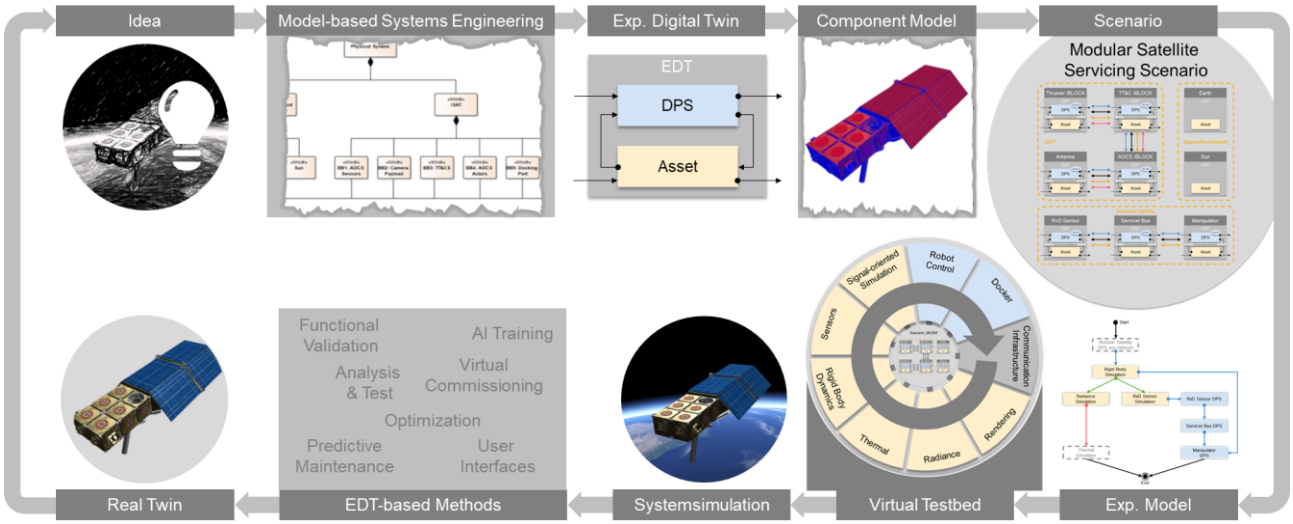

Fig. 14. The EDT-based development process applied to modular spacecraft

The simulations of different operating scenarios are used by various **EDT-based methods**. The simulation of EDT Scenarios allow for in-depth **analysis and test** of various operating scenarios. The iSAT configuration and component parameters can be **optimized** by investigating thousands of iSAT setups and configurations in massive-parallel simulations. Pure virtual **functional validation** of the entire iSAT can be used to check if the chosen setup meets the customer requirements. The logging of the overall system behavior might be used to generate data sets for the **training of AI algorithms**. For **virtual commissioning**, EDTs can be replaced by their Real Twins. In the future, **Predictive Maintenance** applications might compare state variables of the EDT and the Real Twin of the iSAT to detect system anomalies. To command the real iSAT, its EDT can act as a **mediator between ground personnel and satellite** [1]. All this is only possible because the EDT of the iSAT is a 1-to-1 replica of the real iSAT from a structural as well as behavioral point-of-view.

APPLICATIONS

The presented applications are based on an iBOSS-based satellite, the so-called iSAT-1 [5]. Within each of the following sections certain perspectives upon the iSAT-1 are examined in detail. It has to be stressed, that all these applications are based on the same EDTs. No additional configuration of the simulation models or the simulation platform has to be carried out. As the EDTs comprise all the models necessary for all those perspectives, they foster a holistic view on the iSATs with all their facets. It is possible to analyse all perspectives on the EDT Scenario at once, but simulation entities or perspectives can be left out e.g. for performance reasons. This does not affect the EDTs their selves, no extra compilation of simulation models or of the simulation platform is required to execute EDTs with different simulation entities. The result of the application of the EDT methodology for iBOSS is the **Virtual Testbed iBOSS** (VTi). The VTi provides the simulation algorithms and software interfaces to execute arbitrary networks of EDTs under the perspectives of Fig. 5.

Analysis of Detumbling Manoeuvre: To analyse orbital manoeuvres like detumbling, at least the following perspectives need to be considered: rigid body dynamics, orbital dynamics and perturbations, attitude determination sensors, actuator dynamics, and the distributed operational software executing the control algorithms. The EDTs of the satellite's hard- and software components and the EDT of the space environment provide all these perspectives. As already indicated in Fig. 9, the iSAT-1 has four reaction wheels in tetrahedron configuration and two thruster clusters attached to the rear compartment. Additionally, three magnetic torquers in the rear compartment can be used for the attitude control. The detumbling control is based on [33]. The execution of this EDT Scenario offers a deep insight into various aspects of the satellite system (e.g. the actuator torques within the reaction wheels or the control currents within the magnetic torquers, see Fig. 15). Furthermore, the processor and memory loads of each OBC can be examined.

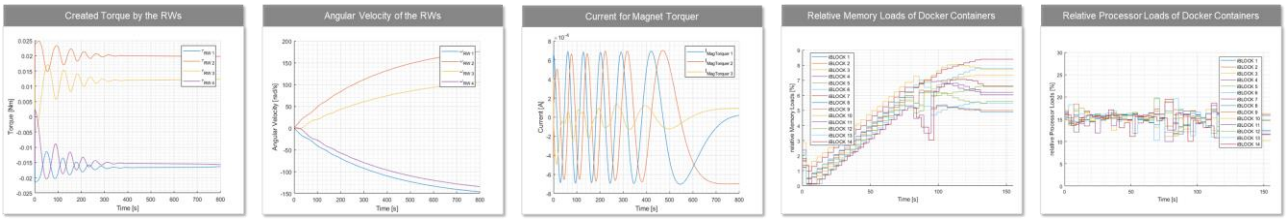

Fig. 15. Analysis of detumbling manoeuvre from the actuator and operational software perspective

Dynamic Interface Loads: Originating from the previous EDT Scenario, dynamic interface loads between the modular building blocks are another crucial point in the layout of modular satellite systems. It has to be ensured that the maximal loads are never exceeded, even that they are never in a critical region close to the limit. Therefore, an EDT Scenario was configured, where all actuators of the iSAT-1 provide full power for a predefined period. This way, the theoretical maximal mechanical loads on the interfaces can be determined. As expected, the mechanical loads far off-centre of the satellite are the most critical ones due to the centrifugal forces. Nevertheless, these forces never exceed a minor fraction of the maximal capable mechanical loads of the interfaces, therefore a safe operation of the satellite is guaranteed.

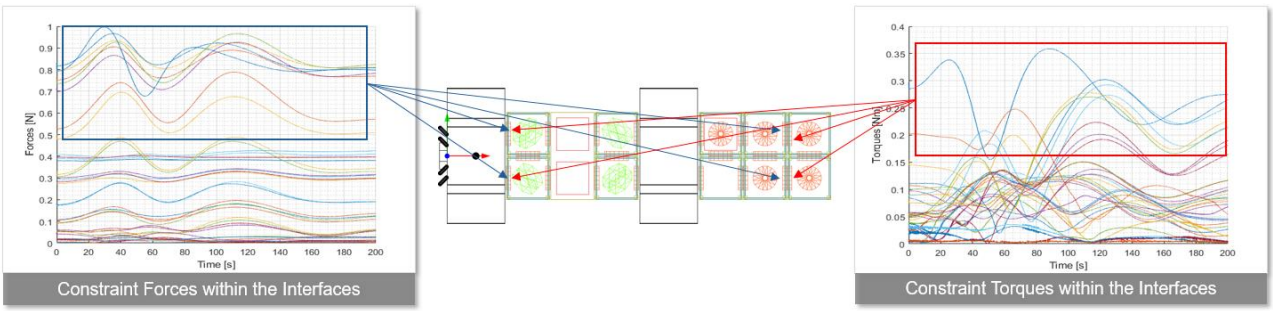

Fig. 16. Analysis of dynamic interface loads with help of the rigid body dynamics perspective

Thermal Layout of the Satellite: Thermal design of a modular satellite is a quite challenging task. All iBLOCKs are connected via a standardized thermal interface. Nevertheless, the internal thermal layout (thermal straps and heat-pipes) of each iBLOCK can be individually adapted to the mission requirements, leading to an almost infinite number of configuration possibilities. Combined with a configurable distribution of computation processes over the iSAT, the thermal heat within each iBLOCK might be highly time-dependent. Thus, the analysis of the EDT Scenario defined above can be extended by a thermal component. Therefore, each EDT of an iBLOCK contains a thermal model. This thermal simulation entity is integrated via FMUs for each iBLOCK. Based on this extended EDT Scenario, a time and iSAT configuration dependent analysis of e.g. the core temperature of each building block can be performed, see Fig. 17.

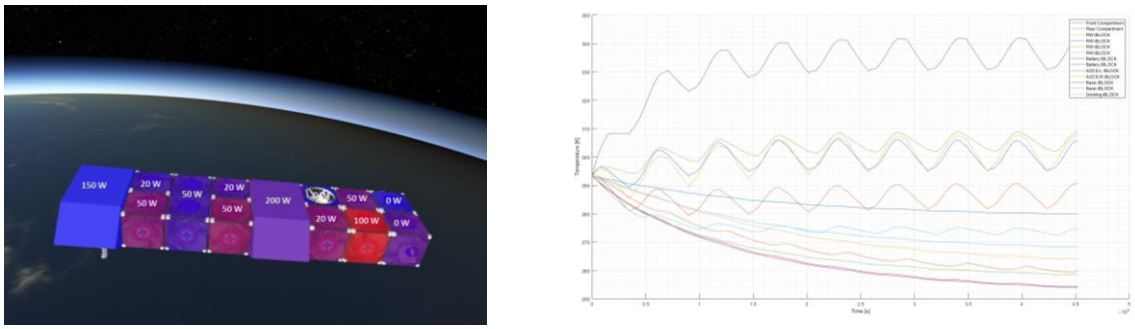

Fig. 17. Thermal perspective on the iSAT-1 (based on works in [34])

Sensor Simulation for Rendezvous & Docking (RvD): When it comes to satellite servicing, a RvD manoeuvre between servicer and target has to be performed. Typically, these manoeuvres are based on and supported by optical sensors. In order to validate such approaches, the aforementioned EDT Scenarios with the iSAT-1 can be extended by an EDT of a servicing satellite equipped with EDTs of various laser scanners. In this application example, the relative localisation between servicer and target is realised by an iterative closed point (ICP) algorithm for laser scanner point cloud matching. In the context of the decision support for sensor equipment design of the servicing satellite, different EDTs of laser scanners can be evaluated regarding their localisation accuracy during rendezvous and docking manoeuvres.

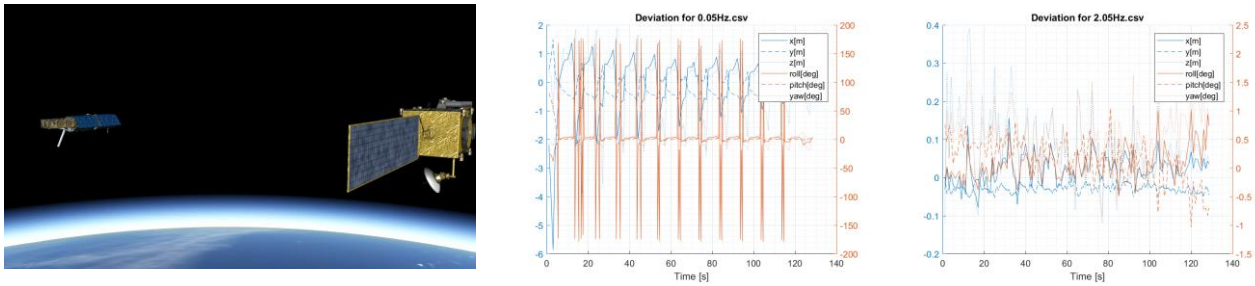

Fig. 18. Sensor perspective on rendezvous and docking (based on works in [32])

Holistic Reconfiguration Scenario: All the previous presented perspectives have been combined into a single holistic reconfiguration scenario, covering all perspectives depicted in Fig. 5. A servicing satellite is equipped with an iBLOCK that should be replaced at the target satellite. Once the phasing manoeuvre is done, a RvD phase is attached. Finally, the actual reconfiguration process is performed, considering the robot kinematics, dynamics and all relevant sensors. Throughout the whole simulation, all states of all EDTs can be analysed, facilitating a deep insight into the physical properties of each component and the state of the operational software or control algorithms. Thereby, a functional verification of the interplay of all subsystems can be performed, providing a benefit for the development of modular satellite systems.

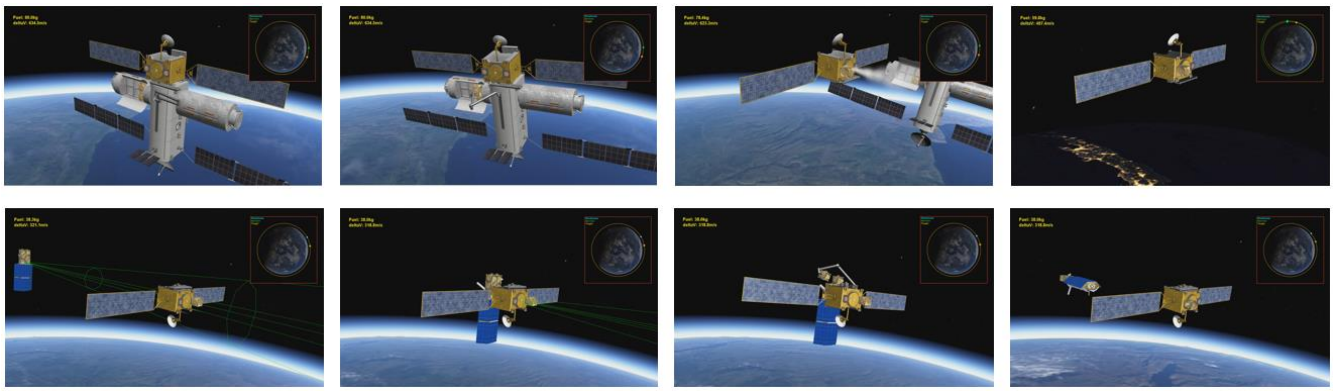

Fig. 19. Comprehensive perspective on a reconfiguration scenario (based on works in [34])

CONCLUSION

In this paper, we introduced the methodology of Experimentable Digital Twins and its use for the development of modular spacecraft. The EDT methodology provides the technical and organisational basis for a comprehensive and integrated Simulation-based Systems Engineering approach for the iBOSS concept including all methods, tools and processes needed. In this context, EDTs provide a structuring element and a semantic entity to bring together developers from various domains and other stakeholders – all of them organized in distributed and interdisciplinary teams. Therefore, EDTs integrate all the information available and necessary throughout the life-cycle (digital models, communication, services, operating data, behaviour). The Virtual Testbed iBOSS (VTi) provides the necessary infrastructure to specify iSATs, to model EDTs of iBLOCKs, to analyse iSATs in coupled simulations in various operating scenarios incorporating different perspectives at the same time and to apply various EDT-based methods like optimization, validation, commissioning and training. All these methods are accessible to different stakeholders – not only simulation experts – and can be used in different VTi setups/configurations on different platforms.

As illustrated in Fig. 20, EDTs are the central structuring element and semantic entity in the iBOSS life-cycle. They are organized in a central EDT repository providing all building blocks necessary to configure new iSATs. In the future, all iBOSS-specific developments will benefit from the current EDT status and result in new or updated EDT. However, iBOSS is only one application field for EDTs. EDTs are used in various areas e.g. for industrial production plants, for autonomous cars, for construction, forestry machines as well as for cities or buildings.

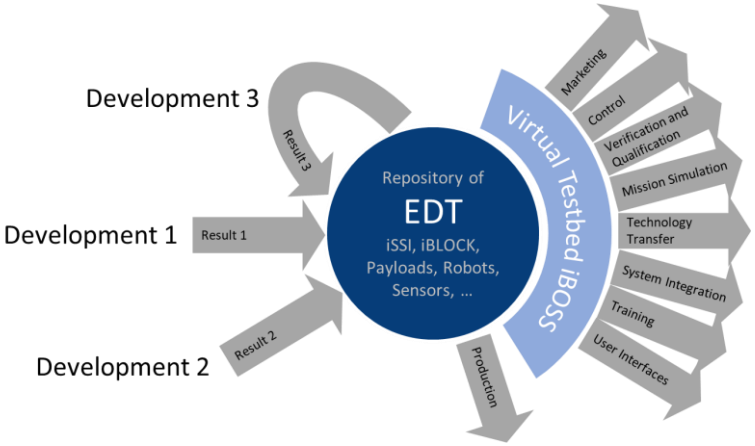

Fig. 20. EDTs as semantic entity and structuring element for development and operation of modular spacecraft

What are the consequences for the development of components, systems, or systems of systems? The answer is threefold [26]: First, **use EDTs for the development of future systems**. Providing EDTs and using simulation should become as natural as developing the hardware and/or software itself (e.g. during the MBSE process, for development, characterization, verification, validation, and optimization). 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. Third, **continue with the development of the overall approach**. Today, we can provide a reference implementation, which serves as the basis for the Virtual Testbed iBOSS. Nevertheless, there is still a lot of work to do e.g. concerning the development of simulation algorithms (such as flexible or soft bodies), the integration of simulation standards like SMP2, for the integration and (weak/strong) coupling of different simulation algorithms, to integrate VTBs in development infrastructures, and so on.

ACKNOWLEDGEMENTS

This work is part of the project “iBOSS-3”, supported by the German Aerospace Center (DLR) with funds of the German Federal Ministry of Economics and Technology (BMWi), support code 50 RA 1203.

We like to thank our colleagues for providing some of the figures and charts presented within this paper. Thanks to Ulrich Dahmen for providing Fig. 5 and Fig. 9, as well as the fundamentals of Systems Engineering. Thanks to Arthur Wahl for providing the thermal analysis used in the applications section. Thanks to Jörn Thieling for providing the sensor-based RvD scenario. Thanks to André Kupetz und Gregor Jochmann from RIF e.V. for providing the Reconfiguration Scenario used in the applications section.

REFERENCES

[1] T. Cichon, J. Roßmann (2018). Digital Twins: Assisting and Supporting Cooperation in Human-Robot Teams. ICARCV, Singapur

[2] T. Bauernhansl, J. Krüger, G. Reinhart, G. Schuh (2016) WGP-Standpunkt Industrie 4.0. Wissenschaftliche Gesellschaft für Produktionstechnik WGP e.V.

[3] U. Dahmen, T. Osterloh, J. Roßmann, Development of a Modular Configuration Framework for Digital Twins in Virtual Testbeds, ASIM Workshop, Arbeitsgemeinschaft Simulation, 2019, ISBN Online 978-3-901608-06-3

[4] J. Weise, K. Bries, A. Adomeit, H.-G. Reimerdes, M. Göller, R. Dillmann (2012). An Intelligent Building Blocks Concept for On-Orbit-Satellite Servcing. i-SAIRAS, Turin

[5] T. Osterloh, U. Dahmen, J. Roßmann (2018). Full lifecycle support for modular satellite systems provided by comprehensive Virtual Testbeds. i-SAIRAS, Madrid

[6] P. Fritzson (2003): Principles of object-oriented modeling and simulation with modelica 2.1., Wiley: 2003

[7] MATLAB and Simulink (2019). [Online]. Available: https://de.mathworks.com/products/matlab.html and www.mathworks.de/products/simulink/

[8] J. Banks (2010): Discrete-Event System Simulation, Prentice-Hall

[9] COMSOL (2019). [Online]. www.comsol.com

[10] Adams (2019) [Online] http://www.mscsoftware.com/de/product/adams

[11] The Unreal game engine (2019). [Online]. www.unrealengine.com

[12] Simmechanics (2019). [Online]. www.mathworks.de/products/simmechanics/

[13] Open Source Robotics Foundation, Gazebosim (2019). [Online]. gazebosim.org

[14] Coppelia Robotics Software, V-rep (2019). [Online]. www.coppeliarobotics.com

[15] S. Salas Solano, J. Marigo, F. Manon, A. Strzepek (2014). BASILES: A common simulation platform to promote model and simulation reuse. SpaceOps Conferences, 5-9 May 2014, Pasadena, CA

[16] L. Cohen, A. Mollier, S Vinay (2015). SMP2 Modelling using the K2 Simulation infrastructure. SESP 2015, Noordwijk, Netherlands

[17] SIMSAT Simulators, Terma. (2019). [Online]. https://www.terma.com/media/148537/simsat_simulators.pdf

[18] Eurosim (2019). [Online]. http://www.eurosim.nl/about/index.shtml

[19] Simulation model portability 2.0 handbook, 2005, EGOS-SIM-GEN-TN-0099 , issue 1, revision 2, 2005/10/28

[20] Functional Mockup Interface (2014). [Online]. Available: www.fmistandard.org

[21] Plattform Industrie 4.0 (2019). [Online] www.plattform-i40.de/I40/Navigation/DE/Service/Glossar/Functions/glossar.html

[22] M. Grieves (2014). Digital Twin: Manufacturing Excellence through Virtual Factory Rep-lication - A Whitepaper

[23] M. Shafto, M. Conroy, R. Doyle et al (2010). DRAFT Modeling, Simulation, Information - Technology and Processing Roadmap, Technology Area 11.

[24] Panetta K. (2018). 5 Trends Emerge in the Gartner Hype Cycle for Emerging Technologies. [Online] www.gartner.com/smarterwithgartner/5-trends-emerge-in-gartner-hype-cycle-for-emerging-technologies-2018

[25] VDI-Richtlinie 3633 (2000). Simulation von Logistik-, Materialfluß- und Produktionssystemen. Düsseldorf.

[26] M. Schluse, M. Priggemeyer, L. Atorf (2018) Experimentable Digital Twins – Streamlining Simulation-based Systems Engineering for Industry 4.0. In: IEEE Transactions on Industrial Informatics, Volume 14, Issue 4

[27] E. A. Lee, H. Zheng (2007). Leveraging synchronous language principles for heterogeneous modeling and design of embedded systems. Design, 114–123. https://doi.org/10.1145/1289927.1289949

[28] D. Baraff (1996). Linear-time dynamics using Lagrange multipliers, 23rd annual conference on Computer graphics and interactive techniques, ACM, 1996

[29] W. Ley, K. Wittmann, W. Hallmann (2008). Handbook of Space Technology, 2008, Wiley

[30] Docker (2019) [Online]. https://www.docker.com/resources/what-container

[31] U. Dahmen, J. Roßmann, Experimentable Digital Twins for a Modeling and Simulation-based Engineering Approach, ISSE 2018, Rome

[32] J. Thieling, J. Roßmann. Highly-Scalable and Generalized Sensor Structures for Efficient Physically-Based Simulation of Multi-Modal Sensor Networks, ICST 2018, Limerick

[33] F. Aghili., Time-Optimal Detumbling Control of Spacecraft, Journal of Guidance Control and Dynamics, Sep. 2009

[34] J. Roßmann et al, The Virtual Testbed Approach towards Modular Satellite Systems, 69th International Astronautical Congress (IAC), Bremen, 2018