Distributed Co-Simulation for Collaborative Analysis of Power System Dynamic Behavior

Claudio David López, Miloš Cvetković and Peter Palensky
Department of Electrical Sustainable Energy
Delft University of Technology
The Netherlands

Abstract—**Given the complexity, scale and heterogeneity of modern power systems, many comprehensive simulation tasks are almost impossible to carry out without collaboration from multiple institutions. One way to approach collaborative simulation is through co-simulation. In this approach each institution contributes a model and the simulation is run distributedly. Co-simulation environments can be centrally orchestrated or decentrally orchestrated, each having its own set of advantages and challenges. In this paper we analyze the merits and challenges of co-simulation for collaborative simulation between institutions through two simple co-simulation environments, one centralized and the other decentralized. We argue that a co-simulation environment for collaboration between institutions should provide functionality like standard co-simulation interfaces for a variety of simulation tools, remote management and configuration, model compatibility checks, and failure detection and recovery. An environment with these characteristics can be easily adopted by a wide range of institutions, which would greatly aid in tackling the complexity of modern electrical power systems.**

Index Terms—**Co-simulation, distributed simulation, collaboration, dynamic simulation, ØMQ, PowerFactory**

I. Introduction

As power systems evolve, their complexity, scale and heterogeneity make them increasingly harder to design and analyze without the collaboration of multiple parties or institutions that provide the necessary know-how and have access to the required information (e.g. TSOs, DSOs, universities, research institutes, consulting firms). This situation manifests itself in simulation tasks as well, where models from engineering branches other than power engineering need to be accounted for (e.g. communication systems) and interactions between systems that used to be regarded as relatively independent must be factored in (e.g. grids of neighboring countries, transmission and distribution grids). As an example of this situation, let us consider a case where a TSO needs to evaluate the impact that the increasing penetration of energy generation and storage technologies at the distribution level will have on its transmission grid. Naturally, it would be difficult for the TSO to carry out said study without models of the relevant distribution grids. However, this difficulty can be easily circumvented if the involved institutions (i.e. the TSO and the DSOs) agreed to simulate collaboratively.

There are two distinct approaches to collaborative simulation: model exchange and co-simulation. In the former, different parts of a system are modeled independently but simulated as one large, merged model. In the later, each part of the system is modeled and simulated independently by having each simulator share selected variables with the others at runtime. A so-called co-simulation master is often in charge of orchestrating the variable exchange and keeping the simulation time of each simulator consistent with that of the others.

Co-simulations can be classified according to the way they are orchestrated in centralized and decentralized. Centralized co-simulations are orchestrated by a co-simulation master whereas decentralized co-simulations are orchestrated by the simulators themselves. The differences between both types of co-simulation manifest themselves in the way they are managed and the way they operate, and can have consequences that are worth examining from the point of view of collaborative simulation.

Standards like the Functional Mock-up Interface (FMI), originally developed for the automotive industry, provide the means for collaborative simulation through model exchange and co-simulation [1], but the lack of power engineering tools that support the standard has been an obstacle to its widespread adoption. Another standard that does enjoy more support from power system simulation tools is the Common Information Model (CIM), which facilitates the exchange of model information. Recently, ENTSO-E began to implement the Common Grid Model (CGM) [2], a pan-European grid model for accurately forecasting electricity flows across the European grid, which requires TSOs to collaborate by submitting their system's data using the CIM. Yet, co-simulation as a medium for collaborative simulation has not been given the same attention in the electrical power industry as it has in other industries.

Examples of co-simulation in power engineering applications abound. The approach has been successfully used to analyze coupled transmission and distribution systems [3], multi-area systems [4], and interactions between power systems and ICT [5], among others. Nevertheless, few of these examples are intended for the specific purpose of collaboration between institutions.

In this paper we analyze the merits of distributed co-simulation as a tool for collaborative dynamic simulation of electrical grids. For this purpose we present two co-simulation environments, one centrally orchestrated and the other decentrally orchestrated, and use them to guide our analysis. Both environments are distributed in the sense that each simulator can run on a different computer, and these computers can be located wherever it is most convenient for each collaborating institution.

The paper is structured as follows: Section II discusses the merits and drawbacks of co-simulation for collaborative simulation, Section III introduces the centralized and the decentralized co-simulation environments and provides an example of their operation, Section IV compares both environments from the point of view of collaborative simulation, and Section V concludes the paper.

II. Co-Simulation for Collaborative Simulation

Let us expand the description of the case from the introduction where a TSO needs to evaluate the impact of storage and generation at the distribution level on its transmission grid. The TSO commissions the required study to a consulting firm that determines that there are two distribution grids in particular that must be considered in detail. The consulting firm requests collaboration from the corresponding DSOs, but out of privacy concerns they are reluctant to sharing their grid data. For a situation as this one, using co-simulation for collaborative simulation between institutions has several advantages with respect to model exchange. We identify the following:

- *Privacy:* Collaborating institutions do not need to share their models, or disclose any private information about them, since co-simulation only requires selected variables to be exchanged between simulators at run time, while each model remains within the institution that owns it.
- *Tool freedom:* Each collaborating institution can continue to develop models for and run simulations with the tool of its choice, since information is exchanged between simulators through a standard, tool-independent protocol.
- *Reduced workload:* Tool freedom removes the need for translating models between tools or languages and for merging them, which is a highly labor intensive task [6]. This is especially true for dynamic models, which are considerably more detailed than their static counterparts.
- *Shared computational load:* The computational load of the co-simulation can be shared among collaborating institutions, since each simulator can run on a different computer if data is exchanged over a network. This can be particularly advantageous for dynamic simulations, since these models are much larger and computationally expensive to simulate than their static counterparts.
- *Up-to-date models:* Model updates can be quickly available to all collaborating institutions; as soon as an institution updates its model locally, the co-simulation is updated as well.

However, co-simulation also poses certain challenges to collaborative simulation that model exchange does not. We identify the following:

- *Implicit assumptions:* Any implicit assumptions that an institution makes about the models and simulators of the others can cause problems if these assumptions are wrong. Examples of this are when one simulator expects three phase inputs while the other provides three sequence outputs, and when the interface variables have mismatching units.
- *Compatibility verification:* The lack of a model merging process carried out by a single institution increases the difficulty of verifying that no wrong implicit assumptions were made and that all models are compatible.
- *Interface specification:* Collaborating institutions must place special care in specifying their model interfaces to make up for the absence of a compatibility verification process and to reduce the chances of making implicit assumptions.
- *Numerical considerations:* Co-simulations require numerical considerations related to accuracy and stability that simulations using one merged model do not [7].
- *Unavailable functionality:* Some co-simulations can require functionality that certain simulation tools cannot provide. Functionality like time roll back or time step repetition for iterative co-simulation methods are common examples of this problem [7]. If at least one collaborating institution uses a simulation tool that lacks the required functionality, collaboration will be hindered.
- *Environment management:* In the absence of a coordinating institution, tasks like loading models, starting simulators and retrieving results are more difficult to execute. The co-simulation environment can potentially provide functionality to allow any collaborating institution to execute these tasks remotely, but this opens additional questions about privacy and security since in this case each institution would have to grant access to their computing infrastructure to other institutions.
- *Failure recovery:* If any component in a co-simulation fails at run-time, progress will be lost. This can be costly for co-simulations that take a long time to complete. Failure detection and recovery mechanisms are valuable in such cases, but detecting the failure of one component in a distributed simulation is more challenging than detecting the failure of a monolithic simulation.

Despite these challenges, there are situations where co-simulation is better suited for collaboration than model exchange. In the case of the TSO and the two DSOs, co-simulation can be justified by the need for privacy. In other cases, a model merging process might be too costly, especially if model updates are frequent, which would turn the model merging process into a bottleneck in the simulation workflow.

III. Co-Simulation Environments

We developed two simple co-simulation environments, each following a different approach to co-simulation orchestration. The environments were built using only widely adopted and well supported technologies. In the first one a co-simulation master centrally orchestrates the co-simulation, whereas in the second one the simulators themselves orchestrate it distributedly. Although more advanced co-simulation environments that follow each of these approaches already exist (e.g. mosaik for centralized orchestration [8] and FNCS for decentralized orchestration [9]), building these two environments using the

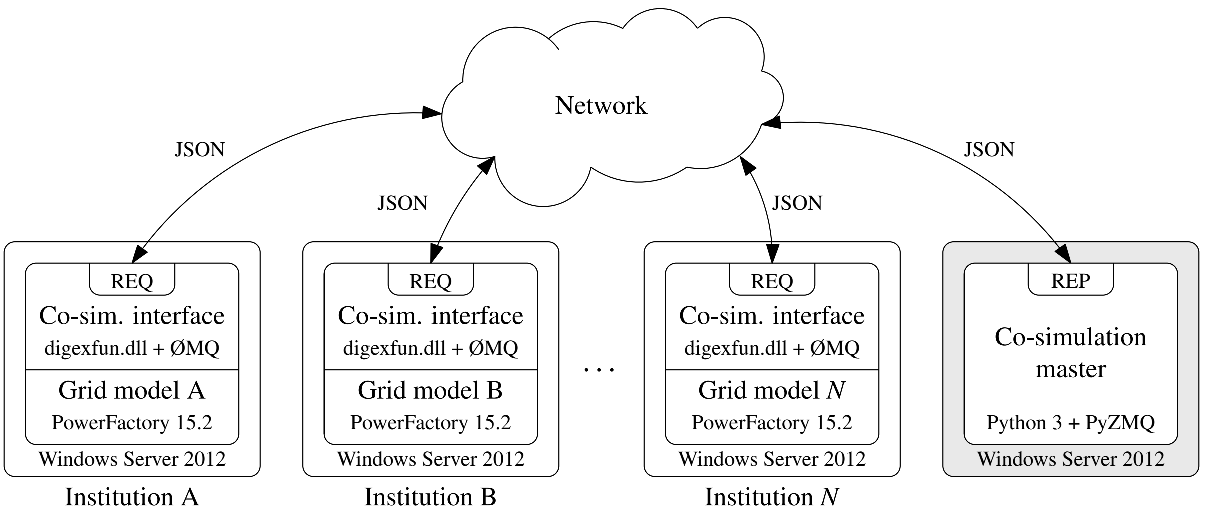

Fig. 1. Diagram of the centrally-orchestrated co-simulation environment. Each institution is represented by a different server running DIgSILENT PowerFactory. A co-simulation master is in charge of orchestrating the co-simulation. The communication is implemented with request (REQ) and repy (REP) sockets from the ØMQ library.

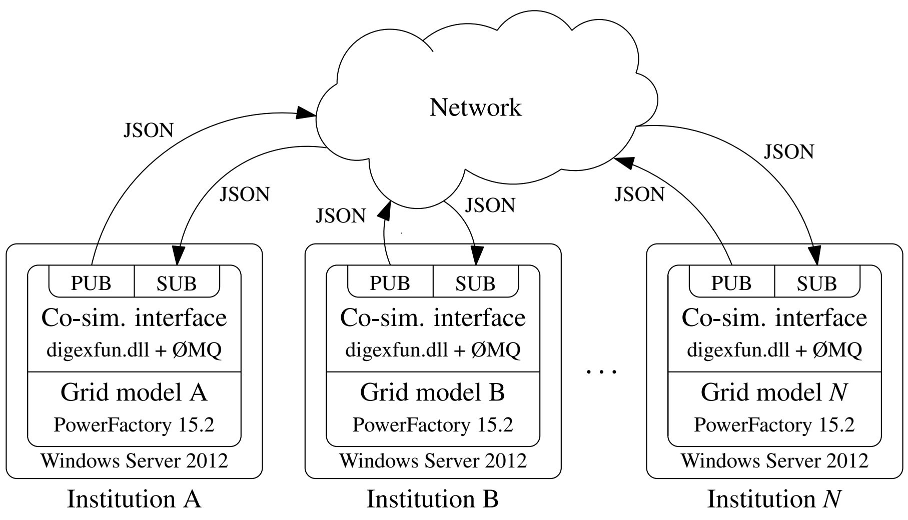

Fig. 2. Diagram of the decentrally-orchestrated co-simulation environment. Each institution is represented by a different server running DIgSILENT PowerFactory. The co-simulation orchestration is done by the co-simulation interfaces. The communication is implemented with publish (PUB) and subscribe (SUB) sockets from the ØMQ library.

same set of technologies makes the comparison straightforward.

A. Co-Simulation with Centralized Orchestration

Fig. 1 shows a diagram of the centrally-orchestrated co-simulation environment. The environment is composed of a set of Windows 2012 virtual servers, each running a different instance of DIgSILENT PowerFactory 15.2. A co-simulation master implemented in Python 3 runs on another virtual server. Each simulator uses a co-simulation interface to interact with the co-simulation master.

The co-simulation interface is composed of a DIgSILENT Simulation Language (DSL) model and a Dynamic-Link Library (DLL) written in C++. PowerFactory loads the DLL at startup making its functionality accessible to the DSL model through DSL function calls. The DSL model is in charge of getting the output variables from the grid model, and passing them to the DLL. The DLL then uses these variables to create an output message which it sends to the master. All output messages are time-stamped with the current simulation time. Once the DLL receives an input message from the master containing input variables for the grid model, the DSL model gets the variables from the DLL and sets them in the grid model so PowerFactory can solve the next time step.

The communication between the co-simulation interface and the co-simulation master is implemented with ØMQ request/reply sockets [10] and messages encoded in JavaScript Object Notation (JSON). In the implemented request/reply pattern, each co-simulation interface opens a request-type socket and the master opens a reply-type socket. Every request socket connects to the reply socket, and sends request messages (output messages) to the master containing the output variables of the corresponding PowerFactory instance. When the master receives all the requests for a given time step, it creates reply messages (input messages) containing the inputs that every PowerFactory instance needs to solve the next time step, and sends each of the messages to the corresponding instance.

B. Co-Simulation with Decentralized Orchestration

Fig. 2 shows a diagram of the decentrally-orchestrated co-simulation environment. This environment is also composed of a set of Windows 2012 virtual servers running PowerFactory 15.2, but in this case there is no co-simulation master. The co-simulation interface differs from the one in Fig. 1 in the functions it performs and in the number and type of sockets it uses.

Each interface uses one publish-type and one subscribe-type socket. In the implemented publish/subscribe pattern, each publish socket makes output messages public to any PowerFactory instance that needs them, while each subscribe socket subscribes only to the messages the corresponding PowerFactory instance requires. As opposed to the environment from Fig. 1 where it is a task of the co-simulation master to create input messages with all the inputs each PowerFactory instance needs, and each PowerFactory instance receives only one message per time step, in the decentrally-orchestrated environment the co-simulation interface can receive messages from several simulators at every time step. It is thus a task of the co-simulation interface to determine when all the required input messages have arrived and to extract the necessary inputs from them so that PowerFactory can solve the next time step.

C. Integration of Other Simulation Tools

Although the presented environments only consider PowerFactory as a simulator, the technologies used to implement the environments make it feasible to integrate other simulators as well. There are several reports of co-simulations that include other popular power engineering simulation tools, such as [11] that couples PSS/E and PSCAD and [12] that couples PowerFactory and PSS/E. However, most of these examples are incompatible with each other. This means that the main challenge to seamless integration of simulation tools into a co-simulation environment that can be used for collaboration between institutions is not just to interface the simulators, but to create compatible interfaces for each of them, so that no matter the simulation tool, collaboration is still possible. In this sense, standardization is fundamental to successful collaborative co-simulation.

D. Co-Simulation Example

To provide an example of both co-simulation environments working, let us go back to the case of the TSO and the two DSOs described in Section II. We now set up a co-simulation using the nine bus system from Fig. 3 to represent the transmission grid of the TSO and two 13 bus systems as the one from Fig. 4 to represent the distribution grids of the DSOs.

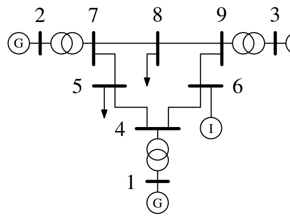

Fig. 3. One-line diagram of the nine bus transmission system. The current source at bus 6 represents the distribution grids.

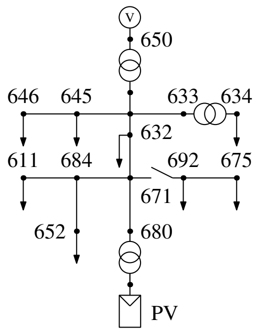

Fig. 4. One-line diagram of the 13 bus distribution system. The voltage source at bus 650 represents the transmission grid.

The grids are coupled using ideal voltage and current sources. From the transmission grid point of view, the distribution grids are current sources (see bus 6 in Fig. 3), whereas from the distribution grid point of view, the transmission grid is a voltage source (see bus 650 in Fig. 4). Thus, at every time step the transmission grid simulator sends the voltage at bus 6 to the distribution grid simulators, while the distribution grid simulators send the current through bus 650 to the transmission grid simulator.

Fig. 5 shows some sample results from a co-simulation as the one described above, where a short circuit occurs in one of the distribution grids after 0.05 s. The results shown in Fig. 5 are the three-phase voltage at bus 6 in the transmission grid, and the current flowing between the transmission and distribution systems as measured at the same bus. According to Fig. 5 the results obtained with both environments match. This is because in both cases the messages exchanged between simulators are exactly the same; the only difference is the mechanism employed to deliver them. The implications of this difference are discussed in the next section.

IV. Centralized versus Decentralized Environments for Collaborative Simulation

Differences between the centralized and the decentralized co-simulation environments appear at different stages in their development and use. Some of these differences are particularly relevant for collaborative co-simulation.

A. Implementation

Arguably the most important activity that takes place before co-simulation execution is the implementation of the co-simulation environment. Correctly implementing a centralized

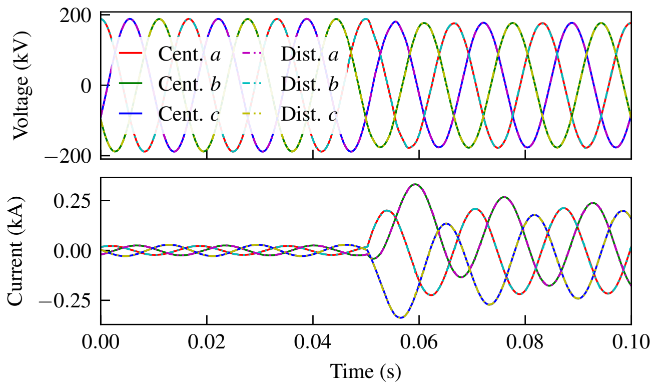

Fig. 5. Voltage and current flowing through the interface bus as measured on the transmission grid. The figure shows phases a, b, and c for centralized and decentralized orchestration. As expected, the results from both co-simulation environments match.

environment is easier than a decentralized one. This is because in a centralized environment the co-simulation master has a complete overview of the co-simulation state. With this information it is straightforward for the co-simulation master to decide how to proceed. In contrast, in a decentralized environment each simulator only interacts with its neighbors and must make decisions based exclusively on the information it exchanges with them. Thus, the complexity of the algorithms required to make the same decisions in the decentralized case is higher.

B. Master Allocation

In the case of a centralized environment one institution must run and manage the co-simulation master. This institution would incur the costs that stem from both of these activities, but would also have privileged access and control over the co-simulation environment. However, this privilege could raise additional privacy and security concerns from the other involved institutions. If one of the collaborating institutions or a third party is trusted by all (e.g. ENTSO-E in the case of European TSOs), this problem can be easily solved. If that is not the case, a decentralized environment might be a better alternative.

C. Configuration

Centralized and decentralized co-simulation environments are configured differently. In the centralized environment each co-simulation interface and the co-simulation master must be configured, whereas in the decentralized environment there is no master to configure. In the centralized environment from Fig. 1, each interface requires an address to connect its request socket to the reply socket in the master. At the same time, the master requires a representation of the co-simulation topology to determine how to create input messages for each simulator from the output messages it receives. In the decentralized environment from Fig. 2 each co-simulation interface requires a list with the addresses of the simulators it must subscribe to, including which specific outputs it needs from each simulator. In all cases certain modifications need to be introduced in the models so the co-simulation interface can get their outputs and set their inputs. In the co-simulation example from Section III, these modifications are adding the ideal voltage and current sources in the systems from Fig. 3 and Fig. 4. These modifications can be considered part of the configuration process as well.

In the simplest case, each institution would be in charge of configuring its own co-simulation interface (and model). If a centralized environment is used, the institution in charge of the master would also be in charge of configuring it. The disadvantages of this configuration procedure are that it is prone to error and that errors are difficult to trace, as the configuration information is split among different institutions. Additionally, both the likelihood of an error and the difficulty of tracing it increase with the number of coupled simulators. Furthermore, as the number of collaborating institutions grows, the configuration procedure becomes increasingly harder to manage, since all of the institutions need to proceed in coordination with their peers.

Alternatively, only one institution could remotely configure the co-simulation environment. This way the configuration information can remain concentrated, making the configuration procedure simpler, less error prone and more scalable. However, before adding this functionality to a co-simulation environment intended for collaboration, the level of access that should be available through remote configuration must be defined. Certain configuration-related activities might only raise security concerns whereas others might raise privacy concerns as well. An example of the former is interconnecting simulators in a certain topology, since this would require access to the computing infrastructure of another institution. An example of the later is defining model inputs and outputs (and introducing the associated model modifications), since this would required access to the models themselves. If remote configuration functionality is implemented, whether the environment is centralized or decentralized becomes irrelevant from the configuration point of view; in both cases the procedure would be the same.

D. Synchronization at Start-Up

In the centralized environment, when a simulator is started it sends a request (output) message to the co-simulation master and waits for a reply (input) message before it solves the next time step. The master does not send reply messages to the simulators until all of them have sent their first requests, which ensures correct synchronization between simulators at start-up. However, in the decentralized environment any simulator could potentially publish an output message before the subscriber simulators are running. If this happens, the first output message would be lost and the synchronization between simulators at startup would fail. To avoid this, an additional synchronization mechanism is necessary at start-up, which requires that simulators exchange synchronization messages until they determine that all of their subscribers are running. The added complexity of this synchronization mechanism exemplifies why a distributed environment is more difficult to implement.

E. Scalability

Both environments have different scalability properties. In the centralized environment all output messages are sent to the co-simulation master. The master must process those messages and then create and send input messages to every simulator. Thus, in the centralized case all messages pass through the co-simulation master, whereas in the decentralized case messages can be exchanged directly between simulators. For co-simulations that couple a large number of simulators, the master can become a bottleneck. The importance of scalability depends on the involved institutions. If the collaborating institutions were the TSOs that belong to ENTSO-E, scalability would be critical given the large number of participating institutions, but in a case as the one from our example co-simulation with only one TSO and two DSOs, scalability would not be the main concern.

F. Summary

Table I summarizes the different characteristics of centralized and decentralized co-simulation environments and their consequences for collaborative simulation.

TABLE I
CENTRALIZED VERSUS DECENTRALIZED CO-SIMULATION FOR COLLABORATIVE SIMULATION

Characteristic	Centralized	Decentralized
Implementation	Less challenging	More challenging
Resource allocation	For simulators and master	For simulators only
Configuration	Challenging without special functionality	Challenging without special functionality
Scalability	Less scalable	More scalable
Collaboration	Trusted institution Small scale	No trusted institution Large scale

V. CONCLUSION

This paper discussed the merits of co-simulation for collaborative simulation of electrical power systems through two co-simulation environments: a centralized and a decentralized one. Co-simulation has several characteristics that make it an appealing tool for collaboration between institutions like TSOs, DSOs, research institutes, universities and consulting firms. As a tool it is especially useful in cases where model privacy and simulation tool freedom are necessary, and when the simulation must be implemented quickly by avoiding model migration and cumbersome model update procedures. In some cases, the collaborating institutions can also benefit from sharing the computational load of the co-simulation.

Regarding the choice between a centralized co-simulation environment (orchestrated by a co-simulation master) and a decentralized co-simulation environment (orchestrated by the simulators themselves) the main aspects to consider are scalability with respect to the number of collaborating institutions and whether the use of a co-simulation master is appropriate; when scalability is unimportant and there is a trusted institution willing to run and manage the master, the simplicity of a centralized environment is preferable. Otherwise, a decentralized environment over which every institution has the same level of control and access is preferable.

However, using co-simulation for collaborative simulation comes with challenges related to simulation tool diversity, model compatibility, environment management, configuration procedure and failure recovery. Although it is possible to implement simple yet functional co-simulation environments for collaboration despite these challenges, as the two environments presented in this paper show, an environment that is intended for widespread adoption must provide functionality that addresses said challenges. Thus, co-simulation interfaces for as many popular electrical power engineering simulation tools as possible, automatic model compatibility checks, remote environment management and configuration, and failure detection and recovery should be available. If an environment with these characteristics is developed, collaborative simulation can become more accessible to a wider range of institutions, which can be a great asset as electrical power systems become more complex.

REFERENCES

[1] Modelisar, "Functional mock-up interface for model exchange and co-simulation," MODELISAR, Tech. Rep., 2014.

[2] ENTSO-E, "ENTSO-E's common grid model," ENTSO-E, Tech. Rep., 2017.

[3] R. Huang, R. Fan, J. Daily, A. Fisher, and J. Fuller, "Open-source framework for power system transmission and distribution dynamics co-simulation," *Transmission Distribution IET Generation*, vol. 11, no. 12, pp. 3152–3162, 2017.

[4] C. D. López, A. A. v. d. Meer, M. Cvetković, and P. Palensky, "A variable-rate co-simulation environment for the dynamic analysis of multi-area power systems," in *2017 IEEE Manchester PowerTech*, Jun. 2017, pp. 1–6.

[5] K. Mets, J. Ojea, and C. Develder, "Combining power and communication network simulation for cost-effective smart grid analysis," *IEEE Communications Surveys Tutorials*, vol. 16, no. 3, pp. 1771–1796, 2014.

[6] M. Cvetković, H. Krishnappa, C. D. López, R. Bhandia, J. Rueda Torres, and P. Palensky, "Co-simulation and dynamic model exchange with consideration for wind projects," in *16th Wind Integration Workshop*, Sep. 2017.

[7] P. Palensky, A. A. van der Meer, C. D. López, A. Joseph, and K. Pan, "Cosimulation of intelligent power systems: Fundamentals, software architecture, numerics, and coupling," *IEEE Industrial Electronics Magazine*, vol. 11, no. 1, Mar. 2017.

[8] S. Schütte, S. Scherfke, and M. Tröschel, "Mosaik: A framework for modular simulation of active components in smart grids," in *2011 IEEE First International Workshop on Smart Grid Modeling and Simulation (SGMS)*, Oct 2011, pp. 55–60.

[9] S. Ciraci, J. Daily, J. Fuller, A. Fisher, L. Marinovici, and K. Agarwal, "FNCS: A framework for power system and communication networks co-simulation," in *Proceedings of the Symposium on Theory of Modeling & Simulation - DEVS Integrative*, ser. DEVS '14. San Diego, CA, USA: Society for Computer Simulation International, 2014, pp. 36:1–36:8.

[10] iMatix. ØMQ. [Online]. Available: zeromq.org

[11] Siemens Power Technologies International. PSS/E-PSCAD co-simulation module.

[12] M. Zamroni, "Development of EMT/TS co-simulation using PowerFactory and PSS/E," Master's thesis, Delft University of Technology, 2017.