European Research Infrastructure supporting Smart Grid and Smart Energy Systems Research, Technology Development, Validation and Roll Out – Second Edition

Project Acronym: **ERIGrid 2.0**

Project Number: **870620**

Technical Report Lab Access User Project

Unified Infrastructure-as-a-Service Platform for Distributed Co-Simulation of Networked Microgrids (IaaS Platform)

Access Duration1: 17/02/2025 to 28/02/2025
Access Duration2: 17/02/2025 to 28/02/2025

Funding Instrument:	Research and Innovation Action
Call:	H2020-INFRAIA-2019-1
Call Topic:	INFRAIA-01-2018-2019 Integrating Activities for Advanced Communities
Project Start:	1 April 2020
Project Duration:	54 months
User Group Leader:	Nauman Beg (German Aerospace Center Institute of Networked Energy Systems)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 870620.

Report Information

Document Administrative Information	
Project Acronym:	ERIGrid 2.0
Project Number:	870620
Access Project Number:	207
Access Project Acronym:	IaaS Platform
Access Project Name:	Unified Infrastructure-as-a-Service Platform for Distributed Co-Simulation of Networked Microgrids
User Group Leader:	Nauman Beg (German Aerospace Center Institute of Networked Energy Systems)
Document Identifier:	ERIGrid2-Report-Lab-Access-User-Project-IaaSPlatform-final
Report Version:	v2.0
Contractual Date:	28/03/2025
Report Submission Date:	18/03/2024
Lead Author(s):	Nauman Beg (DLR VE)
Co-author(s):	Henning Schlachter (DLR VE), Philipp Gottfried (DLR VE), Mats Buchholz (DLR VE), Saikrishna Vallabhaneni (DLR VE)
Keywords:	Distributed Grids, EV profiles, Co-Simulation, RTDS, 5G Communication, Smart Grids, Mosaik, Web Services, Rest APIs
Status:	__ draft, _x_ final

Change Log

Date	Version	Author/Editor	Summary of Changes Made
29/09/2024	v1.0	VTT Visit-1	Draft report
18/03/2025	v2.0	VTT Visit-2	Full report

Table of Contents

List of Figures

List of Tables

List of Abbreviations

IaaS	Infrastructure-as-a-Service
CPPS	Cyber-Physical Power System
LV	Low Voltage
GFMI	Grid Forming Inverter
EV	Electric Vehicle
VTT	Valtion Teknillinen Tutkimuskeskus
RTD	Round Trip Delay
SOA	Service-Oriented Architecture
VM	Virtual Machine
EMT	Electro-Magnetic Transient
MES	Multimodel-Energy System
RTDS	Real Time Digital Simulator
dApp	Distributed Application
LA	Lab Access
UP	User Project
REST	Representational State Transfer
API	Application Programming Interface
MS	Mosaik Slave

Executive Summary

The objective of the two short-term stays under ERIGrid 2.0 Lab Access Program is to simulate a user test-case of a synthetic Low Voltage (LV) grid in the unified Infrastructure-as-a-Service (IaaS) platform through networked co-simulation and digital twins. The primary objective of the conducted tests is to show a proof-of-concept of the IaaS platform for simulation of Cyber-Physical Power Systems (CPPSs) under constrained lab infrastructures and ease of development of the new models. The grid model simulated in the Digsilent PowerFactory environment incorporates a digital twin simulation model of a Grid Forming Inverter (GFMI) and consumption from an Electric Vehicle (EV) charging station running in dedicated remote Matlab instances. A resistive load emulated by the RTDS system connected to an on-site power amplifier at the IntelligentEnergy Testbed in VTT is also connected to the modeling platform. The consumption of the resistive load is fed to the simulation model in real-time. The overview of the complete co-simulated grid model is shown in Figure 1

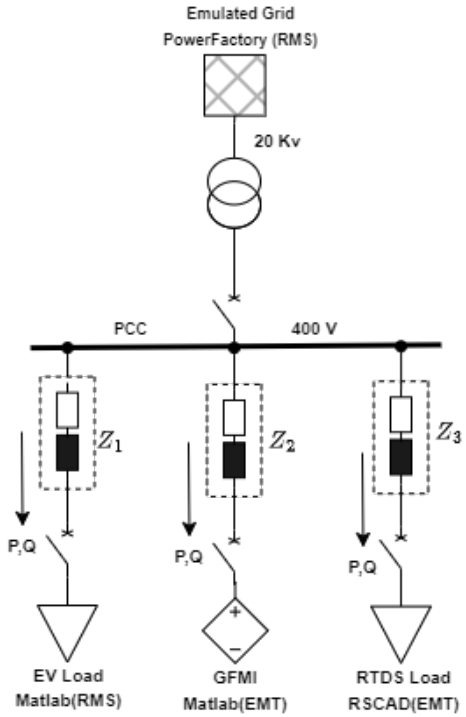

Figure 1: Overview of the simulated grid.

The inputs and outputs of both simulated as well as real-time models are interfaced to the grid model by a methodology referred to as Composite Modeling in the IaaS. Composite Modeling requires the following necessary descriptors that are written for each component in the IaaS.

1. Remote web service for communication between each simulator and the the IaaS
2. protocol converter for communication between the each remote web service and the instance of the model

Originally, 3 scenarios (Scenario 1 - Scenario 3) listed in Table 1 are simulated for various discrete co-simulation step-sizes. The model is extended during our 2^{nd} visit with a 5G communication link between the RTDS and the IaaS and 3 additional scnenarios (Scenario 4 - Scenario 6) are simulated with 5G link.

Table 1: Examples of simulation times for the IaaS platform in different scenarios.

Scenario	$t_{simulation}$	Δt_{co-sim}	Elapsed time					5G
			$t_{Grid,PF}$	$t_{GFMI,Matlab}$	$t_{EVLoads,Matlab}$	$t_{elapsed}$	$t_{platform}$	
	(s)	(s)	(s)	(s)	(s)	(s)	(s)	
1	30	3	47	150	5	301	99	OFF
2	30	2	58	153	2	335	122	OFF
3	30	1	87	150	5	416	174	OFF
4	30	3	48	147	1	303	107	ON
5	30	2	61	143	6	337	127	ON
6	30	1	84	149	7	417	177	ON

The following target metrics described in the canvas document are evaluated for the successful testing of the IaaS. The description of target metrics is as follows:

1. Functionality test described by the consistency of the simulation results of digital twins.
    - Functionality test in validated from the response of digital twins to a frequency disturbance event generated in the synthetic grid model for simulated and real-time components in each scenario. The details of the simulation results including illustrations are described in the complete report.
2. Round Trip Delay (RTD) of the IaaS described by the co-relation between the elapsed time $t_{elpased}$ and the co-simulation time step Δt_{co-sim} for the simulated scenarios.
    - $t_{elpased}$ is evaluated for each simulation scenario and summarized in Table 1.

The following key statements are valid from the evaluated target metrics in the project test-case.

- Increasing the co-simulation step-size (Δt_{co-sim}) reduces the observability of the simulation
- Increasing the co-simulation step-size (Δt_{co-sim}) reduces the elapsed time $t_{elapsed}$ of the simulation
- Increasing the co-simulation step-size (Δt_{co-sim}) increases the time delay between the actual response and observed response of the digital twins in the simulation.

It is concluded that the co-simulation step-size (Δt_{co-sim}) is a design parameter specific to the application. There exists a trade-off between system observability and system delay. Higher delays and elapsed times are generally acceptable for co-simulation setup with only simulated components whereas lower delays are preferred for co-simulation setup with real components.

It is also concluded that the 5G Link has a minimal effect on the results of the digital twins and the elapsed times of the IaaS platform.

1 Lab-Access User Project Information

1.1 Overview

The lab-access project "Unified Infrastructure-as-a-Service (IaaS) for Distributed Co-Simulation of Networked Microgrids" was conducted at the host lab of VTT in the Intelligent Energy Testbed in the time period from 16.09.2024 to 25.09.2024. The second visit took place from 17.02.2025 to 28.02.2025. The user group for the first visit included Nauman Beg, Henning Schlachter, and Philipp Gottfried from the German Aerospace Center Institute of Networked Energy Systems. For the second visit, the user group comprised Nauman Beg, Mats Buchholz, and Saikrishna Vallabhaneni also from the same institute .The user group was supported by the host lab's scientific personnel, Ville Ollikainen and Atte Saarni, during the first visit, and Ville Ollikainen and Heli Kokkoniemi-Tarkkanen during the second visit.

1.2 Research Motivation, Objectives, and Scope

1.2.1 Motivation

The research motivation behind the development and extension of the IaaS platform is to address challenge of simulation and testing of diverse CPPS under constrained lab environments using advance lab testing methods. The proposed approach reduces the modeling effort required to simulate the dynamics of grid assets by directly incorporating their response with composite modeling technique in the platform. The platform leverages Service-Oriented Architecture (SOA) to establish coherent integration and communication framework between individual assets in the platform.

1.2.2 Objectives

The primary objective of the User Project is to extend and simulate a synthetic LV grid user test-case within a distributed co-simulation framework involving networked grid assets using digital twin implementation. This goal is pursued through the development of a unified IaaS platform that facilitates the integration and simulation of various grid assets.

To validate the functionality of the platform, the user test-case is implemented with real-time components integrated into the Intelligent Energy Testbed, which is hosted at the VTT lab. The validation process ensures that the platform operates effectively in a real-world environment, confirming its robustness and reliability.

The core objective of the project is to demonstrate a Proof-of-Concept (PoC) of a Service-Oriented Architecture (SOA) for networked co-simulation. This PoC will establish the feasibility of performing co-simulations between both simulated and real grid assets, using digital twin technology. The demonstration will serve to validate the potential of this approach for improving grid asset management and control in real-time settings.

1.2.3 Scope

The scope of this user project focuses on networked co-simulations between distributed grid assets using digital twins. The digital twins of individual assets are used to simulate the be-

havior of each asset in grid simulations. Each digital twin is integrated in the simulated grid with a composite model. The composite model defines the specification framework necessary for establishing communication between the digital twin and the actual asset in the IaaS platform. The method is well suited for loosely coupled simulations of quasi-dynamic systems and is targeted towards analysing ancillary grid services and control of grid assets for stable power system operation.

1.3 Structure of the Document

This document is organized into several sections, each serving a specific purpose. The structure is as follows:

- **Section 1**: This section provides user information about the lab access.
- **Section 2**: This section provides a brief outline of the state-of-the-art/state-of-technology that provides the basis of the realised Lab Access (LA) User Project (UP).
- **Section 3**: This section outlines the methodology and the experiments that were conducted and also discusses about the automation script of the models.
- **Section 4**: The results from the conducted experiments are summarized, and key conclusions are drawn based on the findings.
- **Section 5**: This section highlights the potential open issues and suggestions for improvements.

2 State-of-the-Art/State-of-Technology

Digital real-time simulators are intensively deployed in various academic research activities to emulate the behaviour of real components in a simulated environment. Typical application ranges from control prototyping of components to system stability analysis and typically requires advanced real-time simulators with high speed hardware interfaces [2, 3]. For precise analysis (EMT), most real-time simulators rely on controlled monolithic frameworks. Few co-simulation interface algorithms enabling co-simulation between real-time systems exist. [2, 4] They generally rely on synchronous phasor measurement systems to improve the accuracy of transmitted analogue signals. Other methods rely on phasor extraction methods [5] to improve dynamic simulation scalability of co-simulations. These frameworks are relevant for strongly coupled co-simulations but their implementation for real-time applications is still restricted due to their high bandwidth requirements (network jitter) and computational effort (EMT domain analysis).

Another domain of co-simulation framework for analysis of Multimodel-Energy System (MES) with communication networks are the so-called Cyber-Physical Energy System (CPES) [6, 7]. These platforms take advantage of modern Information and Communication Technology (ICT) infrastructure to enable interoperability among different models and virtual or physical devices through facilitation of information flow [6, 7]. Typically, CPES platforms leverage ICT infrastructure for providing general purpose services in smart grids [7] and are more heterogeneous in nature.

The co-simulation approach for distributed control applications in a multi-physics model co-simulation setup has recently gained significance in smart grids [8], especially in distribution grids with high penetration of decentralized generation units. The scope of such frameworks is control application in weakly coupled co-simulation environments including soft real-time systems. One typical use-case is agent-based decentralized control in microgrids. Co-simulation frameworks, like Mosaik, have been developed to ease such integration cases. A new challenge is the different temporal and spatial scales that are involved in the real world models and the software simulators that must be addressed during co-simulation.

In addition to the co-simulations in the electrical domain, new developments also consider coupling of power grids with other domains. An example for a co-simulation framework of smart grids with ICT is given in [9]. Furthermore, in [10, 11] investigations related to smart homes and buildings were conducted, in which co-simulation approaches were applied to integrate different software components via the functional mock-up interface.

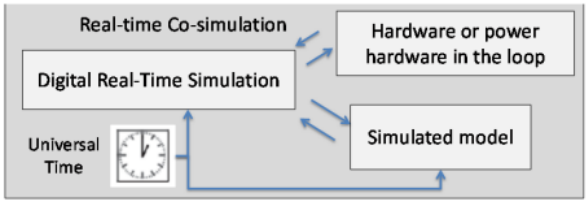

Figure 2: Overview of the real-time co-simulation [1].

Particularly, the latter publication presents an approach using the Mosaik platform from [12], which is also part of this research project.

Experiments on electrical and thermal co-simulation with geographically distributed real-time simulators are reported in [13] with the focus on accuracy, latency and stability. Additional case studies discussing geographical distributed simulations can be found in [14], while further

investigations on accuracy, latency and stability of real-time simulations are presented in [15, 16, 17].

The literature review reveals that the implementation of sub-systems in publicly available co-simulation platforms for MESs is not very coherent and still requires detailed information about the behaviour of individual platform as well as technical knowledge of the communication protocols. The unified IaaS platform addresses this issue using SOA with standardized implementation to integrate each asset.

Another interesting work dealing with the same idea is to standardize integration of additional components into the co-simulation from [18] and was named SEAS engine, which is based Hierarchical Engine for Large-scale Infrastructure Co-Simulation (HELICS), a comparable software tool to Mosaik. HELICS integrates the Simulators from different programming languages. While SEAS is focused on the standardization process itself, the IaaS platform additionally provides the capability of distributed co-simulation.

3 Methodology

3.1 Procedure and Methodology

To conduct the experiments and test the integration of new simulators into the IaaS-Platform, the first step was to install the required softwares PowerFactory, MATLAB/Simulink and Python as well as to setup the mosaik orchestrator including required models on the computers. The inputs and outputs of both simulated as well as real-time models are interfaced to the IaaS over a modeling methodology referred to as Composite modeling in the IaaS. The Composite models require the following necessary descriptors that are written for each model in the IaaS.

1. Remote web service for communication between each simulator and the the IaaS
2. Protocol converter for communication between the each remote web service and the instance of the model
3. Debug environment based on Representational State Transfer (REST) Application Programming Interface (API) for functionality tests of each web service during development phase.

As for the components PowerFactory and MATLAB/Simulink the necessary remote web services and protocol converters were already written in the platform. The next step required the incorporation of the RTDS. To operate and control the system, the software RSCAD is used to establish the connection to the real-time simulator as well as executing and analysis of real-time simulations. In the first attempt to integrate this simulator into the co-simulation platform, an additional web service based on TCP-based socket communication was chosen to control the execution of a real-time model including the operations of starting and stopping simulations as well as reading and writing values from components. The methods implemented in the web service are summarized in Figure 3.

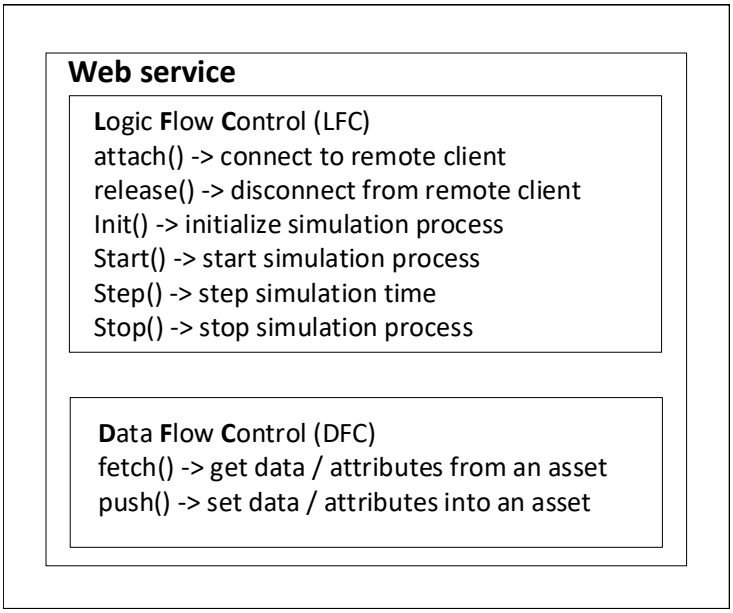

Figure 3: Web service methods for RTDS integration in IaaS over RSCAD.

3.2 Test Set-Up(s)

3.2.1 Experimental test-setup

The test setup to investigate the IaaS consisted of a laptop and two Virtual Machines (VMs) that were running the required softwares for the co-simulation. This is shown in figure 4.

The following software environments and services were running on the VM1:

- Mosaik orchestrator on a linux operating system

The second computer was running the following services:

- PowerFactory grid model

The VM2 was running the following services in a windows:

- RSCAD interface
- MATLAB/Simulink grid forming inverter model (EMT)
- MATLAB/Simulink electric vehicle fleet charging model (RMS)

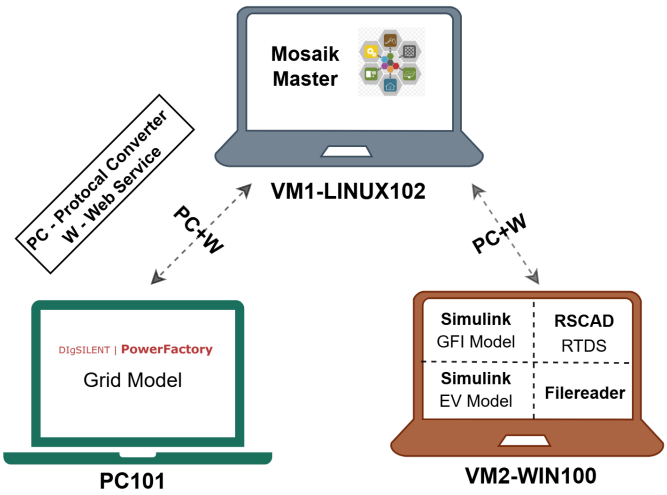

Figure 4: Test workflow overview.

The PowerFactory model runs the underlying grid model and provides voltage and frequency to the other simulation platforms as well as RSCAD. The electric vehicle fleet charging model does not require voltage and frequency information as it directly outputs active and reactive power based on a predefined time series it reads in. The second simulink instance running a grid forming inverter as Electro-Magnetic Transient (EMT) model receives voltage and frequency from the PowerFactory model over the mosaik platform and sends active and reactive power information back. The signal flow and exchanged values between different components of the co-simulation platform are shown in Figure 6.

3.2.2 Circuit diagram of the test grid

The circuit diagram of the emulated grid model in PowerFactory featuring a transformer, loads and a GFMI is shown in Figure 5. The main grid is connected to the primary side of the

transformer with 20 kV and the secondary winding to the bus with loads and GFMI at 400 V. The impedance of each line connected to the bus is denoted by Z_1, Z_2,Z_3 respectively.

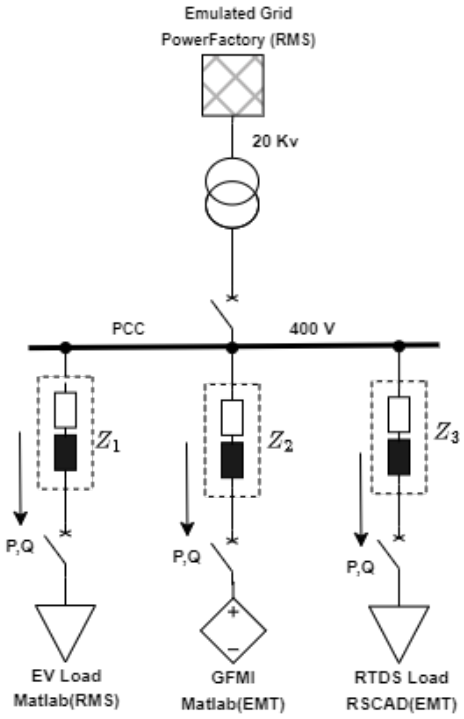

Figure 5: Overview of the simulated LV grid.

3.2.3 Co-Simulation models

Originally, three co-simulation scenarios are simulated for the grid model shown in Figure 5 with the test-setup in Figure 6. In each scenario, co-simulation time-step Δt_{co-sim} is modified as given in Table 2. The total simulation time $t_{simulation}$ and the simulation time-step in each simulation model is fixed. The details of simulated scenarios are also summarized in Table 2.

Table 2: Simulated scenarios in the project test-case.

	$t_{simulation}$ (s)	$\Delta t_{Grid,PF}$ ($ms_{[RMS]}$)	$\Delta t_{GFMI,Matlab}$ ($ms_{[EMT]}$)	$\Delta t_{EVLoads,Matlab}$ ($s_{[RMS]}$)	Δt_{co-sim} (s)	$t_{elapsed}$ (s)
Sc1	30	0.1	0.1	0.5	1	475
Sc2	30	0.1	0.1	0.5	2	382
Sc3	30	0.1	0.1	0.5	3	329

The emulated grid model is described by a mathematical model of actual installed capacities of synchronous, non-synchronous generation and connected loads in the German grid. In each scenario, a fixed power shortfall event of $\Delta P_{disturbance}$ =100 MW is simulated at $t_{sim} = 1.5\ s$ in the grid model to simulate dynamic grid frequency. The dynamic grid frequency and grid voltage at the respective grid nodes is communicated over the platform to the grid assets as shown in Figure 6 and their power response is communicated back to the their respective digital twins in the simulated grid in Figure 5.

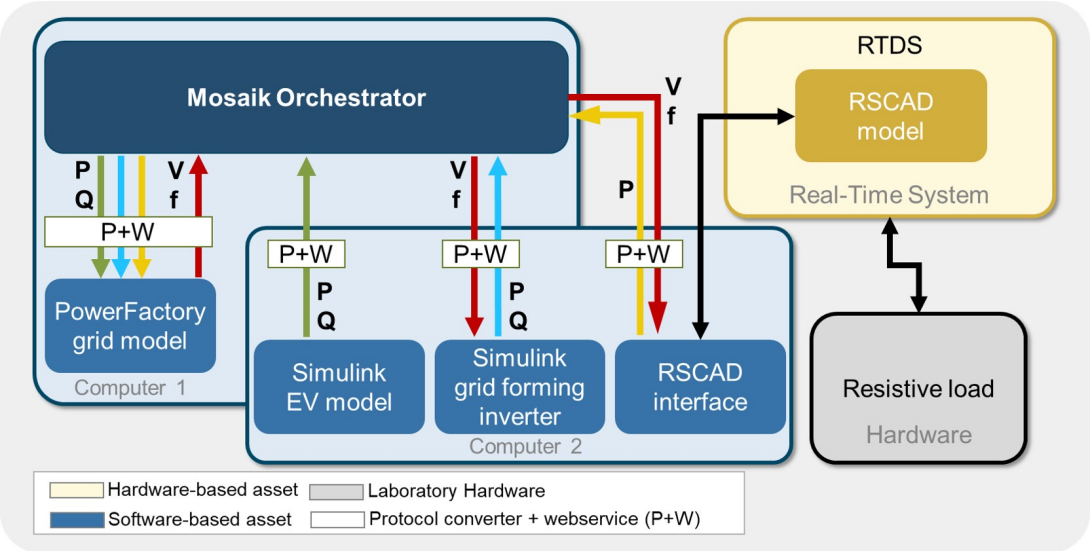

Figure 6: Overview of the integrated softwares and signal flow.

3.2.4 Extended Co-Simulation models of the second visit

The original simulation test-setup in section 3.2.3 is extended with the addition of a 5G communication link in between RTDS and Mosaik orchestrator as shown in Figure 7. The 5G emulator with pre-determined latency is added to the communication interface between RTDS and IaaS to observe the effect of network delays in the simulated results in the IaaS. This setup also includes the updated web services and protocol converters explained in detail in the section 3.3.

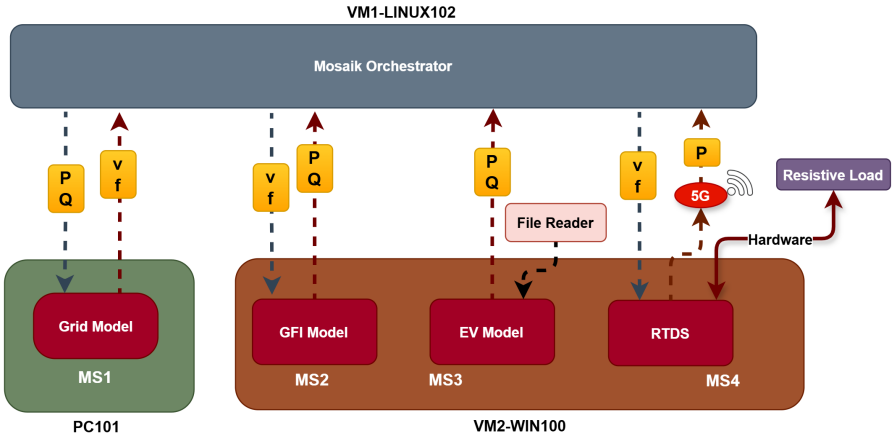

Figure 7: Overview of the signal flow in extended test-setup.

3.3 Code Development for Creation of a Co-Simulation

The general development workflow as used by our team is described in Figure 8. It explains the order in which the components of the IaaS framework were usually constructed. The following subsections are dedicated to explain each of the components' internal workflow in detail.

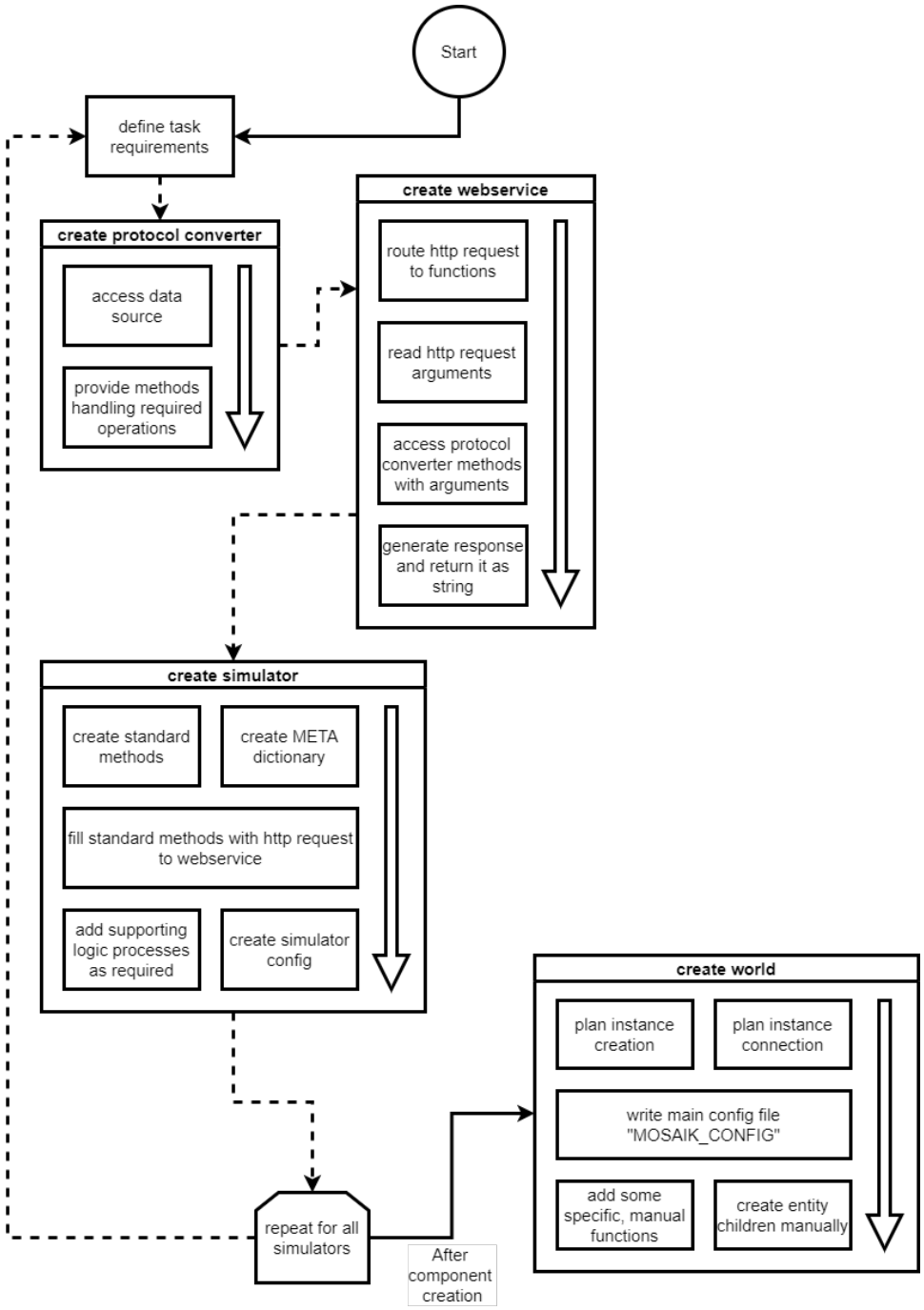

Figure 8: Overview of general workflow of creating a co-simulation.

This however is not a a rigid concept and can be approached in a different order if a task calls for it. As our code is usually developed in Python, the environment also has plenty of opportunities to include other libraries or self-made modules not described in this documentation.

3.3.1 Protocol Converters

The protocol converter is the most flexible building block of the IaaS framework. Its task is to provide the interface between the distributed data source and its web service. As such the

protocol converter has to adapt to the data source and use methods dependent on the source's properties to extract its data. These methods can range from an already existing API library to individual specific communication protocol messages. In the end the programming efforts should result in a class object providing methods to access the data source.
The development of most protocol converters was already completed in the previous work and was reused. During the second visit a new protocol converter for connection to a smart home located at the VTT in Oulu was written. It's using a python class provided by the VTT to access the sensors in Oulu via http requests over the internet.

3.3.2 Developing Web Services

The webservice is running on the remote location and provides routing of http requests to access the methods provided by the protocol converter. They are written using the Flask library for Python to set up the sessions listening for REST API requests from the orchestrator.

The development of new web services and the connected protocol converters is made more comprehensible by using the Swagger module of the Flasgger library. Flasgger is a python library applying the Swagger web API style for Flask objects in Python. It is used to provide a web interface providing access to all the routed functions of the running web service session as shown in Figure 9

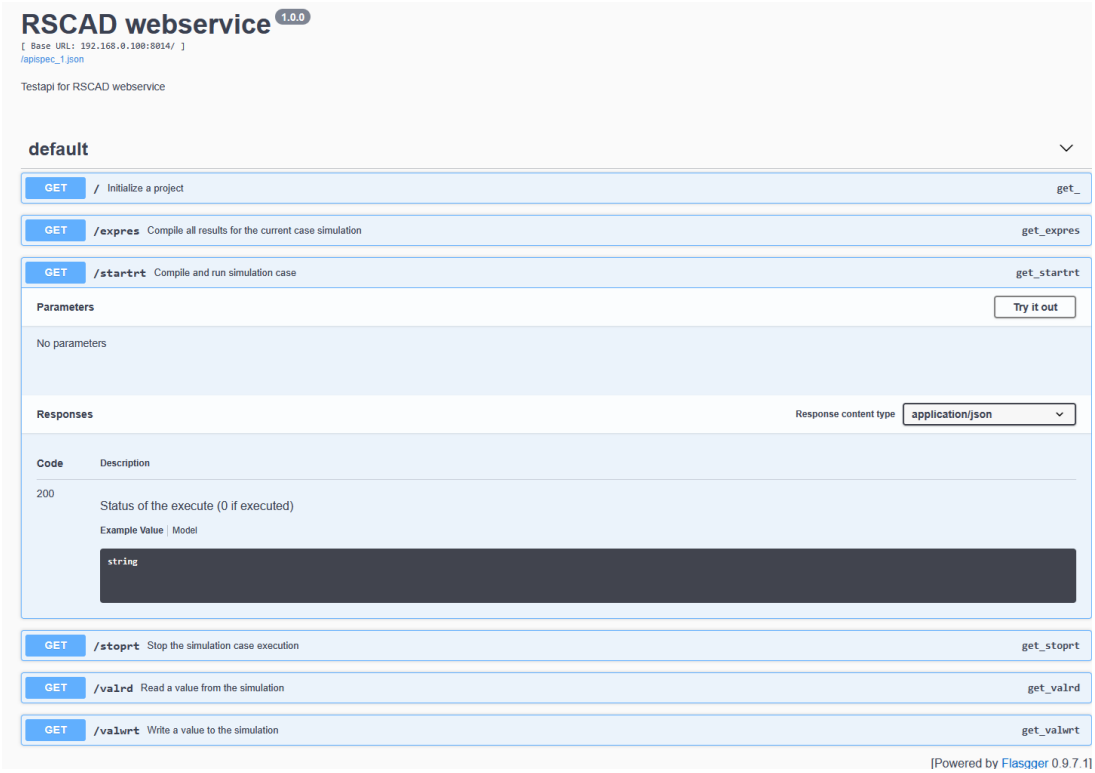

Figure 9: Overview of the Swagger WebAPI.

This provides a remote debugging interface which can be used to send http requested directly to the routed functions of the web service. You can input the parameters of the web service function and will receive the http response in the web interface. This way you can manually check the remote web services for proper operation or their behaviour under certain input parameters.

3.3.3 Creating mosaik simulators

The mosaik simulators are children of the simulator objects provided by the mosaik library. Their structure is pre-defined by the algorithms of the mosaik system and must conform to those standards. Every simulator must provide the four standard methods: init(), create(), step() and get_data(). Each of these methods is accessed at different points during the execution of the mosaik framework.
Further work on our continued development on the construction of new simulators has expanded the process from writing the Python algorithms to also creating a configuration setup for each simulator. The configuration describes an agent of the simulator class and specifies how it will be build as part of the mosaik system. It takes the form of a mapping, in our case a Python dictionary which is read from a .yaml file. Alternatively you could also write the agent config dictionary as a variable directly in the code or import it from a different file system. The template of a simulator config is shown in Figure 10.

```
sim_name: 'NAME_OF_SIMULATOR' # this key is always 'sim_name', the value has to be the same ID as in the SIM_CONFIG
world_starters: # this key is always 'world_starters'
  DICT: [0,F]
  PARAMETERS: USED
  FOR:
    SIMULATOR: {INIT: FUNCTION}
models: # this key is always 'models'
  MODELID: # the model-id has to correspond to the name
    UNIQUE_ENTITY_NAME:
      instantiators: # this key is always 'instantiators'
        DICT: [0,F]
        PARAMETERS: USED
        FOR:
          SIMULATOR: {CREATE: FUNCTION}
```

Figure 10: Agent config template.

The contents of an agent config file are made up of:

- The "sim_name" specifies the name of the target simulator from which the agent entity is created. The name must be the same as the simulator ID by which it is known to the "SIM_CONFIG".
- The "world_starters" dictionary contains the parameters given to the simulators init() function and their values.
- The "models" contain a dictionary mapping the "instantiators" to the unique IDs of individual entities and the specific model-ID of the target model to create an entity from, as it can be found in the "META" dictionary of the simulator under the "models" key. The "instantiators" key maps to another dictionary, which contains the parameters given to the create() function of the simulator and their values.

3.3.4 Setting Up Co-simulations

The automation of scripts and streamlining of the co-simulation setup is also one of the main goals to make the development of the simulators, web services and protocol converters for new models more approachable. To further this objective the "world builder" python object was developed, which can set up a mosaik co-simulation environment from a set of configuration dictionaries. This has greatly reduced the complexity of the main python code, by moving the system logic into the configuration files. With the new automation approach the python code can stay mostly unchanged and varying system setups can be realized by feeding different configs to the world builder.

The configuration of an entire co-simulation system is encompassed by three config dictionaries. We grouped this as three documents in one .yaml-file called "COSIM_CONFIG". Figure 11 shows the template for a "COSIM_CONFIG".

```
--- # Document 1: Contains the mosaik world configs required by mosaik itself.
# Includes the SIM_CONFIG and END as simulation duration
SIM_CONFIG: # key is always 'SIM_CONFIG', value is a dict describing the SIM_CONFIG dictionary required by mosaik
  'NAME_OF_SIMULATOR1':
    'TYPE_OF_SIMULATOR1': 'NAME_OF_FILE_CONTAINING_SIMUATOR:NAME_OF_SIMULATOR_CLASS_CHILD' # f.e.'PFSim': {'python': 'powerfactory_simulator:pfsimulator'}
  '...'

END: 3 # key is always 'END', value is an integer describing the simulation duration

--- # Document 2: Contains configs of all simulators.
# Ordered in a dict where the key is the name of the simulator
# and the value is its configuration according to the AGENT_CONFIG_TEMPLATE
SIMULATOR_ID1:
  sim_name: 'NAME_OF_SIMULATOR' # this key is always 'sim_name', the value has to be the same ID as in the SIM_CONFIG
  world_starters: # this key is always 'world_starters'
    DICT: [0,F]
    PARAMETERS: USED
    FOR:
      SIMULATOR: {INIT: FUNCTION}
  models: # this key is always 'models'
    MODELNAME:
      UNIQUE_ENTITY_NAME:
        instantiators: # this key is always 'instantiators'
          DICT: [0,F]
          PARAMETERS: USED
          FOR:
            SIMULATOR: {CREATE: FUNCTION}
SIMULATOR_ID2: {Simulator2_config}

--- # Document 3: Contains the connection config, describing the information exchange
# Ordered in a dict where the key is the name of the source entitiy_id
# and the value is another dict describing connection parameters keyed to the destination entity_id
'NAME OF SOURCE ENTITY1':
  'NAME OF DESTINATION ENTITY1': 'CONNECTED ATTR' # one connected attribute as a string
  'NAME OF DESTINATION ENTITY2': ['CONNECTED ATTR1', 'CONNECTED ATTR2'] # multiple connected attributes as a list of strings
  'NAME OF DESTINATION ENTITY3': {src_attr: 'CONNECTED ATTR1', dest_attr: 'ATTR_ALIAS1'} # one connected attribute with different attribute names at source and destination
NAME OF SOURCE ENTITY2:
  NAME OF DESTINATION ENTITY1: [{src_attr: 'CONNECTED ATTR1', dest_attr: 'ATTR_ALIAS1', weak: True, initial_data: {out: 0}}, 'CONNECTED_ATTR2'] # list of multiple connected attr
```

Figure 11: Co-Simulation config template.

Its contents are:

- The "WORLD_CONFIG" contains the "SIM_CONFIG" as required by the mosaik world and the parameter "END" describing the simulation duration.
- A dictionary of all the simulator agent configurations that are part of the simulation system. This is documented in a dictionary where each key maps top a agent config of a simulator agent as specified in Figure 10.
- The "connection_config" which maps keys of source entities to dicts of connections made up of the destination entities as keys and the connected attributes as values. It's template is shown in Figure 12.

```
'NAME OF SOURCE ENTITY1':
  'NAME OF DESTINATION ENTITY1': 'CONNECTED ATTR'
  'NAME OF DESTINATION ENTITY2': ['CONNECTED ATTR1', 'CONNECTED ATTR2']
  'NAME OF DESTINATION ENTITY3': {src_attr: 'CONNECTED ATTR1', dest_attr: 'ATTR_ALIAS1'}
NAME OF SOURCE ENTITY2:
  NAME OF DESTINATION ENTITY1: [{src_attr: 'CONNECTED ATTR1', dest_attr: 'ATTR_ALIAS1', weak: True, initial_data: {out: 0}}, 'CONNECTED_ATTR2']
```

Figure 12: Mosaik connection config template.

3.3.5 Time logging

During the second visit to Espoo we added time logging to the IaaS platform. This refers to the tracking of simulation events as timestamps during the execution of a model. It helps to monitor the progress of the simulators, web services and simulation softwares and analyze the performance of different components or agents in the model over time.

Time logging plays an important role in measuring the actual time taken by each element versus the total time recorded by the co-simulation orchestrator. There's always an inherent time delay due to communication lags, as well as the runtime of the Python script logic in different software simulators. This time delay can affect the accuracy of the recorded times.

The time is logged during the execution of each simulator by creating and appending to a .csv file. Time is recorded in a line of the .csv whenever the simulator runs and in case of a software model simulator also recorded again, after the connected simulation software has concluded its simulation step. Figure 13 shows a sample csv-file where the output of the time logging for one simulator is shown.

```
simulation_time;timestamp_before;timestamp_after
0.0;25-02-25-18-12-36;25-02-25-18-12-38
2.0;25-02-25-18-13-00;25-02-25-18-13-02
4.0;25-02-25-18-13-22;25-02-25-18-13-25
6.0;25-02-25-18-13-45;25-02-25-18-13-48
8.0;25-02-25-18-14-08;25-02-25-18-14-11
10.0;25-02-25-18-14-31;25-02-25-18-14-34
12.0;25-02-25-18-14-54;25-02-25-18-14-58
14.0;25-02-25-18-15-18;25-02-25-18-15-21
16.0;25-02-25-18-15-41;25-02-25-18-15-45
18.0;25-02-25-18-16-04;25-02-25-18-16-08
20.0;25-02-25-18-16-28;25-02-25-18-16-32
22.0;25-02-25-18-16-51;25-02-25-18-16-55
24.0;25-02-25-18-17-15;25-02-25-18-17-19
26.0;25-02-25-18-17-38;25-02-25-18-17-43
28.0;25-02-25-18-18-02;25-02-25-18-18-07
```

Figure 13: Example result file of time logging.

The flow diagram in Figure 14 describes the logic used for the time logging for software simulators such as Matlab, PowerFactory, as well as real-time systems such as RTDS.

Modeling Software Simulators

The logic time in each software simulator Mosaik Slave (MS) for a single co-simulation time-step is calculated by the time difference noted in the each stamping before and after the execution of the logic:

$$TS_2 - TS_1 = \Delta TS$$

The total Communication Time (CT) is the difference of the post logic Time Stamp (TS_2) in MS and the timestamp recorded by the Mosaik master (TS_3) for the each simulator.

$$TS_3 - TS_2 = CT$$

Real-Time Simulators

The calculation of the total communication time for the real time systems(CT_{RTS}) is slightly different to that in the software simulators. Its the difference between the timestamp observed by MS (TS_4) before readout command to the timestamp value (TS_5) of the Mosaik master when it recieves the data value.

$$TS_5 - TS_4 = CT_{RTS}$$

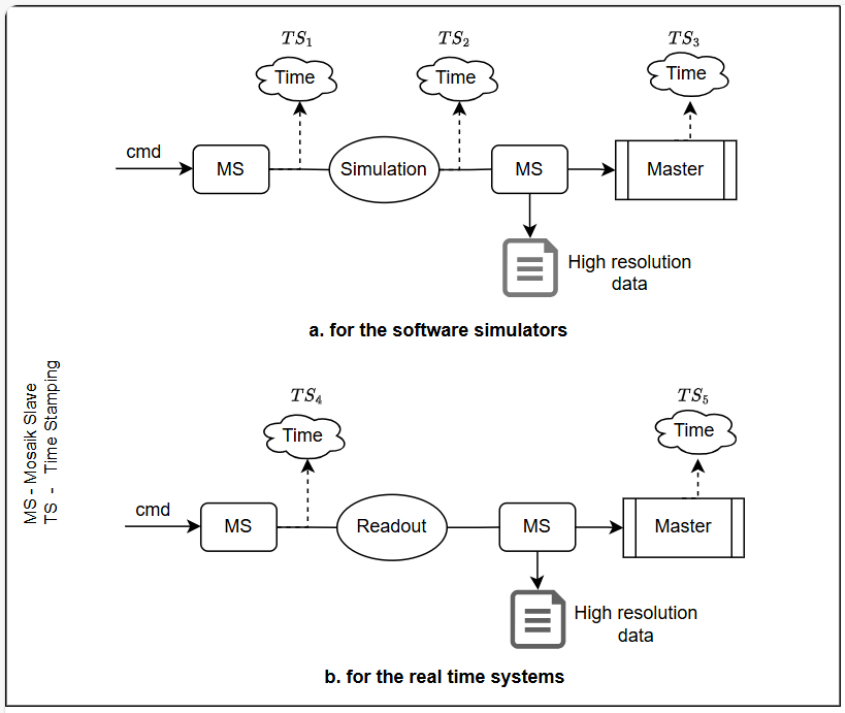

Figure 14: Structure for the time logging.

3.4 Data Management and Processing

A data collector is implemented in the Mosaik orchestrator that is responsible for collecting and storage of all data exchange during runtime between individual simulation or emulation model instances in the IaaS. During each co-simulation time-step Δt_{co-sim}, the collector module saves the actual model outputs from individual instances. The output file includes the absolute time, values for the exchanged attributes over the platform and the co-simulation time-steps.

In addition to the collection of all data exchanges between individual instances at Δt_{co-sim}, the detailed results for each model are stored locally on the respective remote systems where model instances are running. The detailed results are collected to benchmark the actual response of the models with their observed response in the PowerFactory grid model.

4 Results and Conclusions

4.1 Discussion of Results

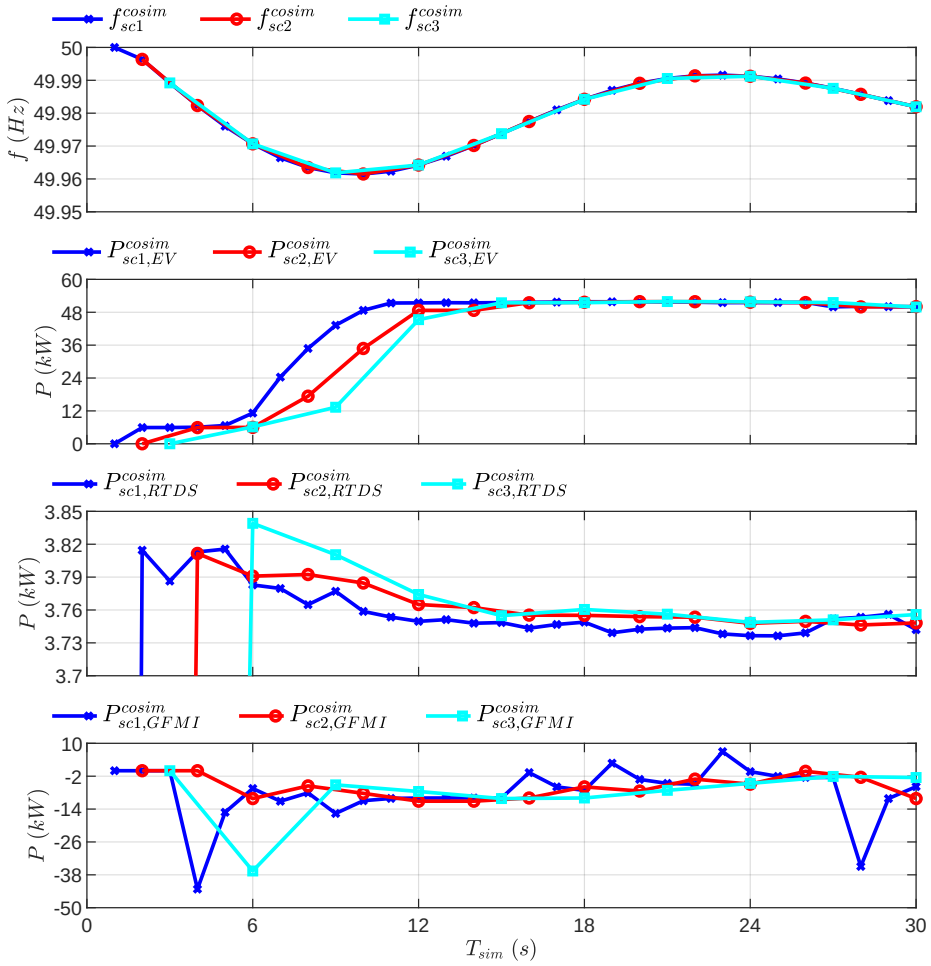

Figure 15: Results of the co-simulation scenarios from visit-1.

In each scenario described in the original test-setup in section 3.2.3, grid frequency and active power of digital twins in the simulated grid model at the respective Δt_{co-sim} time-steps is plotted against the $t_{simulation}$ and shown in Figure 15.Two target metrics described in the canvas document are evaluated from the simulation results for the successful testing of the IaaS:

1. Functionality test described by the consistency of the simulation results of digital twins.
    - Functionality test validated from the response of digital twins to a frequency disturbance event generated in the synthetic grid model for simulated and real-time components in each scenario.
2. RTD of the IaaS described by the co-relation between the $t_{elpased}$ and the Δt_{co-sim} for the simulated scenarios.

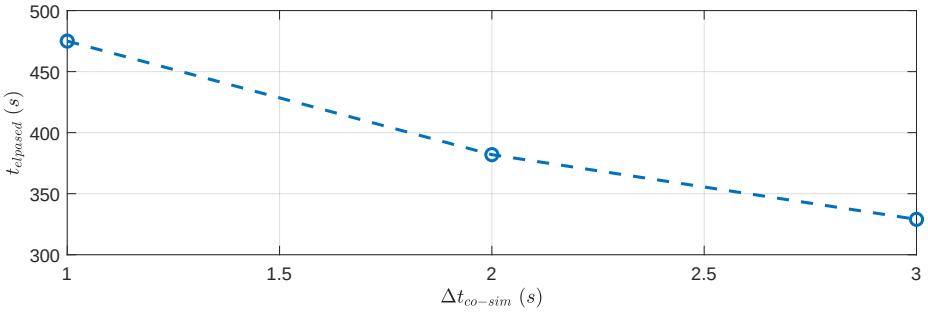

Figure 16: $t_{elapsed}$ vs Δt_{co-sim}.

- $t_{elapsed}$ is evaluated for each simulation scenario and summarized in Table 2.

Based on the first target metric, the following observations from Figure 15 are presented:

- It is noted that the observability of the co-simulation results is dependent on the Δt_{co-sim}. The peak transients in the GFMI active power in response to the grid disturbance are only observed in the digital twin response in $P^{cosim}_{sc1,GFMI}$ (blue) and not captured in the $P^{cosim}_{sc2,GFMI}$ (red) and $P^{cosim}_{sc3,GFMI}$ (sky blue).
- A large Δt_{co-sim} introduces larger delays in the observed response in digital twins. This is observed in the the simulated power response of each digital twin in the grid model. The reason is delayed update of the actual value in the digital twin from its respective grid asset. The frequency response is not affected as the its actual value is updated directly in the grid simulation model.
- The response of digital twins converges to the same values in all scenarios for quasi-dynamic / stationary conditions. This is observed from the power response of EV and RTDS digital twins after initial transients are damped. The dynamics of the GFMI EMT model is highly sensitive to grid frequency and hence only the average power is similar for all scenarios.

Based on the second target metric, the following observations are presented:

- The total elapsed time $t_{elapsed}$ for each scenario is summarized in Table 2. It is noted that $t_{elapsed}$ increases for small Δt_{co-sim} due to frequent data exchange between the IaaS and grid assets.
- The reduction in $t_{elapsed}$ is not linear with Δt_{co-sim} since the internal simulation time of each model is independent of the Δt_{co-sim}. This is shown in Figure 16 where the total elapsed time converges to the sum of total simulation time of individual simulators with increase in the Δt_{co-sim}.

4.2 Discussion of Results in the Extended Test-Setup

The extended test-setup in section 3.2.4 is simulated for six different scenarios listed in Table 3. The first three co-simulation scenarios are without 5G communication link, meaning, there are no additional network delays from 5G. The last three scenarios are with the 5G communication link. Scenario 5 is further classified into two sub-scenarios: 5a, with minimum network delays from 5G, and 5b, representing 5G with high network delays (500 ms). In each scenario, the

Table 3: Stamped time example for the simulators and the master.

Scenario	$t_{simulation}$	Δt_{co-sim}	Elapsed time					5G
			$t_{Grid,PF}$	$t_{GFMI,Matlab}$	$t_{EVLoads,Matlab}$	$t_{elapsed}$	$t_{platform}$	
	(s)	(s)	(s)	(s)	(s)	(s)	(s)	
1	30	3	47	150	5	301	99	OFF
2	30	2	58	153	2	335	122	OFF
3	30	1	87	150	5	416	174	OFF
4	30	3	48	147	1	303	107	ON
5	30	2	61	143	6	337	127	ON
6	30	1	84	149	7	417	177	ON

frequency (f_{PF}^{grid}) response of each co-simulated model to the power imbalance event in the PowerFactory grid model already introduced in the section 3.2.3 is recorded. The observed response in the digital twin of each component of the PowerFactory grid is presented with its actual response measured in the corresponding model instance. Each graph is plotted against the simulation time of the PowerFactory grid. For simplification, the report presents the results for the most relevant scenarios only.

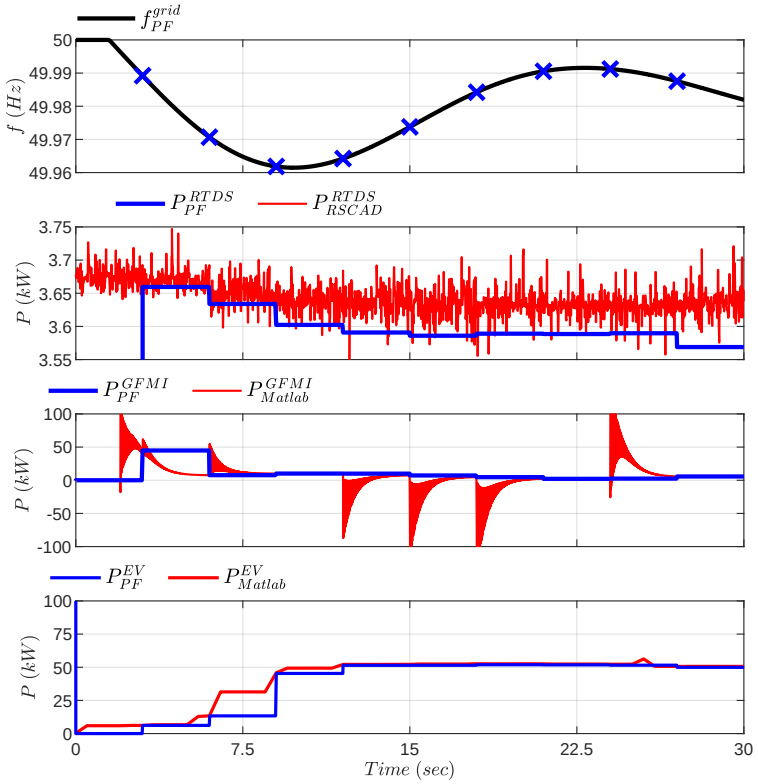

Figure 17: Results of the scenario 1.

Figure 17 presents the results for Scenario 1. The first subplot shows the disturbed grid fre-

quency ($f_{\text{PF}}^{\text{grid}}$) in the simulated grid. The blue cross marks on the curve indicate time instances at which the model output data is exchanged between simulator instances over IaaS. In Scenario 1, this interval is 3 seconds as given in Table 3. In the subsequent subplots, the blue curve represents the response of the co-simulation's digital twin whereas the red curve represents the actual response of each model recorded locally on the remote systems.

The response of the digital twin is always delayed compared to the actual response of each component. The delay is obvious in the simulated response of the EV model. In Figure 18, results of Scenario 3 in Table 3 are presented, where the step-size (Δt_{co-sim}) is reduced to one second. With a smaller step-size, the digital twin response is improved compared to the actual response. The drawback to it is the increase in total elapsed time for the co-simulation due to frequent data update requests over the platform that increases the elapsed time of the platform t_{platform} (see. Table 3). Also, a larger Δt_{co-sim} (Scenario 1) results in larger delay compared to Scenario 3.

A close comparison between the two scenarios reveals that a larger Δt_{co-sim} leads to less detailed observations, possibly missing subtle behaviors, especially in the RTDS ($P_{\text{RSCAD}}^{\text{RTDS}}$) and the GFMI ($P_{\text{GFMI}}^{\text{Matlab}}$).

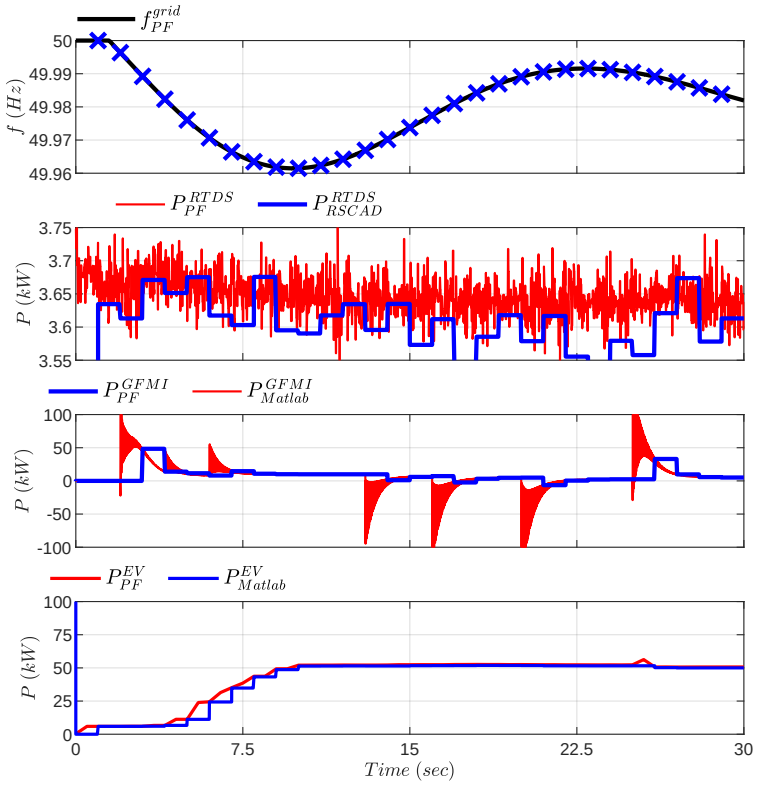

Figure 18: Results of the scenario 3.

The Figure 19 shows the results for Scenario 4 with a Δt_{co-sim} of three seconds and 5G communication enabled. The yellow curve in the RTDS plot represents the actual power of RTDS

without 5G ($P^{\text{RTDS}}_{\text{RSCAD,No5G}}$) link in addition to the red curve that represents the actual power of the RTDS with 5G link. The digital twin response of the RTDS connected load between Scenario 1 (Figure 17) and Scenario 4 (Figure 19) is minimally effected with 5G communication delays.

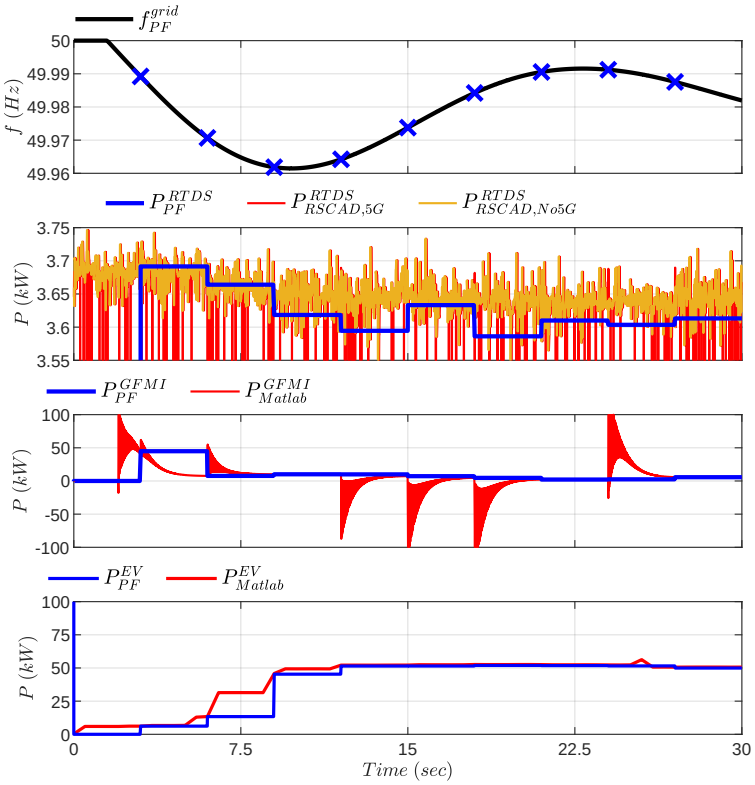

Figure 19: Results of the scenario 4.

Finally, the relationship between the elapsed time of the platform and the co-simulation step-size (Δt_{co-sim}) is plotted in Figure 20. In the current implementation of the IaaS, the elapsed time of platform does not necessarily scales linearly with Δt_{co-sim}. It can potentially limit the application of the platform in real-time applications with hardware components and a requirement for a smaller Δt_{co-sim}. The IaaS is more suited for co-simulations with quasi-dynamic or steady-state models.

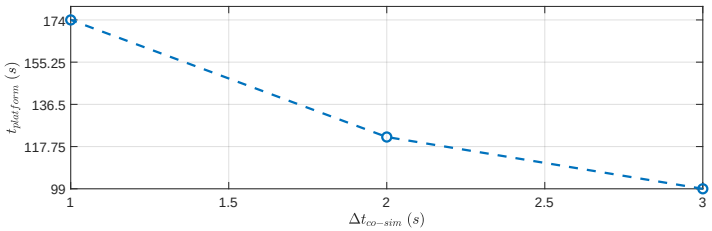

Figure 20: Platform delays with Δt_{co-sim}.

4.3 Conclusions

The following key statements are valid from the evaluated target metrics in the project test-case.

- Increasing the co-simulation step-size (Δt_{co-sim}) reduces the observability of the simulation
- Increasing the co-simulation step-size (Δt_{co-sim}) reduces the elapsed time $t_{elapsed}$ of the simulation
- Increasing the co-simulation step-size (Δt_{co-sim}) increases the time delay between the actual response and observed response of the digital twins in the simulation.
- Enabling 5G communication has no significant effect on the platform delays.

It is concluded that the co-simulation step-size (Δt_{co-sim}) is a design parameter specific to the application. There exists a trade-off between system observability and system delay. Higher delays and elapsed times are generally acceptable for co-simulation setup with only simulated components whereas lower delays are preferred for co-simulation setup with real components.

5 Open Issues and Suggestions for Improvement

- Due to the limited number of visit days, use-case of geographically distributed co-simulation through lab coupling between grid assets in Espoo and Oulu over IaaS is not simulated.
- The improvement in $t_{elapsed}$ by reduction in platform overhead with code optimization in IaaS is still open.
- Implementation of Distributed Application (dApp) for live visualization of platform data is open.

References

[1] X. G. Christian Rehtanz, "Real-time and co-simulations for the development of power system monitoring, control and protection," *International Journal of Electrical Power & Energy Systems*, pp. 1–20, Dec. 2019. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7541030

[2] "Presentation Slides - RESERVE Mainpage — re-serve.eu," http://re-serve.eu/presentation-slides.html, [Accessed 10-10-2024].

[3] "RWTH Aachen | Urban Energy Lab 4.0 - Echtzeitsimulationslabor — uel4-0.de," https://www.uel4-0.de/Infrastruktur/Echtzeitsimulationslabor/, [Accessed 10-10-2024].

[4] S. V. Marija Stevic, "CoSiF — fein-aachen.org," https://www.fein-aachen.org/projects/cosif/, [Accessed 10-10-2024].

[5] F. Marten, "OpSim — iee.fraunhofer.de," https://www.iee.fraunhofer.de/de/schnelleinstieg-wirtschaft/themen/opsim.html, [Accessed 10-10-2024].

[6] T. S. Theodoro, M. A. Tomim, P. G. Barbosa, A. C. Lima, and M. T. Correia de Barros, "A flexible co-simulation framework for penetration studies of power electronics based renewable sources: A new algorithm for phasor extraction," *International Journal of Electrical Power & Energy Systems*, vol. 113, p. 419–435, Dec. 2019. [Online]. Available: http://dx.doi.org/10.1016/j.ijepes.2019.05.047

[7] L. Barbierato, A. Estebsari, L. Bottaccioli, E. Macii, and E. Patti, "A distributed multimodel cosimulation platform to assess general purpose services in smart grids," *IEEE Transactions on Industry Applications*, vol. 56, no. 5, p. 5613–5624, Sep. 2020. [Online]. Available: http://dx.doi.org/10.1109/tia.2020.3010481

[8] K. Johnstone, S. M. Blair, M. H. Syed, A. Emhemed, G. M. Burt, and T. I. Strasser, "Co-simulation approach using powerfactory and matlab/simulink to enable validation of distributed control concepts within future power systems," *CIRED - Open Access Proceedings Journal*, vol. 2017, no. 1, p. 2192–2196, Oct. 2017. [Online]. Available: http://dx.doi.org/10.1049/oap-cired.2017.1175

[9] H. Lin, S. Sambamoorthy, S. Shukla, J. Thorp, and L. Mili, "Power system and communication network co-simulation for smart grid applications," in *ISGT 2011*. IEEE, Jan. 2011, p. 1–6. [Online]. Available: http://dx.doi.org/10.1109/isgt.2011.5759166

[10] J. A. Dominguez, L. Rueda, N. Henao, K. Agbossou, and J. Campillo, "Distributed co-simulation for smart homes energy management in the presence of electrical thermal storage," in *IECON 2022 – 48th Annual Conference of the IEEE Industrial Electronics Society*, 2022, pp. 1–6.

[11] D. S. Schiera, L. Barbierato, A. Lanzini, R. Borchiellini, E. Pons, E. Bompard, E. Patti, E. Macii, and L. Bottaccioli, "A distributed multimodel platform to cosimulate multienergy systems in smart buildings," *IEEE Transactions on Industry Applications*, vol. 57, no. 5, pp. 4428–4440, 2021.

[12] C. Steinbrink, M. Blank-Babazadeh, A. El-Ama, S. Holly, B. Lüers, M. Nebel-Wenner, R. P. Ramírez Acosta, T. Raub, J. S. Schwarz, S. Stark, A. Nieße, and S. Lehnhoff, "Cpes testing with mosaik: Co-simulation planning, execution and analysis," *Applied Sciences*, vol. 9, no. 5, 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/5/923

[13] M. O. Faruque, M. Sloderbeck, M. Steurer, and V. Dinavahi, “Thermo-electric co-simulation on geographically distributed real-time simulators,” in *2009 IEEE Power & Energy Society General Meeting*. IEEE, Jul. 2009, p. 1–7. [Online]. Available: http://dx.doi.org/10.1109/pes.2009.5275631

[14] M. H. Syed, E. Guillo-Sansano, Y. Wang, S. Vogel, P. Palensky, G. M. Burt, Y. Xu, A. Monti, and R. Hovsapian, “Real-time coupling of geographically distributed research infrastructures: Taxonomy, overview, and real-world smart grid applications,” *IEEE Transactions on Smart Grid*, vol. 12, no. 2, pp. 1747–1760, 2021.

[15] L. Barbierato, E. Pons, E. F. Bompard, V. S. Rajkumar, P. Palensky, L. Bottaccioli, and E. Patti, “Exploring stability and accuracy limits of distributed real-time power system simulations via system-of-systems cosimulation,” *IEEE Systems Journal*, vol. 17, no. 2, pp. 3354–3365, 2023.

[16] E. Bompard, S. Bruno, A. Cordoba-Pacheco, C. Diaz-Londono, G. Giannoccaro, M. L. Scala, A. Mazza, and E. Pons, “Latency and simulation stability in a remote power hardware-in-the-loop cosimulation testbed,” *IEEE Transactions on Industry Applications*, vol. 57, no. 4, pp. 3463–3473, 2021.

[17] L. Barbierato, E. Pons, E. F. Bompard, V. S. Rajkumar, P. Palensky, L. Bottaccioli, and E. Patti, “Exploring stability and accuracy limits of distributed real-time power system simulations via system-of-systems cosimulation,” *IEEE Systems Journal*, vol. 17, no. 2, pp. 3354–3365, 2023.

[18] S. Helman, K. Panda, D. Vaidhynathan, B. Knueven, D. Sigler, W. Jones, and J. King, “Seas communication engine: An extensible, flexible wrapper for cosimulation agents,” in *2024 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, 2024, pp. 1–5.

Disclaimer

This document contains material, which is copyrighted by the authors and may not be reproduced or copied without permission.

The commercial use of any information in this document may require a licence from the proprietor of that information.

Neither the Lab Access User Group as a whole, nor any single person warrant that the information contained in this document is capable of use, nor that the use of such information is free from risk. Neither the Lab Access User Group as a whole, nor any single person accepts any liability for loss or damage suffered by any person using the information.

This document does not represent the opinion of the European Community, and the European Community is not responsible for any use that might be made of its content.

Copyright Notice

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 870620.