Using Cossembler for Rapid Prototyping of Co-simulations for Power System Operations

Cvetkovic, Milos; Gusain, Digvijay; Palensky, Peter

DOI
10.1109/PESGM40551.2019.8973884

Publication date
2019

Document Version
Final published version

Published in
2019 IEEE Power & Energy Society General Meeting (PESGM)

Using Cossembler for Rapid Prototyping of Co-simulations for Power System Operations

Miloš Cvetković, Digvijay Gusain, Peter Palensky
Faculty of Electrical Engineering Mathematics and Computer Science
Delft University of Technology
Delft, The Netherlands
Email: {m.cvetkovic,d.gusain,p.palensky}@tudelft.nl

***Abstract*—Improved modeling and simulation of power and energy systems has become increasingly important in the face of energy transition. The main challenge is to capture the complexity that heterogeneity of technologies and uncertainty of renewable resources bring along. One approach to improve simulation modeling capabilities, that relies on reusing existing expertise and legacy tools, is a so-called combined simulation (co-simulation). In this approach, well-established tools are combined together resulting in simulation environments with greater capabilities. In this paper, we introduce a new co-simulation rapid prototyping tool called Cossembler (which stands for Co-simulation assembler), whose main benefits are high usability and a variety of potential application domains that could be addressed by it. The paper further presents two use cases which illustrate Cossembler capabilities.**

I. Introduction

Integration of renewable and carbon-free technologies necessitates the need for better modeling and simulation of energy systems. The main scientific and practical challenge is in representing heterogeneity of diverse technology portfolio operated under various market and governance frameworks while capturing the impact of uncertainties in weather-dependent renewable energy resources. All these aspects must be modeled truthfully, using analytical and numerical methods at disposal and in compliance with the industry adopted tools and approaches. In this paper we focus on enhancing energy system representation for simulation purposes, particularly for short-term planning and operations of power systems.

Today's state-of-the-art, and particularly off-the-shelf, simulation tools provide highly compartmentalized services, specializing in studies of particular applications (transient stability, power flow, etc.). As mentioned in [1], the objective of the investigation will entail deployment of different mathematical models, ranging through time- and frequency-domain analysis, linear system theory, optimizations, graph theory, etc. covering a wide range of applications, from transient stability studies to multi-period scheduling of resources. Many of these application-focused approaches have difficulties in accurately representing novel and heterogeneous technologies as well as uncertainty of primary energy sources. Finally, already mentioned compartmentalization does not provide a holistic picture, which is increasingly needed to address integrated challenges of the energy transition.

The main premise of this paper is that a comprehensive simulation model of power and energy systems can be obtained by combining the existing simulators of various power system applications into a co-simulation. A co-simulation (stands for combined simulation) is a methodology of creating new simulation tools by combining already existing ones into new functional units that have greater capabilities when compared to each individual tool [2]. The main benefit of co-simulation is that it avoids re-inventing the wheel. Co-simulation uses tools and simulators developed by domain experts, and hence, it utilizes already existing expertise and knowledge. Although everything from tools, models, simulators, platforms, etc. can be combined in a co-simulation, we use the unifying name simulator to describe any of these entities. In other words, we use the word simulator to refer to any piece of software that solves a particular problem of interest in the area of power and energy systems.

Co-simulation has already been deployed in the power and energy field to tackle challenges of different scope and magnitude. A rainbow of approaches for interfacing power system and Information and Communication Technology (ICT) simulators is found in the overview papers [3], [4]. In most of the references therein, the scientific focus is on coupling of time-domain and discrete event simulations, while the resulting co-simulation is deployed for investigating impact of communication network latencies and cyber-attacks on power system controllers. Another sphere of high interest involves connecting of Electromagnetic Transients (EMT) and transient stability (TS) simulations (see [5] for an overview). In this case, the focus is on coupling of two most commonly used modeling approaches for simulating power system dynamics. The scientific focus of such research is on handling of algebraic loops, which are intrinsic due to transient stability simulations, while its application is typically on validating grid behavior in the presence of converter-based technologies [6].

In many of these approaches, the newly developed co-simulators are presented as final products targeting one particular application. Since their application is even more specialized than that of the individual simulators, the usage of these co-simulators is often short lived.

On the other hand, in recent years we have seen multiple efforts to develop more sustainable power system co-simulation tools that are sufficiently general to be used for multiple purposes. Mosaik [7] has paved the way by introducing a scalable tool that can handle large number of power system modules. Its simple Application Programming Interface (API) makes its usage rather straightforward. About

978-1-7281-1981-6/19/$31.00

the same time Functional Mockup Interface (FMI) gained in popularity [8]. Although it was originally developed to streamline model exchange in automotive and airspace industries, it is becoming more commonly used for co-simulations by packaging models and solvers into Functional Mockup Units (FMUs). An important benefit of FMI standard is that it provides an entry point to using an abundance of Modelica models, that come handy for modeling of heterogeneous power and energy systems and components. Finally, in a search of a high-performance architecture, reference [9] has proposed HELICS for co-simulation of large-scale power systems. Although all these solutions are highly customizable by the user, they also require a high level of user expertise. In particular, the user is often expected to have quite deep understanding of co-simulation concepts and high programming skills to use these software solutions.

Hence, a major challenge of developing a co-simulation tool that is highly-customizable and at the same time easy-to-use remains open.

In this paper, we introduce a co-simulation rapid prototyping tool called Cossembler, which stands for co-simulation assembler. The tool was developed with a particular intention to speed up the co-simulation prototyping process and to make the co-simulation methodology accessible to engineers and researchers with moderate programming skills. Hence, the tool has a rather light learning curve. Since the main design requirement is usability, the tool does not significantly optimize for performance. In addition, the tool is designed to be lightweight, covering only essential capabilities needed for co-simulations of power and energy systems.

In this paper, we demonstrate the use of Cossembler and show its capabilities and features in one particular application domain, i.e. the domain of power system operations.

The rest of the paper is organized as follows. Section II highlights the added value of Cossembler. In Section III we briefly describe the architecture of the tool. In Section IV we outline the main components of the power system operations application domain. And finally, in Section V we demonstrate two related use cases.

II. Benefits of Cossembler Rapid Prototyping Tool

In this section, we highlight the added value of Cossembler by describing its unique features: high usability and application domain variety.

A. High usability

The main hurdle for creating a co-simulation is a steep learning curve for its user. We use the term user to refer to anyone in the need of co-simulation results. This user is typically from power and energy sphere of interests (e.g. power system engineers, grid operators, policy makers, consultants, etc.). However, this user has to overcome multiple challenges in creating a co-simulation, ranging from API control to synchronization of message exchange between simulators. Apparently, these challenges belong to the software development field deeming co-simulation development quite hard.

Although some of the mentioned tools (i.e. mosaik, HELICS, FMU) can ease this journey, they still require a certain level of coding comfort to be used. Both HELICS and Mosaik provide layered architectures in which the user is often reminded of lower service layers which are often of little interest to the user. Such specs, although comfortable for a computer scientist, are still not a natural choice for a user coming from power and energy field.

Cossembler addresses both of these concerns with its user-centered architecture. First, Cossembler is a block modeling tool. Hence, creation of the co-simulation is as simple as linking appropriate blocks together. Since many intended users have used a block modeling tool at some point in their education or career (the most notable example is Matlab simulink) the learning curve of Cossembler is rather light. In addition, the block modeling design lends itself to a straightforward Graphical User Interface (GUI) design (although such GUI has not been developed yet and remains to be implemented in the future).

Second, Cossembler allows the user to work directly with typical power and energy functionalities (such as computing optimal power flow or running a time domain simulation). This is accomplished by Cossembler blocks with power and energy functionalities (for example, a Security Constrained Unit Commitment (SCUC) block). Hence, the Cossembler usage is very intuitive for the intended user.

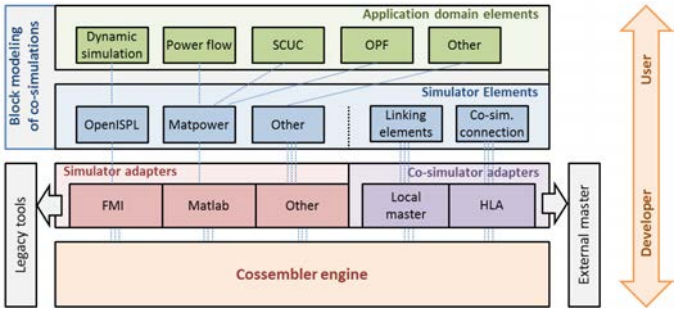

Fig. 1. Sketch of the Cossembler architecture including key components of the Power system operations application domain.

B. Application domain variety

As already mentioned, co-simulation tools are often developed with one particular application in mind. Even the FMI framework, which originated from cyber-physical system community, has a strong inclination towards coupling of time-domain simulations. A dedicated application typically entails optimized architectural design choices. Hence, later extensions and generalizations are sometimes hard to make.

Cossembler is focused on rapid prototyping, and hence, it is optimized for the ease of use and not for a particular application. At the same time, the intention of its creators is to enable rapid prototyping for as many different application domains as possible. Among these domains of interest are the integration of new technologies and energy carriers, validation of novel methods and algorithms for power system operation and planning, cyber-security analysis, real-time co-simulation, etc.

Since developing a tool that covers so many purposes is a considerable task of great magnitude, the development has been planned in stages. Each stage develops one of the application domains. Until now, one application domain has been developed, this is the application domain titled power

system operations and includes blocks with most typical functionalities used in power system operations. This application domain is described in Section IV. Before we proceed with that description, we first describe the generalities of the Cossembler architecture which enable the benefits from this section.

III. Cossembler Architecture

Cossembler is an open source object oriented software developed in Python and its architecture is illustrated in Figure 1.

There are two basic categories of blocks within Cossembler from which every other block is derived. The first one is called Canvas and the second one is named Element. Since Cossembler follows the rules of object oriented inheritance, Canvas is also an incarnation of Element.

Elements are used for multiple purposes. First, they are used to represent simulator blocks within Cossembler. To accomplish this, a simulator adapter is created for each simulator as an Element of Cossembler. For example, a Matlab Element is created in such fashion.

Second, Elements are also used to represent co-simulation master blocks. For now, Cossembler allows integration with CERTI High Level Architecture (HLA) using the built-in HLA Element. The co-simulation master blocks take care of the synchronization and message exchange among simulators if these run as separate processes, at remote machines, under different Python versions or operating systems or if they do not have a Cossembler adapter.

The third purpose of Elements is to represent various basic data manipulation functionalities which could potentially be useful to process the messages exchanged by the simulators. For example, one might wish to multiply the data in the message by a constant factor before this message arrives at its destination. Such functionalities are useful for creating functional interfaces between simulators (examples of really involved ones can be found in literature on coupling of EMT and TS simulations where an EMT signal needs to be converted to a TS signal and vice versa [6]). In addition, various blocks for simulation execution control are implemented as such elements. We call these elements Linking elements.

Finally, the fourth purpose of the Element is to host user defined blocks. These blocks are meant to give the user an opportunity to extend the built-in functionalities of Cossembler by developing their own.

Canvas is the second category of blocks available. It acts as a workspace in which prototyping takes place. A user creates Elements, adds them to a Canvas and connects them in a desired configuration. Besides acting as a container for Elements, Canvas also makes sure that the data propagates from one Element to another in a form of a message. In other words, Canvas acts as a local co-simulation master, directing messages and synchronizing execution of Elements within its jurisdiction.

Since Canvas is one form of an Element, nesting of one Canvas within another is allowed and encouraged to create more advanced functionalities. Elements of interest to the intended user, for example Power flow element or SCUC element, are created as Canvases which contain a simulator element such as Matlab element. Then the Power flow and SCUC elements direct the simulator element to run appropriate functionality (in this case, to run the Matpower toolbox).

Canvases are also used to create local loops and implement conditional execution of certain parts of the code.

With such minimalistic representation, the use of Cossembler becomes simple to master. Figure 2 gives a short code snippet.

```
from cossembler import Canvas
from cossembler import PowerFlow
from cossembler import DynamicSimulation

PFoptions = {...}
DSoptions = {...}

wrld = Canvas('world')
elemDS = DynamicSimulation("DSIPSL",DSoptions)
elemPF = PowerFlow("PF",PFoptions)

wrld.add_element(elemDS)
wrld.add_element(elemPF)

elemPF.connect(elemDS,'Pg','Pg')

wrld.start()
```

Fig. 2. Pseudo code showing the example of connecting two elements in Cossembler

IV. Application Domain - Power System Operations

The first application domain developed with Cossembler is the domain of power system operations. The goal of this domain is to establish blocks typically used in power system operations, such as power flow, optimal power flow, SCUC, economic dispatch (with and without grid constraints), dynamic simulation (in particular transient stability simulation), and the blocks referring to renewable representation (in particular their uncertainty). Application domains are essentially libraries of modeling blocks for that particular sub-area of power and energy field.

With the provided blocks for this application domain, the user is able to represent variety of cases pertaining to power system operations.

Figure 3 gives a general overview of the blocks relative positioning within the power system operations cycle. Note that this is just an illustration, yet of a kind that could be easily implemented as a template for use cases within this domain. The user is free to modify the setup from the image in any desired way, which we do as well when we reach the case studies in the next section.

The green blocks in Figure 3 denote elements that rely on external simulators for execution. Cossembler currently provides all these elements by deploying Matpower (for steady-state computations) and Modelica OpenIPSL library (for dynamic simulations). Other popular legacy tools are to be added in the future. The blue blocks are elements which are developed directly in Cossembler in order to implement desired functionalities.

Also note that the setup in Figure 3 does not use an external co-simulation master algorithm. Hence, all elements are controlled by the overarching Canvas. If one wishes to run this setup across multiple machines or processes, one could assign different parts to different canvases and then run

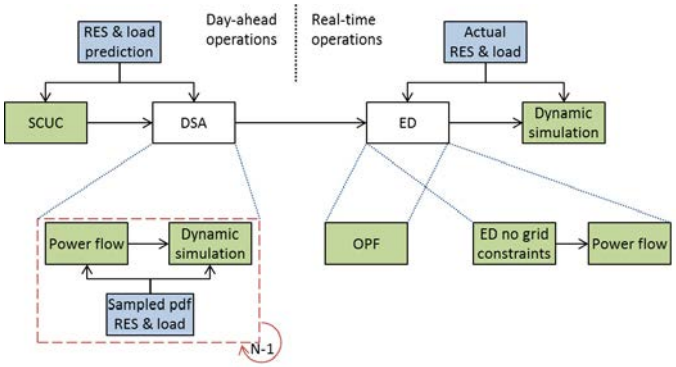

Fig. 3. Illustration of power system operations represented by Cossembler blocks. The day-ahead operations are represented using SCUC and DSA, while the real-time operations are represented using Economic Dispatch (ED) and grid dynamics simulations. In this sketch, DSA is composed of two blocks, while ED can be implemented in two common variants. Finally, both DSA and real-time operations are implemented in cycles to capture their repetitiveness.

these canvases on different machines using a co-simulation connection element to connect them. In this example, the execution of the co-simulation is largely sequential (except for the Dynamic Security Assessment (DSA) section of the simulation), so the division across multiple machines bring little benefits, except for the DSA parallelization. A setup for such paralellization is shown in Figure 4. Connection element for HLA is the only element supported at this time while other co-simulation master solutions will be added in the future.

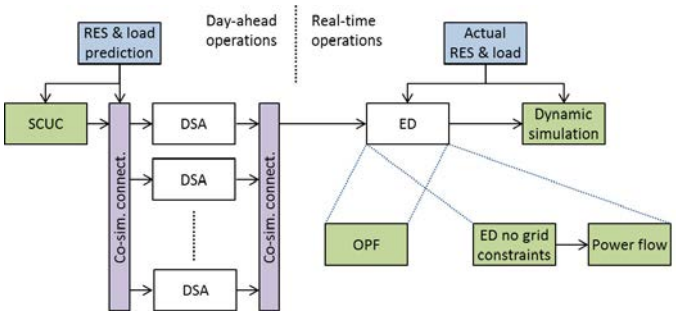

Fig. 4. Illustration of the power system operations building blocks including process parallelization.

V. Use Case Studies

In this section we briefly illustrate two possible use cases within the application domain of power system operations. The first use case is a probabilistic Monte Carlo-based dynamic security assessment which accounts for variability in renewable generation. With this use case, we show that Cossembler can be deployed to truthfully represent grid operational procedures. We also use this use case to illustrate the main benefits of Cossembler. The second use case is the one of co-optimization, which we use to demonstrate that Cossembler can also be used to devise new algorithms by combining the existing ones. Note that these examples are intentionally simple; it is our wish to showcase the capabilities of Cossembler and not to claim accuracy or the realism of the examples.

In both use cases, power system is modeled as IEEE 9 Bus system which is augmented with one wind farm connected to one of the load buses (Bus 9).

A. Use case 1: Stochastic dynamic security assessment

In this use case, the grid operator runs an SCUC to create operation schedule for the next day. After the schedules are created, the operator runs dynamic simulations for several scenarios of interest in order to ensure transient stability of the grid. In this setup, we put particular emphasis on scenarios with different renewable penetration levels. By sampling the probability density function of the renewable generation, we run Monte Carlo analysis to assess the system frequency excursions and possible frequency-band violations.

The left hand side of Figure 3, which refers to day-ahead operations, is the exact representation of the co-simulated system in this use case. As already explained, Matpower SCUC and OpenIPSL are used in this co-simulation. An interesting aspect to elaborate upon is the implementation of the loop that handles Monte Carlo simulation. Among diverse linking elements, Cossembler provides two block-elements for control of execution loops. These elements are ForLoop and WhileLoop. Their usage is straightforward as shown in Figure 5 and fits within overall paradigm of Cossembler. Such elements were purposely created to allow the user to control simulation execution in a much easier way.

```
whielem = WhileLoop('WLoop', elem, "A>B", 'in')
forelem = ForLoop('FLoop', elem, N)
```

Fig. 5. Example of using simulation execution control elements in Cossembler. Variable elem is a modeling block on which the loop is deployed.

Figure 6 shows the deviation of system frequency as a result of Monte Carlo simulation runs. It is notable that the maximum and minimum frequency excursions go beyond the band of 49.8Hz-50.2Hz which is reserved for primary frequency response in Continental Europe. Hence, the grid operator would have to deploy secondary frequency reserves 2.2% of time in order to preserve stability. In the next subsection, we look at co-optimization of reserve scheduling that can address this challenge posed to a grid operator.

B. Use case 2: Co-optimization of frequency reserves

The second use case shows the co-optimization of OPF and dynamic simulation in order to achieve highly optimal reserve requirements. We start by replacing the Power flow block from the left hand side of Figure 3 with the Optimal power flow block which includes optimization of reserve requirements. This block, as many others, engages computation services of Matpower. The updated sketch is given in Figure 7.

The reserve requirements are usually set a priori much ahead of time (sometimes even on seasonal basis), and hence, they are increasingly found to be either overly conservative or overly optimistic in presence of large amounts of renewable generation. Using co-optimization, a grid operator can adjust the required level of reserves on the fly while

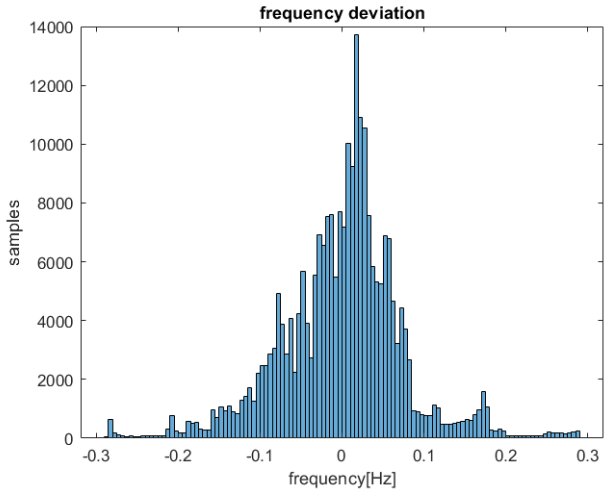

Fig. 6. Frequency deviation histogram from Monte Carlo simulations in dynamic security assessment use case. In this case, the frequency excursion violates the bounds 2.2% of time, assuming that bounds are set at 49.8Hz and 50.2Hz.

observing the true grid conditions and renewable energy levels.

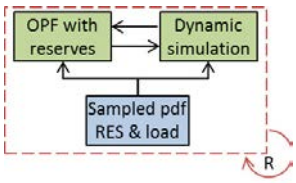

Fig. 7. Co-optimization of frequency reserves using Cossembler.

Figure 8 shows how the frequency requirement changes with each simulation run. By running a sufficient number of scenarios, the frequency requirement will converge to a value which satisfies all frequency deviations and which is not overly conservative.

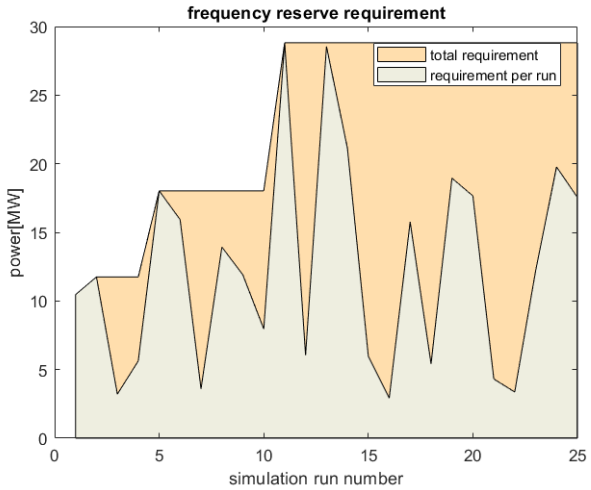

Fig. 8. Frequency requirement as a function of the simulation run.

VI. Conclusions

In this paper, we presented Cossembler - a rapid prototyping tool for co-simulations of power and energy systems. As emphasized throughout the text, the main features of the tool are high usability and variety of potential application domains it can cover. These features result in rather light learning curve for Cossembler users. Cossembler is built on top of legacy tools and connects already existing domain expertise.

The first application domain developed with Cossembler, and also presented in this paper, is the one of power system operations. Within this domain, we showed how modeling blocks of Cossembler can be used to model stochastic dynamic security assessment and how the tool can be used to propose improved operational methods using co-optimization. These examples illustrate effortless use of Cossembler and provide a teaser for variety of possible applications.

The future work on Cossembler will aim to develop block models for other relevant application domains (e.g. cyber-security, multi-energy system modeling, etc.), and also to enhance variety within the existing application domain by integrating other commercial and open source tools.

References

[1] T. Strasser, M. Stifter, F. Andrén, P. Palensky, *Co-Simulation Training Platform for Smart Grids*, IEEE Tran. on Power Systems, vol. 29, no. 4, pp. 1989-1997, July 2014.

[2] P. Palensky, A. van der Meer, C. D. López, A. Joseph, K. Pan, *Applied Co-simulation of Intelligent Power Systems: Implementing Hybrid Simulators for Complex Power Systems*, IEEE Industrial Electronics Magazine, vol. 11, no. 2, pp.6-21, June 2017.

[3] IEEE Task Force on Interfacing Techniques for Simulation Tools et al., *Interfacing Power System and ICT Simulators: Challenges, State-of-the-Art, and Case Studies*, in IEEE Tran. on Smart Grid, vol. 9, no. 1, pp. 14-24, Jan. 2018.

[4] W. Li, M. Ferdowsi, M. Stevic, A. Monti, F. Ponci, *Cosimulation for Smart Grid Communications*, in IEEE Tran. on Industrial Informatics, vol. 10, no. 4, pp. 2374-2384, Nov. 2014.

[5] V. Jalili-Marandi, V. Dinavahi, K. Strunz, J. A. Martinez, A. Ramirez, *Interfacing Techniques for Transient Stability and Electromagnetic Transient Programs IEEE Task Force on Interfacing Techniques for Simulation Tools*, IEEE Tran. on Power Delivery, vol. 24, no. 4, pp. 2385-2395, Oct. 2009.

[6] A. A. van der Meer, M. Gibescu, M. A. M. M. van der Meijden, W. L. Kling, J. A. Ferreira, *Advanced Hybrid Transient Stability and EMT Simulation for VSC-HVDC Systems*, IEEE Tran. on Power Delivery, vol. 30, no. 3, pp. 1057-1066, June 2015.

[7] S. Rohjans, S. Lehnhoff, S. Schütte, S. Scherfke, S. Hussain, *mosaik: a modular platform for the evaluation of agent-based smart grid control*, IEEE/PES Innovative Smart Grid Technologies Europe, 2013.

[8] A. A. van der Meer, et al., *Cyber-Physical Energy Systems Modeling, Test Specification, and Co-Simulation Based Testing*, 2017 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES), Pittsburgh, PA, 2017, pp. 1-6.

[9] B. Palmintier, D. Krishnamurthy, P. Top, S. Smith, J. Daily, J. Fuller, *Design of the HELICS high-performance transmission-distribution-communication-market co-simulation framework*, 2017 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES), Pittsburgh, PA, 2017, pp. 1-6.