Proceedings of the 2022 Winter Simulation Conference
B. Feng, G. Pedrielli, Y. Peng, S. Shashaani, E. Song, C.G. Corlu, L.H. Lee, E.P. Chew, T. Roeder, and P. Lendermann, eds.

MODELING AND SIMULATION OF CYBER-PHYSICAL SYSTEMS USING AN EXTENSIBLE CO-SIMULATION FRAMEWORK

Jan Reitz
Tobias Osterloh
Jürgen Roßmann

Institute for Man-Machine Interaction
RWTH Aachen University
Ahornstraße 55
Aachen, 52074, GERMANY

ABSTRACT

Modern systems increasingly utilize software and especially computer networking to create new functionality, resulting in Cyber-Physical Systems (CPS). Due to the rapid progress in hardware as well as software technology, the range of applications of CPS is ever increasing. Despite the potential of simulation technology, there are several distinct challenges arising when simulating CPS, which mostly stem from their heterogeneity. In this paper, we present an approach to modeling and simulation of CPS that embraces this heterogeneity by integrating component models realized in domain-specific simulation tools in a co-simulation framework using a flexible plugin system. The framework includes an active simulation database that is used for modeling and communication, and a DE scheduler to orchestrate the co-simulation scenario. The approach is applied to a modular spacecraft where computational hardware is emulated and computer networks are simulated. This results in a comprehensive simulation of both physical and information processing systems.

1 INTRODUCTION

The development and operation of modern systems is significantly improved by the digitization of processes and information, resulting in Cyber-Physical Systems (CPS). Due to the rapid progress in hardware as well as software technology, the range of applications of CPS is ever increasing. Lee defines CPS are integration of computation and physical processes (Lee 2008). When using this definition there is a large similarity to the term embedded systems. Some authors differentiate CPS from embedded systems by adding that embedded systems are typically self-contained, while CPS have a focus on their connection with computer networks and the interaction with other CPS (Sadiku et al. 2017). An important feature of CPS with regard to modeling and simulation is that they are hybrid systems in the sense that they consist of physical processes whose state typically changes continuously in value and time, such as electrical motors and thermal processes, and computational hardware and computer networks whose state typically changes discretely in value and time. Physical processes are usually modelled using differential equations that are solved in continuous-time simulations. Computational hardware and computer networks on the other hand are usually modelled and simulated within the discrete event (DE) formalism. Modeling and simulations are important tools for the development and validation of complex systems in general, but the heterogeneity of CPS poses special challenges for both the Simulation software and the simulation engineers using it. Derler et al. list some of the key challenges (Derler et al. 2012):

1. Models with solver-dependent, non-determinant or zeno behavior
2. Divergence of model components throughout the development
3. Prevention of misconnected model components
4. Modeling the interaction of functionality and implementation
5. Modeling distributed behaviors
6. System heterogeneity

It is noteworthy that only the first challenge directly relates to numerical or algorithmic issues, whereas the remaining challenges address model structure and organization throughout their life cycle. Lee further notes that the current state of the art fails to sufficiently deal with real time requirements (Lee 2008), which highlights the need to incorporate timing behavior of computational hardware and computer networks into simulation models. In their survey of the state of the art of co-simulation, which also deals with hybrid co-simulation of CPS, Gomes et al. argue that more emphasis should be put on the extensibility of simulation frameworks, as there is great potential in increased combination of heterogeneous Simulation Units (SU) (Gomes et al. 2018).

In this contribution we discuss an approach to the modeling of co-simulation scenarios based on integrating external SUs into a common environment, called the Virtual Testbed (VTB), using a plugin system. The VTB consists of a simulation database that holds the scenario configuration and state, and a DE-scheduler. The developed concept will be applied to the modeling and simulation of a modular spacecraft as representative of a complex CPS. Figure 1 shows a screenshot of the application example inside the simulation framework. We believe this can alleviate challenges $2, 3, 5$ and 6.

In this approach CPS are decomposed into individual components with clear interfaces and modelled and simulated using appropriate tools. Representations of the individual components are integrated in the versatile simulation database (VSD) using adapter plugins and recomposed by connecting their ports using a unified user interface. They are then executed in a co-simulation scenario using a DE scheduler. This has been demonstrated in the application of a modular spacecraft, which demonstrated this approach by integrating virtual and real hardware into the simulation scenario. This approach provides a holistic view of the model, even if it is distributed over multiple simulation tools.

By keeping component models in their specialized tools, model divergence throughout development can be mitigated. The creation of explicit interfaces on the representations in the VSD using typed ports can prevent misconnected model components. Integrating multiple CPS and connecting them with the built-in network simulator eases the modeling of distributed behavior. System heterogeneity is still a challenge, but the flexible plugin system allowing the seamless integration of external simulation functionalities and the common data model of the VSD have proven to be a helpful. It allows for great extensibility, as was called for by Gomes et. al..

In the following Section 2 we summarize other approaches to modeling and simulation of CPS. In Section 3 we present our approach and demonstrate it using the example of a modular spacecraft in Section 4. Section 5 concludes this contribution.

2 RELATED WORK

Neema et al. developed a testbed to run simulations with a focus on Security and Resilience of CPS (Neema et al. 2018). The testbed uses a model-based integration approach, where models are not only used to model the system of interest but also to configure and parameterize simulations. This is done via WebGME, a collaborative general modeling tool. Data exchange between SUs and their execution is orchestrated using IEEE High Level Architecture. The necessary configuration and interface code between SUs and the High Level Architecture Simulation Bus is generated using the Command and Control Wind Tunnel framework (Hemingway et al. 2012). The presented application uses the traffic simulator SUMO, the network simulation framework OMNet++ and Matlab/Simulink. Using different external simulators requires an extension of the framework to provide the necessary code generation. All of the software components

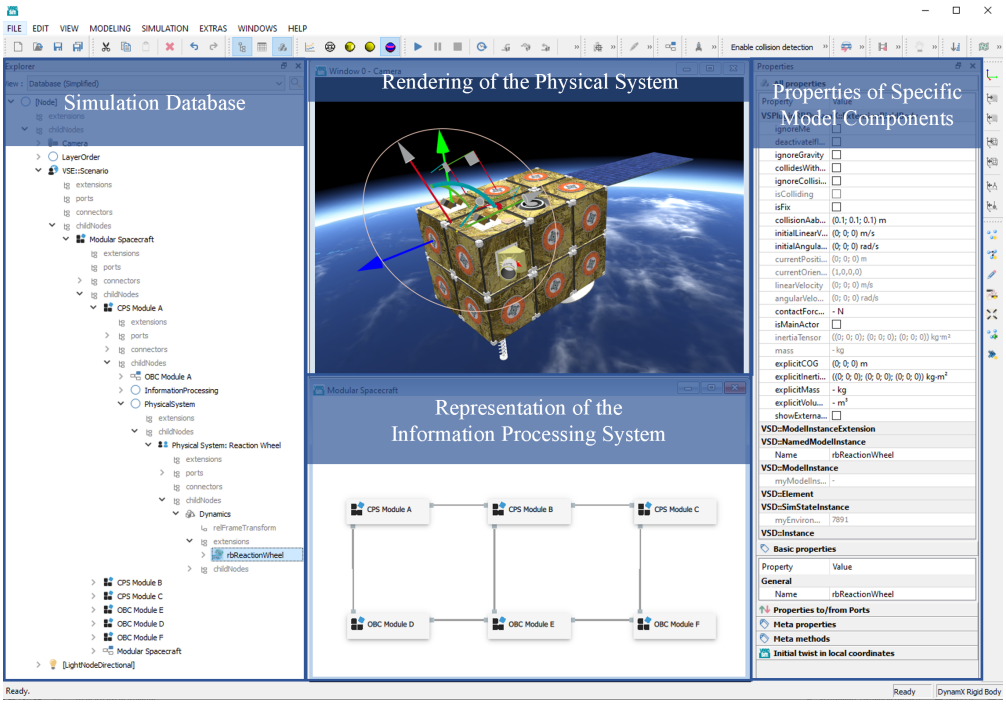

Figure 1: Comprehensive simulation of distributed CPS, applied to the simulation of a the modular spacecraft within the co-simulation framework.

are hosted in a cloud environment, which allows easy horizontal scaling, i.e. launching multiple simulation runs simultaneously. Simulation results are streamed into the time-series database InfluxDB, which is queried by WebGME to display live simulation results to the user.

CyPhySim (Lee et al. 2015) is an open-source simulator that aims to support a number of technologies useful to the simulation of CPS. It is based on the discrete event simulation engine, continuous-time solvers and state machine modeling of Ptolemy II (Eker et al. 2003). In the discrete event case, model components, called actors, communicate via timestamped events, that are emitted from and received by input and output ports. Superdense time is used to support causally related instantaneous events. Next to discrete event modeling CyPhySim supports two separate methods of including continuous systems in the simulation. Next to a classical ordinary differential equation solvers, Quantized State System (QSS) solvers are included which offer an alternative continuous-time simulation that integrates well with discrete event simulation as they require neither backtracking nor iterative solving.

Novak et al (Novak et al. 2017) also incorporate QSS solvers in their proposed method of modeling CPS using an agent-based approach. They note that the modeling of system dynamics in the time domain is more familiar to most engineers and in line with already existing models. However, when coordinating multiple agents, the setting of appropriate synchronization time steps can be difficult. Therefore they propose modeling individual agents either based on time or value quantization and use a QSS formulation on the synchronization level. They report a hardly predictable overloading of computational resources when facing fast state transitions. They patch this issue with an adapted synchronization algorithm that controls the trade off between interaction frequency and accuracy based on continuously updated performance models built from observations of the simulation.

Suzuki et al. present CPS-Sim, a co-simulation framework for CPS simulation (Suzuki et al. 2018). Its main function is the coordination and synchronization of the physical system simulation, in their case Matlab/Simulink, and communication network simulation, in their case OMNeT++. They employ a variable-stepped synchronization method to eliminate inaccuracies due to synchronization delay. This

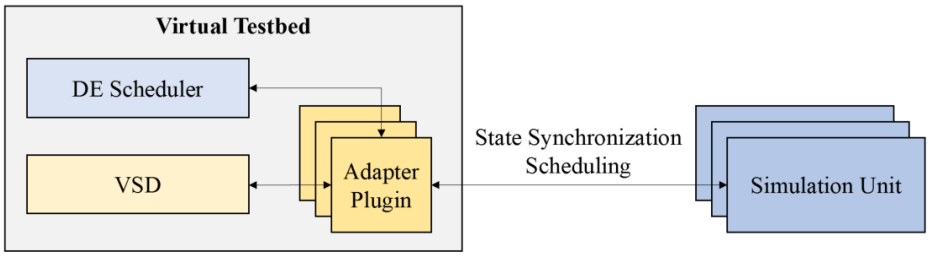

Figure 2: General simulation architecture.

requires the physical system simulator to support arbitrary step-size variation during simulation. They claim that, depending on the simulation scenario, this method is more efficient at producing accurate simulation results than regular time-stepped methods, whose synchronization step sizes would need to be very small in order to reach the same accuracy. The method appears to be limited to two SUs, as no information regarding generalization of the method is provided.

3 ARCHITECTURE

A guiding principle in developing the architecture is to embrace the heterogeneity of CPS by supporting the inclusion of specialized modeling and simulation tools into a central simulation framework while maintaining a unified user interface for modeling on the system level and executing co-simulation scenarios. We call this framework the Virtual Testbed (VTB). This is achieved by three technologies: The versatile simulation database (VSD)(Rossmann et al. 2013), a DE scheduler and a plugin system. Figure 2 displays the general architecture with the interaction of these technologies. Their details are discussed in the remainder of this section.

3.1 Versatile Simulation Database (VSD)

The VSD is an active, object-oriented, self-reflecting graph database. It provides data-management, persistence, meta-information and an event system. In the following we will discuss these features in more detail.

The VSDs meta-system is based on the object modeling groups meta model hierarchy. The meta-model occupies the highest level of the hierarchy (M2), defining concepts such as meta instance, meta property and meta method and how they relate to each other. On the level below (M1) is the simulation data model, describing concrete classes in terms of the concepts defined in the layer above. These classes represent model elements, e.g. rigid bodies with the properties mass, position and velocity. The lowest layer (M0) consists of instances of these concrete classes which constitute the actual simulation models. The reflection capabilities provided by the meta system allow the following functionality for all components storing their data in the VSD:

- Serialization for persistence (saving models to disk) and distributed simulation
- A unified User Interface for modeling and interacting with the VSDs elements (see the right and left side of Figure 1)
- Flexible interaction between model elements

The base class for all classes in the simulation data model is the *Node*. It has a name, and the property *childNodes* which is a list of owning references to other *Nodes* and sub classes thereof. This property defines the main model hierarchy, a tree. *Node* sub classes can define further value properties, as well as owning and non-owning reference and reference list properties. The structure of the simulation model is represented as a tree of *Node*s connected by owning references, e.g. a spacecraft module owns its on-board computer and payload. Non-owning references represent functional dependencies between model components, e.g. a

hinge joint may reference the two interconnected rigid bodies. The tree structure of the VSD can be seen on the left side of Figure 1, while the right side shows the properties of the currently selected element. Another built in class used for structuring models is the *Include*, which allows the insertion of models by reference. This aids in decomposing models into reusable components and allows the creation of component libraries.

The VSD is active, in the sense that it features a notification mechanism. Elements within the VSD can subscribe to events and emit them. There is a number of built-in events, mainly related to state changes, e.g. changes of properties, node creation and node deletion. Additionally, nodes can define custom events which others can also subscribe to. In this way, the VSD provides two communication mechanisms: Communication via the shared state of the VSD and the notification system. Both mechanisms are useful, but are not explicitly part of the model, i.e. they are invisible to the user. This can become a problem for readability and understanding of models, as implicit interactions of nodes can lead to unexpected simulation behavior.

This mechanism only works though, as long as all relevant data is kept in the VSD. It is impossible to react to changes in the private state of external SUs.

To make interactions between model components explicit, the data model has been extended by elements that allow the modeling of interactions between model components. These components are the *Port*, an interface which can be extended by domain specific implementations, and a general *Connector*. The user interface for this type of modeling is displayed in the middle of Figrue 1. Extending the port interface allows the developer to constrain the types of other ports that can be connected to it, both with regards to type and multiplicity. For example, an RJ45 Socket can be modelled to only allow zero or one connections from an RJ45 Plug. The communication is still based on the event system and shared state provided by the VSD, but modeling guidelines and these clear structuring elements aid in decomposing the system of interest into components with clearly defined interaction points. A future research direction is the decomposition of simulation models into largely independent components by analyzing these connection networks.

3.2 Orchestrating the Co-Simulation Scenario

After integrating the relevant data of external SUs into the VSD data model, the joint execution of external SUs needs to be orchestrated. This is the responsibility of the DE scheduler. Both time-stepped and event-driven models are supported. Figure 3 displays both scheduling models in a single application example. This approach can be classified as Hybrid DE (Gomes et al. 2018) as all simulators are wrapped into the DE formalism.

The periodic time-stepped interface consists of a *Load Calculate Save* cycle. In the *Load*-step all relevant data is loaded from the VSD. During the *Calculate*-step, each SU performs the required calculation without side effects. Lastly, values are written back to the VSD in the *Save*-step. With these constraints simultaneous events are concurrent, i.e. they do not require a total ordering such as super dense time. In Figure 3 the blocks *Virtual Hardware* and *RBD* share the same time-stepped events 1,3 and 5. The drawback is that all SUs work with the state of the previous iteration, even though updated information could be available. In the worst case, these potentially relevant changes in the VSD will be visible with a delay of one simulation step. Therefore, in order to simulate systems with precise timing requirements, the simulation step time has to be chosen appropriately.

Apart from periodic time-stepped events the scheduler allows the scheduling of events at arbitrary times. In this case a total ordering is required to guarantee simulation determinism and to enable the modeling of causal dependencies between simultaneous events. Instead of relying on a special formalism to order simultaneous events, the VSDs notification system is employed to model causal dependencies between simultaneous events. This can be done by subscribing to relevant state changes or custom events. In the displayed example, the block *Sensors* is realized in an event-driven style, that reacts to changes in the measured value via the notification mechanism. This is not an event in the sense that it never enters the scheduler event list, but rather an extension of the event that triggered the change, in this case *Step 1*.

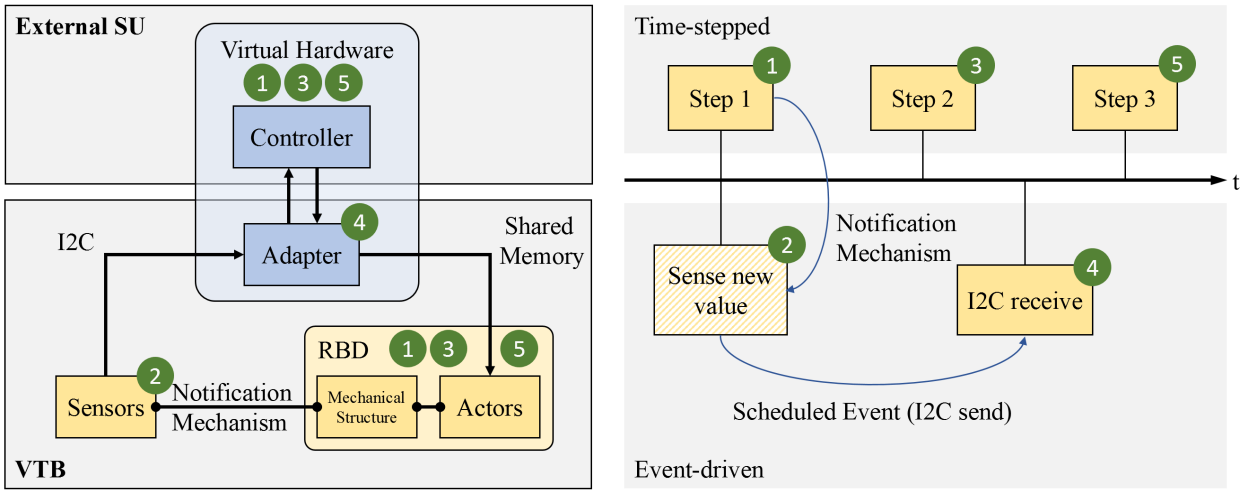

Figure 3: Example model (left) with simulation timeline (right) showing time-stepped and event-based scheduling, green numbers indicate execution order.

After being notified of the value change, the *Sensors* block sends an I2C message to the *Virtual Hardware* block by scheduling the *I2C receive* event.

The work of Lee et al. and Novak et al. highlighted, that the use of QSS solvers for CPS simulation has potential benefits. Providing a QSS solver as core functionality of the Virtual Testbed appears as another promising future research direction.

3.3 Extending the VTB using Plugins

The VTB is meant to be extended by domain specific functionality and offers a plugin system to simplify this process. Most of the existing functionality, such as the scheduler, has been added as plugins. Plugins can either directly implement additional functionality, like a custom rigid body simulation engine, or serve as adapter to include existing functionality in the form of external libraries, processes or services into the VTB. Existing adapter plugins include for example plugins to integrate Modelica, the traffic simulator SUMO or CAN Dongles to interact with real devices over real CAN networks. Plugins can also depend on each other, so common functionality can be provided as plugin libraries. The system currently features over 400 active plugins ranging from GUI functionality for interactive modeling, 3d rendering, ray-tracing based sensor simulation, rigid body and orbit dynamics to geo-data driven forest growth applications.

Extending the simulation functionality of the VTB with a new plugin typically involves a mapping of the domain specific data onto the VSD data model by implementing *Node* sub classes with appropriate properties and methods. These new elements can then be used when creating or editing models. Add simulation functionality usually also involves interaction with the scheduler, either by sub classing the *TaskStep* class to access the time-stepped interface or by using the schedulers API to directly schedule events.

This approach allows enormous flexibility and extensibility, while maintaining a common core and interaction mechanism in the form of the VSD. In practice this creates a network effect, as for example GUI functionality, like a plugin to render the simulation scene to a VR headset, is not tied to specific applications but can be employed by all users based on the common data model.

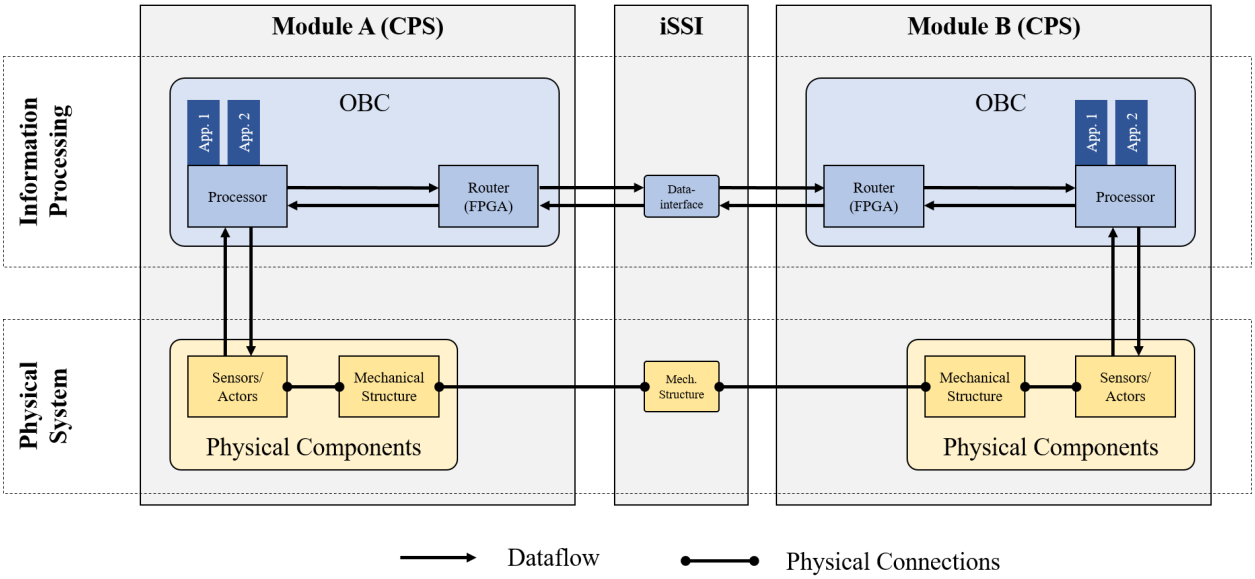

Figure 4: Physical architecture of the modular spacecraft consisting of two interconnected modules.

4 APPLICATION

The immediate application of this concept is the simulation of a modular spacecraft, as initially proposed in the iBOSS project (Weise et al. 2012). According to the iBOSS concept, a modular spacecraft is characterized by individual modules with independent on board computers (OBC) that are interconnected via standardized interfaces (i.e. for mechanical, data, power and thermal coupling).

Figure 4 displays an exemplary configuration of two modules. Each module is a CPS consisting of physical structure (yellow) and computation (blue). Each module contains a Xilinx Zynq-7000 SoC ZC706 as main OBC. This System on Chip (SoC) contains both an ARM Cortex CPU and a Field Programmable Gate Array (FPGA). The CPU executes individual applications and the FPGA is programmed to become a SpaceWire Router, which is connected to the spacecrafts SpaceWire network. CPU and FPGA interact via shared access to the SoCs RAM. Module specific payloads are connected to the CPU via payload specific technologies such as UART or CAN. The modules are mechanically and thermally connected by the intelligent Space Systems Interface (Schervan et al. 2021), which also provides electrical connections for power distribution as well as glass fiber connections for data exchange. Based on system design, control loops are distributed across multiple modules, which can cause critical timing requirements on the network layer, as well as potential data collisions. Within this application example we intend to demonstrate the application of the developed modeling and simulation approach for CPS on a conceptual level.

Past efforts towards this goal mainly focused on the simulation of physical processes, incorporating the satellite and orbital dynamics fused with the simulation of robotic manipulators, as well as thermal simulation including thermal input via radiation (Rast et al. 2016), (Rossmann et al. 2018). Information processing and communication within the satellite was only modeled on a functional level using a simplified publish-subscribe based communication (ROS2) and Docker virtualization (Rossmann et al. 2018). So far, the available simulation approach is not capable of mirroring the physical network topology, which consequently means that some important test cases cannot be replicated in simulation, e.g. failure of connections and investigation of routing algorithms. Figure 5 shows the structure of the exemplary configurations simulation model, where *Module A* is simulated in a hardware in the loop setup and *Module B* is simulated using hardware emulation. The remainder of this section discusses these two approaches and the symmetry between them.

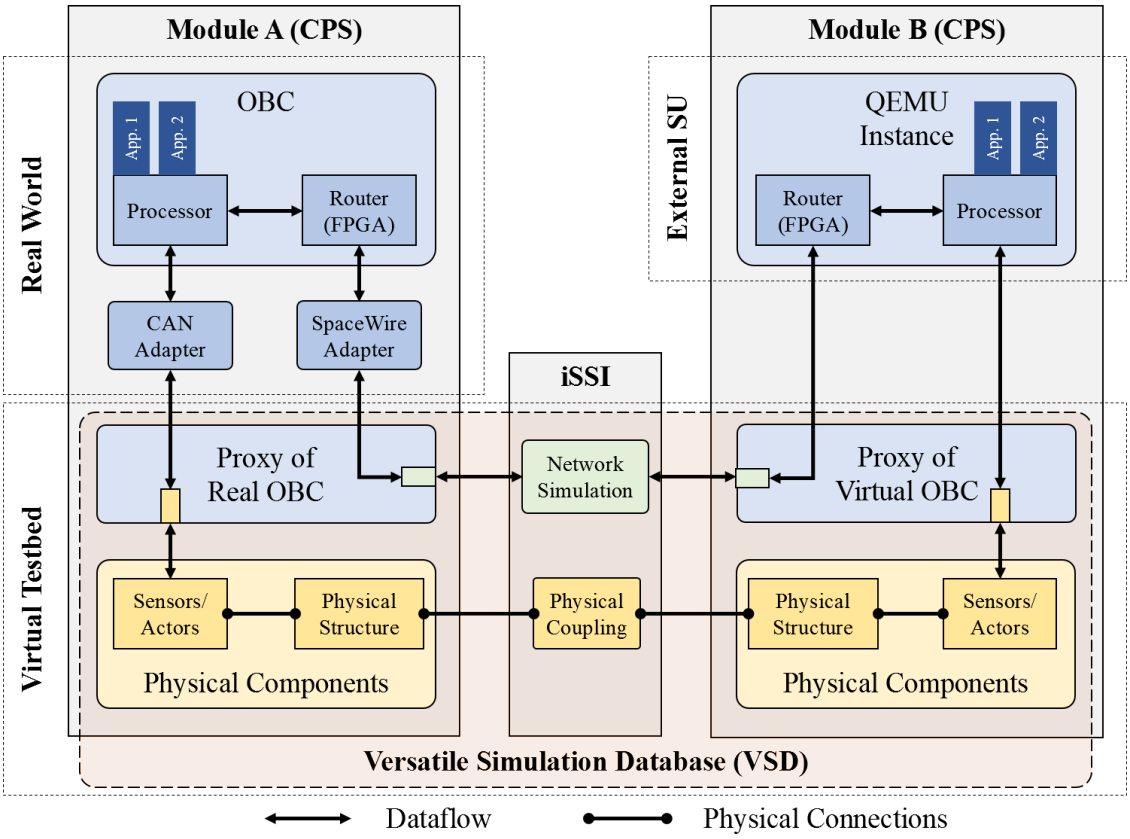

Figure 5: Simulation architecture for a modular spacecraft consisting of two connected modules. The *Physical Components* of both modules are simulated in the *Virtual Testbed* (VTB). The on-board computer (*OBC*) of both modules is represented in the VTB by proxies *Proxy of Real OBC* in a hardware in the loop setup (left), and *Proxy of Virtual OBC* based on a co-simulation with a *QEMU Instance* (right).

4.1 Holistic Simulation of the Modular System

In order to simulate the modular spacecraft in its entirety, we employ the presented VTB approach. The physical components of the system can be modelled and simulated using existing functionality as discussed above. An adapter plugin for the emulator QEMU (Bellard 2005) is used to additionally incorporate an emulation of the OBC, including both CPU and FPGA, into the simulation scenario. The adapter plugin handles the data transfer between the VTB and QEMU as well as the integration of the execution of this external SU into the VTB scheduler. Consequently, model and software mismatches between simulation and reality can be avoided. Additionally, the functionality of the SpaceWire router can effortlessly be used, without the need to develop a specialized SU replicating the FPGA routing functionality. Using virtual hardware removes the real-time requirements imposed by hardware in the loop setups while maintaining the benefit of executing the target binaries without modification. This prevents errors due to the divergence of models and real software implementations.

The virtual hardware is incorporated into the simulation model by a specialized *Node* that acts as *Proxy of Virtual OBC* in the VSD. It is equipped with ports representing the real OBCs ports that can be connected with the remaining simulation model components. This depicted on the right side of Figure 5. The yellow port represents payload specific connections to *Sensors/Actors*. So far, this has been discussed in prior in the context of a self-balancing robot (Reitz et al. 2020). This application extends the use case presented in prior work by adding appropriate network interfaces to the proxy. These green ports represent connections

to other OBCs via a simulated SpaceWire network. Integrating the virtual hardware that is being executed by an external SU with a proxy in the VTB maintains the unified view of the simulation model mentioned in Section 3. All components and connections of the real system are represented in the VSD and therefore available to the user. The connection between the physical layer and the data processing layer is realized according to the real world data protocols, aiming for a seamless transition to hardware in the loop setups.

4.2 Hardware in the Loop Setup

Emerging from the purely virtual simulation setup, it is possible to substitute the simulated modules by their real world counterpart, resulting in a hardware in the loop setup as depicted on the left side of Figure 5. The setup is similar to an external SU and highlights the modularity of the proposed approach. Additional hardware is required to connect the simulation computer to the real OBC, in the case of our application using a STAR Dundee SpaceWire Brick and a Peak CAN adapter.

5 CONCLUSION

Within this paper, we presented an approach to the modeling and simulation of CPS and applied it to a modular spacecraft. In this approach CPS are decomposed into individual components with clear interfaces and modelled and simulated using appropriate tools. Representations of the individual components are integrated in the versatile simulation database (VSD) using adapter plugins and recomposed by connecting their ports using a unified user interface. They are then executed in a co-simulation scenario using a DE scheduler. This has been demonstrated in the application of a modular spacecraft, which demonstrated this approach by integrating virtual and real hardware into the simulation scenario. This approach provides a holistic view of the model, even if it is distributed over multiple simulation tools. We believe this can alleviate challenges $2, 3, 5$ and 6 posed in Section 1. By keeping component models in their specialized tools, model divergence throughout development can be mitigated. The creation of explicit interfaces on the representations in the VSD using typed ports can prevent misconnected model components. Integrating multiple CPS and connecting them with the built-in network simulator eases the modeling of distributed behavior. System heterogeneity is still a challenge, but the flexible plugin system allowing the seamless integration of external simulation functionalities and the common data model of the VSD have proven to be a helpful. It allows for great extensibility, as was called for by Gomes et. al.. Especially the integration of virtual hardware appears as promising research direction. Future work includes simulations with multiple virtual hardware instances interacting over a simulated network.

ACKNOWLEDGMENTS

This work is part of the project "Twins4Space", supported by the German Aerospace Center (DLR) with funds of the German Federal Ministry of Economics and Technology (BMWi), support code 50RA2103.

REFERENCES

Bellard, F. 2005. "QEMU, a Fast and Portable Dynamic Translator". In *Proceedings of the USENIX Annual Technical Conference*. April 10^{th}-15^{th}, Anaheim, California, 41–46.

Derler, P., E. A. Lee, and A. S. Vincentelli. 2012, January. "Modeling Cyber-Physical Systems". *Proceedings of the IEEE: Special Issue Cyber-Physical Systems* 100(1):13–28.

Eker, J., J. W. Janneck, E. A. Lee, J. Liu, X. Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Y. Xiong. 2003. "Taming Heterogeneity - The Ptolemy Approach". *Proceedings of the IEEE* 91(1):127–144.

Gomes, C., C. Thule, D. Broman, P. G. Larsen, and H. Vangheluwe. 2018. "Co-simulation: A Survey". *ACM Computing Surveys* 51(3):1–33.

Hemingway, G., H. Neema, H. Nine, J. Sztipanovits, and G. Karsai. 2012. "Rapid Synthesis of High-Level Architecture-based Heterogeneous Simulation: A Model-based Integration Approach". *Simulation: Transactions of the Society for Modeling and Simulation International* 88(2):217–232.

Lee, E. A. 2008. "Cyber Physical Systems: Design Challenges". In *11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing*. May 5th-7th, Orlando, Florida, 363–369.

Lee, K. H., J. H. Hong, and T. G. Kim. 2015. "System of Systems Approach to Formal Modeling of CPS for Simulation-Based Analysis". *ETRI Journal* 37(1):175–185.

Neema, H., B. Potteiger, X. Koutsoukos, G. Karsai, P. Volgyesi, and J. Sztipanovits. 2018. "Integrated Simulation Testbed for Security and Resilience of CPS". In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*. April 9th-13th, Pau, France, 368–374.

Novak, P., P. Kadera, and M. Wimmer. 2017. "Agent-Based Modeling and Simulation of Hybrid Cyber-Physical Systems". In *2017 3rd IEEE International Conference on Cybernetics*. June 21st-23rd, Exeter, United Kingdom, 1–8: Institute of Electrical and Electronics Engineers, Inc.

Rast, M., A. Kupetz, M. Schluse, O. Stern, and J. Rossmann. 2016. "Virtual Testbed for Development, Test and Validation of Modular Satellites". In *the 13th International Symposium on Artificial Intelligence, Robotics and Automation in Space*. June 20th-22nd, Beijing, China, 1–8.

Reitz, J., A. Gugenheimer, and J. Roßmann. 2020, December. "Virtual Hardware in the Loop: Hybrid Simulation of Dynamic Systems with a Virtualization Platform". In *2020 Winter Simulation Conference*, edited by K. G. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and T. R., 1027–1038. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Rossmann, J., M. Schluse, and R. Waspe. 2013. "Combining Supervisory Control, Object-oriented Petri-Nets and 3D Simulation for Hybrid Simulation Systems using a Flexible Meta Data Approach". In *Proceedings the 3rd International Conference on Simulation and Modeling Methodologies, Technologies and Applications*, edited by T. Ören, J. Kacprzyk, L. Leifsson, M. S. Obaidat, and S. Koziel, 15–23. Setúbal, Portugal: Science and and Technology Publications, Lda.

Rossmann, J., O. Stern, A. Kupetz, G. Jochmann, M. Schluse, D. Kaufmann, U. Dahmen, A. Wahl, J. Thieling, and T. Osterloh. 2018. "The Virtual Testbed Approach towards Modular Satellite Systems". In *the 69th International Astronautical Congress*. October 1st-5th, Bremen, Germany, 1–5.

Sadiku, M. N., Y. Wang, S. Cui, and S. M. Musa. 2017. "Cyber-Physical Systems: A Literature Review". *European Scientific Journal* 13(36):52–58.

Schervan, T., J. Kreisel, K. Schroeder, and D. R. Wingo. 2021, June. "New Horizons for Exploration Via Flexible Concepts Based On Building Blocks Using The Standardized iSSI (Intelligent Space System Interface) Modular Coupling Kit By iBOSS". In *Global Space Exploration Conference*, 14–18. St. Petersburg, Russia: International Astronautical Federation.

Suzuki, A., K. Masutomi, I. Ono, H. Ishii, and T. Onoda. 2018. "CPS-Sim: Co-Simulation for Cyber-Physical Systems with Accurate Time Synchronization". In *7th IFAC Workshop on Distributed Estimation and Control in Networked Systems*. August 27th, Groningen, The Netherlands, 70–75.

Weise, J., K. Brieß, A. Adomeit, H.-G. Reimerdes, M. Göller, and R. Dillmann. 2012. "An Intelligent Building Blocks Concept for On-Orbit-Satellite Servicing". In *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space*. September 4th-6th, Turin, Italy: European Space Agency.

AUTHOR BIOGRAPHIES

JAN REITZ is a researcher at the Institute for Man-Machine Interaction at RWTH Aachen University. He received his M.Sc. in Mechanical and Process Engineering from TU Darmstadt. His research interest is in modeling and simulation of cyber-physical and distributed systems. His email address is reitz@mmi.rwth-aachen.de.

TOBIAS OSTERLOH received M.Sc. degree in electrical engineering with focus on system technology and automation of RWTH Aachen University in 2015. Since 2016 he is research assistant with the Institute for Man-Machine Interaction investigating architectures and methodologies for the multi-body dynamics simulation of digital twin. His email address is osterloh@mmi.rwth-aachen.de.

JÜRGEN ROSSMANN is a professor with the faculty of electrical engineering at RWTH Aachen University and head of the Institute for Man-Machine Interaction. After graduating from the Universities of Dortmund and Bochum, he worked as a research assistant, then as a group leader and finally as a department head at the Institute of Robotics Research at the University of Dortmund. His work in the fields of space and industrial robotics, automation technology and virtual reality has been awarded more than 15 different prizes. He is a visiting professor for robotics and computer graphics at the University of Southern California since 1998. Furthermore, he is a member of the board of Directors of the Dortmund Institute for Research and Transfer (RIF) as well as managing director of VEROSIM GmbH. His email address is rossmann@mmi.rwth-aachen.de.