applied sciences

MDPI

Article

CPES Testing with MOSAIK: Co-Simulation Planning, Execution and Analysis

Cornelius Steinbrink [1,*,†], Marita Blank-Babazadeh [1,†], André El-Ama [1,†], Stefanie Holly [1,†], Bengt Lüers [1,†], Marvin Nebel-Wenner [1,†], Rebeca P. Ramírez Acosta [1,†], Thomas Raub [1,†], Jan Sören Schwarz [2,†], Sanja Stark [1,†], Astrid Nieße [3,†] and Sebastian Lehnhoff [1,†]

1 OFFIS—Institute for Information Technology, Escherweg 2, 26121 Oldenburg, Germany; marita.blank-babazadeh@offis.de (M.B.-B.); andre.el-ama@offis.de (A.E.-A.); stefanie.holly@offis.de (S.H.); bengt.lueers@offis.de (B.L.); marvin.nebel-wenner@offis.de (M.N.-W.); rebeca.ramirez@offis.de (R.P.R.A.); thomas.raub@offis.de (T.R.); sanja.stark@offis.de (S.S.); sebastian.lehnhoff@offis.de (S.L.)

2 Department of Computer Science, University of Oldenburg, Ammerländer Heerstraße 114-118, 26129 Oldenburg, Germany; jan.soeren.schwarz@uol.de

3 Group Energy Informatics, Leibniz University Hannover, Appelstraße 9a, 30167 Hannover, Germany; niesse@ei.uni-hannover.de

* Correspondence: cornelius.steinbrink@offis.de; Tel.: +49-441-9722-404

† These authors contributed equally to this work.

Received: 19 January 2019; Accepted: 26 February 2019; Published: 5 March 2019

Abstract: The complex nature of cyber-physical energy systems (CPES) makes systematic testing of new technologies for these setups challenging. Co-simulation has been identified as an efficient and flexible test approach that allows consideration of interdisciplinary dynamic interactions. However, basic coupling of simulation models alone fails to account for many of the challenges of simulation-based multi-domain testing such as expert collaboration in test planning. This paper illustrates an extended CPES test environment based on the co-simulation framework MOSAIK. The environment contains capabilities for simulation planning, uncertainty quantification and the development of multi-agent systems. An application case involving virtual power plant control is used to demonstrate the platform's features. Future extensibility of the highly modular test environment is outlined.

Keywords: testing; co-simulation; MOSAIK; multi-agent systems; simulation planning; uncertainty quantification

1. Introduction

A cyber-physical energy system (CPES) is an example of so-called systems of systems. Its dynamical behavior emerges from the interactions of various complex components and systems that themselves may display nonlinear dynamics. Additionally, CPES unite formerly separate domains from the physical energy systems, e.g., electric power grids, distributed energy resources (DER), or heat and gas networks, with the digital world of information and communication infrastructure (ICT), data privacy and cyber-security. But also human behavior plays an important role, be it from an economic point of view with energy markets or through the social aspects of consumption behavior. Such human-driven systems are sometimes called transactive energy systems or H-CPES, but in the context of this paper, social and economic dynamics are included in the understanding of CPES. Furthermore, environmental factors play an important role in the forcing (weather) as well as the assessment (sustainability) of CPES. Therefore, CPES should also be considered in regard to their response to meteorological events or techno-environmental feedback.

One of the major challenges of designing and implementing CPES lies in the development and testing of technologies and concepts to guarantee functionality, stability and safety during operation. This is especially true for CPES components with complex behavior and many interfaces with different domains. Due to this complexity and interdisciplinarity, analytical testing on a formal basis is infeasible for CPES solutions (beyond the first conceptualization phase of new solutions). The alternative approach of hardware tests in laboratories or in the field is expensive and inflexible. It is thus applied only in the last stages of development. Therefore, software simulation proves to be valuable as a preparatory test stage before hardware testing, on the one hand, as well as an easily scalable test environment that allows consideration of larger and more complex setups. The role of simulation in the development of e.g., smart grid algorithms has been discussed in detail in [1].

A type of simulation that has become popular in CPES testing is co-simulation (see [2,3]). It allows users to couple independent simulation tools representing different parts of the overall system. This greatly reduces the modeling effort and error-proneness related to monolithic representations of interdisciplinary systems. The different components of a co-simulation setup (called "simulators" in the following) are interfaced with each other and exchange data during runtime so that a simulation process can work in a distributed manner and across heterogeneous runtime environments. Furthermore, generic co-simulation systems can be used to reduce the interfacing effort and increase the transferability of solutions onto different application cases.

Despite its advantages, working with co-simulation involves a number of challenges. A commonly discussed hurdle is the technical coupling of heterogeneous software tools. Identifying the most suitable tools, however, is often overlooked. Such a process requires experts and modelers from different domains to find a common understanding of their tools' capabilities and simulation goals. This is often hindered by the lack of a domain-crossing language for CPES description. Similarly, joining knowledge from different domains will typically involve different degrees of simplification and aggregation. Therefore, the quality and reliability of co-simulation outputs may vary over time and may be difficult to assess even for simulation experts. While these issues are widely recognized in the CPES co-simulation community, most approaches are so far focused on the more technical aspects of simulator coupling.

A popular standard for co-simulation is the functional mock-up interface (FMI) [4], which describes the data and functions a simulator needs to provide in order to participate in a co-simulation. Simulators interfaced via FMI still need a so-called co-simulation master that coordinates and synchronizes the data exchange between them. Several co-simulation frameworks exist already that provide such a master algorithm, often in combination with their own interfaces and further capabilities. The standard high-level architecture (HLA) [5] provides the basis for some frameworks like the CPS wind tunnel [6]. Similarly, the multi-paradigm simulation environment Ptolemy II [7] is also sometimes employed in co-simulation (e.g., [8]). Newer frameworks involve the tool DACCOSIM [9] for distributed, FMI-based simulation, the agent-based framework MECSYCO [10], and the OpSim system [11] related to the simulation message bus (SMB) [12] concept. The focus of this paper lies on the open-source co-simulation framework MOSAIK (http://mosaik.offis.de/) that is used in its current form since 2014.

Comparing or even ranking co-simulation frameworks against each other is a challenging task since the diversity of possible application cases in the CPES domain requires or favors different framework features in different situations. Some attempts at framework classification and comparison can be found in [13,14]. Another, simplified approach to assess frameworks is given by the three attributes performance, flexibility and usability. All three points are important for a good framework, but practical experience suggests that no single one can be maximized without at least one of the other attributes decreasing. The authors thus suggest that the mentioned co-simulation frameworks fill different niches depending on the employed simulators, modeled systems, research goals, and target user group.

The MOSAIK framework is focused on providing high usability and flexibility. It is thus most suitable for users that regularly need to set up different co-simulation experiments and probably work in interdisciplinary teams to conduct tests with a focus on multi-domain interactions. Furthermore, the framework has been created with users in mind that are not necessarily co-simulation experts. This is reflected by the core features of MOSAIK as well as additional modules and processes that help users, among other things, to plan their co-simulation experiments or assess their output accuracy. These modules tackle the challenges associated with the combination of tools and knowledge across different domains, as indicated above.

One of the major purposes of the original MOSAIK design has been the evaluation of new forms of ICT solutions based on multi-agent systems (MAS) [15]. In general, MAS provide the possibility to model components and their interactions in different domains. Example application areas are social sciences with the evolution of societies [16] or pedestrian behaviour in transportation studies [17]. But also in the domain of energy systems, MAS have become popular [18]. They can be used as control mechanisms reaching from local to coordinated optimization and thus allow for distributed control with different objectives, e.g., scheduling for market optimization or grid stability and therefore provide more flexibility and scalability. Thus, MAS solutions are especially beneficial regarding the growing complexity in CPES. Due to the strong link between the MAS approach and MOSAIK, the framework is integrated with features for the development of distributed ICT solutions. This aspect of the testing environment is elucidated further below.

This paper makes a case for test environments that go beyond basic co-simulation frameworks, but in fact integrate toolchains for the realization of more mature and complex validation experiments. This is done by presenting the current state of the MOSAIK environment, including the core framework as well as its extended functionalities and associated tools. The extended environment allows users to not only solve the problem of technical simulator interoperability, but actually face challenges associated with cross-domain expert collaboration. This involves tool selection and test planning, result reliability, and development of novel control solutions for complex systems. All this information is provided with a focus of application in CPES development and testing. The remainder of the paper is structured as follows. In Section 2 the main features of the MOSAIK environment are illustrated, with Section 2.6 providing an exemplary application case. Section 3 presents the results from the demonstration case with a focus on the benefits provided by the MOSAIK toolchain. Section 4 discusses the trade-offs made by the framework, provides an overview of further application examples, and outlines future developments. The paper is concluded with Section 5.

2. Materials and Methods

This section gives an overview of the proposed CPES test environment with Section 2.1 displaying the overall architecture. While the MOSAIK framework as the core of the environment has already been in use for some years, the extended environment includes a number of new tools and concepts that have not yet been described in combination. Together, they allow not only the coordination of heterogeneous simulation models (Section 2.2), but also the structured planning of complex experiments (Section 2.3), analysis of their output accuracy (Section 2.4), and integration of distributed ICT solutions (Section 2.5). The setup presentation is concluded by the outlining of a demonstration case (Section 2.6) to illustrate the various capabilities of the environment.

2.1. MOSAIK *System Architecture*

The MOSAIK co-simulation environment can be divided into three parts: the core framework, the set of adapters, and additional utility software. The core provides the basic co-simulation functionality via a set of interconnected modules, the central two being the sim-manager and the scheduler modules. The sim-manager is responsible for setting up and maintaining the connections between MOSAIK and the simulators. The scheduler, on the other hand, coordinates the data exchange between the simulators during runtime (see Section 2.2). A simulator is always an independently executable piece of software that implements a simulation model. For user interaction with MOSAIK, two API are provided. The component-API has to be used to establish an interface between MOSAIK and a new simulator. It specifies the socket connection used for data exchange and outlines what kind of data the simulator has to accept and provide. The scenario-API, on the other hand, allows users to set up executable co-simulation scenarios by specifying which simulators are to be used and how they should be parameterized and interconnected with each other. During execution, a scenario script employs the functionalities of the sim-manager and the scheduler. An overview of the MOSAIK core architecture is provided in Figure 1.

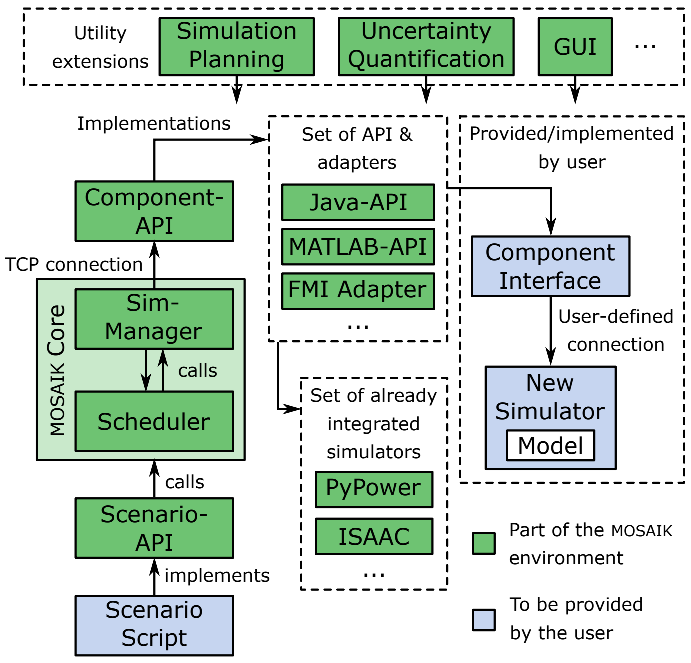

Figure 1. Architecture overview of the full MOSAIK environment.

While the MOSAIK software itself contains no simulation models, there is a constantly growing set of simulators that are interfaced with the framework. This is called the MOSAIK ecosystem. An important part of this ecosystem are open-source simulation tools that can be downloaded with the basic MOSAIK setup, e.g., the PyPower tool (based on the MatPower software [19]). The MAS tool, ISAAC, is to be mentioned here (see Section 2.5). However, a number of commercial tools are also already supported by MOSAIK, like PowerFactory or Opal-RT eMEGAsim [20,21]. Even more so, the component-API specification has been implemented in different programming languages (e.g., Python, Java, MATLAB) to facilitate the integration of a wide variety of different simulators. An FMI adapter is also given so that MOSAIK can serve as a master for all simulators that support the standard.

To improve the usability and performance of MOSAIK, additional software modules and processes are being developed to extend the capabilities of the system beyond data exchange scheduling between simulators. The MOSAIK system architecture follows a modular approach in regard of these utility tools. They build up on the co-simulation core but function as independent software. This way, users of the co-simulation framework have the freedom to install only those additional modules which they actually like to employ. The utility environment is comprised of a GUI for demonstration and education purposes (https://maverig.readthedocs.io/quickstart.html), processes for simulation planning (Section 2.3), an uncertainty quantification system (Section 2.4), and is currently being extended by modules for performance improvement via the application of surrogate modeling (Section 4.3).

2.2. Scheduling Algorithm

The scheduling paradigm employed by MOSAIK can be interpreted as a flexible discrete-time simulation. One of its basic concepts is the so-called data flow graph, a directed graph that indicates the data dependencies between the simulators employed in the co-simulation. The graph is established automatically at the beginning of a co-simulation process. Its structure is based on the connections between simulators that are defined by a user via the scenario-API in a scenario script. For each simulator *S*, the graph indicates which other simulators provide *S* with data (predecessors) and which require data from *S* (successors). After the graph is established, an individual simulation process is started for each employed simulator. The structure of such a process is depicted in Figure 2.

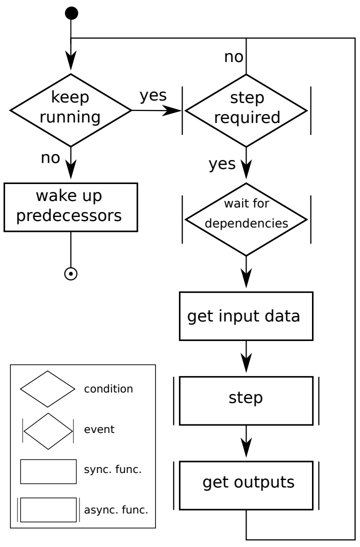

Figure 2. Simulation process for an individual simulator.

The process for a given simulator is structured in steps that indicate its temporal progression. The step size is measured in integers to guarantee comparability between the timing of different simulators. The simulation process iterates for each simulator, each time performing a step, until the end of the co-simulation is reached. The data exchange between simulators is conducted asynchronously to improve co-simulation performance.

At the beginning of each simulation process iteration, it is checked whether a new simulation step is required. If the end of the co-simulation is reached or all predecessors have stopped already, the simulator in question is stopped. Right before stopping, the simulator triggers its (unstopped) predecessors to allow them to check whether they can also stop. Unstopped simulators continue with their simulation process and check whether a simulation step is currently expected from them. This is true if the simulator in questions possesses no successors or if any successor is currently waiting for data input. A simulator that requires a next step then checks whether it needs input data from its predecessors and waits if that is the case. In other words, waiting for input data can propagate across different simulation processes until a simulator with no open dependencies provides the initially needed data.

The waiting between simulation processes is managed via the discrete event simulation (DES) framework *Simpy* (https://simpy.readthedocs.io/en/latest/). This framework enables the creation of wait events to pause processes and trigger them again if certain conditions are resolved (typically data availability). This allows for efficient management of the timing of asynchronous simulation processes.

The actual data exchange between a simulator and MOSAIK is divided into three stages. First, the input data required by the simulator is read from MOSAIK's internal buffer and put into the simulator's input buffer. The data requirement is specified via the data flow graph. Furthermore, the data in the MOSAIK buffer is associated with timestamps to make sure that only valid data is provided. After the data provision, the actual simulation step for the simulator is performed by calling the appropriate function of the simulator interface. All data within the simulator's input buffer as well as its current step time is provided as input. Stepping triggers calculations of the simulator and advances it in time. The call is conducted asynchronously so that it does not needlessly block the overall co-simulation. As a response to the call, the simulator provides its next time step. This way, simulators can adjust their temporal resolution during execution and are not bound to one fixed step size. After the step execution, finally, the simulator receives an asynchronous request for all output data its successors expect. The data is stored in the MOSAIK buffer. This allows for the reuse of old data without the need to request it again from the simulator. Such a feature can be useful if two simulators with different temporal resolution are coupled. In order to prevent storage overflow, the buffer is regularly pruned. After the data has been provided, all of the simulator's successors are triggered. They can then check whether all their required input data is available and they can proceed with their simulation process.

The scheduling algorithm in its presented form only works for data exchange in one direction. If two simulators require data from each other, deadlocks can paralyze the co-simulation. In order to avoid this, a second data flow graph is introduced that handles the second direction of these data flow cycles. To avoid conflicts, the dependencies of the second graph are always resolved after those of the first graph. In other words, cycles are broken by a shift in time. A more detailed description about this and other aspects of the MOSAIK core can be found in the online documentation (https://mosaik.readthedocs.io/en/latest/dev/design.html#design-and-achitecture).

2.3. Simulation Planning

Co-simulation scenarios typically have to be developed in cooperation of experts from different domains and a simulation expert. This cooperation poses potential challenges, because experts from different domains may have problems in understanding each other correctly due to differing terminology and backgrounds. Therefore, the presented co-simulation environment is extended by a process for integration of domain knowledge and structuring of the test scenarios. In this simulation planning process, it has to be defined what the objectives of the simulation are, what kind of objects should be considered, which simulation models are available and used, and which data flows are needed. Additionally, it should be possible to enrich the simulation objective definition with external domain knowledge and definitions of terms (see left-hand side of Figure 3).

An approach for modeling these simulation objectives has been developed in form of a process to assist the planning of co-simulation scenarios for the investigation of energy scenarios [22]. It consists of a sustainability evaluation process (SEP) in combination with an information model as described by Schwarz et al. [23]. In the SEP, the specific use case of co-simulation for the evaluation of possible future states of the energy system is addressed. As a first step, future scenarios are developed. Then these future scenarios are used to parameterize co-simulation setups. The last step of the SEP is the evaluation of sustainability of the simulation results and future scenario descriptions based on multi-criteria decision making.

The information model supports the SEP and allows to model parameters, dependencies, and data flows of the simulation and evaluation [23]. As the process aims to allow the cooperation in an interdisciplinary project team, the focus lies on usability for a diverse group of domain experts. Thus, the modeling takes place in a mind map tree structure and a plug-in has been implemented to transform the content to a resource description framework (RDF) file for further automated processing. The modeling can start with the first rough planning of the objectives of a simulation in a brainstorming in the interdisciplinary project team. As the planning progresses, the structure of the system model has to be refined to comply to the structure of the information model. This allows the transformation of the output into further usable data formats.

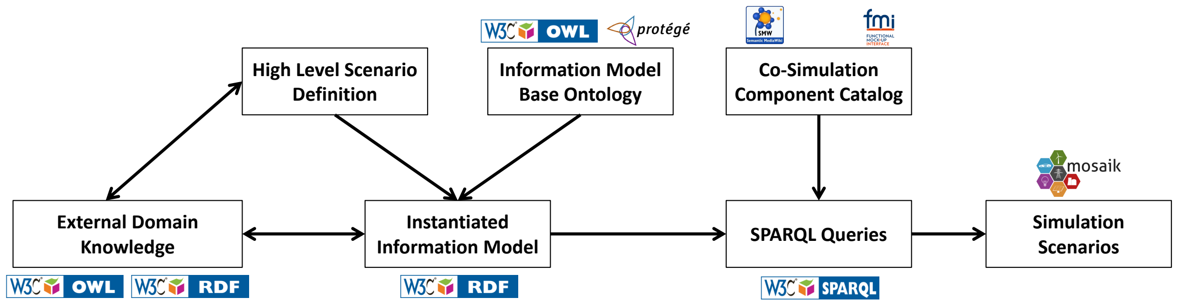
High Level Scenario Definition
Information Model Base Ontology
Co-Simulation Component Catalog
External Domain Knowledge
Instantiated Information Model
SPARQL Queries
Simulation Scenarios
OWL
protégé
RDF
SPARQL
fmi
mosaik

Figure 3. Overview of the simulation planning process.

This high-level scenario description in the information model is the basis for the development of executable simulation scenarios. Especially for large-scale scenarios with simulation models from many different domains, the development of these scenarios is complex and the experts may need assistance. Additionally, a simulation scenario should be validated regarding its completeness, coherence, and correctness. Therefore, semantic web technologies are used to structure the available information and allow the seamless integration of external domain knowledge. The information model structure is described by a base ontology [24]. Thus, the content of the information model is available for querying with the SPARQL query language.

For the development of executable simulation scenarios, the content of the information model has to be combined with information about the available simulation models. For this, a standardized interface description of the simulation models is needed. FMI can offer technical specification of the interface [4], but additional information about the simulation models is required. To collect this information, a questionnaire has been developed which allows users to describe the characteristics of a simulation model. For example, the used MOSAIK interface technology, temporal and spatial resolution, or the modeled domains are described. The questionnaire has been implemented in a semantic media wiki (SMW) (https://www.semantic-mediawiki.org) [25] as a so-called co-simulation component catalog. All available simulation models can be described based on the questionnaire. The information stored in the SMW is also available to SPARQL queries. With these queries, the information model and the co-simulation component catalog can be accessed and the simulation and domain experts can be assisted in choosing suitable simulation models for a scenario.

The information from the co-simulation component catalog and information model can also be used for the validation of scenarios and the coupling of simulation models. For example, the simulation models are described by their inputs and outputs in form of FMI variables with their units and can be checked for reasonable coupling (e.g., a simulation model providing *kW* data is not directly coupled with a model expecting *MW* data).

All in all, the presented simulation planning process provides a usable information model that at the same time allowes users to express complex features of multi-domain systems as well as simulation goals. It thus serves as a basis for expert collaboration and establishes a common understanding between co-simulation experts and researchers from other domains, handling a considerable challenge in CPES research. The concepts of a component catalog and SPARQL queries even provide the basis for a more automated process so that executable co-simulation setups can be derived quickly from first domain expert discussions, leading to faster iteration cycles.

2.4. Uncertainty Quantification

A considerable challenge of simulation-based testing is the handling of uncertainty. In this context, uncertainty describes the discrepancy between simulation output and real-world observations that stems from measurement errors in data, system stochasticity, and the simplified nature of models. Since uncertainty can never be fully eliminated from a simulation, the process of uncertainty quantification (UQ) is instead applied to allow users to make quantitative statements about the accuracy/uncertainty of their simulation results (e.g., [26]).

UQ is especially challenging in CPES co-simulation. On the one hand, the combination of different simulators and data sources can lead to the mixture of different forms and magnitudes of uncertainty. Some simulators, e.g., may produce data high with uncertainty that is propagated through the system and dominates the overall output uncertainty. Furthermore, different simulators may need different types of mathematical uncertainty representation (e.g., [27]). On the other hand, the simulation of highly interdisciplinary systems leads to an additional effort in the assessment of input uncertainties. The UQ process cannot be conducted by a single analyst but needs information from a variety of experts and data sources. Thus, it is likely that different states of knowledge have to be consolidated.

The UQ process can be separated into the two subprocesses of uncertainty assessment and uncertainty propagation. During uncertainty assessment, users identify the uncertainty sources in their simulation setup. Most of these sources typically lie in the input data, but simplifying assumptions made in the modeling process can also be seen as uncertainty sources. After these initial uncertainties have been identified, they must receive a quantitative representation. This is typically done by modeling the data uncertainty as probability distributions, but intervals or probability boxes may also serve as uncertainty models (e.g., [28]). The information required to conduct uncertainty assessment may stem from different sources, be it data analysis, model comparison or expert knowledge. Once the assessment is concluded, the initial uncertainties are propagated through the simulation model. That means that calculations are conducted to trace how initial uncertainties are combined and transformed by model operations to form uncertainty in the simulation output. In CPES co-simulation, this has to be done in a non-intrusive manner since simulation models are treated as black boxes and usually cannot be manipulated on code basis to allow other forms of UQ. The procedure of conducting uncertainty assessment followed by propagation is called a forward problem of UQ. In contrast, the inverse problem, associated with the technique of Bayesian inference, involves comparison of UQ output with measurement data to adjust the initial uncertainties and gradually improve the UQ quality. While this method has gained a lot of attention in the recent years, it is not considered practical for co-simulation of future energy systems due to the lack of the necessary measurement data.

The MOSAIK environment accounts for uncertainty with the help of the module *MoReSQUE* [29] that introduces an ensemble-based UQ process into the co-simulation. The module provides a propagation process that is conducted for each employed simulator individually. This is possible in a non-intrusive manner by replacing single simulation model instances by an ensemble of models that are able to process different input and parameter values. This provides the potential for the application of different propagation algorithms like Monte–Carlo simulation, polynomial chaos expansion (e.g., [30]), or new types of methods [31]. Depending on the employed propagation algorithms different types of uncertainty models can be supported. The corresponding uncertainty information is stored in so-called uncertainty structures that are exchanged between ensembles during runtime. Uncertainty structures can be either constructed from data sets or modeled directly by domain experts. This allows for knowledge integration from simulation model providers and other researchers into a collaborative UQ process.

2.5. Multi-Agent System

A multi-agent system (MAS) is a system that consists of interacting, autonomous agents within a certain environment. According to [32] an agent is a computer system that acts autonomously in order to fulfill its design objectives. It can perceive information from its environment and (re-)act on it, hence, possibly influencing or changing the state of the environment. Intelligent agents may have social abilities meaning they are capable of communicating with other agents in the same environment. Consequently, within a MAS, the agents can interact, i.e., exchange information, coordinate, organize or negotiate with each other in order to achieve their individual but also common goals.

A multi-agent simulation uses a MAS-model in order to investigate the behavior of the MAS. Because of the interaction among the agents and between agents and the environment, dynamics might occur that cannot be foreseen when designing the MAS. When such emergent behavior gets revealed in a multi-agent simulation, it can be analyzed and the root cause be identified. In case the emergent behavior is undesirable, interaction rules and constraints between the agents can be established in order to prevent such unwanted emergence.

An advantage of MAS is their flexible way of modeling different components as well as their interactions. This is especially beneficial in a complex system composed from many heterogeneous interacting components such as the energy system. In CPES, MAS can be used to model and simulate components and their interactions across multiple domains with the aim to understand the system behavior better and analyze possible emergence. The foundational concept of abstracting interdependent components to agents and their interaction makes MAS a fitting programming paradigm for CPES.

Moreover, MAS can be used as an engineering tool for developing optimization strategies of system components such as for instance controllable DER. In more detail, energy units can be represented by agents who can locally optimize the utilization of their flexibility. At the same time, these agents can coordinate among each other to achieve common objectives in a distributed and self-organized manner. In other words, the main goal of MAS is to achieve a desired system state while the individual objectives of agents serve as constraints in the process of reaching this state. One possible organization form is a virtual power plant (VPP), which is an aggregation of different, possibly heterogeneous and distributed DER that coordinate their behavior in order to act as a single power plant. Objectives of VPP can be economic, i.e., market-based provision of energy (see e.g., [33]) but also technical for grid-stabilizing services such as provision of primary control reserve [34] or congestion management [35].

ISAAC (https://github.com/mtroeschel/isaac) is an energy unit aggregation and planning software [36]. It is already coupled with MOSAIK, which allows to simulate scenarios for testing purposes before deploying agent-based control strategies to the field. The coupling is implemented via a flexible API that can be modified given the needs of the study at hand. ISAAC encompasses a MAS based on AIOMAS (https://aiomas.readthedocs.io), a lightweight framework written in Python that supports the implementation of MAS in the understanding of distributed process coordination. ISAAC's agents implement a modified version of the COHDA algorithm, which is a optimization heuristic for distributed agents [37]. The agents each represent an individual energy unit in the negotiation, to which end they need to know the capabilities of their unit. For example, the power output of a wind power plant is limited by the available wind and the unit controller might only have a fixed number of set points. Unit capabilities can be described in multiple ways, one of which is based on the sampling of alternative schedules from a unit model, each denoting a valid schedule under their unit's constraints. This abstraction of unit flexibility is a general solution as it can be applied to virtually any energy unit type. Complementing this generality property, COHDA also shows the scalability necessary to aggregate many small DER, as required by the CPES [36]. The quality of COHDA's solution is measured by its performance, which is defined as the difference between the solution and an optimization target. As the target depends on the use case to be fulfilled by ISAAC, the performance is also a use-case specific metric.

As CPES are critical infrastructures, guaranteed behavior is needed in some application areas (see Section 2.6 for such an example). In order to fulfill such guarantees and prevent undesired behavior, ISAAC embeds COHDA in an observer/controller architecture. This architecture introduces new types of agents in addition to the negotiating COHDA agents. The new agents encompass an observer agent and a controller agent. The observer agent receives data from the agents in the MAS during runtime. This information can be, e.g., the state of the optimization process or the quality of solutions. The controller, on the other hand, is responsible for altering the system behavior. It receives the information provided by the observer and can act on it by deciding on control actions to alter the optimization process. In general, it acts as an interface between the real-world units and their representing agents by informing the agents about the overall objective of the VPP. Objectives do not necessarily have to be fixed but can change over time depending on the system state of the power system. Additionally, the controller is responsible for assuring termination of a negotiation within a desired time. The state of the MAS and the progress of its negotiation is supervised by the observer. The observer/controller architecture preserves the benefits of agent-based control strategies like the ability to run agents distributed in the field with COHDA while avoiding many kinds of unwanted behavior like late termination and convergence towards local optima.

Due to the modular character of MOSAIK, neither changing the control strategy nor the power system setup leads to additional engineering overhead. This has made realizing several studies possible, which use the coupled ISAAC-on-MOSAIK setup (see Section 4.2).

2.6. Example Application Case

The example case study represents a section of a distribution grid in which a technical VPP (TVPP) is located. The TVPP aggregates and optimizes the scheduling units with respect to services that can be offered to the grid operator. In the example at hand, the TVPP consisted of three wind turbines and two industrial loads that can be used for congestion management to prevent transformer overloads. In addition to these controllable and flexible units, some inflexible units like households and industrial loads were situated in the distribution grid.

Figure 4 depicts the example case and the corresponding simulation setup. In the chosen scenario, grid congestions can result from overloading the transformer, which connects the distribution grid to the transmission grid. The threat of transformer overload exists due to excessive feed-in of wind turbines with high simultaneity. To avoid permanent damage by overloading grid equipment, a solution that mitigates grid congestions needs to exist before the equipment suffers irreparable harm.

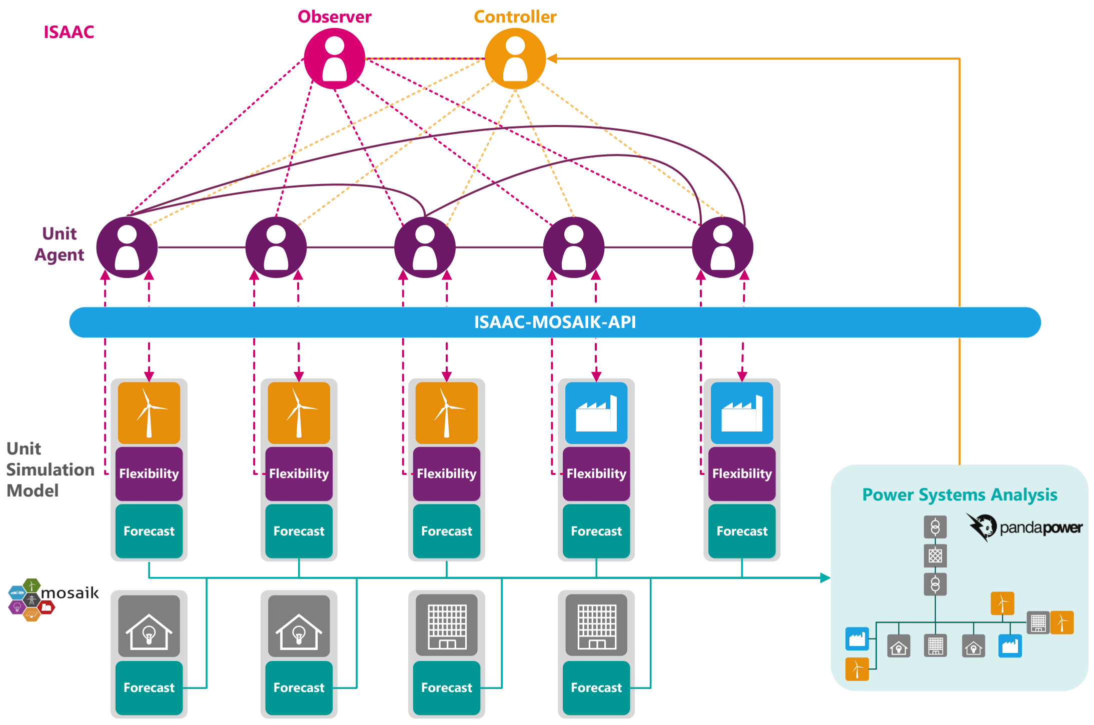

Figure 4. Scenario of exemplary application case.

The example has been implemented as a MOSAIK scenario. Each simulation step includes the simulation of the system operation as well as the planning process of the TVPP.

The planning process is as follows: During a simulation step, a forecast of the upcoming power consumption or production of each of the units is performed and forwarded to the power system analysis tool Pandapower [38]. In this tool, power flow calculations for the grid section are performed on the power forecasts. Based on its calculated results Pandapower passes technical restrictions to the controller agent of ISAAC. In case of an imminent transformer overload, the controller agent initiates a negotiation between the unit agents. The objective of this negotiation is to find an overall aggregated schedule, comprised of the individual schedules of each unit, which mitigates the transformer overload.

To achieve mitigation, all unit agents receive the flexibilities (e.g., a set of potential schedules) from their unit simulation models. The flexibilities only allow schedule changes that take local restrictions into account. In the example case, the congestion can be prevented either by capping of the feed-in of the wind turbines or by load shifting of the industrial consumers. After the negotiation, each agent sends the determined schedule to its corresponding unit simulation model.

In order to simulate the operation of the system, the power values of the units are also transmitted to Pandapower in each simulation step. These inputs are used for a load flow calculation that reveals whether the planning process of the TVPP has been successful.

Given the right parameterization, this example is able to show how highly simultaneous feed-ins of wind energy collectors can cause a grid congestion by provoking an overloading of a transformer and how a TVPP can use the flexibility of its units to mitigate said congestion.

3. Results

3.1. Planning Process

For setting up a co-simulation as in the application case described in Section 2.6, the typical first step is the definition of the objective. In the information model, the evaluation criteria of a co-simulation can be described (for more details about the information model see [23]). In the example application case, this would be the violation of a transformer limit, as shown on the right-hand side in Figure 5. On the left-hand side, the considered domains (in this example only the energy domain), domain objects, and attributes are shown. The arrows describe data flows between the attributes, simulation models, and the evaluation as described in Section 2.6. For the example application case, the modeling was not too complex, but for the planning of large-scale co-simulation scenarios, the manual development gets more complex and demands assistance by querying the information model, e.g., to find evaluation functions without input, get assistance in finding suitable simulation models or find dependencies between different simulation models.

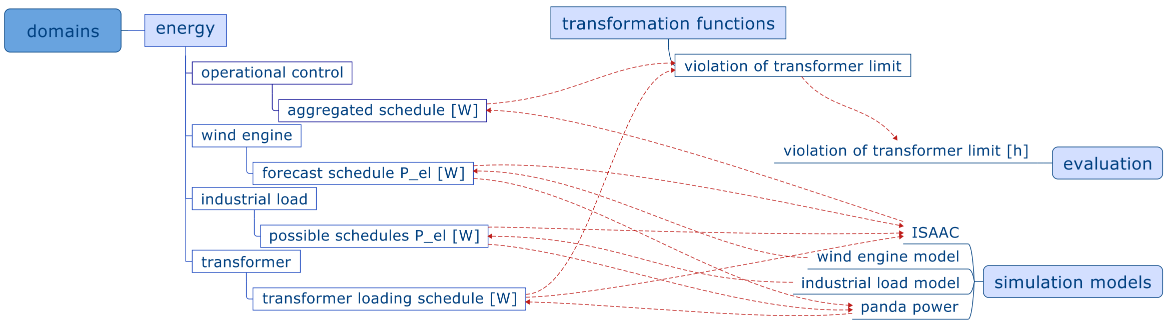

Figure 5. Example of the high-level simulation planning information model.

After the definition of the objective, suitable simulation models have to be found or developed. Therefore, the simulation planning process can assist the user in finding suitable simulators for their use case, mitigating the need to develop new simulation models. The query in Figure 6a shows how the information model and the co-simulation component catalog can be used for this. It is easily imaginable that for a representation of wind turbines no new simulation model should be developed, but an already existing one can be chosen. For that use case, a query can be run to find suitable simulation models in the co-simulation component catalog. The results are filtered by the function "generation/conversion" and temporal resolution of 1 or 15 min.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX cosicoca: <https://mosaik.offis.de/cosicoca/>
SELECT ?component ?energySectors ?usedMosaikAPI ?openSource ?functions ?temporalRes
WHERE {?component rdf:type cosicoca:Component .
         ?component cosicoca:functions ?functions .
         ?component cosicoca:temporalRes ?temporalRes .
         ...
         FILTER regex(?functions ,"generation/conversion")
         FILTER (?temporalRes = "1␣min" || ?temporalRes = "15␣min")
}
```

(**a**) SPARQL code

component	energySectors	usedMosaikAPI	openSource	functions	temporalRes
wikiComponent:solarplantsim	heat	none	false	generation/conversion	1 min
wikiComponent:chpsim	electricity, heat	Python	false	generation/conversion	1 min
wikiComponent:condensingboilersym	heat	none	false	generation/conversion	1 min
wikiComponent:WindTurbine	electricity	Python	true	generation/conversion	1 min, 15 min

(**b**) Results

Figure 6. SPARQL query to find suitable simulation models.

Additionally, the attributes for the considered energy sectors, the used interface technology, and the availability as open source software are shown to decide for a suitable model in the results (see Figure 6b). The result of the query shows that four components meet the filter and the simulation model WindTurbine would be the suitable one for the application case.

3.2. Test Outcome

Figure 7 shows the results of the example case described in Section 2.6. In each plot, the schedules of all units within the TVPP are stacked upon each other by addition, while the blue line shows the aggregated schedule. The red areas describe the transformers upper and lower bound that has been derived by calculations in Pandapower.

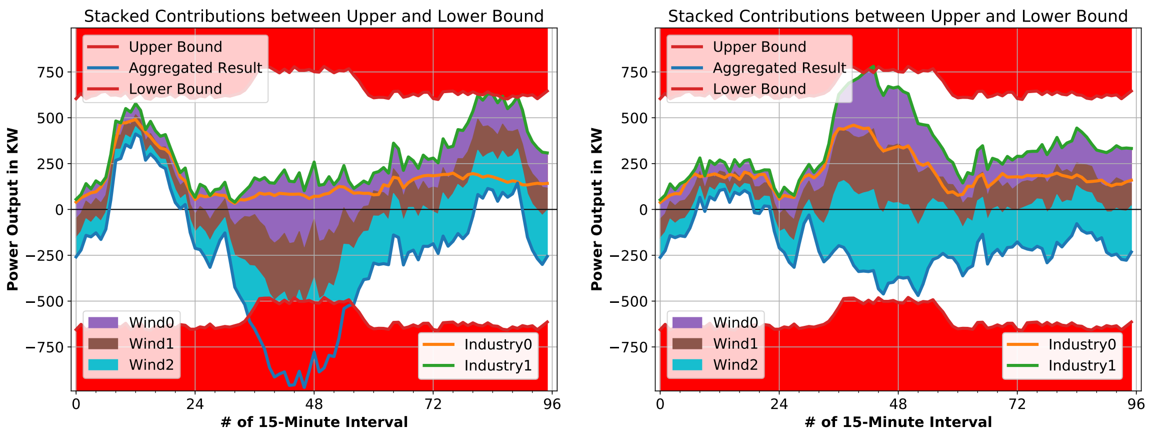

Figure 7. Simulation results of the technical virtual power plant (TVPP) in the example case described in Section 2.6. The left plot shows the situation before optimization of the TVPP and the right plot after optimization.

The left plot shows the case in which the TVPP does not offer any congestion management while the right plot exhibits the results when the TVPP optimizes their scheduled units towards the mitigation of grid congestions. In the latter case, the aggregated load of the TVPP remains within the requested boundaries, because parts of the industrial loads are shifted towards the period in which high feed-in from the wind turbines is forecasted. The results show that the TVPP is able to optimize the schedules of its units such that the requirements calculated in the power system analysis are met.

Load shifting has been conducted via employment of different schedules for the industrial loads. In other words, the representing agent of a load has the information of all acceptable schedules for that load. In this example, all of these schedules are treated as equally valid and no monetary constraints are associated with the changing of schedules. While this simplification allows for a more comprehensible demonstration case, ISAAC still provides all capabilities for more complex modeling of individual DER constraints.

3.3. Uncertainty Analysis

The complexity of an UQ process increases drastically with the number of uncertainty sources in a setup (an effect known as the curse of dimensionality). Therefore, it is a good practice to exclude potential uncertainty sources from the UQ process that are likely to be irrelevant or have a negligible effect on the output uncertainty. This is typically done with sensitivity analysis (SA), but in cases like the presented application case, expert assessment can suffice to reduce the UQ complexity.

First of all, uncertainty of the MAS can be excluded from the consideration since this is the tested algorithm that does not represent a physical real-world system. This leaves potential uncertainty in the load schedules and the wind power feed-in. However, load development can typically be predicted very well. Furthermore, a number of the loads are highly flexible and their schedules may be changed by the MAS. Thus, their uncertainty is not considered to have a significant effect on the outcome. Uncertainty in the wind power data, on the other hand, is considered relevant since it is reasonable to assume that power feed-in data that is transmitted to the MAS may be afflicted with a measurement error that is additionally linked to a prediction error.

In this case, a simple worst case analysis is conducted for demonstration purposes by assuming a maximal data error of ±10%. Therefore, the UQ can be conducted with an interval representation of the uncertainty. The results (Figure 8) show that stronger wind power feed-in can cause the MAS to not fully meet the target schedule – although the resulting congestion is still far less severe than in the case without control.

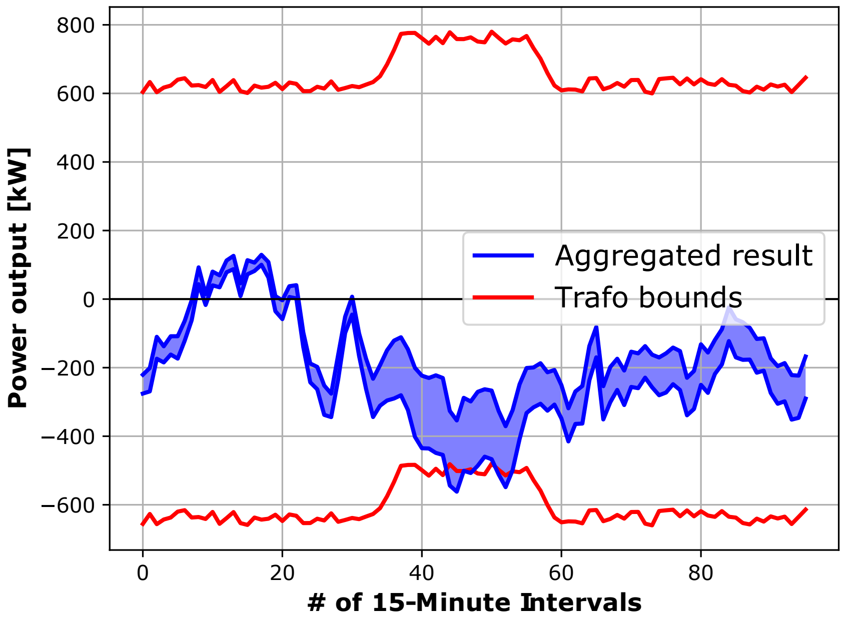

Figure 8. Uncertainty of the aggregated schedule in comparison with load limits imposed by the transformer.

Note that the current ISAAC implementation does not include robust optimization features that work with uncertainty information. In such a case, it would have been reasonable to employ probabilistic uncertainty models to provide the optimization with a better basis for decision making. In the current setup, however, the uncertainty assumptions suffice to demonstrate the importance of UQ even in such simple cases. While UQ with interval uncertainty and a small number of simulators can likely be conducted with simple adjustment of the code, more complex co-simulation scenarios and consideration of probabilistic uncertainty quickly illustrate the need for dedicated tools like MoReSQUE.

4. Discussion

4.1. Benefits and Trade-Offs

As mentioned in Section 1, the quality of CPES testing platforms can be assessed regarding different aspects and measures. Therefore, each platform has a different design goal and fills a different niche. The MOSAIK co-simulation framework is designed to include a broad range of simulation models from different domains (flexibility). Furthermore, the system should be usable by CPES researchers that are not necessarily co-simulation experts (usability).

Usability of co-simulation is partly achieved here through a small number of software dependencies in the framework since this leads to an easier deployment process. It has to be noted, however, that the upcoming concept of simulation as a service (SaaS) presents an alternative solution to this challenge. In SaaS solutions, simulation software is executed on a cloud platform and is only accessible to researchers via a user interface. This has the benefit of eliminating the deployment process and additionally allows the utilization of more powerful, dedicated computation hardware. In the future, MOSAIK will likely also be provided in a SaaS context, but for now a focus is put on local deployment to allow for shorter testing an debug cycles of new co-simulation setups.

In addition to the slim code base, the centralized scheduling concept of MOSAIK is supporting its usability. In a centralized setup, all data exchanged is treated in the same way so that no additional configuration is needed. More distributed systems, like those based on HLA, typically require the user to provide more specification on how one simulator interacts with the others. This obviously has the benefit of supporting a wider range of application cases and allows some performance boosts, but users may at times be overwhelmed by the number of options. MOSAIK, thus, displays a trade-off between usability on the one side and higher performance and flexibility on the other side. Practical experience has shown, however, that the scheduling capabilities provided by MOSAIK are sufficient for a wide range of application cases (see Section 4.2).

All in all, it is important to note that the structure of a co-simulation framework alone cannot satisfy all the usability or performance needs of CPES researchers. Therefore, an extended environment with tools and processes is suggested here that adds functionalities, e.g., in the selection of simulation components or in the development of MAS-based CPES solutions. Further planned extensions are discussed in Section 4.3.

4.2. Further Application Examples

The MOSAIK co-simulation framework has already been applied by different institutions in a variety of CPES test cases [39–41], demonstrating features like interoperability with real-time laboratories [21], the possibility of simulator exchange [20], or distributed coordination of heterogeneous simulation components [42].

In the project NEDS, MOSAIK has been used to simulate different parts of a smart grid in energy scenarios of the years 2050 and transition years. In this setting, ISAAC was coupled with a smart home model, optimizing the flexibility usage of domestic loads with regard to congestion avoidance and cost minimization. For the development of the energy scenarios and their evaluation of sustainability, the SEP has been developed [23]. Supported by an information model, the scenarios have been modeled in this interdisciplinary project with partners from different domains, like electrical engineering, computer science, business administration, economics, and psychology.

In the project Smart Nord (http://smartnord.de/ (only available in German)), MOSAIK has been used for different purposes [43]. One major task has been the testing of agent-based distributed optimization strategies with different objectives along the value chain of a dynamic VPP [33]. Aside from that, market-based re-dispatch solutions have been evaluated with MOSAIK to test whether the associated products can help to prevent congestions in the system [44].

Another approach for congestion management was developed in the project Designetz (https://www.designetz.de/) where a system cockpit (SC) was developed as an intelligent user interface for distribution system operators. It allows to identify congestions and initiate mitigation actions using various flexible units. For testing the functioning, the process of congestion management implemented in the SC architecture with many heterogeneous components MOSAIK is used as a connecting middleware. Those simulators include a flexibility optimizing heuristic (ISAAC), a database encapsulating real-world units, and a planning tool for determining the power system status. For details, see [35].

The project LarGo! (http://www.largo-project.eu/) is focused on the large-scale roll-out of CPES solutions [45]. MOSAIK is used to couple simulators for testing the roll-out and deployment process of smart grid applications. This includes power system simulation and ICT simulation.

While most of the project mentioned so far are focused on purely software based CPES testing, the MOSAIK environment is also employed in the hardware-related European projects uGrip (http://www.ugrip.eu/) and ERIGrid (https://erigrid.eu/). The former is focused on the hardware integrated testing of microgrid solutions while the latter is contributing to the improvement of laboratory and co-simulation platforms for CPES testing.

4.3. Future Work

The current MOSAIK scheduling algorithm can, as mentioned above, be interpreted as a flexible discrete time scheduling. While such a solution works well for a variety of different application cases, it fails to fully support some crucial aspects of CPES, like communication systems. In the time discrete scheduling, each step performed by a simulator predicts the time of the next step. A communication simulation, however, introduces a delay into a system's message exchange. As a consequence, a simulator may not be able to predict its next step on its own but rather needs information from other simulators. The only way of handling this problem in a discrete time setup is to reduce the step size for simulators to the length of the minimal possible delay and check during each step whether data has been received. A more efficient solution, on the other hand, is given by the employment of DES scheduling that allows simulators to be triggered by events and stay idle until then. Such a scheduling option is currently developed for MOSAIK. It is planned to be available as an additional scheduling module for the framework that is interoperable with the current scheduling (similar to the approach seen in Ptolemy II).

It has been discussed in [46] (and above) that different co-simulation frameworks may be more suitable for different application cases. As an example, MOSAIK provides benefits for co-simulation prototyping, but may fall short on strong real-time requirements in some hardware integrated test cases. Therefore, it is planned to have the assets of the outlined testing environment (simulation planning, UQ, ISAAC) be interoperable with other frameworks, e.g., HLA-based systems. Accordingly, a tool-invariant process for test realization is currently developed that will include a classification of co-simulation services to find the best framework for a given application case.

Despite MOSAIK's focus on usability, the improvement of the platform performance is also an ongoing research field. Since co-simulation performance is highly dependent on the employed simulators, an approach is currently developed that allows the integration of surrogate models into co-simulation. In other words, machine learning algorithms are trained to replicate the input-output relationship of more complex simulation models. This way computationally expensive simulators or sets of simulators can be replaced by surrogates to boost the overall process performance. UQ methods are used to ensure a sufficient accuracy in the resulting co-simulation.

The outlined UQ system MoReSQUE is so far limited to analyzing the impact of simulation input uncertainties on the output accuracy. However, there are other aspects of co-simulation uncertainty that may play an important role for the assessment of result reliability. On the one hand, coupling uncertainty between simulators has to be analyzed, e.g., resulting from different model resolutions. Additionally, simulator-internal uncertainty has to be assessed via model validation techniques. This is typically done by comparing simulator output with measurement data or other simulators in a cross validation. However, additional research is needed on methods of model uncertainty representation in co-simulation as well as the combination of validation information from different sources due to the interdisciplinary nature of CPES. These topics will form the next steps in UQ research for the outlined testing environment. Next to that, improved approaches are needed for the integration of expert knowledge into the UQ process. One possible solution is the availability of user-friendly tools for uncertainty modeling as presented in [47]. For a more sophisticated collaboration process, it is planned to include UQ considerations directly into the information model presented in Section 2.3. Finally, validation of UQ with measurement data has to be researched more thoroughly. Such approaches are so far sparse in the energy field due to the limited availability of measurement data from complex CPES. Nevertheless, techniques like Bayesian inference (e.g., [48]) are important to correct potentially false assumptions in input uncertainties and thus avoid biases in the assessment of output reliability.

Regarding ISAAC, one of the main development goals for the future is to increase the system's modularity with respect to the structure of the agents, the components included, the optimization heuristic used and the use case under investigation. Furthermore, an increased focus of ISAAC will be placed on supporting decisions regarding the design of a MAS before it is applied to the field (e.g., the optimal degree of (de-)centrality). ISAAC shall hence serve as a flexible and modular simulative testbed for the application of autonomous and intelligent agents in a smart grid. Aside from that, the tool will be used to test the suitability of MAS solutions for a variety of CPES challenges like the mitigation of cyber-attacks or the restoration of power supply after a blackout.

The long-term goal of the simulation planning process is the automated transformation of a high-level scenario description into an executable co-simulation scenario to allow SaaS applications. By describing the semantics of data, validation of co-simulation scenarios may also be enabled regarding the units of connected simulation models, the domain and resolutions, and the dimension. In its current state, the information model for high-level simulation planning has been used with the SEP for the definition of energy scenarios. For the future, the approach is to be extended to more general simulation planning, e.g., for testing cyber-security solution in the smart grid. Additionally, integrating the simulation planning more strongly with the MOSAIK framework via automated code generation and checking is planned. A more extensive scenario planning and evaluation process is also a key part of decision maker support. In upcoming work, the presented testing environment will be extended by a support tool to analyze the effect of energy system policies. The goal of the research is the testing of realizable policies for CPES under market constraints to minimize uncertainties and support policy makers.

5. Conclusions

The paper at hand suggests a more prominent role of testing environments in the development of CPES setups and components. Especially co-simulation is a promising approach to allow for more prepared and efficient laboratory and field tests. Furthermore, new solutions can be more easily validated under large-scale requirements. The co-simulation framework MOSAIK is designed as a usable and flexible solution for CPES testing across multiple domains, calling for the collaboration of various experts. In its current form, the framework is extended by an environment that includes a process for the selection of the most appropriate simulation tools. Additionally, capabilities for uncertainty assessment of test results are provided. These extensions are aimed at broadening the focus of generic co-simulation approaches, involving not only the technical issues of tool coupling, but also the challenges associated with collaboration of different experts possessing different knowledge and perspectives. Finally, the ISAAC software represents a natively coupled toolbox that allows the development and analysis of MAS-based control and optimization solutions for CPES. In the near future, further extensions of the environment are planned to improve the performance, flexibility and usability of CPES testing.

Author Contributions: C.S. has contributed to the introduction, description of MOSAIK, the discussion and summary. Additionally he has conducted the uncertainty quantification. M.B.B. has contributed to the description of multi-agent systems, ISAAC and further applications. A.E.-A. and T.R. have participated in the software development and contributed to the description of MOSAIK. S.H., B.L. and M.N.-W. have contributed to the description of ISAAC and designed, conducted and analyzed the demonstration case. R.P.R.A., S.S., A.N. and S.L. have contributed to conceptualization and development and the description of further applications and future work. J.S.S. has developed the simulation planning process and contributed in its description and application.

Acknowledgments: Parts of this work received funding in the European Community's Horizon 2020 Program (H2020/2014-2020) under the project "ERIGrid" (Grant Agreement No. 654113) and the ERA-Net Smart Grids Plus "uGrip" (Grant Agreement no. 77731).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

API	application programming interface
CPES	cyber-physical energy system
DER	distributed energy resource
DES	discrete event simulation
FMI	functional mock-up interface
HLA	high-level architecture
ICT	information and communication infrastructure
MAS	multi-agent system
RDF	ressource description framework
SA	sensitivity analysis
SaaS	simulation as a service
SC	system cockpit
SEP	sustainability evaluation process
SMB	simulation message bus
SMW	semantic media wiki
TVPP	technical virtual power plant
UQ	uncertainty quantification
VPP	virtual power plant

References

1. Nieße, A.; Tröschel, M.; Sonnenschein, M. Designing Dependable and Sustainable Smart Grids—How to Apply Algorithm Engineering to Distributed Control in Power Systems. *Environ. Model. Softw.* **2014**, *56*, 37–51. [CrossRef]
2. Palensky, P.; Van Der Meer, A.A.; López, C.D.; Joseph, A.; Pan, K. Cosimulation of Intelligent Power Systems: Fundamentals, Software Architecture, Numerics, and Coupling. *IEEE Ind. Electron. Mag.* **2017**, *11*, 34–50. [CrossRef]
3. Palensky, P.; van der Meer, A.A.; López, C.D.; Joseph, A.; Pan, K. Applied co-simulation of intelligent power systems: implementation, usage, examples. *IEEE Ind. Electron. Mag.* **2017**, *11*, 6–21. [CrossRef]
4. Blochwitz, T.; Otter, M.; Arnold, M.; Bausch, C.; Clauß, C.; Elmqvist, H.; Junghanns, A.; Mauss, J.; Monteiro, M.; Neidhold, T.; et al. The Functional Mockup Interface for Tool independent Exchange of Simulation Models. In Proceedings of the 8th International Modelica Conference, Dresden, Germany, 20–22 March 2011; pp. 173–184. [CrossRef]
5. Dahmann, J.S.; Fujimoto, R.M.; Weatherly, R.M. The Department of Defense High Level Architecture. In Proceedings of the 1997 Winter Simulation Conference, Atlanta, GA, USA, 7–10 December 1997; pp. 142–149. [CrossRef]

6. Neema, H.; Sztipanovits, J.; Burns, M.; Griffor, E. C2WT-TE: A Model-Based Open Platform for Integrated Simulations of Transactive Smart Grids. In Proceedings of the Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES), Vienna, Austria, 11 April 2016.
7. Davis, J., II; Goel, M.; Hylands, C.; Kienhuis, B.; Lee, E.A.; Liu, J.; Liu, X.; Muliadi, L.; Neuendorffer, S.; Reekie, J.; et al. *Overview of the Ptolemy Project*; Technical Report, ERL Technical Report UCB/ERL; University of California: Berkeley, CA, USA, 1999.
8. Wetter, M. Co-simulation of building energy and control systems with the Building Controls Virtual Test Bed. *J. Build. Perform. Simul.* **2011**, *4*, 185–203. [CrossRef]
9. Galtier, V.; Vialle, S.; Dad, C.; Tavella, J.P.; Lam-Yee-Mui, J.P.; Plessis, G. FMI-based distributed multi-simulation with DACCOSIM. In Proceedings of the Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium. Society for Computer Simulation International, Alexandria, VA, USA, 12–15 April 2015; pp. 39–46.
10. Vaubourg, J.; Presse, Y.; Camus, B.; Bourjot, C.; Ciarletta, L.; Chevrier, V.; Tavella, J.P.; Morais, H. Multi-agent multi-model simulation of smart grids in the MS4SG project. In Proceedings of the International Conference on Practical Applications of Agents and Multi-Agent Systems, Salamanca, Spain, 3–5 June 2015; pp. 240–251.
11. Marten, F.; Vogt, M.; Widdel, M.; Wickert, M.; Meinl, A.; Nigge-Uricher, M.; Töbermann, J.C. Real-time simulation of Distributed Generators, for testing a Virtual Power Plant software. In Proceedings of the E-World Energy & Water, Essen, Germany, 10–12 February 2015.
12. Faschang, M.; Kupzog, F.; Mosshammer, R.; Einfalt, A. Rapid Control Prototyping Platform for Networked Smart Grid Systems. In Proceedings of the Annual Conference of the IEEE Industrial Electronics Society, Vienna, Austria, 10–13 November 2013. [CrossRef]
13. Schlögl, F.; Rohjans, S.; Lehnhoff, S.; Velasquez, J.; Steinbrink, C.; Palensky, P. Towards a Classification Scheme for Co-Simulation Approaches in Energy Systems. In Proceedings of the International Symposium on Smart Electric Distribution Systems and Technologies (EDST), Vienna, Austria, 8–11 September 2015.
14. Vogt, M.; Marten, F.; Braun, M. A survey and statistical analysis of smart grid co-simulations. *Appl. Energy* **2018**, *222*, 67–78. [CrossRef]
15. Schütte, S.; Sonnenschein, M. Mosaik—Scalable smart grid scenario specification. In Proceedings of the 2012 Winter Simulation Conference, Berlin, Germany, 9–12 December 2012.
16. Epstein, J.M. Agent-Based Computational Models And Generative Social Science. *Complexity* **1999**, *4*, 41–60. [CrossRef]
17. Klügl, F.; Rindsfüser, G. Large-Scale Agent-Based Pedestrian Simulation. In Proceedings of the 5th German Conference on Multiagent System Technologies (MATES 2007), Leipzig, Germany, 24–26 September 2007.
18. McArthur, S.D.J.; Davidson, E.M.; Catterson, V.M.; Hatziargyriou, N.D.; Funabashi, T. Multi-Agent Systems for Power Engineering Applications—Part I: Concepts, Approaches, and Technical Challenges. *IEEE Trans. Power Syst.* **2007**, *22*, 1743–1752. [CrossRef]
19. Zimmerman, R.D.; Murillo-Sánchez, C.E.; Thomas, R.J. MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education. *IEEE Trans. Power Syst.* **2011**, *26*, 12–19. [CrossRef]
20. Lehnhoff, S.; Nannen, O.; Rohjans, S.; Schlögl, F.; Dalhues, S.; Robitzky, L.; Hager, U.; Rehtanz, C. Exchangeability of power flow simulators in smart grid co-simulations with mosaik. In Proceedings of the 2015 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES), Seattle, WA, USA, 13 April 2015; pp. 1–6.
21. Büscher, M.; Claassen, A.; Kube, M.; Lehnhoff, S.; Piech, K.; Rohjans, S.; Scherfke, S.; Steinbrink, C.; Velasquez, J.; Tempez, F.; et al. Integrated Smart Grid simulations for generic automation architectures with RT-LAB and mosaik. In Proceedings of the 2014 IEEE International Conference on Smart Grid Communications (SmartGridComm), Venice, Italy, 3–6 November 2014; pp. 194–199.
22. Grunwald, A.; Dieckhoff, C.; Fischedick, M.; Höffler, F.; Mayer, C.; Weimer-Jehle, W. *Consulting With Energy Scenarios: Requirements for Scientific Policy Advice*; Series on Science-Based Policy Advice; acatech—National Academy of Science and Engineering: Munich, Germany, 2016.

23. Schwarz, J.S.; Witt, T.; Nieße, A.; Geldermann, J.; Lehnhoff, S.; Sonnenschein, M. Towards an Integrated Development and Sustainability Evaluation of Energy Scenarios Assisted by Automated Information Exchange. In *Smart Cities, Green Technologies, and Intelligent Transport Systems*; Donnellan, B., Klein, C., Helfert, M., Gusikhin, O., Pascoal, A., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 3–26. [CrossRef] ,
24. Schwarz, J.S.; Lehnhoff, S. Ontology-Based Development of Smart Grid Co-Simulation Scenarios. In Proceedings of the EKAW 2018 Posters and Demonstrations Session (EKAW-PD 2018), Nancy, France, 12–16 November 2018; pp. 21–24.
25. Krötzsch, M.; Vrandecic, D.; Völkel, M.; Haller, H.; Studer, R. Semantic Wikipedia. *J. Web Semant.* **2007**, *5*, 251–261. [CrossRef]
26. Magnusson, S.E. *Uncertainty Analysis: Identification, Quantification And Propagation*; Department of Fire Safety Engineering, Lund Institute of Technology, Lund University: Lund, Sweden, 1997.
27. Ferson, S.; Ginzburg, L.R. Different methods are needed to propagate ignorance and variability. *Reliab. Eng. Syst. Saf.* **1996**, *54*, 133–144. [CrossRef]
28. Bruns, M.; Paredis, C.J.J. Numerical Methods for Propagating Imprecise Uncertainty. In Proceedings of the ASME Design Engineering Technical Conferences and Computers and Information Engineering Conference, Philadelphia, PA, USA, 10–13 September 2006.
29. Steinbrink, C. A Non-Intrusive Uncertainty Quantification System for Modular Smart Grid Co-Simulation. Ph.D. Thesis, Carl-von-Ossietzky Universität Oldenburg, Oldenburg, Germany, 2017.
30. Feinberg, J.; Langtangen, H.P. Chaospy: An open source tool for designing methods of uncertainty quantification. *J. Comput. Sci.* **2015**, *11*, 46–57. [CrossRef]
31. Steinbrink, C.; Lehnhoff, S. Quantifying Probabilistic Uncertainty in Smart Grid. In Proceedings of the Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES), Vienna, Austria, 11 April 2016.
32. Wooldridge, M. *An Introduction to MultiAgent Systems*; John Wiley & Sons Ltd.: Chichester, UK, 2002.
33. Nieße, A.; Beer, S.; Bremer, J.; Hinrichs, C.; Lünsdorf, O.; Sonnenschein, M. Conjoint Dynamic Aggregation and Scheduling Methods for Dynamic Virtual Power Plants. In Proceedings of the 2014 Federated Conference on Computer Science and Information Systems, Warsaw, Poland, 7–10 September 2014.
34. Lehnhoff, S.; Klingenberg, T.; Blank, M.; Calabria, M.; Schumacher, W. Distributed Coalitions for Reliable and Stable Provision of Frequency Response Reserve—An Agent-based Approach for Smart Distribution Grids. In Proceedings of the IEEE International Workshop on Intelligent Energy Systems, Vienna, Austria, 14 November 2013.
35. Erlemeyer, F.; Schmid, D.; Rehtanz, C.; Lüers, B.; Lehnhoff, S. Coordination and live testing of flexibility on distribution grid level. In Proceedings of the CIRED 25th International Conference on Electricity Distribution, Madrid, Spain, 3–6 June 2019; submitted.
36. Nieße, A.; Tröschel, M. Controlled Self-Organization in Smart Grids. In Proceedings of the 2016 IEEE International Symposium on Systems Engineering (ISSE), Edinburgh, UK, 3–5 October 2016.
37. Hinrichs, C.; Sonnenschein, M. A distributed combinatorial optimisation heuristic for the scheduling of energy resources represented by self-interested agents. *Int. J. Bio-Inspir. Comput.* **2017**, *10*, 69–78. [CrossRef]
38. Thurner, L.; Scheidler, A.; Schafer, F.; Menke, J.H.; Dollichon, J.; Meier, F.; Meinecke, S.; Braun, M. Pandapower-an open source Python tool for convenient modeling, analysis and optimization of electric power systems. *IEEE Trans. Power Syst.* **2018**, *33*, 6510–6521. [CrossRef]
39. Wang, K.; Siebers, P.O.; Robinson, D. Towards Generalized Co-simulation of Urban Energy Systems. *Procedia Eng.* **2017**, *198*, 366–374. [CrossRef]
40. Chromik, J.J.; Remke, A.; Haverkort, B.R. A Testbed for locally Monitoring SCADA Networks in Smart Grids. *Energy-Open* **2017**. [CrossRef]
41. Mirz, M.; Razik, L.; Dinkelbach, J.; Tokel, H.A.; Alirezaei, G.; Mathar, R.; Monti, A. A Cosimulation Architecture for Power System, Communication, and Market in the Smart Grid. *Complexity* **2018**, *2018*. [CrossRef]
42. Steinbrink, C.; Köhler, C.; Siemonsmeier, M.; van Ellen, T. Lessons learned from CPES co-simulation with distributed, heterogeneous systems. *Energy Inform.* **2018**, *1*, 38. [CrossRef]
43. Hofmann, L.; Sonnenschein, M. (Eds.) *Smart Nord—Final Report*; Hartmann GmbH: Hannover, Germany, 2015.

44. Wissing, C. Marktbasiertes Redispatch mit Flexibilitäten von Netznutzern für das Verteilnetz. Ph.D. Thesis, Carl von Ossitzky Universität Oldenburg, Oldenburg, Germany, 2015.
45. Kintzler, F.; Gawron-Deutsch, T.; Cejka, S.; Schulte, J.; Uslar, M.; Veith, E.M.; Piatkowska, E.; Smith, P.; Kupzog, F.; Sandberg, H.; et al. Large Scale Rollout of Smart Grid Services. In Proceedings of the 2018 Global Internet of Things Summit (GIoTS), Bilbao, Spain, 4–7 June 2018.
46. Steinbrink, C.; van der Meer, A.A.; Cvetkovic, M.; Babazadeh, D.; Rohjans, S.; Palensky, P.; Lehnhoff, S. Smart grid co-simulation with MOSAIK and HLA: a comparison study. *Comput. Sci. Res. Dev.* **2018**, *33*, 135–143. [CrossRef]
47. Ferson, S.; Hajagos, J.; Myers, D.S.; Tucker, W.T. *Constructor: Synthesizing Information about Uncertain Variables*; Technical Report; Sandia National Laboratories: Albuquerque, NM, USA, 2004.
48. Kennedy, M.C.; O'Hagan, A. Bayesian calibration of computer models. *J. R. Stat. Soc. Ser. B (Stat. Methodol.)* **2001**, *63*, 425–464. [CrossRef]