Politecnico di Torino

ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation
Doctoral Program in Control and Computer Engineering (34^{th}cycle)

Modelling and Co-simulation of Multi-Energy Systems

Distributed Software Methods and Platforms

By

Luca Barbierato

******

Supervisor(s):
Prof. Enrico Macii, Supervisor
Prof. Ettore Francesco Bompard, Co-Supervisor

Doctoral Examination Committee:
Prof. Davide Brunelli, Referee, Università degli Studi di Trento
Prof. Ana-Maria Dumitrescu, Referee, Politehnica University of Bucharest
Prof. Marco Aiello, Committee Member, University of Stuttgart
Prof. Milos Cvetkovic, Committee Member, Technische Universiteit Delft
Prof. Andrea Calimera, Committee Member, Politecnico di Torino

Politecnico di Torino
12 September 2022

Declaration

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Luca Barbierato
12 September 2022

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

I would like to dedicate this thesis to my loving parents,
and to my loved ones.

Acknowledgements

I would like to acknowledge the EDA Group of the Department of Control and Computer Engineering for their warm welcome in 2015 and their essential help during this three-year of Ph.D. activities. In particular, I would like to thank Prof. Enrico Macii for giving me the opportunity of undertaking my Ph.D. career and Prof. Edoardo Patti for his trust and guidance that allow my competencies to flourish in the unexplored field of Computer Science. Last, but not least, I would like to express my gratitude to Lorenzo Bottaccioli for working with me on a daily basis, sharing his knowledge, and leading me in this research. Finally, I would like to acknowledge the Energy Center LAB colleagues, Prof. Romano Borchiellini, Prof. Ettore Francesco Bompard, Prof. Enrico Pons, Prof. Andrea Lanzini, and my Ph.D. fellow Daniele Schiera that have collaborated with me in developing energy aspects of the software infrastructure solutions proposed in this dissertation.

Abstract

The ongoing energy transition to reduce carbon emissions presents some of the most formidable challenges the energy sector has ever experienced, requiring a paradigm change that involves diverse players and heterogeneous concerns, including regulations, economic drivers, societal, and environmental aspects. Central to this transition is the adoption of integrated Multi-Energy Systems (MES) to efficiently produce, distribute, store, and convert energy among different vectors. A deep understanding of MES is fundamental to harness the potential for energy savings and foster energy transition toward a low-carbon future. Unfortunately, the inherent complexity of MES makes them extremely difficult to analyze, understand, design, and optimize.

This dissertation addresses this problem by applying Information and Communication Technology (ICT) and proposing different distributed software methods and platforms for modelling and co-simulating MES. These solutions will enable the definition of a virtual representation of the real world as a composition of models by applying co-simulation techniques, permitting the analysis of a MES scenario from multiple viewpoints and at different spatio-temporal scales, providing a structured basis to design, develop and validate novel solutions and technologies for MES.

Firstly, this dissertation proposes GAMES, a General-purpose Architecture model for MES, which helps designers in describing a MES scenario by applying Model-Based System Engineering (MBSE) and Model Driven Architecture (MDA) strategies. GAMES builds a solid conceptual framework of the energy sectors and aspects involved in MES offering a guided and semi-automated definition of complex MES scenarios. Then, it proposes three variants of a co-simulation infrastructure to address the interconnection of heterogeneous General-purpose Programming Language (GPL), software, and hardware simulators. Depending on the interconnected simulators, a MES designer could choose among: *i)* the Pure Software Co-simulation

Infrastructure when the MES scenario presents models developed in GPL and software simulators with slow temporal evolution, i.e. their time step duration is equal or greater than one second; *ii)* the Digital Real-Time Co-simulation Infrastructure when the MES scenario presents models developed with hardware simulators with fast temporal evolution, i.e. their time step duration is around tens of microseconds with strict real-time requirements for Hardware-In-the-Loop (HIL) and Power Hardware-In-the-Loop (PHIL) testing; and, finally, *iii)* the Hybrid Co-simulation Infrastructure that joins together the previous infrastructures to simulate complex MES scenarios with both slow and fast temporal evolution of involved simulators. To conclude, the Distributed Event-Driven Infrastructure is proposed with the purpose of scaling a simulated MES application and/or service from a testing environment to a real-world scenario.

The proposed solutions will ultimately promote and accelerate the energy transition by making the simulation of complex MES scenarios easier, more accurate, and more efficient. Ease of use of such a solution comes from tools that automate and validate the definition of involved energy scenarios. Accuracy instead comes from the inclusion of heterogeneous simulators that capture the multifaceted nature of the scenario under analysis. Finally, efficiency comes from distributed simulations and support for dedicated hardware.

This promises a broad impact for both academic researchers and practitioners working in the energy sector. Indeed, the solutions proposed in this dissertation offer a common playground where both researchers and companies can share knowledge and assess their solutions. Different actors may benefit from the outcome of this dissertation: *i)* distribution system operators, who manage the distribution grids; *ii)* energy aggregators, who manage the resources and loads of both consumers and prosumers to generate savings and offer ancillary services to the grid; *iii)* public administrators and decision-makers, who are in charge of drawing policies to plan the energy transition; *iv)* energy managers, who manage energy flows in a building/small district; *v)* energy communities, which are a new emerging actor of the energy sector consisting of single citizens, companies, and public administration, and aim to adopt renewable energy sources and storage systems to produce and consume energy locally and share it in a peer-to-peer market.

Contents

List of Figures

List of Tables

List of Acronyms

AC	Alternating Current
AIO	Analog Input/Output
API	Application Programming Interface
BAU	Business As Usual
BEMS	Building Energy Management System
BIM	Building Information Modelling
CPU	Central Processing Unit
CHP	Combined Heat and Power
CIM	Computational Independent Model
COE	Co-simulation Orchestrator Engine
CPES	Cyber-Physical Energy System
CPS	Cyber-Physical System
CSC	Control System Core
CT	Continuous Time
DB	Database
DC	Direct Current
DE	Discrete Event
DER	Distributed Energy Resource
DIO	Digital Input/Output
DMG	Distributed Multi-Generation
DMS	Distribution Management System
DR	Demand Response
DRTS	Digital Real-Time Simulator
DSL	Domain Specific Languages
DSM	Demand Side Management

DSO	Distribution System Operator
DSP	Digital Signal Processor
DUT	Device Under Test
E2E	End-to-End
EHP	Electric Heat Pump
EMS	Energy Management System
EMT	Electro-Magnetic Transients
EMTP	Electro-Magnetic Transients Program
EU	European Union
EV	Electric Vehicle
FDIR	Fault Detection Isolation and Restoration
FMI	Functional Mock-up Interface
FMI-CS	Functional Mock-up Interface for Co-simulation
FMI-ME	Functional Mock-up Interface for Model Exchange
FMU	Functional Mock-up Unit
FPGA	Field Programmable Gate Array
GHG	Greenhouse Gas
GHI	Global Horizontal Irradiance
GIS	Geographic Information System
GPPL	General-Purpose Programming Language
GPS	Global Positioning System
GUI	Graphical User Interface
HIL	Hardware-In-the-Loop
HLA	High Level Architecture
I/O	Input/Output
IA	Interface Algorithm
ICT	Information and Communication Technology
IDE	Integrated Development Environment
IDMT	Inverse Definite Minimum Time over-current relay
IEC	International Electrotechnical Commission
IED	Intelligent Electronic Device
IEEE	Institute of Electrical and Electronics Engineers
INCOSE	International Council on Systems Engineering
IoT	Internet of Things

IP	Internet Protocol
IPR	Intellectual Property Right
ISO	International Standardization Organization
ITM	Ideal Transformer Method
JSON	JavaScript Object Notation
LAN	Local Area Network
LV	Low Voltage
MAN	Metropolitan Area Network
MAS	Multi-Agent System
MBSE	Model Based System Engineering
MDA	Model Driven Architecture
MES	Multi-Energy System
MQTT	Message Queuing Telemetry Transport
MV	Medium Voltage
NIST	National Institute of Standards and Technology
NOP	Normally OPen
OMG	Object Management Group
GAMES	General-purpose Architecture for MES
OMT	Object Management Template
OPF	Optimal Power Flow
P2P	Peer-to-Peer
PDM	Platform Definition Model
PHIL	Power Hardware-In-the-Loop
PID	Proportional-Integral-Derivative
PIM	Platform Independent Model
POD	Point of Delivery
PSCC	Power System Component Core
PSI	Platform Specific Implementation
PSM	Platform Specific Model
PSUT	Power System Under Test
PTP	Precision Time Protocol
PV	Photovoltaic
RCP	Rapid Control Prototyping
RES	Renewable Energy Source

REST	REpresentational State Transfer
RISC	Reduced Instruction Set Computer
RMS	Root Mean Square
ROS	Rest Of the System
RTI	Run-Time Infrastructure
SCADA	Supervisory Control and Data Acquisition
SFP	Small-form Factor Pluggable
SGAM	Smart Grid Architecture Model
SOC	State Of Charge
SoS	System-of-System
SysML	Systems Modeling Language
TC	Transparent Clock
TCP	Transmission Control Protocol
TRL	Technological Readiness Level
TSO	Transmission System Operator
UDP	User Datagram Protocol
UML	Unified Modeling Language
VCS	Voltage Control System
VDI	Voltage Deviation Index
VPP	Virtual Power Plant
VRS	Voltage Regulation Service
XML	eXtensible Markup Language

Chapter 1

Introduction

Smart Cities and their energy systems mainly contribute to climate change due to high levels of energy consumption and Greenhouse Gas (GHG) emission [1]. According to the United Nations Habitat, cities consume about 78% of global energy demand and generate more than 60% of GHG emissions primarily through the consumption of fossil fuels for energy supply and transportation [2]. To reach the ambitious goals of the Glasgow agreement [3], a drastic reduction in carbon emission is needed. To achieve such a reduction, a transition from classic fossil fuels to Renewable Energy Source (RES) as well as the adoption of integrated energy system components, such as micro co-generators, is required.

Nowadays, close cooperation between different energy vectors is timidly taking place and increasing in energy systems management through Distributed Multi-Generation (DMG). For instance, Combined Heat and Power (CHP) systems, in which electricity, heat exchanging fluids, and gas networks interact with each other, are gaining great interest from the research community as a solution to optimize the involved energy vectors' consumption guaranteeing secure and affordable energy systems and reduce the overall amount of GHG emission and local pollution to meet the challenging European Union (EU)'s 2030 Climate Target Plans. Other DMG examples that are characterizing this transformation of our energy systems are Electric Heat Pumps (EHP), air conditioning devices, tri-generation of electricity, heat, and cooling, and so on. In the same way, different energy sectors are widely taking advantage of the interaction with energy vectors to ensure their operation. Like the interactions between electricity, the fuel chain, and the transport sector ensured by

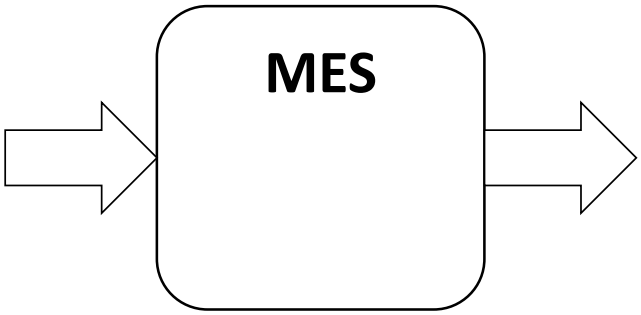


Fig. 1.1 System Engineering description of a Multi-Energy System (MES)

Electric Vehicles (EV), bio-fuels and hydrogen-based transport that is reducing the environmental footprint of road transport. So, the vertical approach of facing energy system innovation, design, and development must evolve in an integrated approach. This transition will change completely the structure of the energy systems from standalone hierarchical centralised energy systems to cooperative and distributed energy systems, the so-called Multi-Energy System (MES) vision highlighted by the European Green Deal's action plan [4] and Clean Energy for all Europeans package [5].

MES are complex systems in which different energy vectors (e.g. electricity, heat exchanging fluids, natural gas, and hydrogen), energy sectors (e.g. fuels, transport), and their respective generation, transformation, transmission, and distribution systems interact with each other to optimise the overall resulting System-of-System (SoS) with a specific objective, also considering each of the interconnected subsystem operational management [6]. While MES components interact in such a multi-faceted way that they are very difficult to be analysed comprehensively, such systems and the ones with DMG can be the major option to decarbonize the energy sector. In fact, MES means the coordinated planning and operation of the energy system across multiple energy carriers, energy markets, end-use sectors, and above-mentioned technologies at different functional layers to maximise efficiency, minimise waste, power a climate-neutral circular economy, and achieve a low carbon future [4, 7]. Moreover, this vision increases technical, economical, and environmental performance at both the operational and the planning stage with respect to "classical" energy systems whose sectors are treated independently.

The systemic description of a MES could be analyzed as a black-box with external interaction determined by an input and output equivalent model following a

system engineering approach (see Figure 1.1). Starting from a literature analysis, the main aspects to analyse in-depth these systems have been identified:

- the **multi-fuel** perspective represented in Figure 1.2a. It analyses the different inputs that a MES requires, including "classical" natural gas to biomasses and RES for both electrical and thermal energy, that will be integrated together to optimize the supply of the demand for the service offered by the MES;
- the **multi-service** perspective represented in Figure 1.2b. It identifies the output of MES operations, in terms of service offered by the MES itself (e.g. optimization of energy pattern), with particular attention to the supply level. It is worth noting that one of the most relevant use cases is the combined production of multiple energy carriers;
- the **multi-scale** perspective represented in Figure 1.2c. It scales the analysis from small environments up to large-scale scenarios (e.g. house, district, city, region, state). MES could be intended in different ways of aggregations, such as components involved in the MES or even just conceptually;
- the **multi-time** perspective represented in Figure 1.2e. It harmonizes and synchronizes different operational timings. Each energy vector is characterized by different transitional states that happen with their own temporal resolution. Once an operational analysis of a MES is taken into account, this perspective requires particular attention in managing a shared temporal resolution for all energy vectors involved;
- the **energy network** perspective represented in Figure 1.2d. It takes into account different vector's distribution and transmission systems (e.g. electrical lines, gas, and district heating pipes). The principal objective is to determine innovative multi-energy technologies that could optimize and reduce the systems cost of the networks involved;
- the **digitalization** perspective to monitor and manage the overall MES components through Information and Communication Technologies (ICT). This perspective determines a transition from an old analogic semi-automated to digital fully-automated management of energy vector's operations, taking into account each single energy system digitalization composing a MES;

- the **economic and business** perspective to study the impact of new solutions and services on the marketplace, taking into account different stakeholders, national policies, and energy vector's regulation.

Thus, MES complexity is difficult to be understood without exploiting models merging cyber-physical, economic, and social perspectives. Moreover, such analysis must also consider constraints and feedback from regulators, economic drivers, social and environmental behaviours [8]. Therefore, the major challenges associated with the design, simulation, and management of MES use cases lie in an effective holistic analysis based on collaboration among different domain experts and integration of their specialised modelling and simulation tools. Indeed, experts from different domains may misunderstand each other due to different terminology and backgrounds and, usually, cannot communicate and exchange information with each other due to data format incompatibility and license restrictions of tools.

To foster this energy transition, ICT tools are required to simulate and test such a heterogeneous SoS before deploying innovative solutions in real-world scenarios that fulfil these complex interactions. The energy research community concentrated its effort on developing standalone simulation tools to cope with the needs of analysing this disruptive transition. Different simulation tools have been proposed to analyse specific aspects of these complex systems. For instance, Digital Real-Time Simulator (DRTS) have been proposed to analyse Electro-Magnetic Transients (EMT) of innovative equipment within electricity power transmission and distribution lines. However, these solutions focus only on some of the above-mentioned perspectives. Other solutions are more complete and allow analysis of all the aspects required by MES. Often, these solutions follow a vertical design in different fields of technology (e.g. electrical and thermal engineering, distribution and transmission grid management, and energy market analysis). Hence, MES scenario developers must dedicate a steep learning curve to master these solutions.

So, standalone simulation is failing in describing the multi-disciplinarity, multi-domain, and multi-model vision of these complex systems with the required spatio-temporal scalability to face the energy system integration with composite and intelligent control strategies [4]. Further efforts are required to extend standalone capabilities with a distributed simulative approach, taking into account heterogeneous aspects of the aforementioned SoS. In the literature, co-simulation techniques have been proposed to interconnect different domain-specific software, General-

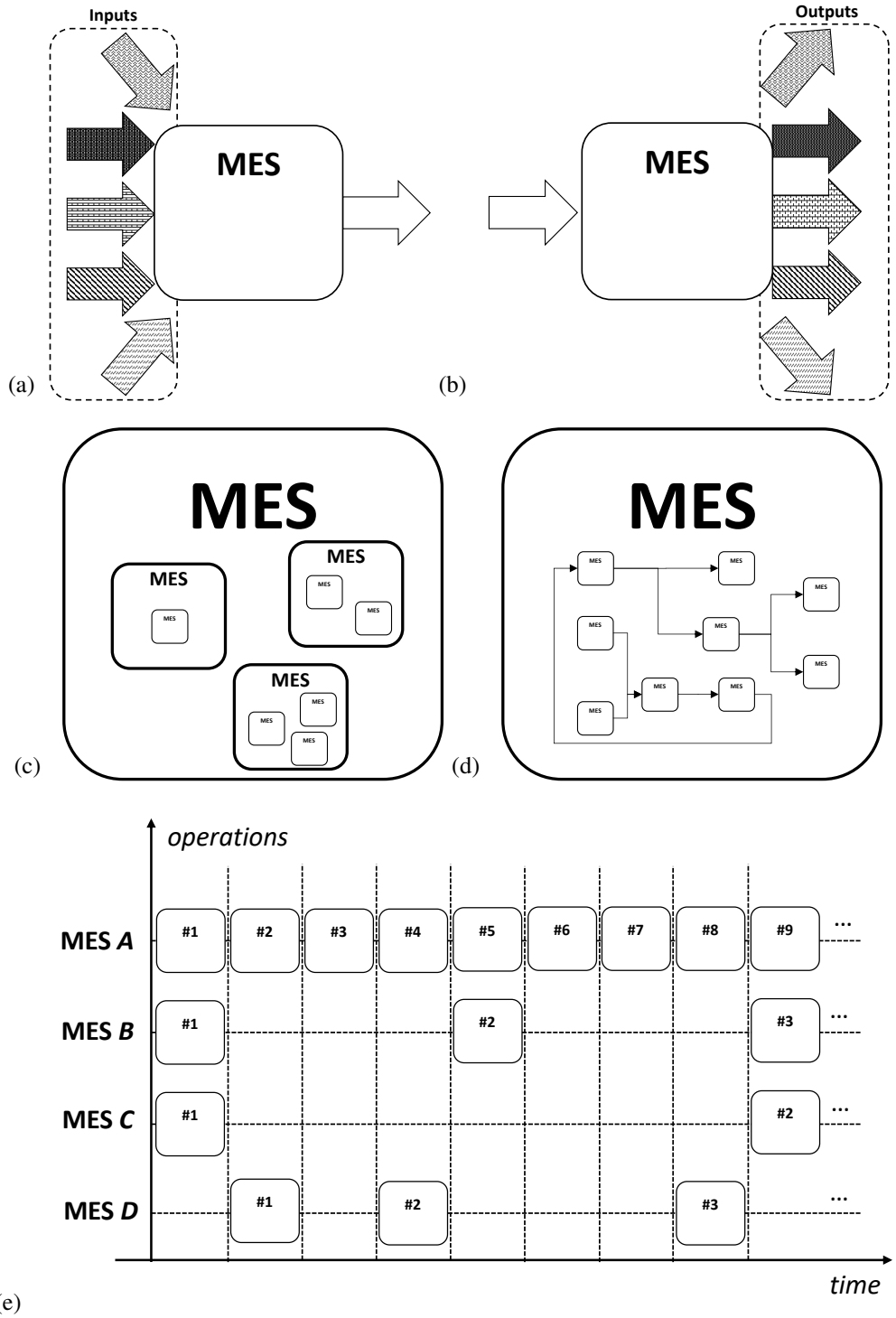


Fig. 1.2 Some descriptive aspects of a MES: (a) the multi-fuel, (b) the multi-service, (c) the multi-scale, (d) the network, and (e) the multi-time perspectives.

Purpose Programming Language (GPPL), and hardware simulators, exploiting each of the respective tool peculiarities in a shared simulation environment capable of analysing concurrent aspects of a MES.

1.1 Open Challenges

The complex nature of Multi-Energy System (MES) leads to the essential need of new framework to plan, develop and test these large-scale SoS. A MES designer demands proper engineering and validation approaches, methods, concepts, and corresponding tools. ICT could acknowledge these requirements and provides a structured basis for the design, development, and validation of new solutions and technologies. This section tries to list different challenges and objectives still open in this field. The challenges addressed by this dissertation are grouped into four main pillars: *i)* Knowledge Integration, *ii)* Design Framework, *iii)* Automated Composability, and *iv)* Co-simulation Technologies.

1.1.1 Knowledge Integration

A Multi-Energy System (MES) is categorised as a *complex system* from a system engineering perspective [9]. Complex systems are composed of a great number of different elements employing dynamic connections. In Figure 1.3, two principal viewpoints can be assessed: *i)* system dynamics and changeability [10] in term of parameters, effects and mechanisms, and *ii)* variety, multiplicity, and size to represent the scalability of the system. Specifically, the elements that compose a MES should be grouped by energy vector membership, represented by the third axis. Following this interpretation, MES can be stated as heavily interconnected SoS with composite system dynamic interaction belonging to a single energy vector.

Due to the complexity of the analysis, we can define two different perspectives by designing a MES: i) *Vertical Knowledge Integration*, and ii) *Horizontal Knowledge Integration*. *Vertical Knowledge Integration* represents the different scientific contexts specific to an individual energy vector. For instance, each electricity vector could be split into generation, transmission, distribution, distributed energy resources, and consumer/prosumer premises. Each of these contexts requires specific expertise to be comprehensively analysed.

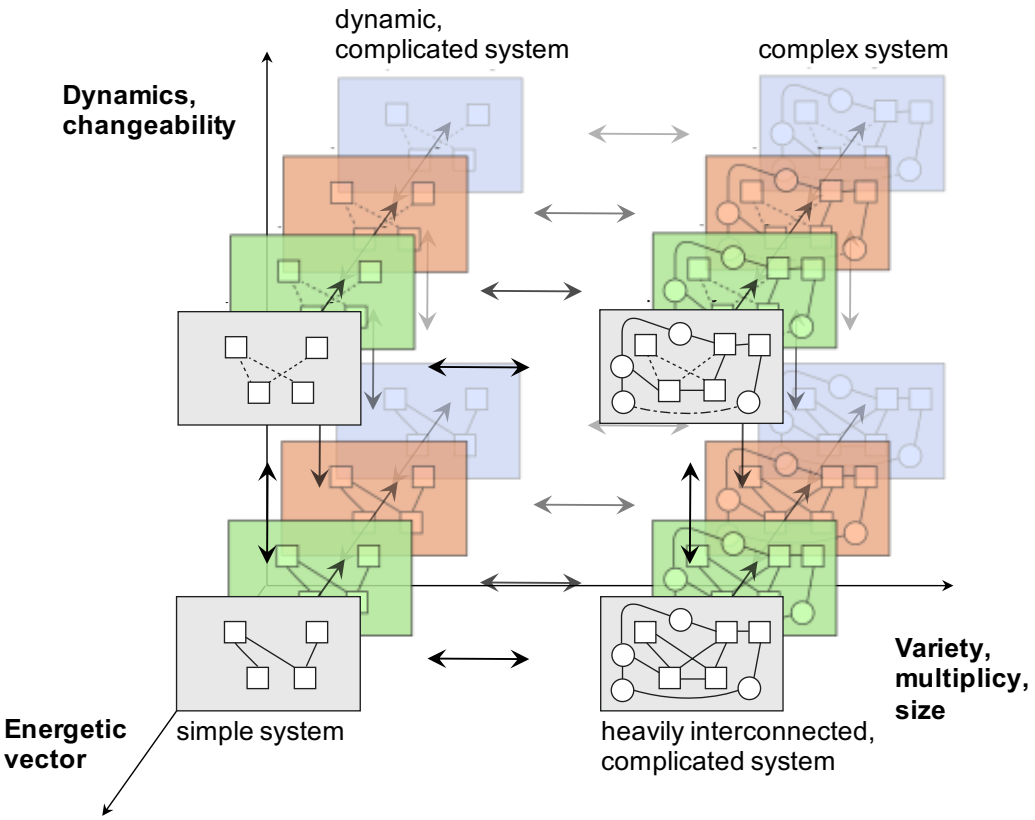


Fig. 1.3 Vertical and Horizontal Knowledge Integration of a generic MES

Instead, *Horizontal Knowledge Integration* is aiming at a broader MES analysis by avoiding or reducing complexity for a specific energy vector, towards the concept of a Simple System. Nevertheless, such knowledge integration expresses the complexity of the interaction between the energy vectors. From the point of view of the energetical aspect, these perspectives generate two mutually exclusive objectives: i) reducing the MES details design to address a more scalable system, or ii) increasing the MES details, reducing the dimension of the overall system.

Integrating monitoring, management, and control aspects in a MES analysis dramatically increase the order of complexity. Data signalling through ICT engineering is a requirement to enable such functionalities. It requires a wider vision to define the exchange of information between all the elements composing a MES. Moreover, data signalling must be enabled by a communication network (i.e. Internet) involving all the protocol suite stacks elements that allow information exchange in the analysis. Dealing with ICT rises different needs: *i)* a common information model description evenly distributed among MES entities, *ii)* a dedicated application protocol to monitor, manage and control such MES entities, *iii)* network management of the data signalling, and iv) cyber-security and data privacy management to increase the resilience of MES against cyber attacks and threats.

Lastly, business, economics, and regulation aspects play a role of great importance to join the energy market and offer new commercial services (e.g. ancillary services, Demand Response (DR) and Demand Side Management (DSM)) exploiting innovative business models and ICT. To face decision-making assessment and economic feasibility of such services, a MES designer must choose among different alternatives: *i)* outline a business model to commercialise a service, *ii)* select different MES-related technology to provide such service, and *iii)* enable the ICT component to fulfil the service in an optimal and secure fashion. All three requirements impact the economic feasibility of the service.

Figure 1.4 shows the representation of concurrent MES aspects as a complex Cyber-Physical Energy System (CPES) with overlapping structures. Each of the possible stakeholders of a MES design introduce its own aspect of the SoS infrastructure. Thus, the first challenge highlights the need for unified tools to support the design and analysis of MES architectures. These tools must allow effort parallelisation of MES architecture design, permitting each stakeholder to design its own aspect exploiting the others' high-level SoS structures.

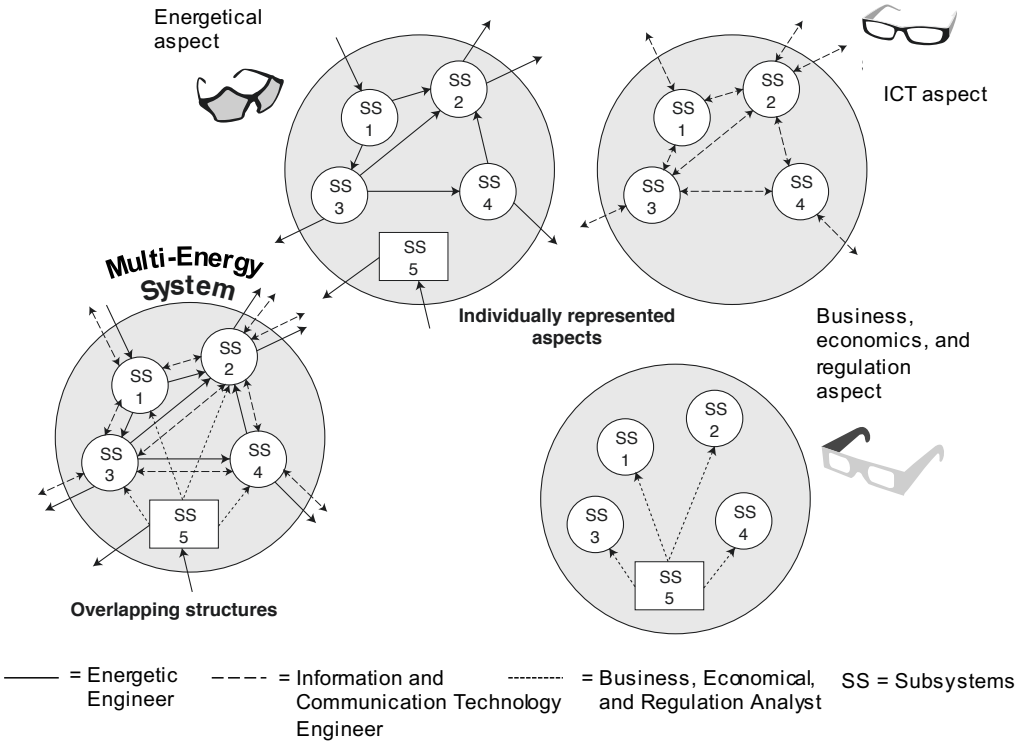


Fig. 1.4 Aspects of a generic MES design

1.1.2 Design Framework

The major concern regarding MES design lies in the development and testing of innovative technologies to ensure operational functionality, stability and safety. Systematic analysis on a formal basis is impossible when dealing with the high-level complexity of MES design. Also, hardware testing in laboratory or fields trial are expensive and inflexible tools to assess a MES scenario. On the other hand, software simulation could be a viable solution which is scalable and cost-effective.

Operational analysis of a MES requires collaboration of expert in each different aspect presented in Figure 1.4. To achieve these objectives, each analysed field of technology introduces knowledge and terminology from different MES domains. To deal with such heterogeneity, a *Design Framework* is needed to allow the interconnection of executable interdisciplinary models without involving their detailed complexity. The manual configuration of such multi-faceted scenarios in nowadays design framework can be error-prone. For instance, holonic component with complex behaviour and many interfaces with different domains (e.g. Communication Network) requires handling a complex simulation engine. Moreover, scaling the analysis from small environments up to large-scale scenarios (e.g. house, district, city, region, state) could increase the complexity of different components' interconnection.

Thus, the design framework must be focused on the high-level description of the MES scenarios avoiding details of model coupling and interfacing. It should rely on a simple and viable modelling language shared among MES scientific community. This will ensure a reduction in coding effort to achieve a valid analysis result. Models, software, simulators and hardware must be accessible among experts in the fields through a disposable shared library to avoid their design from the scratch and enhance their reliability. Moreover, a discovery service must be integrated into the design framework to foster the reuse of items already compliant with the co-simulation infrastructure.

Finally, the interconnection of models into scenarios must be automated allowing to respect a set of interconnection rules defined for each library item. Avoiding manual interconnection reduces the possible erroneous link between models, leading to a faultless and reliable scenario generation.

These requirements will reduce the gigantic modelling effort needed to cope with such a complex scenario.

1.1.3 Automated Composability

This challenge reflects the complex effort needed by ICT engineering to deal with the physical interconnection of MES software simulators and hardware (e.g. micro CHP, Thermal Storages, and EHP).

Normally, software simulators employ a Domain Specific Languages (DSL) to parametrise simulated entities in a grey-box modelling approach. DSL allows usage of Application Programming Interface (API) to describe different aspects of a system (i.e. inputs, attributes, control variables and outputs).

Often, these software simulators follow a vertical design in different fields of technology (e.g. electrical and thermal engineering, distribution and transmission grid management and energy market analysis). Hence, MES scenario developers must dedicate a steep learning curve to master these solutions.

Furthermore, simulations are commonly achieved in a standalone environment without taking into account information from distributed or third-parties components. Nevertheless, this process should be implemented by considering the different interactions between the simulation environment and external sources. For instance, a simulated model could retrieve input data from interconnected hardware to generate the desired system stimulus.

Hardware set-ups are of great value in hardware assessment for a MES scenario but intrinsically originate criticalities in the test-bed configuration. Depending on the hardware, strict real-time constraints are introduced increasing the interconnection complexity. Moreover, the ICT integration of distributed software simulators and hardware entities requires expertise in distributed network interconnection.

An *Automated Composability* process could face these issues. Relying on the above-mentioned design framework, a dedicated compiler could generate each specific DSL code related to a particular component of the MES scenario, either hardware or software, and could offer a self-regulating distributed network interconnection of the relationship among the different participants.

1.1.4 Co-simulation Technologies

Co-simulation is a challenging task when dealing with heterogeneous ICT simulation tools. It must preserve high efficiency and accuracy in each single subsystem simulation. Furthermore, the complex dynamic system of systems simulation obtained by coupling different simulators may not cause instability and inaccuracies. The main challenges in this regard are Time Synchronisation and Regulation, and Communication.

Time Synchronisation is mandatory when the distributed co-simulation infrastructure interacts in a time-dependent manner. It refers to the algorithm used to ensure temporally correct ordering among events generated by various simulators. So, time synchronisation must share among the simulation entities a common time reference to respect the aforementioned time step evolution constraint. *Time Regulation* instead refers to the need of instituting a policy to regulate how individual simulators evolve over time. For instance, a particular simulator could be a leader of the distributed environment (i.e. time-regulating), and some others could be followers (i.e. time-constrained). Depending on the application, a policy must be created using a correct time regulation scheme for the simulators involved, which can have a major impact on the performance and correctness of the distributed co-simulation environment. By following this time regulation policy, the co-simulation framework must ensure the correct ordering of simulators' internal state evolution respecting each of the simulator time step evolution.

On the other hand, *Communication* refers to data exchange among different interconnected simulators, normally carried out by telecommunication protocols. Choosing the right protocol is fundamental to design an accurate and reliable co-simulation infrastructure, capable of ensuring the stability of the numerical solution. The main issues are normally generated by latency (or lag). In a co-simulation context, latency is the time delay between the sending procedure of data retrieved from a simulator engine and the receiving procedure that provides the received data to the solver of other distributed simulator environments to fulfil their numerical calculation. Latency is the main cause of instability and inaccuracies for a distributed co-simulation infrastructure and must be mitigated by applying specific techniques. So, the communication strategy must ensure the data exchange management via communication protocols, such as Transmission Control Protocol (TCP) and User Datagram Protocol (UDP), and must consider the complex ordering of the internal

state evolution of each interconnected simulator with the aim of reducing the latency experienced by the exchanged messages.

Finally, for software and GPPL simulators, the entire co-simulation process may require a master algorithm that ensures the correct evolution of the simulation environment, so-called Co-simulation Orchestrator Engine (COE). For hardware simulators (i.e. DRTS), the simulation process instead cannot be controlled by an external entity since their time regulation must respect the real-time constraint to permit Hardware-In-the-Loop (HIL) and Power Hardware-In-the-Loop (PHIL) interconnection for testing purposes, so-called hard real-time environments. So, the interconnections among software and GPPL and hardware simulators raise a complex issue, which has not yet been tackled by the research community.

1.2 Goals and Achievements

A first literature analysis revealed that applying co-simulation solutions could reduce the complex effort needed for modelling the heterogeneous set of components and equipment required for a MES scenario simulation with respect to standalone simulation software and/or GPPL simulation libraries. In fact, co-simulation techniques have been taken into account in this dissertation to propose a solution to relieve MES designer from the complex effort needed to analyse both operational and planning aspects of a MES.

Co-simulation is a flexible approach to integrate different domain-specific simulators together in a shared and distributed simulation environment. Following this paradigm, a complex scenario is decomposed in a SoS topology in which each node (i.e. subsystem) is simulated by a different simulator engine (or solver). This decomposition allows choosing among a set of domain-specific simulation tools to find the best solution that enhances numerical calculation and boosts the computational time of a single subsystem. For instance, DRTS is a plus to fulfil a Smart Grid simulation in a distributed co-simulation infrastructure.

As a result, this dissertation proposes different modelling and co-simulation techniques designed, developed, and tested over real-world scenarios to demonstrate the feasibility of applying co-simulation in such a complex system analysis. The

application of these techniques will foster the energy transition towards integrated energy systems, so-called MES vision.

The main achievement is proposing a set of comprehensive tools that a MES designer could choose to simulate a MES scenario, depending on the perspectives and aspects that must be taken into account for the scenario analysis. These tools will avoid the complexity of employing standalone simulation software or state-of-the-art co-simulation solution to represent such a heterogeneous set of components and equipment required by MES analysis.

Firstly. this dissertation proposes a comprehensive architectural description for modelling and co-simulate MES, so-called General-purpose Architecture for MES (GAMES), following *"from the black-box to the white-box"* approach. The presented architectural model integrates and extends the Smart Grid Architecture Model (SGAM) to deal with the definition of MES use cases. Thus, the proposed solution enables a modular methodology where different aspects of a MES can be analysed altogether with high-level details of the use case description, exploiting Unified Modeling Language (UML) to characterise each component involved as a *black-box*. Furthermore, GAMES integrates Systems Modeling Language (SysML) to enable a systemic description of the MES component and their interconnections, following a *grey-box* approach. This will allow translating the systemic description of components involved in MES use cases into DSL code of each software simulator involved in the co-simulation scenario and hardware configuration files for the interconnection of real-world devices, allowing a *white-box* analysis of software and hardware involved in the MES use case. GAMES is considered a solution of the former open challenges *Knowledge Integration*, *Design Framework*, and *Automated Composability*.

Then, this dissertation proposes three different co-simulation infrastructures that allow a MES designer to develop, test, and deploy a MES simulation interconnecting different GPPL, simulation software, and hardware simulators in a unique simulative tool. Depending on the set of simulator entities interconnected, a MES designer could choose among the three configurations of the proposed architecture, namely:

- The Pure Software Co-simulation Infrastructure capable of interconnecting GPPL and simulation software related to specific operational and planning assessment of a MES use case with slow temporal evolution;

- The Digital Real-Time Co-simulation Infrastructure capable of interconnecting different hardware simulators, i.e. DRTS, when a MES designer needs to analyse in real-time demanding MES operational use case with fast temporal evolution (e.g. power grids), allowing HIL and PHIL testing;
- The Hybrid Co-simulation Infrastructure capable of analysing a MES use case with concurrent slow and fast temporal evolution of its operational and planning aspects.

Furthermore, this dissertation proposes an event-based infrastructure, so-called the Distributed Event-Driven Platform, that allows near real-time testing of MES use case offering the possibility to switch easily from a testing environment to a real-world application of a particular MES functionality. These two infrastructures are considered as a solution to the former open challenges *Automated Composability*, and *Co-simulation Techniques*.

Finally, a scalability assessment of the existing co-simulation framework is presented to determine whether or not to apply one of the co-simulation solutions present in the state of the art, depending on the MES use case under analysis and its requirement.

1.3 Structure of the Dissertation

The structure of this dissertation is as follows. Chapter 2 provides a theoretical background on the system engineering tools used to analyse and decompose a MES scenario into a System-of-System (SoS) structure to then apply Model Based System Engineering (MBSE) techniques to standardise the proposed scenario description among the energy research community by applying a model-based approach. Then, a literature review of co-simulation standards, techniques, and solutions is presented in Chapter 3, highlighting the proposed infrastructure novelties. Chapter 4 introduces the innovative General-purpose Architecture for MES (GAMES) architecture and describe its main structure that facilitate a MES scenario description. Chapter 5 presents the co-simulation infrastructures proposed in this dissertation in its three variants, namely *i)* the Pure Software, *ii)* the Digital Real-Time, and *iii)* the Hybrid Co-simulation Infrastructures. Each of their layered infrastructures has been presented and analysed and tested over a specific MES scenario. Chapter 6 introduces

instead the distributed event-driven platform that allows testing innovative MES services in a co-simulation environment to then scale to a real-world application. The platform novelties have been tested over a real-world MES scenario. Then, a scalability comparison has been presented in Chapter 7 to determine which of the literature co-simulation orchestrator perform better in rising the simulated entities composing a MES. Finally, Chapter 8 reports concluding remarks.

Chapter 2

Enabling Technologies

2.1 Systems Engineering

Many phenomena in daily life are delineated as systems, such as MES. Following the fundamentals of Systems Engineering, systems are defined as a composition of *elements* that represent their fundamental building blocks. Each element is characterized by *properties* and *functions*. Properties are understood as parameters of an element. For instance, a Photovoltaic (PV) panel represent an element of a photovoltaic system and one of its property is its peak power. Functions, on the other hand, represent the purpose of an element. For a PV panel, one of its functions is the production of electricity from solar radiation.

The elements of a system could also be viewed as a system itself. This process, called system decomposition, yields several subsystems that normally have the same properties and functions. Thus, PV panels that are constituent elements of a photovoltaic system could be seen as subsystems. Each of the generated subsystems has the same properties and parameters, which fits the above definition of a system. In fact, a PV panel is composed of solar cells, its constituent elements, which present particular properties and functions. It is worth noting that this process could be iterated again on PV panels by applying system decomposition to generate different subsystems (i.e. their solar cells).

Elements of the system are associated with each other through *relations*. The elements and their relations define the *structure* of a system. Following the photovoltaic system example, different PV panels (i.e. its elements) can be combined

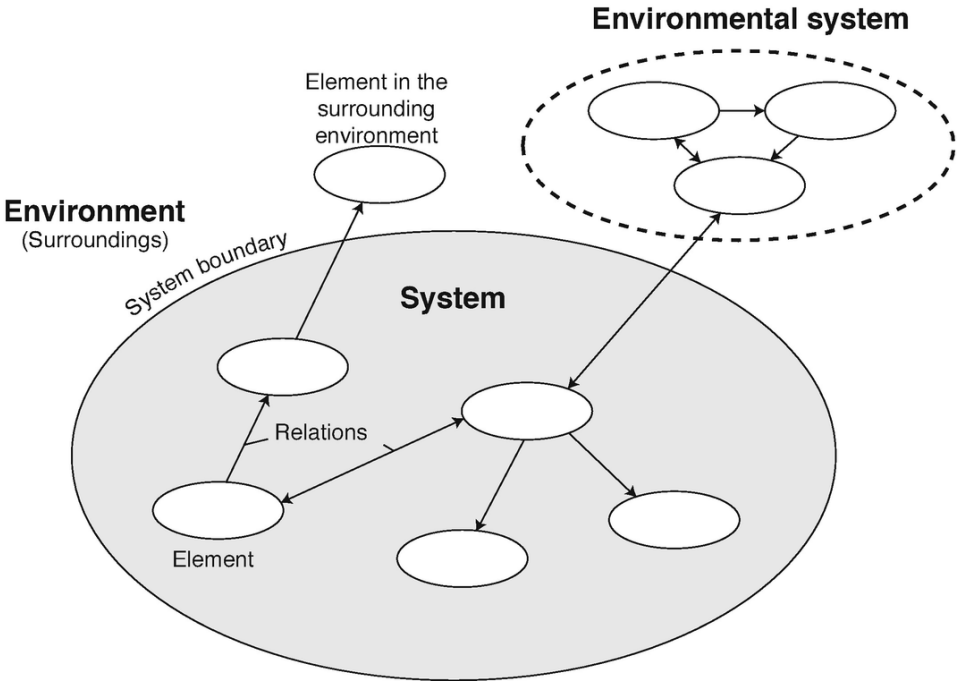


Fig. 2.1 Fundamentals basic terms of a system

in parallel interconnections or in series with copper cables (i.e. their relations) in order to increase the electricity production of the overall system. The schema of the interconnection of the different elements defines the system structure.

Finally, a system composed of its elements and relations allows the definition of its *boundary* with the outside world, the so-called *environment*. Other elements or systems that affect or are affected by the system are located outside the boundary of the system. In the literature, these systems are referred as the *environmental system*. Figure 2.1 provides a graphical representation of all of the above definitions that allows a clear understanding of the basic terms of a system.

When dealing with systems that are composed of heterogeneous elements, it is fundamental to define further notions, such as System-of-System (SoS) which extends the concept of system decomposition. It describes a decomposition of a system in which the generated subsystems (i.e. its heterogeneous elements) can operate independently of each other. Thus, each of the subsystems that make up a SoS has its own purpose that is independent of the purposes of the other subsystems. Together, they will implement additional or greater benefits than any single system could provide. To conclude this introduction to the Systems Engineering notions, the

SoS or a single system is described by its *system hierarchy* that orders and relates all the entities that satisfy the benefits of the SoS or system.

2.1.1 Black box, grey box, and white box systems

It is of paramount importance to choose from a set of possible complexities of a system, depending on the advantage exploited by the completeness of detail that a system designer would like to reach. For example, a system could be described by its transfer function f applied to inputs I to recover outputs O without complicated models or complex internal dynamics, as described in Figure 2.2a. Systems in which the internal construction or phenomenon is meaningless or unknown are called *black box* systems [9]. This level of description is essential when trying to reduce the complexity of a system description.

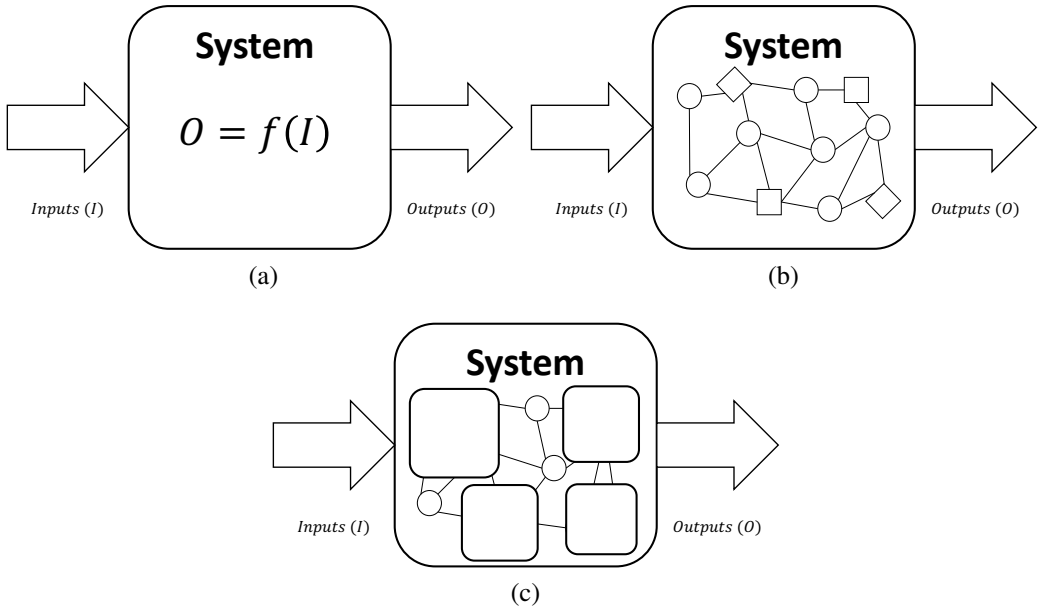


Fig. 2.2 Generic representation of different levels of description of a system: (a) black box, (b) white box, and (c) gray box systems

A *white box* system [9], on the opposite, analytically expresses the real connections between inputs and outputs that must be observed in detail or are of great significance for the purpose of the system definition (e.g. for simulative purposes), as depicted in Figure 2.2b. In these systems, all the smallest heterogeneous ele-

ments and relationships are detailed and described without the possibility of further breaking up the structure of the system.

A *grey box* system [9] instead is used when its description is roughly or partially structured, as described in Figure 2.2c. Roughly structured means that the level of detail is low. Partially structured alternatively indicates that the level of detail varies depending on the definition of the elements: some system parts could be structured in an elementary way or not at all, and others are structured with complete details.

These definitions can be applied in an arbitrary fashion by the system designer since we can define the transition from a black box to a white box system passing from a variable number of grey box systems in different, multi-faceted ways.

2.1.2 System aspects

All systems and SoS can be seen from different perspectives and viewpoints. Each perspective and viewpoints define a *system aspect*, that typically enhances a particular analysis of the system and the composition of its elements.

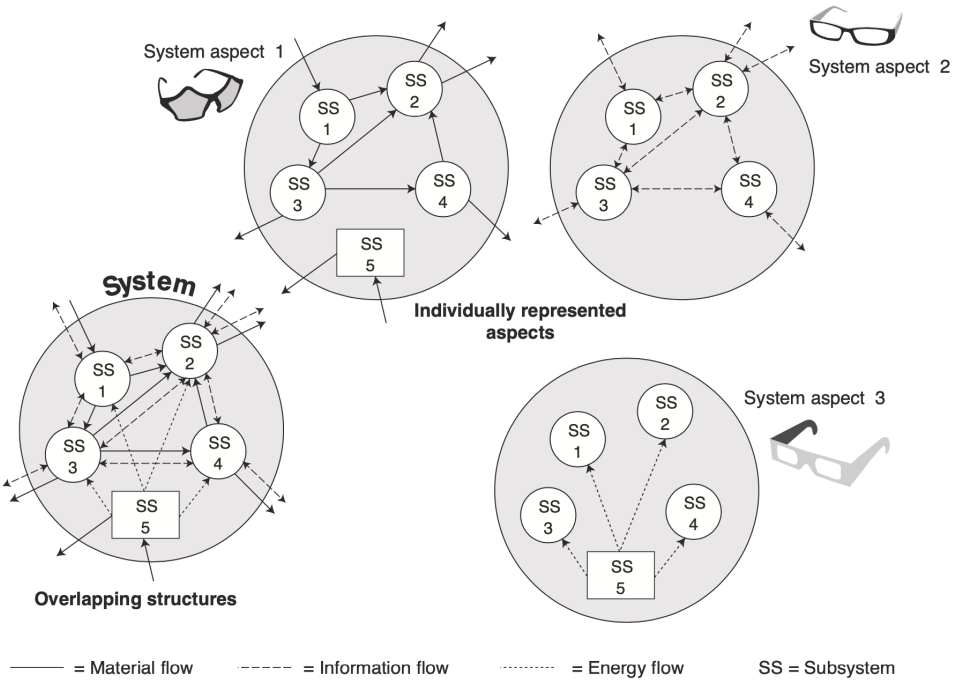


Fig. 2.3 System aspects representation

In Figure 2.3, three different aspects of a SoS are presented, each one depicting a different type of relation between subsystems (i.e. Material, Information, and Energy flows). The resulting system presents a typical overlapping structure that is capable to join all the previous aspects together. Multi-Energy Systems (MESs) are a clear example of these systems where different energy, ICT, financial, and business perspectives add their own vision of the relations among subsystems and of subsystems themselves, modelling and highlighting a particular behavior, quality, or property of the MES.

2.1.3 System types

When dealing with a particular aspect, a systems designer must choose between different levels of detail needed to analyze a system. For example, a system might be analyzed using a black box or gray box approach, resulting in a low number of elements and relationships. Conversely, the system could be described with high resolution by various types of elements with complex and dynamic relationships through a white box approach. In this sense, it is useful to define different types of systems according to their variety, multiplicity, and size, and on the other hand by their dynamics and changeability.

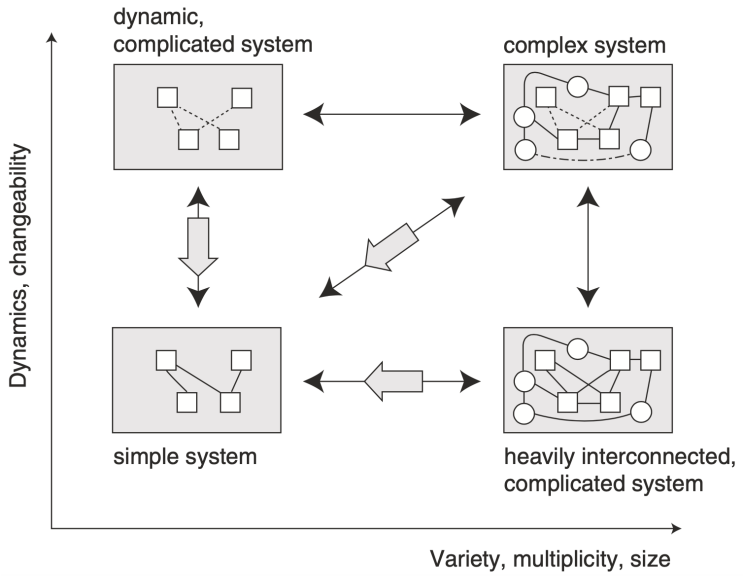


Fig. 2.4 System Types

These two axes are represented in Figure 2.4, where four definitions of system types have been introduced as follow:

- A *Simple System* consists of a low number of elements that exhibit permanent relationships with low intensity [9]. These systems can be explicited in their entirety - in special cases even through mathematical or analytical methods. So, they present a small variety, multiplicity, and size, and low dynamics and changeability.
- A *Massively Interconnected, Complicated System* has a high number of elements. These elements have a large variety and are interconnected by simple, static relationships [9]. Due to their size, it could be very difficult to explicitly describe these systems that normally require computer simulation to depict their behavior.
- A *Dynamic, Complicated System* presents elements with temporal and, often, non-linear behavior of relationship with other elements [9]. Although they present normally a really low number of elements, it results difficult to describe quantitatively or to predict the system behavior due to their dynamic character.
- A *Complex System* joins together Massively Interconnected and Dynamic, Complicated Systems. These systems present a great number of elements and composite relationships with high dynamic connections [9]. This type is the most difficult to be analysed. In some cases, representing and describing the complexity of these systems may be an impossible task and simplifications must necessarily be made.

As much as possible, a systems engineer should try to describe a system by reducing the complexity of the necessary aspects, and try to represent them as simple systems.

2.2 Model Based System Engineering

The former definition of MBSE from International Council on Systems Engineering (INCOSE) in [11] is "the formalized application of modeling to support system requirements, design, analysis, verification, and validation activities beginning in the

conceptual design phase and continuing throughout development and later life cycle phases. MBSE is part of a long-term trend toward model-centric approaches adopted by other engineering disciplines, including mechanical, electrical, and software. In particular, MBSE is expected to replace the document-centric approach that has been practiced by systems engineers in the past and to influence the future practice of systems engineering by being fully integrated into the definition of systems engineering processes."

In 1997, Object Management Group (OMG) presents its first attempt of standardization of model-based approach specification, so calledUML [12]. UML is a General-Purpose Programming Language (GPPL) for modelling software that provides a standard to visualize its design following a system engineering perspective. In 2005, UML become an International Standardization Organization (ISO) standard for software engineering and has been extended to a second revision in 2017 [13]. It offers two main diagram types, called Structural and Behavioral UML Diagrams. Although UML offers different simple software to deploy the overall model-based approach, the common practice of software engineers has been to choose among a set of these diagrams and draw them in an informal fashion. These informal diagrams are then shared with other colleagues to describe the structure of the software under development.

The main technical implementation of MBSE in the field of system analysis instead has been SysML [14] that is a GPPL for modelling systems that, similarly to UML, follow a system engineering perspective. It offers the possibility to support the specification, analysis, design, verification and validation from simple to complex system studies. In 2012, SysML become an ISO standard for system engineering [15]. SysML is an extension of some UML schema to support system engineering applications.

2.2.1 Model Driven Architecture

Model Driven Architecture (MDA) was presented in 2001 by the OMG as standard for applying model-based approach to software development.

The main goals of the standard are portability, interoperability, and reusability of software among different hardware and GPPL through an architectural vision. It embraces the complete lifecycle development of a software, from the analysis,

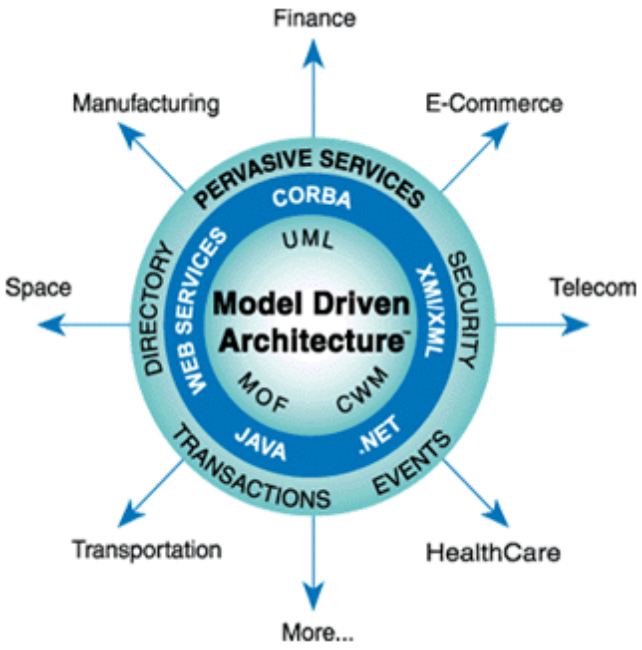


Fig. 2.5 MDA vision

design, and testing to the deployment and maintenance of software. In brief, it is an approach that applies existing OMG specification (e.g. UML) to optimize software development taking care of the above-mentioned concerns and avoiding technological definition, enabling the interoperability required for a commercial software solution. MDA general vision is depicted in Figure 2.5, which presents a diagram that illustrates the possible applications and the proposed solutions by the standard.

The two main concepts of MDA are models and transformations. When dealing with the first step of the MDA process, the software developer must create via UML diagrams *i)* a Computational Independent Model (CIM) that represent a model of the overall business rules that encompass the core business of specific software, and *ii)* a Platform Independent Model (PIM) that instead represent a model to maps the CIM entities into the relative services offered by the software in a platform independent way. The PIM can be transformed in different specific software platforms (e.g. .NET Framework) through Platform Specific Model (PSM) that contains the specification and entities of the PIM with added details to determine how these specifications and entities are mapped in a specific platform described by the Platform Definition Model (PDM). A PIM could have one or more PSM, depending on the number of platforms

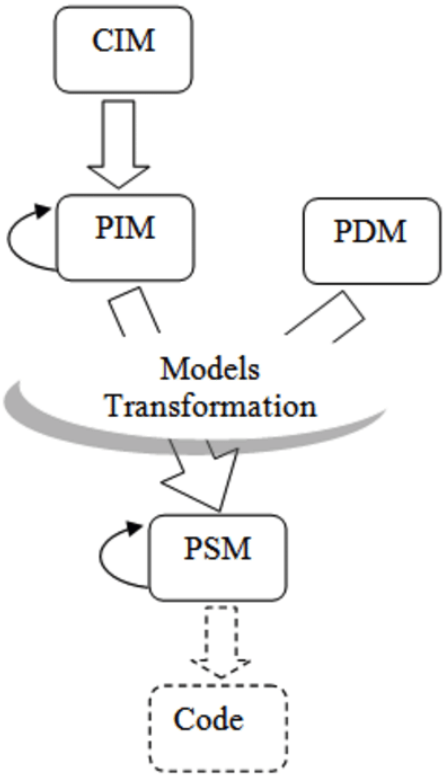


Fig. 2.6 MDA process

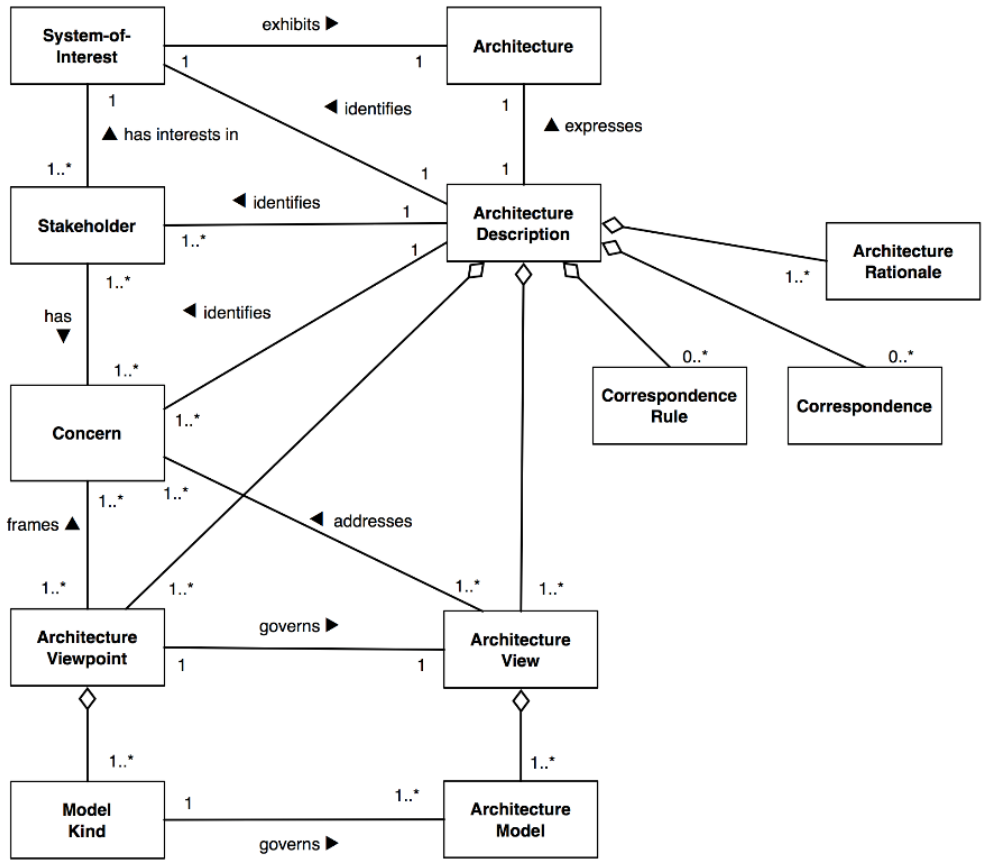


Fig. 2.7 Model-Based System Engineering (MBSE) conceptual model of an architecture description according to ISO 42010

in which a software developer wants to deploy its software solution. Finally, the PSM is compiled into code to be run by the specific platform defined in the PSM. It is worth noting the relations among models (i.e. CIM, PIM, PDM, and PSM) and their transformations till reaching the code in Figure 2.6.

2.2.2 ISO42010 and Architecture Design

Another huge effort of the scientific community to foster the application of MBSE in architectural description of system and software is the IEC/ISO/Institute of Electrical and Electronics Engineers (IEEE) 42010:2011 standard [16], ISO 42010 for short.

ISO 42010 defines architecture description for the creation and analysis of architectures of systems by providing a comprehensive architecture ontology. The

application of the standard ensures a coherent practice for system and software architecture descriptions and description languages, taking into account also the architecture framework development. It can also be used to assess compliance to the standard of an already developed system and software architecture to its provision.

In Figure 2.7, the standard introduces the conceptual model for architecture description that is commonly used to define an abstract of the architecture by means of concepts (e.g. stakeholder, concern, architecture view and viewpoint) and properties (e.g. relations among concepts). More details can be found in [16].

2.3 Smart Grid Architecture Model

SGAM [17] is one of the most promising model-based tools in the field of energy engineering, in particular for power grid evolution and innovation. It is a tool to design and validate Smart Grid use cases in an architectural viewpoint based on ISO 42010 standard [18]. As depicted in Figure 2.8, SGAM is a 3D structure with three main axis for the dimension of: i) *Domains* , ii) *Zones*, and iii) *Interoperability Layer*.

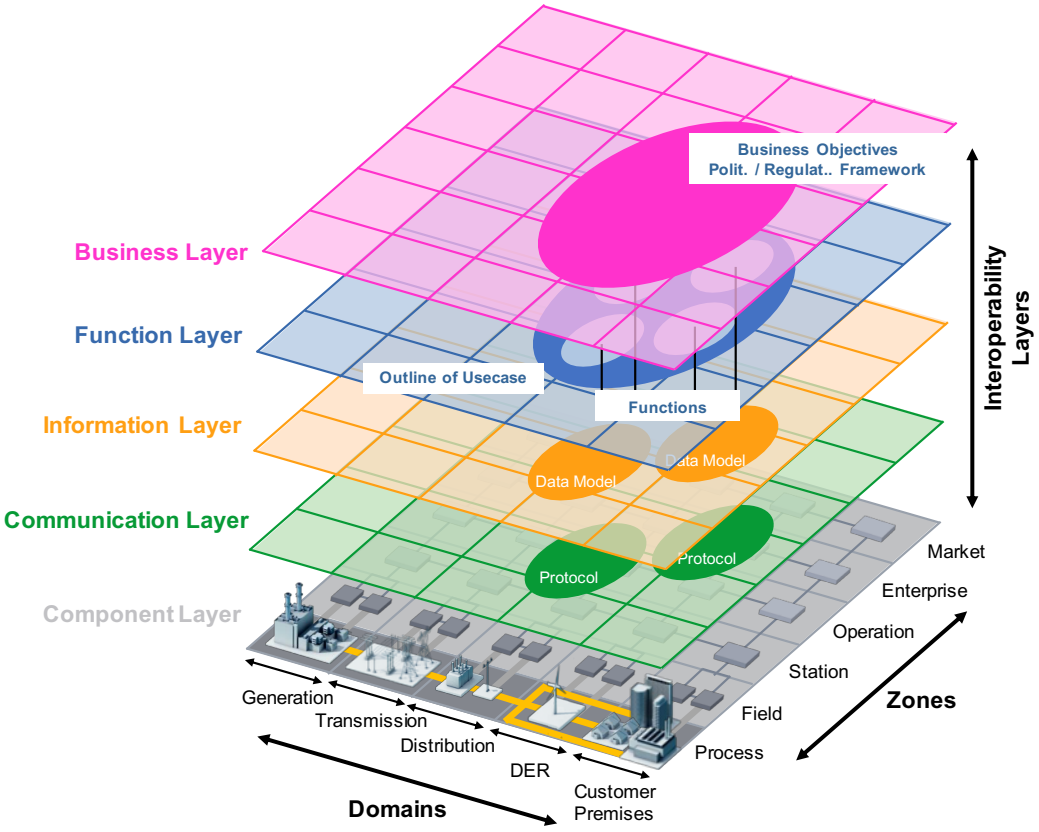


Fig. 2.8 Smart Grid Architecture Model (SGAM)

Domains represents the overall conversion chain of electricity following the *National Institute of Standards and Technology (NIST) Conceptual Model* [19] based upon IEC 62890 Value Stream Chain [20]. Domains are described as follows:

- ***Bulk Generation*** represents the production of electricity in bulk quantities typically connected to the transmission system, such as fossil, nuclear, hydropower plants, off-shore wind farms, large-scale solar plants (e.g. PV);

- ***Transmission*** represents the infrastructure and organization allowing transport of electricity over long distances;
- ***Distribution*** represents the infrastructure and organization permitting to distribute electricity to customers;
- ***Distributed Energy Resource (DER)*** represents the entities applying small scale generation technologies, typically between 3kW and 10MW, directly connected to distribution infrastructure (e.g. PV plants);
- ***Customer Premises*** represent the end users of the conversion chain like industrial, commercial, and residential customers, either consuming or producing electricity (e.g. rooftop PV installation and micro-turbine).

Zones represents the hierarchical levels of ICT control system implemented along the Domains to control the conversion chain of electricity based upon IEC 62264 [21] and IEC 61512 [22] Hierarchical Level for Automation. Zones are described as follows:

- ***Market*** represents the market operation enabled by the overall electric conversion chain (e.g. mass market, retail market, and related energy trade actions);
- ***Enterprise*** encompasses commercial and organizational processes, services, and infrastructure for enterprises, like utilities, service providers, or energy traders (e.g. asset management, logistics, billing, and customer relationship management);
- ***Operation*** hosts power system operative control in the respective domain (e.g. Distribution Management System (DMS), Energy Management System (EMS), and Virtual Power Plant (VPP) management);
- ***Station*** represents the aggregation equipments for a geographical area of interest (e.g. data concentrators, substation automation, and local Supervisory Control and Data Acquisition (SCADA) system);
- ***Field*** represents the protection, control, and monitor equipment of the power system, usually called Intelligent Electronic Device (IED) (e.g. protection relay, bay controller, and measure acquisition system);

- ***Process*** represents the physical equipment directly involved in the physical, chemical, and spatial transformation of energy (e.g. generators, transformers, cables, solar plants, and storage).

Zones reflect a functional separation of data aggregation. For instance, real-time measurements system are typically in the Field and Station zones. Instead, functions that cover a large geographical area are usually located in the Operation or Enterprise zone.

Finally, *Interoperability Layers* are relevant to the mission of integrating and interoperating different systems in the electrical conversion chain [23]. Interoperability Layers are fund on MBSE and follow IEC 42010 standard [17]. In fact, they are architectural viewpoints to frame business, functional, informational, communication, and physical concerns. Thus, they are divided into five main layers:

- ***Component Layer*** represents the physical distribution of all participating equipment in the Smart Grid context;
- ***Communication Layer*** describes protocols and mechanisms with the purpose of data signalling;
- ***Information Layer*** describes information exchanged between functions, services, and components in terms of data models and information models;
- ***Functional Layer*** includes functions and services with their relationships and links independently from actors and physical implementations;
- ***Business Layer*** maps regulatory, market and economic structures, policies, and business models of market stakeholders involved. It supports business executives in decision-making related to new business models and cases.

Thus, SGAM identifies the right solution for a high-level architectural analysis of a Smart Grid solution and offers an effective response when dealing with reference architecture in a smart context [24]. SGAM adoptions have been proposed in the literature to couple Smart Grid use case with other areas. For instance, Hurtado et al. [25, 26] couple a Building Energy Management Systems with the Smart Grid (SG-BEMS). In [27], Schuh et al. present the E-Mobility Information System Architecture (EM-ISA) to deal with human interaction with electric vehicles in Smart

Grid use cases. Moreover, Shafiullah et al. [28] extend SG-BEMS to redefine the prosumer domain as the Neighborhood domain (SG-NEMS).

2.4 Co-simulation Techniques

The co-simulation approach is effective when dealing with multi-domain complex systems in which analytical assessments are no longer feasible considering their complexity. Co-simulation is often related to Cyber-Physical System (CPS) [29] and, in particular, Cyber-Physical Energy System (CPES) [30], of which the most prominent example can be found in the MES concept. General notions about co-simulation are thoroughly reported in [31] and [32]. In these literature definitions, co-simulation is a flexible approach that allows the integration of heterogeneous domain-specific simulators creating a shared simulation environment. Therefore, this paradigm allows decomposing a complex system in a SoS structure by applying system engineering. Each of the identified subsystems deals with a well-defined problem and is simulated by a specific simulator engine (or solver) that interacts with other subsystems' simulator engines (or solvers). This decomposition allows choosing among a set of domain-specific simulation tools to find the best solution that enhances numerical calculation and boosts the computational time of a single subsystem. For instance, Digital Real-Time Simulator (DRTS) is a plus to fulfil a Smart Grid simulation in a distributed co-simulation infrastructure.

The co-simulation approach must preserve high efficiency and accuracy in each individual subsystem simulation. Furthermore, the complex dynamic SoS simulation obtained by coupling different simulators may not cause instability and inaccuracies. In fact, computer-aided power system analysis with DRTS could be error-prone in a co-simulation environment. The effect of interconnecting different simulators together could lead to results that differ from the corresponding standalone simulation, affecting the numerical stability and accuracy of the solution.

Besides the several implementations of co-simulation frameworks, a shared general architecture can be highlighted. The main components required to build a co-simulation framework are depicted in Figure 2.9a: *i)* the *Scenario*, *ii)* the *Orchestrator*, *iii)* the *Simulator*, and iv) the *Model Instance*.

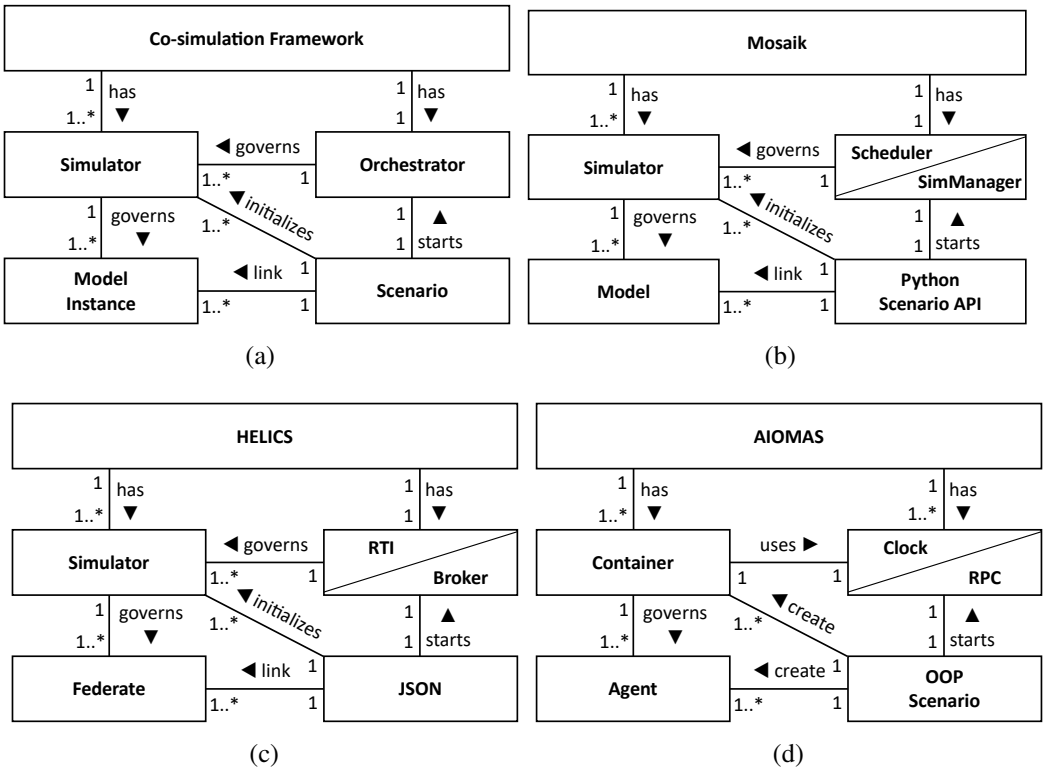


Fig. 2.9 Component relational schema of a general co-simulation framework (a) and its declination for Mosaik (b), HELICS (c), and AIOMAS (d)

The *Scenario* is a representation of the simulated environment that contains the formal knowledge of the entire complex system. It is not an actual physical component of the co-simulation framework. In fact, it represents the configuration offered by the co-simulation framework that manages the startup of the Orchestrator, the initialization of the Simulators, and states the relationships that occur between Model Instances. The *Orchestrator* is the main component of a co-simulation framework and manages the exchange of data from the Simulators and their time regulation and synchronization. *Simulators* instead contain a specific Model Instance class and have different functionalities (e.g. solvers) to perform their domain-specific computations. Simulators instantiate their Models multiple times and govern the resulting collection by acting as a communication adapter with the Orchestrator. In fact, Simulators transmit inputs received by their peers via the Orchestrator and the Orchestrator commands to their Model Instance collection. In return, Simulators receive outputs from Model Instances that are sent to the Orchestrator. Finally, *Model Instances* are representations of multiple homogeneous physical entities. They contain a physical model that could belong to different mathematical types, ranging from pure algebraic equations to differential equations, as well as finite element methods or behavioural models [29].

In addition, the arrangement of these components addresses three main tasks that represent the most important challenges to ensure a reliable, accurate, and stable co-simulation framework, which are *i)* the *Initialization*, *ii)* the *Time Regulation and Synchronization*, and *iii)* the *Data Exchange Management*.

Initialization

The *Initialization* task is performed by the Scenario that initiates the Simulators with the proper parameters setting (e.g. time step duration and start date) and communicates the number of Model Instances that compose their collection. The Initialization process finally sets up the co-simulation environment by establishing all the relationships and connections among Model Instances of all Simulators involved in the co-simulation environment.

Time Synchronization and Regulation

The *Time Regulation and Synchronization* tasks manage and regulate the time step progression of each individual Simulator. *Time Synchronisation* is mandatory when the distributed co-simulation infrastructure interacts in a time-dependent manner. It refers to the algorithm used to ensure temporally correct ordering among events generated by various simulators. *Time Regulation* instead refers to the need of instituting a policy to regulate how individual simulators evolve over time. For instance, a particular simulator could be the leader of the distributed environment (i.e. time-regulating), and some others could be a follower (i.e. time-constrained). Depending on the application, a policy must be created using a correct time regulation scheme for the simulators involved, which can have a major impact on the performance and correctness of the distributed co-simulation environment.

Co-simulation can be classified according to its time regulation paradigms [32], which are: i) Discrete Event (DE) or event-based regulation, and ii) Continuous Time (CT) or time-stepped regulation. The DE paradigm proceeds in time by exploiting events that trigger an evolution of the dynamics of the co-simulated environment. Thus, Model Instances communicate via Simulators with each other using events that might change their internal state or trigger other events. Conversely, the CT paradigm determines the evolution of the time step with a constant time interval in which the Simulators evolve their internal states by exchanging inputs and forwarding outputs at the end of each time step. Some co-simulation frameworks are able to handle both paradigms, resulting in a hybrid regulation paradigm. This case requires a complex time regulation algorithm where the synchronization task becomes even more critical.

Data Exchange Management

The *Data Exchange Management* task handles the communication among Model Instances, Simulators, and the Orchestrator by implementing telecommunication protocols that are usually the most effective solution for this task. Choosing the right protocol is fundamental to designing an accurate and reliable co-simulation infrastructure, capable of ensuring the stability of the numerical solution. In data exchange management, the main issue is related to the communication latency that usually affects telecommunication protocols. More specifically, communication

latency in co-simulation frameworks refers to the amount of time elapsed from the forwarding of the output variables of one Model Instance to the reception of the variable as input by another Model Instance. Latency is the main cause of instability and inaccuracies for a distributed co-simulation infrastructure and must be mitigated by applying specific techniques. Large latency can compromise the overall co-simulation environment when dealing with strict time constraints of a particular Simulator that could internally trigger a time step overflow.

2.4.1 Co-simulation Standards

The two main standardization efforts for co-simulation are i) High Level Architecture (HLA), and ii) Functional Mock-up Interface (FMI). In the next two paragraphs, their main infrastructures, components, and offered services are introduced.

High-Level Architecture (HLA)

The High Level Architecture (HLA) [33] is the first major generic standard for distributed simulations and provides rich set of services that are essential for creating, executing, and managing a co-simulation environment. It was originally developed by the US Department of Defence for military distributed simulations but it was designed as a completely generic architecture that can work with any simulations. HLA was taken over by IEEE in the recent IEEE1516 HLA [34] standard. It is organized into three parts: *i)* the Interface Specification that defines the key interface API that simulators use for integration, *ii)* the Object Management Template (OMT) that specifies the information that simulators communicate, and *iii)* the HLA-Rules that specifies the set of rules that individual simulators must follow to be HLA-compliant.

The individual simulations are called *federates* and the composed simulation environment with many parallel running federates is called a *federation*. Federates exchange data via data structures that are stateless or maintain states. Any federate may publish and/or subscribe to these data structures. It specifies different delivery methods such as reliable delivery or best-effort delivery. The key difference between the two methods is that with reliable delivery, a message will definitely be delivered, whereas with best-effort delivery, the message will be carried with the best effort, but there is no guarantee that it will be delivered. As reliable delivery uses various

network protocols for guaranteeing delivery of messages, it could involve large propagation delays, which in turn could increase the run-time of the overall simulation. On the other hand, best-effort delivery can be faster to execute, but it could lead to some lost messages. Thus, there is a trade-off between using reliable or best-effort message delivery methods.

To manage time regulation, synchronization among federates, HLA provides Run-Time Infrastructure (RTI), its main component. It offers time management services to the federation. Federates internally maintain their own internal logical time, which they can choose to synchronize with the RTI time or wall-clock or both or none. Furthermore, each federate can be *i) time regulating* if their time controls time progression of all time constrained federates, *ii) time constrained* if their time is controlled by time regulating federates, or iii) both time regulating as well as time-constrained, or iv) neither time regulating or time constrained. This provides significant flexibility in designing time synchronization among simulators varying from no-time synchronization to full-time synchronization, where all federates run in a lockstep manner. A number of RTI implementations exist in both the open-source and commercial domain such as Portico RTI [35], Pitch RTI [36], and Mak RTI [37].

Functional Mock-up Interface (FMI)

Functional Mock-up Interface (FMI) FMI is a recent effort by the ITEA2 project MODELISAR [38] and is widely used for model exchange and co-simulations, especially in the automotive industry. The FMI standard provides a well-defined set of function calls to specify simulation input and output variables and to control its execution and state updates. FMI-compliant simulations pack shared libraries that can be executed using the standardized function calls. These function calls span all stages of the model execution, like initialization, configuration, access, modification, and manipulation.

The FMI standard consists of two main parts. The first part is Functional Mock-up Interface for Model Exchange (FMI-ME), which standardizes the distribution of a dynamic system model in the form of generated C-Code as an input/output block to other simulation environments. The second part is Functional Mock-up Interface for Co-simulation (FMI-CS), which standardizes the mechanisms for coupling of two or more simulation tools in a co-simulation environment.

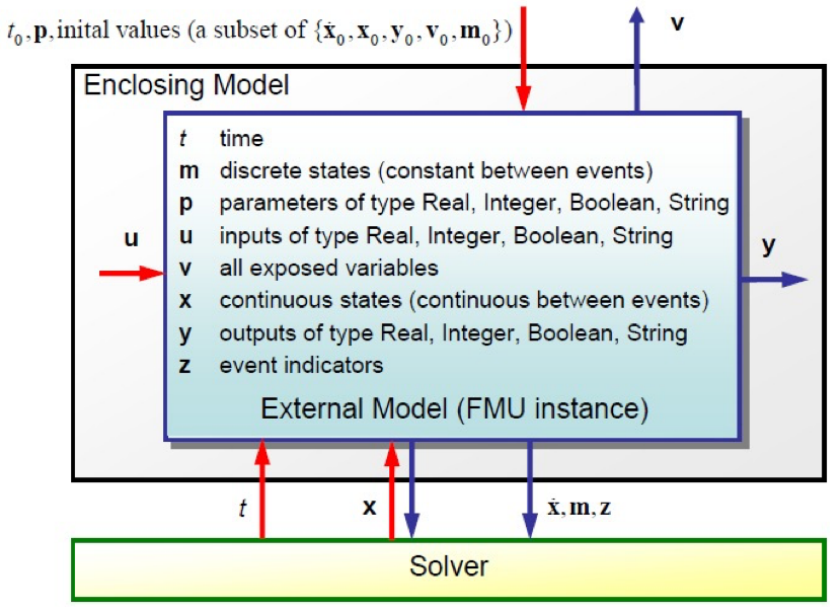


Fig. 2.10 Functional Mock-up Interface for Model Exchange (FMI-ME)

As shown in Figure 2.10, a dynamic system model is enclosed in the form of a Functional Mock-up Unit (FMU). The FMU is a zip-archive that mainly contains an eXtensible Markup Language (XML) file that provides meta-data and further details of the model such as default start and stop times, variable types, units, tool-specific data, parameter and variable names and attributes. It also contains shared library files in executable format (e.g. .dll or .so), which conform to the function call specifications given in the standard. When FMI-ME is used, the FMU contains only the system model, but no solver. Here the dynamic system model is solved through numerical integration using an external solver (see Figure 2.10). When FMI-CS instead is used, the FMU contains both the system model and a solver.

FMI assumes a discrete set of communication points, which represent the only times when the subsystems can exchange data. Outside of these points, the subsystems are executed independently. Finally, the FMI standard does not specify how to implement an orchestrator to keep the mechanism of initialization, time regulation, synchronization, and data exchange management. So, the FMI user must provide a valuable orchestrator solution to manage the FMU participating to the co-simulation environment.

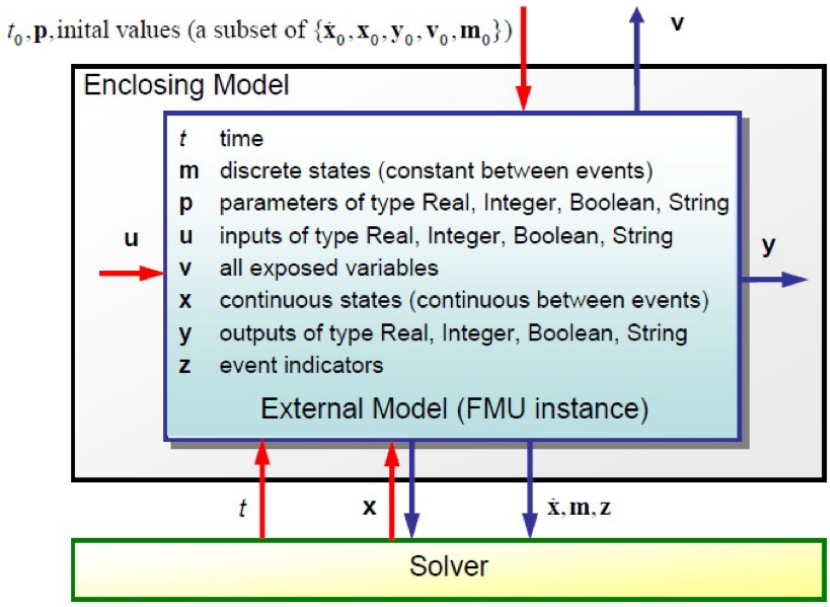


Fig. 2.11 Functional Mock-up Interface for Co-simulation (FMI-CS)

2.4.2 Co-simulation Solutions

In this Section, the analysis of two of the most widely adopted co-simulation frameworks in literature is presented. The analysis is performed by applying the above-mentioned definitions, and identifying the principal differences among the co-simulation frameworks. The co-simulation frameworks analyzed are Mosaik and HELICS. Both frameworks provide all the above-mentioned co-simulation functionalities and are exploitable with Python language.

The co-simulation approach is shared with the Multi-Agent System (MAS) concept and, indeed, several studies implement co-simulation frameworks exploiting MAS tools [39–43]. The concept of Agents in MAS applications can easily comply with the definition of SoS covering the needs of a co-simulation framework and, in particular, to its required components. AIOMAS was included in the analysis. AIOMAS is a Python library for developing MAS that could be useful in deploying a MES co-simulation environment. It can be considered a viable alternative solution to co-simulation frameworks, as demonstrated in [41]. In the following sections, the main infrastructure of these three frameworks and the details of their main components are described to better highlight their peculiarities.

Mosaik

Mosaik is a Python co-simulation framework developed to couple existing Simulators in the Smart Grid field. Its general architecture does not preclude other domain applications. Mosaik provides different Application Program Interfaces (APIs) and components for the main functionalities of a co-simulation framework. Firstly, the *Python Scenario API* allows for creating a Python script Scenario that instantiates and establishes input/output relationships between Model Instances and Simulators. The High-level Simulators API instead provides an abstract class with communication, time regulation, and synchronization features already implemented. They are language agnostic, thus allowing the integration of different programming languages (i.e. Python, C++, and JAVA) and Simulator software (e.g. MATLAB). The Low-level API instead offers the possibility to establish a plain network socket for exchanging serialized JSON data to extend Mosaik Simulators integration capabilities. The implementation of this API requires a meta description of the Simulator that states its parameters and the exchanged variables.

Figure 2.9b depicts the relational entities in Mosaik architecture. The Orchestrator role is fulfilled by two components: the *SimManager* and the *Scheduler*. These two components respectively share the tasks of Data Exchange Management and Time Regulation and Synchronization. The SimManager starts the Simulators that govern their Model Instance collection and, subsequently, handles their data exchange. Mosaik manages multiple Simulators that can create Model Instance collection by instantiating their Models. The Scheduler instead synchronizes the Simulators' time regulation and could manage both CT and DE paradigms.

HELICS

Hierarchical Engine for Large-scale Infrastructure Co-Simulation (HELICS) is a co-simulation framework based on IEEE1516 HLA standards [44, 45]. It integrates Simulators from different programming languages (i.e. Python, C++, JAVA, Nim) and simulation software (e.g. MATLAB) in a scalable and distributed environment.

The HELICS architecture and its relational entities are presented in Figure 2.9c. The Scenario in this framework consists of a JSON configuration file in which all the necessary links and parameters for the instances are made explicit. HELICS introduces a different terminology with respect to Figure 2.9a. It retains the concept

of Simulators, which in this case, are generic executables that can instantiate a multitude of Federates. Federates represent specific entities defined in the Scenario that executes their respective physical models. HELICS architecture is distributed so each Federate can communicate with others through a publish/subscribe [46] approach via Cores. Cores are components embedded in Simulators that allow their Federates to join Federations and enable communication with the HELICS architecture. The Data Exchange Management task among Federation is guaranteed by the *Broker* component that coordinates the exchange among different Federation. A Broker could also communicate with other Brokers, and consequently with other Federations, enabling the possibility of deploying a hierarchical architecture. Finally, the Orchestrator is managed by the *Run-Time Infrastructure* (RTI), a component inherited by HLA standard, to ensure a proper Time Regulation and Synchronization of the overall co-simulation environment in both CT and DE paradigms.

AIOMAS

AIOMAS is a Python library to implement MAS. It has been chosen to present parallelism between MAS and co-simulation frameworks. At an higher level of abstraction, AIOMAS provides four main classes: *i)* the Container, *ii)* the Agent , *iii)* the Remote Procedure Call (RPC), *iv)* the Clock and *v)* the OOP Scenario.

Figure 2.9d shows AIOMAS relational entities following the generic co-simulation infrastructure described above. The Object Oriented Programming (OOP) Scenario component in this configuration does not have a specific implementation. In fact, its design and development are completely up to the end user who can decide to create a specific general Python script or distribute Agents linking inside their Python classes (i.e. OOP Scenario). The tasks required to establish the co-simulated environment are similar to the aforementioned frameworks and are: *i)* the Container creation, *ii)* the Agent collection generation, *iii)* and the distributed orchestration infrastructure start-up (i.e. Clocks/RPC). Agents incorporate specific models of the physical entities they describe and execute their behaviour. Containers, on the other hand, host Agents and communicate with them via RPC servers that handle the Data Exchange Management task. In a Container, Agents implement RPC clients that manage remote communication, using the Container as a gateway to reach Agents that belong to other Containers. Each Container implements the task of Time Regulation and Synchronization through the implementation of a shared distributed Clock. The time

evolution of the co-simulation environments could follow the CT or DE paradigms, depending on the user's implementation choice.

Chapter 3

Related Works

In the last few decades, a robust research effort has been given to develop domain-specific simulation tools that have been used to simulate the behaviour of MES and solve the problems of a particular domain with high efficiency and accuracy [47]. However, the analysis of complex MES requires the collaboration of researchers from different disciplines in the energy, ICT, social, economic, and political sectors. These perspectives reflect on the difficult effort needed to simulate MES to assess their efficiency from an operational and planning viewpoint. Standalone solutions have been proposed in the literature to analyse and simulate MES. However, these solutions focus only on some of the above-mentioned perspectives. Other solutions are more complete and allow the analysis of all the aspects required by MES. Often, these solutions follow a vertical design in different fields of technology (e.g. electrical and thermal engineering, distribution and transmission grid management, and energy market analysis). Hence, MES scenario developers must dedicate a steep learning curve to master these solutions.

A growing effort appears to focus on combining two or more modelling frameworks to integrate aspects of different domains and functional layers with the exploitation of novel methodologies, standards, and tools [48], such as co-simulation infrastructures. Co-simulation has been widely applied to integrate several models in order to represent and describe the complexity of MES [49, 50]. In particular, co-simulation platforms have been developed for studying applications such as new strategies for city energy supply and demand, urban energy planning, or distribution

networks analysis and stability for an efficient DER penetration in a new smart citizen-centric energy system [51, 52].

The energy research community has concentrated its effort on applying co-simulation covering three main sectors of MES, also called integrated energy systems: i) Smart Grid, ii) Cyber-Physical Energy System (CPES), and iii) MES District and Smart MES Building.

3.1 Smart Grid Co-simulation

The first and most promising sector in which co-simulation has been adopted as a valuable solution to assess innovative components, equipment, control strategies, and services is the Smart Grid context. Following the Smart Grid Architecture Model (SGAM) definition, different domains of interest can be identified, such as generation, transmission, distribution, DER, and customer premises. The most demanding simulations are related to the transmission and distribution grids where the security, resilience, and safety of their operations is of paramount importance to avoid power grid overloads and, in the worst cases, blackouts of portions of the power grid.

When dealing with pure software co-simulation for power systems, different solutions are proposing co-simulation technologies to interconnect heterogeneous simulation environments by means of an orchestrator entity that manages time evolution, regulation, and data exchange between different simulator entities. For instance, Bharati et al. [53] implement a complex transmission and distribution co-simulation framework based on HELICS to effectively provide innovative algorithms for future power grid planning, validation of controls under critical contingencies, and validation of various wide-area monitoring and controls. However, these solutions cannot deal with the time-domain analysis (e.g. EMT) of large Alternating Current (AC) power systems that requires significant computational power to reduce the simulation time-step, enlarge network sizes, and accurately capture the fast transients. For EMT analysis, a widely accepted and well-used pure software solution is the Electro-Magnetic Transients Program (EMTP) [54] that implements the Dommel algorithm for the network solution.

Owing to the limits of pure software-based simulations, rising interest in testing real-world hardware has focused power system researchers' attention on real-time simulation [55]. Such a paradigm refers to a software model of a physical system that can execute at the same rate as the real-world physical system following the *wall clock* time. Simulations of such a paradigm are performed in a discrete constant time-stepped environment (i.e. fixed time step simulation) in which they must solve the internal state equation of the system under test in less time than the fixed time step duration. Conversely, an over-run error occurs. The time constraint of a real-time simulation varies depending on the application: transient stability studies, for example, can be performed with phasors-based simulations with a time step duration in the range of 10 ms. On the other hand, Electro-Magnetic Transients (EMT) simulations require around tens of microseconds fixed time step duration to depict the detailed dynamics of large AC systems [56]. To this purpose, innovative multiprocessor architecture (e.g. IBM® Power8) and Field Programmable Gate Array (FPGA) have been proposed as a suitable solution to ensure hardware acceleration of EMT analysis [57] to respect real-time constraints. Different works analysed multi Digital Signal Processor (multi-DSP) [58–60], multi Reduced Instruction Set Computer (multi-RISC) [61], PC-cluster architectures [62, 63] and FPGA solutions. For instance, Chen et al. [64] present an FPGA-based real-time EMTP simulator based on a deeply pipelined paralleled Dommel algorithm.

Moreover, such technologies ensure fast Digital and Analogue Input/Output (I/O) facilities to create the closed-loop interface with a real power system component, allowing Hardware-In-the-Loop (HIL) and Power Hardware-In-the-Loop (PHIL) to test its functionalities in a protected environment (i.e. laboratory setup). PHIL avoids huge costs in deploying such a component in the real world and shortens the design cycle. Nevertheless, PHIL is subject to stability and accuracy issues due to the latency of communication and power amplifier harmonic distortions between the power Device Under Test (DUT) and the simulated Rest Of the System (ROS). For this reason, different Interface Algorithms (IA) have been proposed in the literature to mitigate the effect of communication latency and stabilise the overall system under test [65], such as the Ideal Transformer Method (ITM).

In the last decades, different commercial real-time solutions have gained the interest of power system designers to address the real-time constraint and apply HIL and PHIL testing, so-called Digital Real-Time Simulator (DRTS). The most important DRTS producers for power system analysis are RTDS Technology and OPAL-RT.

In particular, RTDS Technology proposes the NovaCor chassis, a POWER8 RISC 10-core architecture, capable of continuous real-time EMT. Different plug-and-play external boards enable Digital I/O, Analogue I/O, and standard communication protocols for power systems (e.g. PMU, GOOSE, SV, MODBUS, etc.) according to Standards IEEE C37.118 [66] and IEC 61850 [67], widening its scope of application. Finally, RTDS provides RSCAD, a comprehensive software graphical interface to design complex power system scenarios.

OPAL-RT instead propose a different chassis solution (e.g. OP5700) that implements FPGA-based real-time simulator to meet the requirements for the most demanding PHIL and Rapid Control Prototyping (RCP) application. Also, in this case, OPAL-RT furnishes Digital and Analogue I/O and different standard communication protocols on board. Opal-RT proposes RT-LAB as a graphical interface, an extension of the MATLAB Simulink environment dedicated to the control of their simulator chassis.

However, both DRTS suffers a limited number of nodes that restricts the scalability of the Power System Under Test (PSUT). Different works have proposed to relax the complexity of the simulation of some parts of the power network in analysis and scale up the PSUT, the so-called multi-rate approach [68]. The multi-rate approach proposes to define different time resolutions for different areas of the PSUT but still, the scalability is limited. To cope with such a limitation, the power system research community starts proposing co-simulation techniques to interconnect together different DRTS exploiting fast high-bandwidth telecommunication protocols based either on TCP or UDP. A detailed overview of methods, test procedures, studies, and experiences in this regard is provided in [69]. This work is presented by members of the Survey of Smart Grid International Research Facility Network task on Advanced Laboratory Testing Methods. Other related co-simulation works are discussed in [70, 71] for distribution networks and Transmission System Operator (TSO)-Distribution System Operator (DSO) customer coordination studies, respectively.

For instance, the VILLAS framework [72] allows the setup of geographically distributed laboratories, interconnecting different DRTS to enable PHIL testing. But even with these optimistic premises, DRTS interconnection suffers a series of inaccuracies due to time latencies, jitter, limited bandwidth, and network interface management of the communication link. These inaccuracies could affect the stability

of a PSUT co-simulation as in PHIL systems when trying to interconnect a DUT to a simulated ROS. In fact, PHIL introduces several criticalities with error (i.e. time delay and power amplifier harmonics distortion) generated by the power interface that may cause severe instability issues or unacceptably inaccurate results. These effects are comparable to DRTS interconnection ones. Similarly, [73] presents a framework for virtual integration of laboratories. This can enable co-simulation and joint experiments involving both hardware and software resources hosted at geographically distributed laboratories. In [74], Ren et al. present the PHIL instability problem highlighting the importance of checking the closed-loop stability and improving it through a particular IA. In [65], the most interesting IA are compared together: *i)* the ITM and its variants [75–77], and *ii)* the Damping Impedance Method with different estimation algorithms of the damping impedance [78, 79]. The outcome of this comparison highlights that ITM is the straightforward and the simplest IA to implement PHIL application.

More general HIL interconnection for IED testing (e.g. PMU) could exploit other ICT to mitigate the effects of desynchronisation among DRTS internal clock and the IED DUT in HIL configuration. In this regard, IEEE1588 Precision Time Protocol (PTP) [80] is one of the most used synchronisation protocols in Smart Grid testing. PTP aligns with high precision different internal clocks avoiding effects of time misalignment of ROS and DUT during real-time simulations. For instance, Blair et al. [81] propose a real-time test-bed for measurements and analysis of distributed PMU systems for Smart Grid control and protection exploiting PTP to synchronise the simulated ROS and the PMU IED DUT. Similarly, a co-simulation test bench using OPAL-RT with its modelling software RT-LAB and FPGAs for analysing photovoltaic power generation systems is presented and discussed in [82]. EMT-Root Mean Square (RMS) co-simulation involving OPAL-RT's ePhasorSim and RTDS is covered in [83]. This work demonstrates the applicability of this co-simulation for HIL testing of protective relays. Likewise, [84] presents a hybrid co-simulation setup based on the Mosaik framework for EMT-RMS co-simulations. In this work, the authors employ the FMI for the co-simulations. Power systems and communication co-simulations are discussed in [85–87] for SCADA systems, cyber-physical studies, and cybersecurity investigations, respectively.

3.2 Cyber-Physical Energy System Co-simulation

Cyber-Physical Energy System (CPES) are of great interest to analyse integrated energy systems solution that deals with innovative control strategies and services. In this context, the physical energy system (e.g. power system) and its main components (e.g. transformer, relays, switches) are equipped with Intelligent Electronic Device (IED) that allows *i)* the collection of measurements, data, and information about their operational status, and *ii)* the direct control of their equipment. IEDs are interfaces for the cyber world that allow communication via Internet protocols to distributed intelligence (e.g. control room) permitting the optimal monitoring and control of the physical energy system.

When dealing with CPES co-simulation of innovative control strategies, an energy system simulation (e.g. power system) is interfaced with a communication network (e.g. Metropolitan Area Network (MAN)) to implement innovative services and test the data exchange among physical components and equipment by means of innovative communication protocol (e.g. IEC/ISO 61850). These co-simulations have been mainly focused on the study of innovative Smart Grid solutions. A comprehensive review of CPES Smart Grid solutions has been presented in Table 3.1. It reports a comparison of the reviewed smart-grid co-simulation environments. It highlights: i) what kind of co-simulation is performed; ii) if the framework is compliant with SGAM; iii) if and what network simulator has been integrated into the framework; iv) if a DRTS is used; v) which are the integrated software frameworks; vi) if HIL simulation are supported; vii) if the simulation environment interacts with Internet of Things (IoT) devices.

For instance, Hopkinson et al. [88] present a co-simulation environment, called EPOCHS, that is based on HLA standard [33]. EPOCHS is a breakthrough technology since it is the first known co-simulator for realising simulations of power systems and communication networks. To evaluate electromagnetic scenarios involving communication networks, EPOCHS integrates i) ns-2 [89], a communication network simulator; ii) PSCAD/EMTDC, a commercial electromagnetic transient simulator; and iii) PSLF, a commercial electro-mechanical transient simulator. INSPIRE [90] is another co-simulation platform based on HLA technology. It interconnects a power system simulator, called DIgSILENT PowerFactory, with a communication network simulator, called OPNET [91], and a continuous time-based application modelled in MATLAB, JAVA, GNU R and C++. It is based on both the IEC/ISO

61850 standard and the IEC/ISO 61968/61970 common information model. Thus, INSPIRE is limited on simulating the interactions between both *Components* and *Communication Layers* in SGAM. Shum et al. [92] presented a more complex ad-hoc co-simulation platform based on HLA and JADE, a popular multi-agent platform. The platform integrates OPNET with the commercial electromagnetic transient simulator PSCAD/EMTDC. The framework aims at easing in understanding, evaluating, and debugging of distributed smart grid software.

Compared to [88, 90, 92], GECO [93] exploits a different approach to create a co-simulation environment between PSLF and ns-2. In GECO, co-simulation is achieved through a global event-driven mechanism to manage the co-simulation orchestration instead of HLA technology. Yang et al. [94] presented a co-simulation framework for validating distributed controls in Smart Grids, based on both Hardware- and Software-In-the-Loop approaches. Controls are designed by exploiting a model-driven approach based on the IEC/ISO 61499 standard. The communication among power plants and the controllers, that are developed in MATLAB Simulink, is achieved through UDP and TCP sockets. Mosaik [100] is a framework for modular simulation of active components in Smart Grids that also follows an event-driven approach. Mosaik has been used by Nguyen et al. [95] to integrate DRTS (i.e. RTDS) with Omnet++ (a communication network simulator) and perform Power Hardware-in-the-Loop simulations.

For analysing the performance of Smart Grid technology, Bian et al. [96] presented a real-time co-simulation platform that integrates Opal-RT and OPNET. The platform provides the possibility to interface the OPNET simulation environment with the real world through Internet protocols. Venkataramanan et al. [97] developed an ad-hoc co-simulation platform that can be used as a test-bed for microgrid cyber-physical analysis. It combines: i) an RTDS, ii) the Common Open Research Emulator (a.k.a. CORE), and iii) Open Platform Communication based on FreeOPCUA. OOCoSim [98] is another co-simulation framework that integrates both OPNET and OpenDSS (a power system simulator). OOCoSim framework has been applied to study only demand response applications in a smart-grid scenario. Finally, Mirz et al. [99] proposed a co-simulation architecture for power system communication and market analysis in smart grids. The authors proposed a data model to integrate and allow the communication among i) market models in python, ii) power system simulations in Modelica, and iii) network models in ns-3.

	Co-Simulation Purpose	SGAM Compliant	Multi model				IoT
			Network Simulators	DRTS	Software Framework	HIL	
EPOCHS [88]	Investigating electromagnetic scenarios involving communication	No	ns-2	No	PSCAD/EMTDC PSLF	No	No
INSPIRE [90]	Analysing real-time performance of wide area monitoring, protection and control application	No	OPNET	No	DIgSILENT PowerFacory MATLAB Java C++ GNU R	No	No
Shum et al. [92]	Facilitating to understand, evaluate and debug distributed smart grid software	No	OPNET	No	PSCAD/EMTDC JADE DecompositionJ	No	No
GECO [93]	Investigating wide area measurement and control schemes	No	ns-2	No	PLSF	No	No
Yang et al. [94]	Validation distributed controls in smart grids considering communication networks	No	None	No	Matlab Simulink Functional Blocks	Yes	No
Nguyen et al. [95]	Holistic evaluation of smart grid solutions addressing mainly power and ICT domains	No	Omnet++	Yes	Matlab	Yes	No
Bian et al. [96]	Analysing performance of smart grids in real-time	No	OPNET	Yes	Java Eclipse	Yes	No
Venkataramanan et al. [97]	Co-simulation test-bed for cyber attacks on micro-grids	No	CORE	Yes	Python	Yes	No
OOCoSim [98]	Simulating the integrated scenario on power and communication networks to evaluate the effects on networked control in the smart grids	No	OPNET	No	OpenDSS	No	No
Mirz et al. [99]	Co-simulation of smart grid scenarios by integrating power grid, network and market models	Yes	ns-3	No	Python Modelica	No	No

Table 3.1 Co-simulation solution in the literature for CPES.

3.3 MES District and Smart MES Building Co-simulation

Finally, the last sectors in which the energy research community applied co-simulation techniques are the smallest unit of this innovative integrated energy systems vision: MES district and Smart MES Building. In this context, new enabling technologies have been adopted to support the co-simulation techniques, such as FMI [101] and Mosaik co-simulation framework [102]. In [51], authors demonstrated that FMI [101] is essential in co-simulating MES district models. In particular, they performed a comparative analysis between canonical standalone simulation and co-simulation through FMI with the aim of assessing performance and scalability. They demonstrated that co-simulation can run up to 90 times faster than the integrated simulation for 24-dwellings in a district. Meanwhile, authors in [52] showed the capability of the Mosaik framework by developing a Cyber-Physical Energy System (CPES) test environment for simulation planning, uncertainty quantification, and the development of Multi-Agent System (MAS). Nevertheless, due to the complexity and heterogeneity of MES, the co-simulation platforms in literature seem to still face technical challenges in providing an easy-to-use and cross-domain scenario composition and co-simulation environment.

Although energy district offers great potential for saving, the smallest entity that could compose a MES are buildings and these entities offer the most feasible and suitable innovation for the integrated energy vision. In fact, the great potential in energy-saving and grid balancing that Smart MES Buildings could offer has attracted many researchers to developing co-simulation frameworks tools specialised to model in detail these buildings by describing their individual components and their complex interactions [104–106, 110, 111, 107–109]. Indeed, they play a crucial role as they are responsible for roughly 40% of the overall energy consumption [112]. Table 3.2 reports a comparison of reviewed co-simulation frameworks for MES, with particular attention to Smart MES Buildings, highlighting the co-simulation techniques, the scenario design and composition procedures, and the use of standalone models.

Commonly, the co-simulation techniques were used to enhance the modelling of individual components of Smart MES Building by using EnergyPlus, which is one of the most widely used building energy modelling software [104–106, 108, 109]. Fewer researchers have performed co-simulation with EnergyPlus by use of a middleware coupling [106, 109], such as Building Controls Virtual Test Bed (BCVTB) [113] that allows connecting different simulation programs, such as Modelica, Radiance,

Refs	CF	FMI	COE	DC	SDC	TM	EM	BM	PM	SM	HCCS	TPS	Are stand-alone models?
This work	×	×	Mosaik	×	Scenario Builder	×	×	×	×	×	×	×	Yes
Schiera et al. [103]	×	×	Mosaik		Single YAML setup	×	×	×	×		×	×	Yes
Chapman et al. [104]	×	×	No-MASS/FMI		Individual models setup	×		×			×		Partially
Thomas et al. [105]	×	×	CitySim/FMI		Individual models setup	×		×			×	×	Partially
Kwak et al. [106]	×		BCVTB		Individual models setup	×	×	×			×	×	Yes
Nicolai et al. [107]	×	×	SimulationX		Individual models setup	×		×			×		Yes
Wang et al. [108]	×	×	EnergyPlus/FMI		Individual models setup	×		×			×		Partially
Jia et al. [109]	×		BCVTB		Individual models setup	×		×			×		Yes

Acronyms used in table. CF: Co-simulation Framework, FMI: Functional Mock-up Interface, COE: Co-simulation Orchestration Engine, DC: Distributed Commputing, SDC: Scenario Design & Composition, TM: Thermal Model, EL: Electrical Model, BM: Behavioural Model, PM: energy Production Model, SM: energy Storage Model, HCCS: Heating and Cooling Control System, TPS: Third-Party Services integration.

Table 3.2 Co-simulation solution in the literature for MES, with particular attention to MES District and Smart MES Building.

and MATLAB Simulink. However, BCVTB is not standardised and is limited to the use of ad-hoc integrated modules. Many other researchers have chosen a more flexible and standardised approach using FMI coupled with EnergyPlus and various software and tools [104, 105, 108] or FMI coupled with another building energy simulation tool in a Modelica-based environment [107].

3.4 Novelties

This dissertation faces the challenges still open in the modeling and co-simulation of Multi-Energy System (MES) by proposing different solutions that could assist a MES designer facing different MES scenarios:

- Following a system engineering approach, ***General-purpose Architecture for MES (GAMES)*** is proposed as a reference architecture for MES use case description. GAMES supports a MES designer with the following tasks: *i)* describe the MES use case definition through UML application; *ii)* outline the systemic description by extending the UML use case with SysML diagrams of the complex interactions among cyber-physical components and entities; and *iii)* deploy the co-simulation infrastructure for testing of the MES scenario with an automated compilation of the most basic models of the components and entities that interact in the MES use case.

- The ***Pure Software Co-simulation Infrastructure*** is proposed when the MES scenario under analysis could be simulated by means of common GPPL (e.g. Python, Java, C++) and/or simulation software (e.g. EnergyPlus, MATLAB Simulink, Modelica) and does not require real-time simulation (i.e. DRTS) to run fast time stepped models (e.g. EMT analysis of power system).

- The ***Digital Real-Time Co-simulation Infrastructure*** is proposed when the MES scenario concentrates on real-time fast transient analysis of a MES (e.g. EMT analysis of power grid).

- The ***Hybrid Co-simulation Infrastructure*** is proposed to join the Pure Software and Digital Real-Time Co-simulation Infrastructure when a MES designer requires both slow and fast co-simulation environments.

- Finally, the ***Distributed Event-Driven Co-simulation Infrastructure*** that allows CPES analysis of MES to test disruptive control strategies in a co-simulation environment that then could scale to real-world applications.

In the following sections, the novelty with respect to the state-of-the-art solutions of co-simulation techniques and infrastructure are highlighted for every contribution.

3.4.1 General-purpose Architecture for Multi-Energy Systems

To the best of our knowledge, a reference architecture for MES is not present in literature. None of the above mentioned SGAM extensions in Section 2.3 deal with MES use case design. Moreover, the presented literature solution on co-simulation does not follow reference architecture models backing a high-level use case description. Thus, they do not assist a use case designer in analysing a dynamic complex system scenario to deploy a reliable operational analysis of a MES. Thus, an integrated approach is needed to: i) propose a reference architecture model for MES use case description to enhance knowledge integration among MES designer community, ii) allow the integration of a systemic description of a MES use case into the above-mentioned reference architecture to deal with a grey-box description of the component involved, and iii) incorporate an automated process that permits to translate the systemic grey-box description into specific simulator DSL and interconnect them following a co-simulation approach.

This dissertation propose General-purpose Architecture for MES (GAMES). GAMES follow the innovative approach *"from the black box to the white box"*. The presented architectural model integrates and extends the SGAM to deal with the definition of MES use cases following a black-box approach. Thus, the proposed solution enables a modular methodology where different aspects of a MES can be analysed altogether with high-level details of the use case description, exploiting UML to characterise each component involved as a black-box. In its core, it implements a SysML coupling with the proposed architectural model to deal with a grey-box systemic description of the MES use case describing i) single components structure (i.e. input, attribute, parameter, and output), ii) each component specific simulation target (i.e. DSL or HIL specification), and iii) interconnection between components towards a operational description of the MES use case. This will allow translating the systemic description of components involved in MES use cases into

DSL code of each software simulator involved in the co-simulation scenario and hardware configuration files for the interconnection of real-world devices, enabling a *white-box* analysis of software and hardware involved in the MES use case and a secure set-up of a co-simulation framework to run MES use cases.

3.4.2 Pure Software Co-simulation Infrastructure

Although the reviewed co-simulation frameworks are suitable for analysing specific MES solutions, they seem to lack compelling flexibility and usability on easy and collaborative integration of several standalone domain-specific models for a broader analysis of MES use cases [104–106, 110, 111, 107–109]. Indeed, these frameworks generally use circumscribed co-simulation master algorithms or delimited co-simulation environments that make demanding and effort the integration of models and scenario composition [104, 105, 107, 108]. Moreover, in most cases [104, 105, 108], the implemented models are not standalone. Still, they are mainly inside of a few comprehensive simulators, thus partly losing the potential and advantages of the co-simulation framework. Hence, none of these literature solutions integrate different domain-specific MES models together.

On these premises, this dissertation presents a Pure Software Co-simulation Infrastructure that exploits a multi-modeling approach in order to simulate and assess energy performance in MES. The presented platform integrates heterogeneous simulation models by exploiting the flexible Mosaik co-simulation framework [102] that has been extended to embed FMI [101] allowing the interoperability among various simulation engines and tools (i.e. Energy Plus, Modelica, and MATLAB Simulink). The proposed platform permits a modular integration of several domain-specific models and simulation engines with a plug-and-play fashion through an effective cross-domain scenario configuration procedure. This solution tackles the challenges associated with the combination of software and knowledge across various domains, enhancing cooperation among the domain experts and making it easier to implement their models. Furthermore, our solution allows locating each model in different network nodes which communicate with the master node (i.e. Mosaik) through Internet protocol suite (i.e. TCP), providing a distributed co-simulation environment.

With respect to literature solutions compared in Table 3.2, our platform simultaneously integrates different heterogeneous and distributed models. The platform has been assessed with a scenario that simulates the electrical and thermal energy demand and behaviour of a smart MES building, which includes: *i)* models for household occupancy, thermal demand, and indoor temperature scheduling, *ii)* models to realistically emulate appliances' load consumption, *iii)* photovoltaic simulator based on geographic information system, *iv)* an electric heat-pump with integrated control strategy, *v)* smart meters, *vi)* an electric storage, and *vii)* weather information provided by third-party services through web-service communication.

3.4.3 Digital Real-Time Co-simulation Infrastructure

With respect to literature solutions, none of the above-mentioned methods is capable of enhancing the scalability of the PSUT without affecting its numerical stability and accuracy. Hence, this dissertation proposes the Digital Real-Time Co-simulation Infrastructure that uses the Aurora 8B/10B protocol. This ensures low communication latencies and consequently, the least nonlinear effect on the numerical solution of the PSUT. Moreover, the proposed co-simulation architecture exploits IEEE1588 PTP that synchronises and regulates continuous-time behaviour with high precision. This is achieved by aligning the internal reference clocks of the different interconnected DRTS with a common Global Positioning System (GPS) clock signal. Furthermore, the additional scientific novelty in this work is the application of the ITM IA to the proposed distributed Digital Real-Time Co-simulation Infrastructure. Exploiting ITM IA, we obtain a decoupled PSUT numerical solution. This is demonstrated by following the Nyquist principles of frequency-domain analysis, commonly used in PHIL context to determine the stability of IA. The proposed frequency-domain and time-domain analyses prove the basis for the application of the proposed distributed digital real-time co-simulation infrastructure for power system analysis.

3.4.4 Hybrid Co-simulation Infrastructure

None of the above-mentioned literature solutions is capable of interconnecting pure software co-simulation with digital real-time co-simulation. For software and GPPL simulators, the entire pure software co-simulation process may require a master algorithm that ensures the correct evolution of the simulation environment, so-called

COE. For digital real-time co-simulation, the simulators (i.e. DRTS) process instead cannot be controlled by an external entity since their time regulation must respect the real-time constraint to permit HIL and PHIL interconnection for testing purposes, so-called hard real-time environments. So, the interconnection among software and GPPL and hardware simulators raises complex issues, which have not yet been tackled by the research community.

This dissertation presents the innovative Hybrid Co-simulation Infrastructure that allows the interconnection among multi-model simulation software (e.g. MATLAB Simulink, Modelica, EnergyPlus), General-Purpose Programming Language (GPPL) (e.g. Python, C++, Java), and commercial Digital Real-Time Simulator (DRTS) (e.g. OPAL-RT). To couple pure software co-simulation with the hard real-time world of DRTS, it exploits a soft real-time approach where the pure software co-simulation environment runs at the wall-clock time, mimicking a real-world scenario. The soft real-time approach is not obliged to precisely respect the real-time constraints and must allow the possibility to run slightly in overrun since a normal software co-simulation environment and its behavior (e.g. a MES building) does not impact the fast transient of a power grid. The soft and hard real-time environment communication is ensured by a near real-time middleware that exploits VILLAS-framework [114] to ensure the correct data exchange among simulation models and entities. By employing this disruptive strategy, the Hybrid Co-simulation Infrastructure ensures the correct wall-clock time evolution of the co-simulated MES scenario and respects the real-time constraint of the interconnected DRTS that permits to include fast time-stepped simulation of a power grid model into the MES scenario. Moreover, the DRTS capabilities enable the HIL and PHIL testing of real-world hardware, creating a powerful testbed for innovative power grid technologies and components. It is worth noting that normally the real-time world requires a small time step duration, around tens of microseconds for EMT analysis. To cope with these strict real-time constraints, the pure software co-simulation environment has been stressed and accelerated to reach low time step duration, around one hundred milliseconds, to deploy a realistic MES scenario where a building is capable of offering demand response and ancillary services (e.g. voltage regulation) to the power grid.

3.4.5 Distributed Event-Driven Infrastructure

The presented literature solutions, with the exception of [99], do not follow SGAM that supports the design of services highlighting the interoperability among the actors in the MES context. Moreover, [88, 93, 97] are designed and implemented for a specific use-case scenario with particular requirements. Whilst, these kinds of co-simulation platforms must be designed to be flexible as much as possible to simulate general-purpose services, even concurrently, in MES.

Thus, a multi-model approach is needed to allow cooperation among different simulation models, possibly distributed over the Internet and communicating by exploiting standard protocols and data formats. This makes easier the definition of more complex MES scenarios. Finally, literature solutions lack the integration of IoT devices and third-party platforms that can feed the algorithm to be tested with real-world data, even in (near-) real-time. Furthermore, the integration of IoT communication paradigms and protocols enable the required flexibility and scalability in performing HIL co-simulations. Thus, a platform that integrates IoT protocols can be ready to integrate next-generation devices (e.g. real-world smart meters).

This dissertation proposes a distributed multi-model co-simulations platform to asses and evaluates general purpose services in MES by implementing an event-driven approach. It exploits communication paradigms peculiar to IoT platforms, to implement a flexible framework where different MES scenarios and simulations can be executed. In its core, it implements *i)* different software simulation environments, *ii)* DRTS, and *iii)* three different communication network simulators (to be used alternately according to co-simulation requirements). Thanks to the integration of DRTS, the platform is also ready to perform HIL and PHIL simulations. Whilst by using IoT protocols, it allows interoperability with real-world IoT devices and third-party platforms.

The strength of this platform comes from its capability to host new different use cases, input data, or models by means of its modules in a plug-and-play fashion, with a minimum effort to change the overall setup. As an example, several DMG can be integrated as either independent simulators or standalone physical generators thanks to our IoT-based architecture which is a novel contribution to the co-simulations of MES with respect to conventional setups. The proposed platform is designed

for general purpose services, but its distinguished capability, compared to many existing platforms, is related to co-simulation and real-like analysis of the use cases in which low communication latency is crucial and electromagnetic transient analysis is required. Coordination of switching of converters connecting emerging DMG to low inertial systems for voltage or frequency control, and improving the reliability of networks by advanced outage management system are two main examples of such use cases. In these applications, a large number of devices are also involved, which makes their fast coordination and connectivity a challenge. Regarding advanced outage management systems, which is the context of the use case discussed in the use case presented in Section 6.2, a newly developed self-healing scheme named De-centralised Fault Detection and Isolation, and Centralised Restoration (De-FDI-CR) has been simulated by using the proposed platform. Successful deployment of such schemes creates self-healing distribution power grids with high reliability and customer satisfaction.

Chapter 4

GAMES: General-purpose Architecture model for Multi-Energy Systems

Several improvement attempts are trying to extend reference architectures (e.g. SGAM for electric use case descriptions) to deal with a systemic description of dynamics involved in complex MES use cases. However, a reference architecture for MES is not present in the literature. Moreover, the presented literature solution on co-simulation do not follow reference architecture models backing a high-level use case description. Thus, they do not assist a use case designer in analysing a dynamic complex system scenario to deploy a reliable operational analysis of a MES. Thus, an integrated approach is needed to: i) propose a reference architecture model for MES use case descriptions to enhance knowledge integration among MES designer community, ii) allow the integration of a systemic description of a MES use case into the above-mentioned reference architecture to deal with a grey-box description of the components involved, and iii) incorporate an automated process that permits to translate the systemic grey-box description into specific simulator DSLs and interconnect them following a co-simulation approach.

In this Chapter, the General-purpose Architecture for MES (GAMES) is proposed as an architectural modelling methodology that allows knowledge integration of different energy, ICT, financial, business, and regulatory frameworks to perform

MES modelling, use case definition and simulation extending SGAM to cope with MES.

4.1 *"From the black box to the white box"* approach

GAMES follows the innovative approach *"from the black box to the white box"*. It exploits and extends SGAM to deal with MES use cases following a black-box approach. In its core, it implements a SysML coupling with the proposed architectural model to deal with a grey-box systemic description of the MES use case describing i) single components structure (i.e. input, attribute, parameter, and output), ii) each component specific simulation target (i.e. DSL or HIL specification), and iii) interconnection between components towards an operational description of the MES use case. Furthermore, GAMES automatizes the generation of each software component specific DSL code and HIL specification allowing a reliable and secure set-up of a co-simulation framework to run MES use cases.

Moreover, it allows a systemic description of a MES use case exploiting UML and SysML to describe cyber and physical components in depth and their interconnections. Finally, it will automatise the generation of each specific DSL code or HIL configuration related to a particular MES component. Furthermore, it will interconnect them using co-simulation techniques.

Following the scheme in Figure 4.1, GAMES is structured in three main layers: *i) Conceptualisation and Specification*, *ii) Component Design Development*, and *iii) Domain-specific Implementation*. Each layer addresses one of the challenges defined in Chapter 1.

4.2 Conceptualisation and Specification

Software Engineering offers tools to address architectural descriptions, use cases, scenarios, and case studies. The aim of such a tool is to transfer tacit knowledge when trying to document experience gained in a vertical field of technology (e.g. electrical and thermal engineering, distribution and transmission grid management, and energy market analysis). Formal and explicit knowledge (as opposed to tacit knowledge) must be avoided to offer a clear high-level viewpoint of a use case.

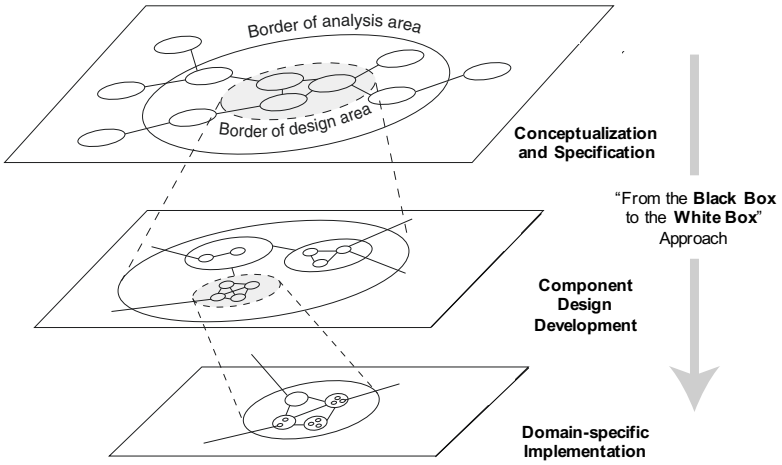


Fig. 4.1 GAMES hierarchical structure of the System-of-Systems.

4.2.1 GAMES Architectural Design

Following the hierarchical structure of GAMES shown in Figure 4.2, GAMES *Conceptualisation and Specification* is based upon an extended version of the SGAM to cope with MES use cases description. One contribution of GAMES is to expand the SGAM Component Layer to three (or more) dimensions, each one for every energy vector. GAMES Conceptualisation and Specification extends SGAM considering that other energy vectors (e.g. heat exchanging fluids and gas) share a common structure with the electrical one. Following the MBSE structure in Figure 2.7, each energy vector identifies a different *architecture view* of physical layer *concern*. Moreover, the physical management of each energy vector supply chain involves different *stakeholders* from a structural and regulatory framework perspective addressing such *concern*. Consequentially, each energy vector requires a different *architectural viewpoint*, governing the *architectural view* of the physical interconnection between components.

It may be argued that the same division should apply to other layers (e.g. Communication and Information Layer). However, MES designers must focus their effort on developing communication, informational, functional, and business solutions shared among all energy vectors involved in the use case. So, other layers are inherited by SGAM and extended to cope with all energy vector interactions.

Unified Modeling Language (UML) is considered as *de facto* standard language in the field of Software Engineering, and it has been implemented into GAMES.

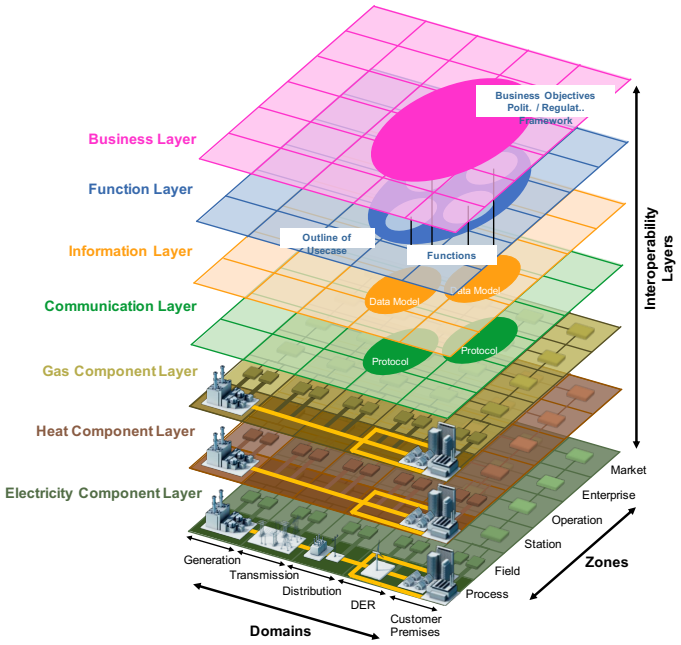


Fig. 4.2 General-purpose Architectural model for MES engineering application (GAMES).

The usage of UML in GAMES allows sharing of use cases universally among MES scientific community fostering reuse and extension of already developed use cases. A MES designer could choose among different Integrated Development Environment (IDE) supporting UML to scratch their MES use cases. Use Case formalization and MES conceptualization are processed into Business and Functional layers and they are suitably described by Use Case, Activity, and Sequence UML diagrams. While MES specifications are handled into Information, Communication, and Component layers and they are commonly described by Class, Object, Timing, Interaction and Communication UML diagrams.

4.3 Component Design Development

A conceptual mapping between SGAM Interoperability Layers and MDA reveals a lack of SGAM approach to technological representation. MDA is a design approach for software systems separating functionality and technology. The abstraction defined by MDA are:

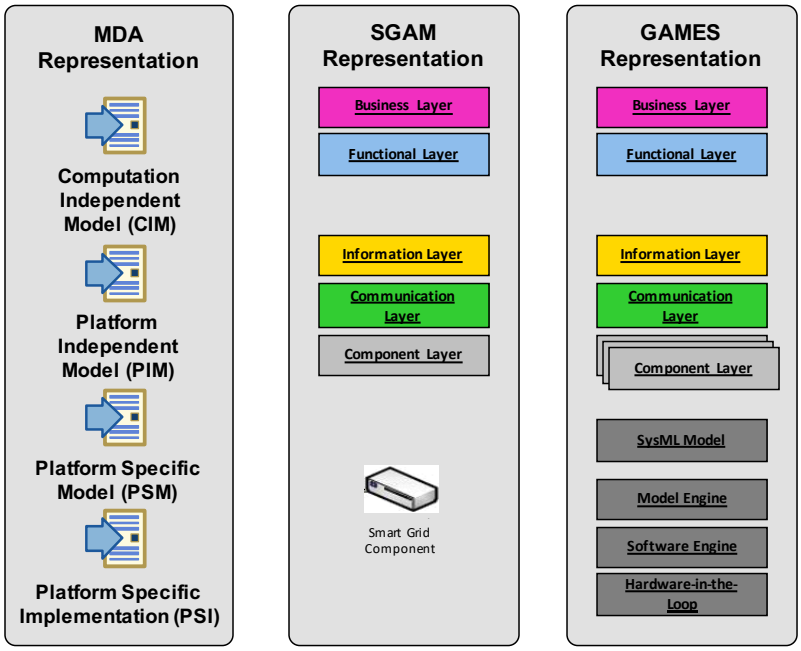


Fig. 4.3 SGAM and GAMES parallelism with Model Driven Architecture.

- ***Computational Independent Model (CIM)***: which is a systematic level describing functionality perspective;
- ***Platform Independent Model (PIM)***: which is an architectural level that decomposes the system in subsystem approaching it in a black-box fashion;
- ***Platform Specific Model (PSM)***: which involves the technological platform description of each component and the whole system, necessary for the actual implementation;
- ***Platform Specific Implementation (PSI)***: which is the implementation of the physical component, either hardware or software (i.e. source code).

A parallelism between Model Driven Architecture (MDA) and SGAM representation is shown in Figure 4.3. Business and Functional Layers express the CIM level describing the functional perspective by the means of economical and service description. Furthermore, Information, Communication, and Component Layers are represented by the PIM level concerning the SoS from ICT and physical equipment perspective. Finally, PSM and PSI are related to the Smart Grid constituent components considered as black-boxes. As shown in Figure 4.3, GAMES Conceptualisation

and Specification behaves as SGAM. However, GAMES *"from the black box to the white box"* approach allows to expand Conceptualisation and Specification (i.e. black box approach) and zoom in analysis details providing GAMES *Component Design Development*.

Component Design Development enables a design framework to describe a MES use case from a systemic viewpoint covering the needs of the MDA PIM with an in-depth analytical and logical description of the components. It exploits SysML, a general-purpose architecture modeling language for Systems Engineering applications based on UML. SysML is an enabling technology for MBSE. It supports the specification, analysis, design, verification, and validation of a broad range of systems. These systems may include hardware, software, information, and processes. SysML usage fosters the integration of functional aspects of MES components described in GAMES and their interfaces over interoperability layers, taking advantage of the same IDE to deploy the SysML diagrams. Each constituent component of GAMES is operationally, logically, and analytically described by the Component Design Development. Moreover, it describes the component interconnections and data exchange between entities bypassing the specific components' coupling and interfacing. Conversely to SGAM, each component of GAMES is unpacked as a grey box model through SysML application. The component is described by the so-called Four Pillars of SysML, referring to the four essential diagrams of SysML: Requirement, Activity, Block, and Parametric diagrams.

4.4 Domain-specific Implementation

Formal knowledge is rather important when different vertical fields of a MES are focused. In this context, formal knowledge is represented by the effort needed to design a MES component. Commonly, a MES component shows complex behaviour, endogenous and exogenous dynamics, and many interconnections with different energy vectors. When the scope of the analysis covers multiple vertical fields, the challenge becomes quite demanding.

GAMES *Domain-specific Implementation* will relieve MES designers from the formal knowledge required to exploit domain models related to the above-mentioned component technologies. The aim is to combine the UML/SysML diagrams with executable semantics to obtain an high level abstraction, supporting the translation

of PIM into PSM (model to model transformation) and compilation of PSM into PSI (DSL code generation). It will specify the selected underlying technology (i.e. DSL and hardware configuration) in which each component will be deployed, tested and validated. The automation of the process identifies the simplest and smallest description of the block, avoiding complex component behaviour. This operation prevents the manual configuration of each component which can be error-prone. Moreover, a MES designer could access each generated DSL code and integrate complex functionalities of the components in the case of a particular simulation objective. Furthermore, Domain-specific Implementation simplifies the appropriate interconnections of such heterogeneous software and hardware components allowing the interconnection to a co-simulation framework.

Chapter 5

Distributed Multi-Modelling Co-simulation Infrastructure

In this chapter, three different versions of the proposed co-simulation infrastructure are presented: i) the Pure Software Co-simulation Infrastructure, ii) the Digital Real-Time Co-simulation Infrastructure, and iii) the Hybrid Co-simulation Infrastructure. The Pure Software Co-simulation Infrastructure is a platform version that comes to aid in designing, developing, and testing MES scenarios which interconnect models designed with GPPL and simulation software. The platform allows to integrate models that present internal dynamics with a time step duration greater than 1 second and does not requires real-time evolution. This is typically the case of energy vectors and carriers that presents slow temporal dynamics, such as heat exchange fluids for cooling and heating, natural gas, and some electrical behaviour like consumption and production pattern for optimal power flow. The aim of this platform is to offer to the MES research community the possibility to integrate the vertical and horizontal knowledge for SoS design of different energy vectors, as presented in Chapter 4. The platform will integrate heterogeneous models of physical and social phenomena with a simple plug-and-play approach, accurately reproducing MES and the environment in which they operate and supporting the different stakeholders in the energy market to take short and long-term decisions. So, it will simulate phenomena that evolve at different spatio-temporal scales from households to districts and cities, from seconds to years.

The Digital Real-Time Co-simulation Infrastructure instead has been developed for modelling and simulating of MES scenario that requires slow time step duration, in the order of microseconds, to analyse fast transients such as the analysis required for assessing security, reliability, and stability of the electricity networks, like the EMT analysis of transient evolution in case of a fault in the distribution or transmission grid. These analyses commonly require the use of Digital Real-Time Simulator (DRTS) capable of solving the Dommel equations of the power system under analysis in a matter of microseconds. Furthermore, DRTS respect the real-time constraints and permits to interconnect real-world hardware and devices through HIL and PHIL techniques to test in protected environments innovative physical components for MES. However, the complexity of the Dommel equation and the relative amount of computational time required to solve them limit the scalability of the system under analysis to respect the real-time constraints. To cope with that limitation, the Digital Real-Time Co-simulation Infrastructure has been proposed to locally interconnect different DRTS by means of point-to-point fast bandwidth optical interconnection and a proper synchronization protocol to align these complex hardware calculations, allowing the interconnection of real components and equipment exploiting HIL and PHIL techniques.

When dealing with both the above-mentioned MES scenario requirements, the Hybrid Co-simulation Infrastructure glues together the Pure Software Co-simulation and the Digital Real-Time Co-simulation Infrastructure to offer the possibility to analyse slow and fast phenomena in the same platform. To cope with this challenging objective, this infrastructure exploits a coupling methodology that obligates the pure software environment to respect the real-time constraints in a slack fashion (i.e. soft real-time), since the impact of slow temporal dynamics does not impact the evolution of a fast real-time environment (i.e. hard real-time). By means of a near real-time adaptation layer, the soft and hard real-time environments could exchange data and could extend the spatio-temporal scale of the MES scenario under analysis.

In the next sections, each of the above-mentioned platform configurations is discussed in depth by following the proposed analytical structure, which *i)* describes the infrastructure configuration, *ii)* proposes a MES scenario, and *iii)* presents its relative experimental results.

5.1 Pure Software Co-simulation

The pure software co-simulation infrastructure lays the foundations of the hybrid co-simulation infrastructure by integrating different GPPL (e.g. Python, Java, C++) and simulation software (e.g. MATLAB Simulink, Modelica, EnergyPlus) into a co-simulation environment. From an architectural perspective, it enables the application of a SoS structure to analyse a MES scenario by integrating identified subsystems coming from different interoperability layers and domains of knowledge in order to fulfil the co-simulated scenario solution. To join these perspectives and help researchers in design, develop and test new components or solutions in such a complex energy system, GAMES [115] was firstly applied by using a MBSE approach to perform a holistic and systematic analysis of the MES use case, which is subsequently translated into an executable scenario and simulated in the proposed co-simulation platform.

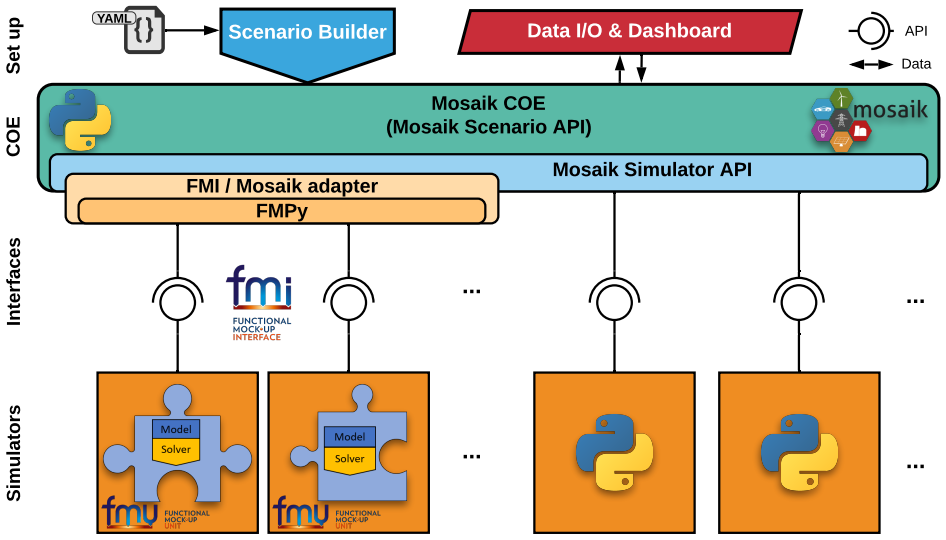


Fig. 5.1 Scheme of the pure software co-simulation infrastructure. It reports the enabling technologies and main elements of the platform: Scenario Builder, Data I/O & Dashboard interface, Mosaik COE including Scenario and Simulator API, FMI/Mosaik adapter, simulation blocks and their interfaces.

This section focuses on the software infrastructure of the proposed co-simulation platform and its embodied elements, as depicted in Figure 5.1, by showing its functionalities, capability, and usability of easily performing scenario design and composition as illustrated in Figure 5.2, integrating domain-specific models, and

simulating complex energy system scenarios. Indeed, each technological field expert could cooperate by deploying its own simulation models shared among the platform research community. In this view, the pure software co-simulation infrastructure is able to integrate already existing models performed by domain-specific simulation software in a shared and distributed co-simulation environment where the different models can be added/removed/replaced in a plug-and-play fashion through a cross-domain scenario configuration procedure. From a software platform perspective, each model is seen as a grey-box model automatically connected with the other models based on the designed scenario. Therefore, each domain expert shall develop its model only, ensuring that it exposes the input/output variables and parameters necessary to build the scenario by establishing a common understanding with the co-simulation expert and the other domain experts. Finally, since the platform has been designed to be highly distributed, the different models can run on different computers/servers connected over the Internet.

To achieve the above-mentioned purposes, the platform was based on three main enabling technologies outlined in the following paragraphs:

- ***Co-simulation Techniques:*** Co-simulation has been identified as a flexible approach to integrate sub-models coming from different domain-specific models and simulation tools in a shared and distributed simulation environment. Essentially, co-simulation techniques allow integration of a system of systems, each one simulated by a different simulator engine (or solver). Indeed, the domain-specific subsystems are modelled by the domain experts and usually simulated by their specialised solvers and modelling tools. Therefore, this approach preserves the use of efficient and suitable subsystems' solvers that are coupled to obtain a more complex dynamic system of systems simulation in terms of scalability, variety and composability of models. Moreover, co-simulation enhances performances in respect to stand-alone simulations thanks to the ability to distribute and parallelise the computation either in a cluster of servers and computers or in a cloud environment, allowing to choose the best simulation engine for each model to be integrated as they will exchange information over the Internet. Finally, co-simulation facilitates the hybrid multi-modelling approach, in which different modelling paradigms (e.g., agent-based, equation-based, discrete-event based) can be easily coupled into the shared environment.

- ***Functional Mock-up Interface (FMI)***: Commonly, domain-specific energy modelling tools and their solvers cannot communicate and exchange information with each other due to data formats incompatibility and license restrictions. This issue often raises in trying to interconnect a proprietary simulation platform (e.g. MATLAB Simulink) that requires a license to run its solver and does not allow interconnection with other simulation software. Moreover, often they do not exchange information among different instances of the same simulation engine. Finally, energy models designers may come across Intellectual Property Right (IPR) restrictions that do not allow them to share their models with the scientific community. To face these issues, FMI [101] has been proposed as a tool-independent standard that allows *i*) to encapsulate model and its simulation engine, *ii*) to support direct control of the model through a standardised interface, and *iii*) to exchange data among different models [101]. In practice, FMI defines an interface based on a set of C-functions with the model that is implemented by an executable, called FMU. In a nutshell, FMU is a ZIP file that contains all the equations used by the model, its resources, documentation, and an XML file that describes the model structure and defines the variables used by FMU. Generally, FMU either embeds the simulation engine that is supplied by the exporting tool, or it requires the simulation environment to perform numerical integrations.

- ***Co-simulation Orchestrator Engine (COE)***: The co-simulation approach requires a master algorithm to create instances of models and manage time evolution and regulation in a shared simulation environment, the so-called Orchestrator. In recent years, different COE have been developed [48] based on specific application cases. Among them, the open-source co-simulation framework Mosaik [102] provides good performance, high usability, and flexibility. Indeed, Mosaik can be integrated with several power grid simulators (e.g. pypower, pandapower) and any other simulators written with various GPPL, such as Python, C++ and Java. Moreover, Mosaik allows the distribution of simulators exploiting TCP/IP communications used for data exchange.

5.1.1 Pure Software Co-simulation Infrastructure

As illustrated in Figure 5.1, the pure software co-simulation infrastructure can be divided into different framework layers, where each layer can be distributed into different network nodes to improve co-simulation performances through such a distributed computation environment. The platform framework layers are: *i)* the set-up layer consisting of a *Scenario Builder* and *Data I/O & Dashboard* modules, *ii)* the Mosaik COE layer with the FMI/Mosaik adapter, *iii)* the interfaces layer between COE and simulators through Mosaik API, and *iv)* the layer of the attached simulators to perform the required scenario. The platform framework layers are outlined in the following paragraphs.

Scenario Builder, Data I/O & Dashboard

Scenario Builder module is a wrapper that simplifies and tailors the interface to Mosaik COE by translating input data and processing information through the different COE levels. The module composes the overall energy scenario through an effective cross-domain scenario configuration procedure that establishes a common understanding among the co-simulation expert and model domain experts, as illustrated in Figure 5.2. In particular, the wrapper parses a YAML configuration file containing all information to set up the entire co-simulation environment and distributes such information throughout the underlying classes and methods of Mosaik COE. YAML is a human-readable data serialisation standard based on nested objects as "*key: value*" structures, commonly used as a configuration file. Figure 5.3 depicts the adopted schema of a YAML file describing the scenario template structure and its main contents. In our co-simulation platform, the schema of the YAML configuration file is based on four root objects: *i) scenario configuration* that contains common scenario settings, *ii) simulator configuration* that includes models' instances with their simulation settings and initial conditions, *iii) connections* that contains topology of the connections among models' instances exploiting a list of run-time data exchanged for each connection, and *iv) scenario outputs* that includes setting to display or save the desired variables.

The use of YAML by the platform's users allows for simplifying the scenario composition using default settings patterns of the objects and focusing only on changing the desired parameters in a plug-and-play fashion. Indeed, it is possible to

compose different YAML files containing template objects that share the adopted schema into a single YAML configuration file. For example, a modelist can focus on setting significant model instance parameters and connections in agreement with the other experts and the scenario requirements, while later can add built-in YAML

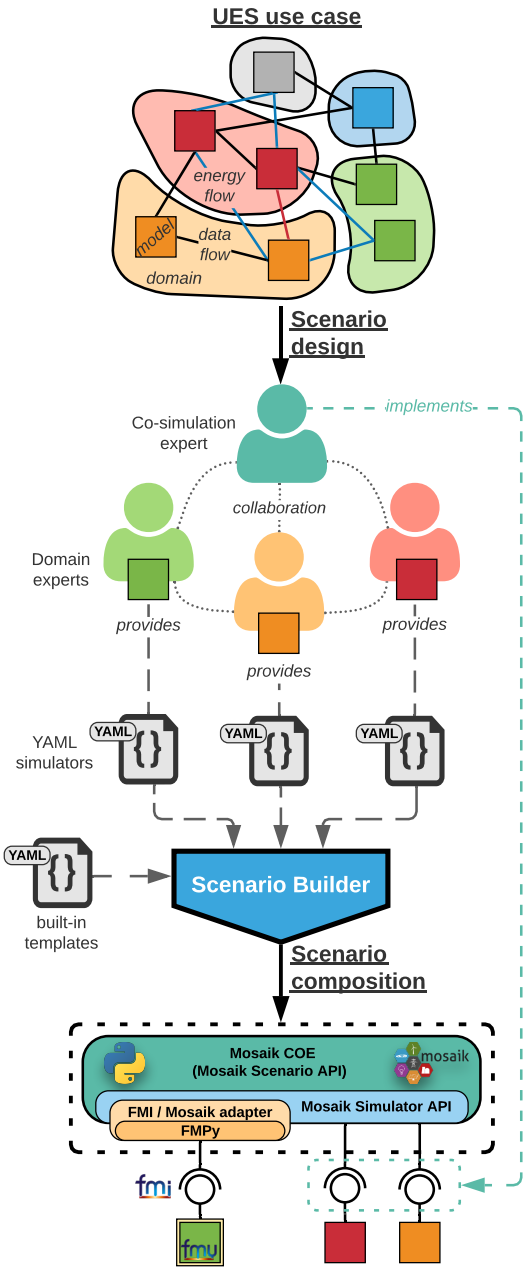


Fig. 5.2 Overview of the main steps to perform scenario design and composition starting from a MES use case within the co-simulation platform.

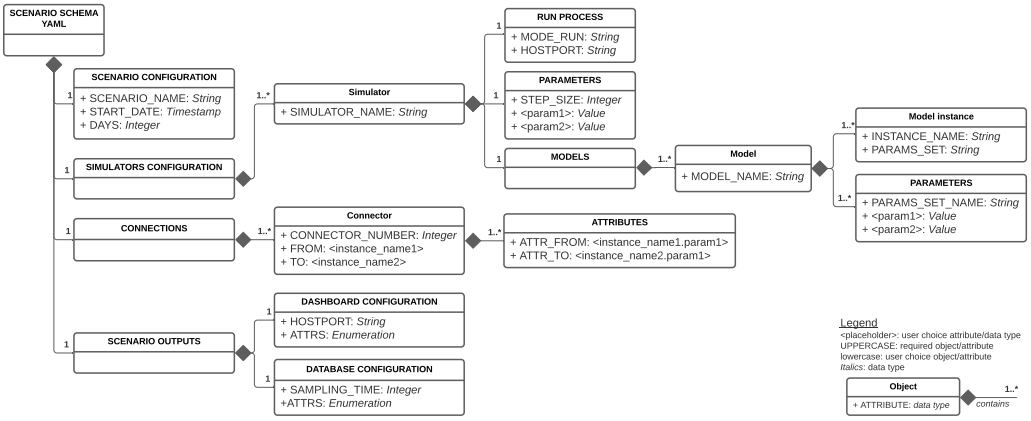


Fig. 5.3 The tree diagram from the YAML template scenario file showing the general characteristics of the scenario schema root element, with main parent elements and children elements of: *i)* scenario configuration, *ii)* simulators configuration, *iii)* connections topology of data flow, and *iv)* scenario outputs.

templates to set up the connection protocol interface and the shared simulation environment interface. Furthermore, if the platform's user explicitly requests the output of desired variables under the YAML root object *scenario outputs*, then the *Scenario Builder* automatically adds and connects back-end components that are necessary to provide the functionalities of the *Data I/O & Dashboard* module. Finally, the *Scenario Builder* module allows the centralisation of the set-up operation in a single network node that distributes the information remotely to the other network nodes.

Data I/O module instead provides an interface to easily set and retrieve requested data coming from the co-simulation infrastructure. In particular the module is made by: *i)* an output interface for storing data through *Hierarchical Data Format* (HDF5), *ii)* an input interface for setting environment, scenario, and tunable parameters through a publish/subscribe pattern [46] using the open-source asynchronous messaging library *ZeroMQ*, as well as *iii)* an interface to communicate with a web app *Dashboard*. The *Dashboard* was developed to provide a user-friendly lab-view interface that provides a data flow graph among scenario models, a run-time simulation view, and a canvas to plot the time series of the desired variables from the output simulation data set.

Mosaik COE

The *Mosaik COE* is the master node of the pure software co-simulation platform and provides time management and synchronisation to orchestrate the co-simulation environment during its run-time execution. Furthermore, it allows control and management of data flow among the models' instances. As shown in Figure 5.2, Mosaik COE receives as input the scenario information from the *Scenario Builder* and set up the co-simulation environment exploiting:

- Mosaik *Scenario API* that allows to start simulators and instantiate their models to generate the co-simulation environment as described in the scenario configuration file, i.e. how the models' instances should be parameterised and interconnected with each other.
- Mosaik *Simulator API* that has to be implemented to establish an interface between Mosaik and a simulator to set up data exchange and orchestrate all stages of the co-simulation framework (i.e. initialisation, step management, data handling by getting/setting them from/to models' instances).

The COE manages the time synchronisation based on a *Discrete Event (DE)* synchronisation method. To do so, Mosaik sets a schedule based on the time-step description provided by each simulator. According to these descriptions, the schedule contains pre-defined synchronisation points and exploits a *directed acyclic schedule graph* to determine the order of step commands, which are sent to each simulator. It is worth noting that this feature allows the integration of simulators with different time step resolutions in the same co-simulation scenario, from one second up to ad lib.

FMI/Mosaik adapter

The FMI/Mosaik adapter was developed to map the Mosaik *Simulator API* with FMI functions by using the *FMPy* library [116], an open-source Python-based library to simulate FMU, as shown in Figure 5.1. Moreover, *FMPy* supports the latest versions of the FMU co-simulation and model-exchange methods, which were validated with cross-check rules defined by the FMI standard steering committee. The adapter allows loading and controlling FMU in the shared co-simulation environment that interacts with the FMU through C-functions to create one or more instances of a

model, to manage simulation evolution, and to exchange data enabling interaction among different simulation engines. Therefore, the adapter automatise the integration of new encapsulated FMU models without the domain expert having to implement the Mosaik *Simulator API* specification, as shown in Figure 5.2. Indeed, a YAML FMU simulator template is provided to standardise the platform's user data input procedure by establishing a common understanding with the other domain experts. Subsequently, the *Scenario Builder* can process and integrate the FMU simulator into the co-simulation environment automating the setup and execution of the FMU instances.

Simulators and Interfaces

Each model has to be integrated into the co-simulation environment by exploiting the *Mosaik Simulator API* specification to handle the COE-simulator communication. This preliminary procedure could be challenging because, normally, the model's domain expert does not know the co-simulation framework, and it would be necessary to collaborate with the co-simulation expert, as illustrated in Figure 5.2. To make this procedure effective, the *Scenario Builder* module provides a standard YAML simulator template that helps to set up the simulation and model requirements through a grey-box modelling approach with minimum effort and cost. Therefore, it subsequently simplifies the model integration process into the co-simulation platform.

The co-simulation platform accepts both Python simulators and encapsulated models as FMU thanks to FMI/Mosaik adapters. The simulators expose their input/output variables and parameters to the shared simulation environment, thus the *Scenario Builder* provides to connect them among each other and with the COE. Hence, the platform guarantees a plug-and-play integration of models and simulators, in which one or more models can be easily replaced without affecting the whole simulation engine. Finally, simulators can reside on different network nodes and communicate through TCP/IP sockets allowing distributed and parallelised computing to enhance co-simulation performances.

5.1.2 Applications and Results

The flexibility provided by the pure software co-simulation platform permits to use it as a virtual test bed for complex MES, as well as conducting experimental

research on enabling technologies. To demonstrate the capability of the presented platform in integrating and synchronising heterogeneous models written in different programming languages and executed with different software in a plug-and-play fashion, a well-consolidated and feasible Smart MES Building scenario was designed and implemented. Indeed, it is out of the scope of the present research to propose new domain-specific models or scenarios since the focus of the dissertation is on the co-simulation infrastructure by showing the effective and simplified use of the platform layers to realise complex scenarios. This section provides a full description of the design of a Smart MES Building scenario and the simulators implemented.

Figure 5.4 illustrates the scheme of simulation blocks, focusing on blocks related to the cyber-physical elements of the system, and the energy and data flow among them. A simulation block represents an encapsulated entity of the system under analysis, which contains the standalone model that is able to interact with the environment exchanging data, parameters, and state variables dynamically. As illustrated in Figure 5.2, the simulation blocks are added, parameterised and connected through the YAML configuration files provided by the domain experts and finally managed by *Scenario Builder* that instances the models and composes the overall scenario automatically.

The scenario consists of an EnergyPlus building model plugged into the platform through a FMU, and linked to a Modelica-based Electric Heat Pump model with a control system model, encapsulated in FMU as well. Another FMU provides a battery system modelled in MATLAB Simulink. Furthermore, a PV system, household behaviour, and weather data are provided to the building by linking stand-alone Python simulators. In addition to the simulation blocks illustrated in Figure 5.4, the *Scenario Builder* automatically adds and connects the back-end blocks as described in the previous section.

i) Solar & Local Weather. Weather data are integrated from third-party data sources, such as Weather Underground [117], through an integrated Python application interface. As shown in Figure 5.4, the application retrieves the information at the time step required by the models and distributes them in run-time through Mosaik which manages the time synchronisation. If the minimum available time step of the weather data sources is greater than the time step required by a model, a linear interpolation is performed.

ii) PV system. The PV system was modelled by integrating the simulation infrastructure presented in [118]. The infrastructure allows to estimate the PV potential and to simulate the solar radiation profiles in real-sky conditions with a high spatio-temporal resolution (depending on the resolutions of the input data). It uses as inputs: *(a)* Geographic Information System (GIS) data to describe building rooftops in terms of slope, orientation, possible obstacles, and shadows; *(b)* weather data provided by *Solar & Local Weather* block such as the Global Horizontal Irradiance (GHI) in real-sky condition *GHI* and the outdoor Dry-Bulb Temperature T_{DryBul}. The PV simulation can be performed with the same time step as the resolution of the *GHI* data. The on-site generated electricity is primarily self-consumed by the household while any surplus is used to charge the battery or sent to the grid. The last case happens if the battery is already fully charged.

iii) Household behaviour. The household behaviour was integrated into the co-simulation platform by using the Python simulator proposed in [119]. The model

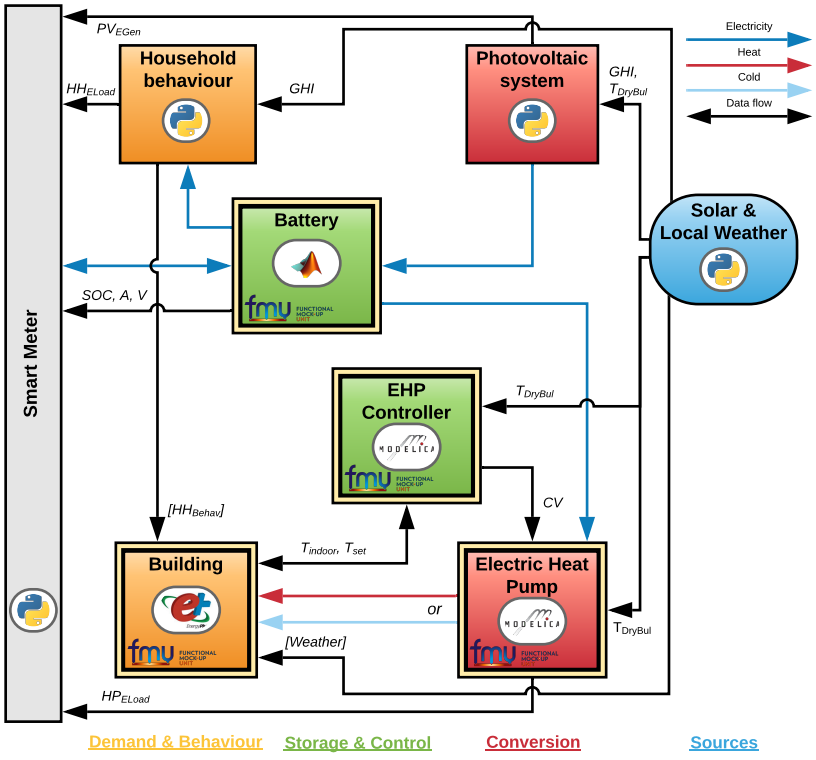


Fig. 5.4 Block diagram of the scenario designed within the co-simulation platform. The diagram shows both the energy- and data-flow connections between simulation blocks.

uses a different kind of input data to create a non-homogeneous semi-Markov model for simulating the household electricity behaviour and thermal gains and retrieving the aggregated electricity and thermal load profiles. The Household behaviour simulator performs activity simulation with a resolution of 10 minutes, and it can provide appliances and aggregate load profiles with a time step of one second up to ad lib.

The model creates the household by specifying the composition of the family, starting from census data. Then, the set of appliances in the households is distributed according to statistics obtained from *Use of Energy* surveys. Whilst, the statistics obtained from the *Time of Use* surveys were used to create a Semi-Markov model and generate each person's behaviour. Finally, the simulator uses the created Semi-Markov model to generate household behaviour in terms of occupancy, type and duration of each activity performed by the household's inhabitants, which is associated with specific usage of electric appliances (e.g., washing machine, dish-washer, vacuum cleaner, fridge, etc.). The simulator also uses weather data, provided by *Solar & Local Weather* block, to compute the energy consumption of domestic lighting systems according to solar radiation *GHI*.

In the end, the Household behaviour parses the number of occupants in a zone and their interactions with lights and appliances, providing aggregated loads, appliances and lights loads and schedules, which are given as input vector (HH_{Behav}) to the *Building* block.

iv) Electric Heat Pump (EHP). The air-to-water EHP has been developed using Modelica's standard components by using the open-source OpenModelica modelling and simulation environment. The EHP model was exported as a FMU for co-simulation that contains the model and exposes inputs and outputs, as shown in Figure 5.4. Moreover, the numerical solver is embedded and supplied by OpenModelica. Finally, the EHP model was interfaced through the FMI/Mosaik adapter.

The EHP model computes the sensible heat gain required to maintain the set-point temperature T_{set} in rooms. The EHP needs as input the regulation of the actuator provided by the EHP Controller. The coefficient of performance of EHP is set by a parametric relationship with the outdoor Dry-Bulb Temperature T_{DryBul} and the outlet flow temperature of the water-based underfloor heating system. The most implemented regulation system for the outlet flow temperature is the use of a climatic curve, which sets the temperature based on the outdoor temperature T_{DryBul} through

a piece-wises linear function. The output of the EHP FMU is the heat requested by the *Building* block through the heating system. The measured variable T_{indoor} provided by the *Building* block is controlled to maintain the desired set-point T_{set} by implementing a Proportional-Integral-Derivative (PID) controller that acts on the water mass flow rate of the heating system through regulation of the control valve actuator (CV). PID is a negative feedback closed-loop control system that calculates the error value as the difference between the desired set-point T_{set} and the measured value T_{indoor} and applies a correction based on proportional, integral and derivative terms on the CV actuator to regulate water mass flow rate and minimise the error over time. It is worth noting that thanks to the platform's flexibility, it is possible to replace the control system in a plug-and-play fashion with more advanced control systems.

v) Building. The building was modelled in EnergyPlus, a well-known open-source detailed building energy modelling engine that performs calculations of energy consumption in buildings, such as heating and cooling loads, disaggregated energy end-uses, and many other building-related features. Furthermore, EnergyPlus allows exporting the IDF file building model as a FMU for co-simulation through the Python package *EnergyPlusToFMU*. It is worth noting that EnergyPlus is a well-established energy simulation engine used as the core engine of many commercial and non-commercial energy simulation software, such as OpenStudio and DesignBuilder. These solutions provide a graphical interface and additional tools to improve the building's design and complexity, such as the import of CAD geometry, 3D model, or the direct import of building models from Building Information Modelling (BIM) tools. Therefore, the potential provided by EnergyPlus and its extensions combined with the possibility of exporting the building model as a FMU unlocks a perfect integration within the co-simulation platform, allowing flexibility and composability of building models with different levels of complexity and design by following the modelist choices and the scenario objectives.

Inputs and outputs of the Building FMU are managed by EnergyPlus through three types of *External Interface* objects being on the IDF file: *i)* the vector of the Household behaviour variables (HH_{Behav}) and heat gain provided by EHP for the water-based underfloor heating system were interfaced as input schedules, *ii)* the indoor temperature T_{indoor} and set-point temperature T_{set} were linked to external interface as output variables, and *iii)* the weather data needed by EnergyPlus (i.e., Dry-Bulb Temperature, Dew-Point Temperature, Relative Humidity, Barometric

Pressure, Direct Normal Radiation, Diffuse Horizontal Radiation, Total and Opaque Sky Cover, Wind Direction, Wind Speed) were interfaced as input actuators passing a vector data (*Weather*). EnergyPlus can perform simulations with a minimum time step of one minute up to one hour.

vi) Battery. The electrical energy storage system was modelled in MATLAB Simulink by using the *generic battery model* provided by the Simscape Library, and it was encapsulated into a FMU for co-simulation. The model describes the dynamic behaviour of the most popular types of rechargeable batteries, and it can be fully parameterised using a commercial battery data sheet. The main parameters are the type of battery (e.g., Li-ion and lead-acid), nominal voltage, and rated capacity. The other detailed parameters are derived from the discharge characteristics and they are already implemented in the Simulink model for the common types of batteries.

The battery model was electrically connected with the production unit and the loads. A simplified controller manages the charge and discharge of the battery under the depth of discharge limit, prioritising self-consumption and, eventually, charging the battery only from the surplus of PV production. The main state variables of the battery (SOC, A and V) are sent to the *Smart Meter* to manage the energy flux of the entire system. For simplicity, in this scenario, it was neglected the temperature and aging effects on the battery. However, based on the capability of the battery model, it is possible to simulate these effects by interfacing it with the other blocks of the scenario, e.g., the *Solar & Local Weather* or the *Building* block to provide the ambient temperature to the battery. The present battery model is set to simulate the charge and discharge of the battery with a resolution of 10 minutes.

vii) Smart Meter. The virtual Smart Meter provides the physical and data interface between the building system and the distribution network. It receives the PV electrical generation (PV_{EGen}), the aggregated household electrical load (HH_{ELoad}), the EHP electrical consumption (HP_{ELoad}), and main state variables of the battery (SOC, A and V). It was used as a data collector manager returning the simulation results either in run-time or at the end of the simulation. The smart meter simulator can perform the simulation with whatever time step resolution without any limitation. Moreover, the smart meter model is equipped with the Message Queuing Telemetry Transport (MQTT) publish/subscribe protocol in order to transmit to external smart metering infrastructures the measured values.

Smart MES Building Simulation and Results

In order to test the capability and usability of the pure software co-simulation infrastructure, the scenario designed in Section 5.1.2 was simulated for a hypothetical and realistic house located in Turin, Italy. The scenario is a test case to evaluate the energy performance of a complex building energy system through the analysis of its dynamics and operation. Therefore, the YAML configuration file (see the schema in Figure 5.3) was filled with all data required to set up a co-simulation environment, models, and connections among them as described in the following.

Our test case consists of a *Building* of about 150 m^2 equipped with a water-based underfloor heating system. It was modelled with the OpenStudio suite that supports whole building energy modelling using EnergyPlus engine. The software includes a SketchUp plug-in to graphically create the geometry needed for EnergyPlus, as depicted in Figure 5.5.

The geometry, materials, and construction layers of the building model have been chosen to be representative of a single-family house in Northern Italy. The main geometric and construction data of the building envelope are shown in Table 5.1.

The *Solar & Local Weather* block provides weather data to the sub-models of the scenario, giving the location and the reference year.

Before starting the co-simulation, the *Household behaviour* block generates the family, starting from the local socio-demographic and energy-related data. It consists of four members: a full-time man worker, a housewife, and two student kids.

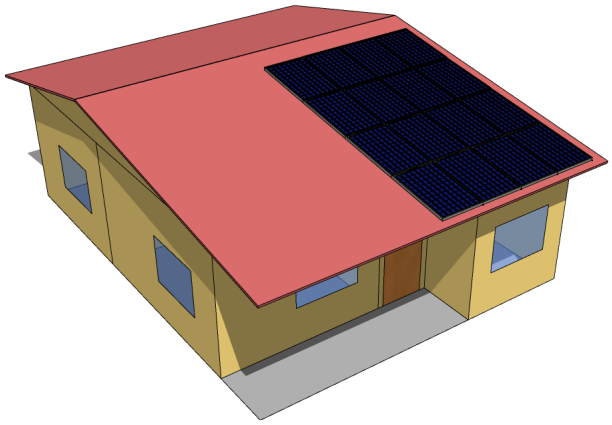


Fig. 5.5 The 3D layout of the house modelled with the OpenStudio suite and SketchUp plug-in.

Quantity	Unit	Value
Conditioned net floor area	m^2	136.82
Conditioned gross volume	m^3	412
Gross Wall Area	m^2	134.64
Gross Window-Wall Ratio	-	0.26
U-value external wall	$Wm^{-2}K^{-1}$	0.427
U-value floor (roof)	$Wm^{-2}K^{-1}$	1.260
U-value window	$Wm^{-2}K^{-1}$	2.674

Table 5.1 Main geometric and construction data of the building envelope.

The PV system is composed of 16 panels on the roof-top south oriented with a 37° tilt angle. It provides a total of 5 kW_p to the premises.

An EHP is installed in order to satisfy the heating demand of the house with a power input of 3 kW_e and a COP of 4 at nominal conditions. The heat pump is equipped with a PID controller and an inertial flywheel to decouple the heat generator with the underfloor heating system. The performance characteristics were retrieved from data sheets of common residential heat pumps. In the winter season, the desired indoor temperature was set to 20 °C[1] from 6:00 to 22:00, and 16 °C for the remaining day's hours.

A Lithium-Ion battery is installed with a rated capacity of 60 Ah and a nominal voltage of 200 V and is located inside the house. The parameters of the discharge characteristics were derived from the built-in Li-Ion battery Simulink model.

The *Scenario Builder* module parses the YAML configuration file to retrieve all the required information to perform the scenario simulation and distributes them to COE through Mosaik API. The simulation blocks are instantiated using the Mosaik *Simulator API* (retrieving model settings, parameters, constants, and time-step) and connected to each other, as shown in Figure 5.4, via Mosaik *Scenario API*. The time-steps Δt were set considering the scenario characteristics, the capability of the solvers, and computational effort: EnergyPlus (i.e. the Building) 10 *min*, Modelica (i.e. the Electric Heat Pump) 5 *min*, MATLAB Simulink (i.e. the Battery) 10 *min*, PV simulator 15 *min* and household behaviour 10 *min*. The *Solar & Local Weather* block provides data to each simulation engine at the requested time step. The scenario

[1]The Italian regulation establishes private homes are not supposed to be heated to more than 20 °C, although the norms allow a margin of 2 °C.

was simulated for a whole thermal season. The distributed co-simulation platform resides in our campus and involves five computers (or nodes) all connected between

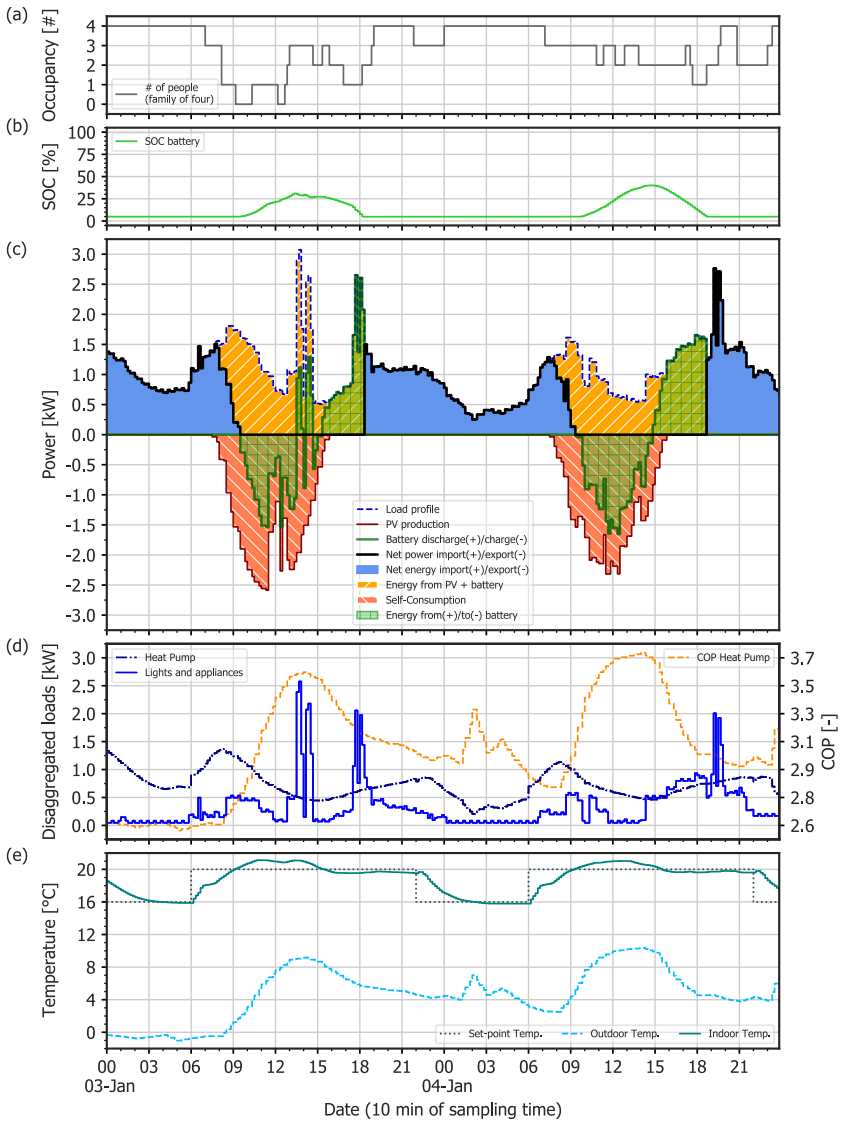


Fig. 5.6 The figure presents the scenario simulation results showing a general time window of two consecutive days i.e., a weekend of January. The time-series plots depict: *(a)* the number of people in the house; *(b)* the State Of Charge (*SOC*) of the battery; *(c)* the profiles of the net power, total load, PV production, and battery, highlighting the self-consumption and the energy covered by the combined PV-battery system; *(d)* the view of load profile in terms of its disaggregated elements, i.e., lights, appliances, and heat pump, the latter linked to its Coefficient Of Performance (*COP*); *(e)* the outdoor, indoor, and scheduled set-point temperatures.

them by using a Local Area Network (LAN). Each network node is an Intel® Xeon® E3-1245v5 Central Processing Unit (CPU)@3.50Ghz with 32GB DDR4@2133MHz

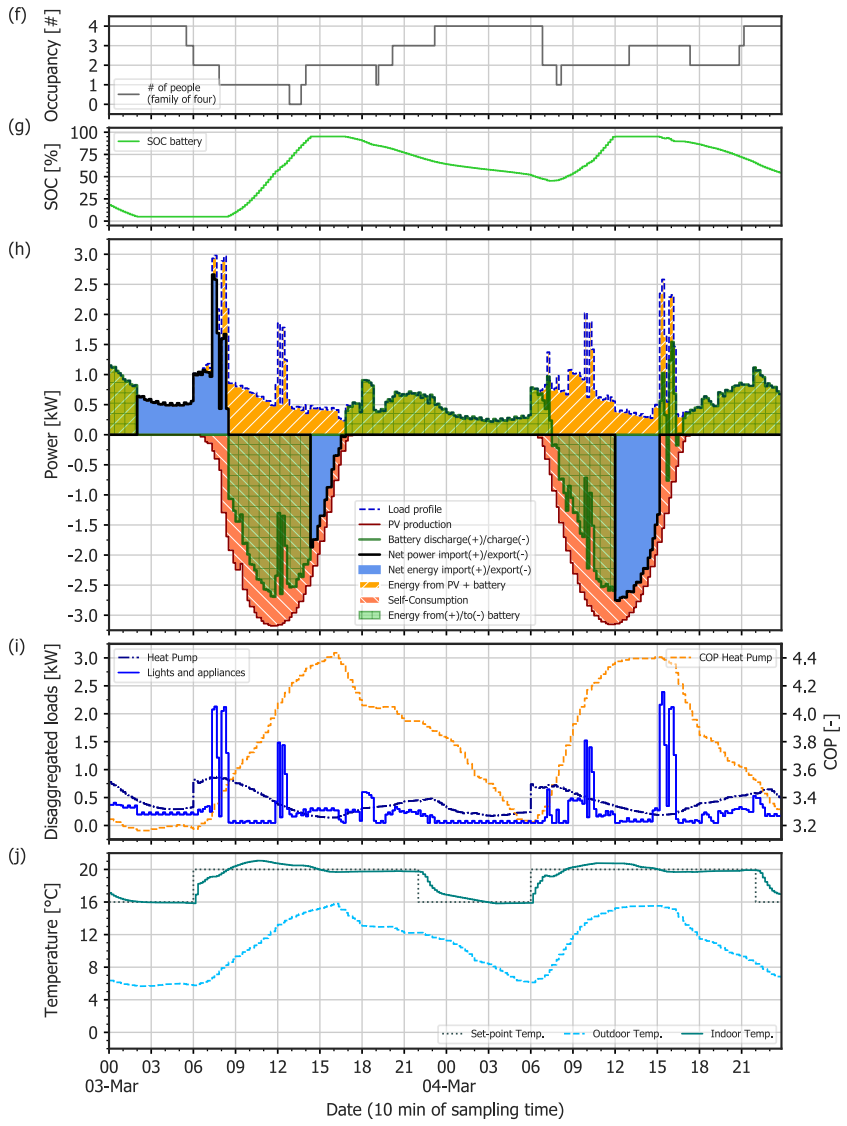


Fig. 5.7 The figure presents the scenario simulation results showing a general time window of two consecutive working days of March. The time-series plots depict: *(f)* the number of people in the house; *(g)* the State Of Charge (*SOC*) of the battery; *(h)* the profiles of the net power, total load, PV production, and battery, highlighting the self-consumption and the energy covered by the combined PV-battery system; *(i)* the view of load profile in terms of its disaggregated elements, i.e., lights, appliances, and heat pump, the latter linked to its Coefficient Of Performance (*COP*); *(j)* the outdoor, indoor, and scheduled set-point temperatures.

RAM. The different simulators and models are distributed across the five computers as follows:

- **Node A** Scenario set-up and Mosaik COE;
- **Node B** Python simulators (i.e. Household behaviour, PV, Smart Meter, and Solar & Local Weather);
- **Node C** EnergyPlus (i.e. Building FMU);
- **Node D** MATLAB Simulink (i.e. Battery FMU);
- **Node E** Modelica (i.e. EHP FMU).

The main results of the simulation are depicted in Figure 5.6 and Figure 5.7. The figure shows a general time window of two consecutive days of two months, i.e. a weekend of January in Figure 5.6(a)-(e) and two working days of March in Figure 5.7(f)-(j). The chosen months represent the colder and hotter month of the thermal season in the given location. The main characterising data for analysing the energy building behaviour are the occupancy, battery profile and State Of Charge (*SOC*), total and disaggregated load profiles, Coefficient Of Performance (*COP*) of EHP, PV production, net power and net energy import/export measured on the meter, self-consumption, energy from/to battery, energy covered from combined PV-battery system and, finally, outdoor and indoor temperatures with the schedule of the desired temperature. The data were collected with a sampling time of 10 minutes. Nevertheless, it can be requested from the co-simulation platform information from either each module that exposes inputs/outputs and parameters or the simulation environment directly at wherever sampling time.

The household behaviour is resumed by the occupancy in Figure 5.6(a) and Figure 5.7(f). Occupancy is one of the most important factors affecting the building energy demand for heating/cooling by varying conditioning periods and house settings, as well as the electricity demand by the use of the EHP itself, lights, and appliances as shown in the disaggregated loads plot in Figure 5.6(d) and Figure 5.7(i). As shown in Figure 5.6(e) and Figure 5.7(j), the set-point temperature's schedule affects the thermal demand from the EHP and, as a consequence, its electrical consumption. Moreover, the electric load requested by the EHP to satisfy the heating demand is considerably higher than the electrical household consumption due to

appliances and lights, especially on colder days. Therefore, the resulting aggregated load profile, shown in Figure 5.6(c) and Figure 5.7(h), almost follows the EHP load profile. For example, on the weekend of January during the morning from 6:00 to 9:00, the EHP consumption increases until it reach a maximum power absorption of about 1.4 kW to cover the increased heating demand due to the change in indoor set-point temperature. Meanwhile, people leave the house and, as a consequence, light and appliance consumption becomes minimum and does not exceed 0.5 kW of absorption. However, on the same morning hours during the working days of March, it takes place different power peaks of about 2 kW caused by more use of appliances and lights.

By comparing the curves of EHP load and its *COP* in Figure 5.6(d) and Figure 5.7(i), the EHP consumption is almost higher when the *COP* drops and vice versa. An example can be seen on the afternoons of January, where the EHP absorbs about 0.5 kW while its *COP* reaches about 3.6. More noticeable on March, where *COP* reaches 4.4 while the EHP absorbs only 0.25 kW. This typical behaviour is linked to the outlet flow temperature of the EHP that is regulated by a climatic curve. Indeed it is regulated considering the outdoor temperature. As a consequence, the *COP* qualitatively follows the outdoor temperature.

During winter days, the outdoor temperature can quickly swing, as shown in the temperatures plot in Figure 5.6(e) and Figure 5.7(j), even reaching temperatures below zero as in January. The requested heating demand is satisfied by the EHP and, thanks to the PID control strategy, the room temperature remains around the set-point temperatures with few little oscillations. The PID controller was tuned during run-time simulation till reaching optimum values and stability for the desired control response. Indeed, the tunable parameters can be changed during execution through *Data I/O* interface, thus directly seeing the feedback results on simulation.

Following the power-related plot in Figure 5.6(c) and Figure 5.7(h), we can see how the combined PV-battery system affects the load profile of the building measured on the meter (net power). The energy management system rules the energy flows by maximising self-consumption: the in-site electricity production is directly used to cover the load reducing the imported power from the grid, and any surplus is used to charge the battery whenever it is not full. Otherwise, the power surplus is exported to the grid. The load profile and PV production curves show the timing imbalance between peak demand and renewable energy production. The battery helps

to reduce the timing imbalance by shifting the load: it is charged during daylight by solar power, then it is discharged, providing power into the premises in the evening, thus reducing imported power and increasing self-consumption. The Figure 5.6(b) and Figure 5.7(g) show the *SOC* curve during the analysed days, which are strictly correlated with the charging and discharging phases depicted in Figure 5.6(c) and Figure 5.7(h). As can be seen in the figures, a change of sign on the first derivative of the *SOC* curve from positive to negative is equivalent to a change of phase of the battery from power charging to power discharging, and vice versa. When the first derivative is null, corresponding to *SOC* equal to the upper (95%) or lower limit (5%), the battery power is null.

During the days of March, the combination of good solar radiation and low EHP load results in high levels of *SOC* up to full charge, thus increasing the opportunity to sell energy to the grid. Moreover, the storage capacity of the battery permits it to cover the entire evening and night loads. As an example, on March 3^{rd} following Figure 5.7(g)-(h), the daily PV production was 19.91 kWh: about 17.68 kWh was self-consumed of which 5.96 kWh directly used to cover the load and 11.72 kWh used to charge the battery, while the remaining 2.23 kWh was sold to the grid. In particular, the battery starts charging at 8:30 until reaching the upper charge limit at 14:00 when the power surplus begins to be sold until the end of PV production. From 18:00 until the next beginning of the daily PV production, the battery succeeds in covering the entire load, even remaining with a *SOC* of about 50%. On the other hand, during the colder and cloudy days of January resulting in less solar radiation, the power export rarely happens and the battery works at low operating levels. In any case, the exploitation of the accumulated capacity permits it to cover a significant part of the evening load. For instance, on January 4^{th} following Figure 5.6(b)-(c), the daily PV production is entirely self-consumed. A part of this energy was exploited to charge the empty battery up to about 40% of *SOC* and later used to cover about 4.54 kWh of the load during the afternoon until 19:00.

The pure software co-simulation platform simulates the complexity of the Smart MES Building dynamics with great details and realism by integrating each subsystem with its solver's flexibility and efficiency. Notably, as depicted in Figure 5.6(c) and Figure 5.7(h), the daily PV production profiles of the two months analysed are affected by the real-sky weather conditions resulting in markedly different. During the two days analysed in January, the power peaks of PV do not exceed 2.6 kW and their profiles result jagged due to the cold and cloudy days of the winter, with a total

in-site electricity production of 22.15 kWh. Going towards spring in March, the PV profiles are more uniform due to warmer and clear days, reaching power peaks of almost 3.5 kW with a total production of 40.38 kWh. On the other hand, the energy management system actively responds to the variability of PV production and load request by charging and discharging the battery.

Testing of a different control strategy

The flexibility and usability of the proposed co-simulation platform permit us to easily implement experiments testing different solutions. Indeed, it is only needed to replace the desired simulator block in a plug-and-play fashion through the *Scenario Builder* module. The module sees the simulator as a grey-box model, thus it will only be necessary to re-link the input/output variables with the other models and set the internal parameters of the new integrated block through the YAML configuration file.

By way of example, the PID controller of the EHP was replaced with a more simpler hysteresis controller that turns ON/OFF the heating system if the T_{indoor} exceeds -2/+2 $^{\circ}C$ with respect to T_{set}. The indoor temperature trends and EHP loads obtained from the simulation of the two control strategies were compared and depicted in Figure 5.8 during the days of March. As it can be seen, the PID controller regulates the indoor temperature better than the hysteresis one, hence reducing the discomfort by maintaining on average the temperature closer to the chosen set-point. From an energy point of view, the EHP regulated with the PID controller performs better than the hysteresis controller by reducing its energy consumption by about 33%. Indeed, during the two days of March, EHP with PID controller consumes 53.92 kWh, while EHP with hysteresis controller consumes 79.77 kWh.

In the end, there is no limitation on implementing more advanced control strategies as their usage is strictly dependent on the use case analysed only. For example, it is possible to implement more advanced control strategies and optimisation algorithms that, for instance, are based on Model Predictive Control or deep Reinforcement Learning. For example, a general control block can be implemented as follow: the emulator could be a detailed EnergyPlus model or even a HIL model of a real building that could share information through a TCP/IP connection with the possibility to control the heating system within the co-simulation platform remotely; to perform the predictions, a validated dynamic model based on reduced order RC

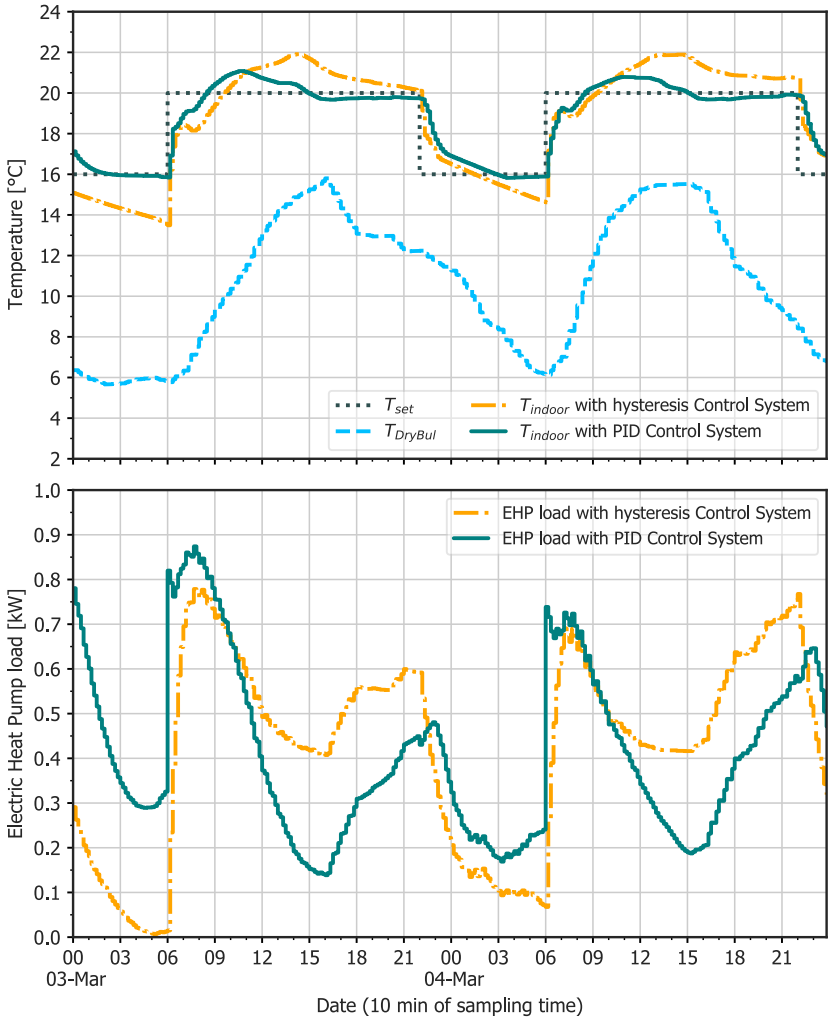


Fig. 5.8 Comparison of the indoor temperatures trends and EHP loads by implementing different control strategies: PID and hysteresis Control Systems.

model can be implemented directly in Python or by using dedicated libraries, which can be easily integrated into the co-simulation platform by implementing the Mosaik *Simulator API*; the optimisation algorithms can be implemented by exploiting plenty of different libraries as well. Finally, the connection of the various elements of the control block with the rest of the system can be made through the *Scenario Builder* module, similarly to what was made with the other simulation blocks. In the same way, the flexibility and usability of the proposed co-simulation platform also permit to effectively integrate algorithms that are able to analyse the system more in-depth, for example, to find optimal control and operating strategies concerning a pre-defined

objective function, e.g., maximising the total self-consumption, which increases the savings by buying less energy from the grid. Regarding the system plants' sizing, in this work, the cost-benefits analysis of the optimal PV and battery sizes was out of the scope. However, the co-simulation platform allows to find the optimal sizes of the subsystems performing optimisation by simulation, or directly using advanced optimisation tools by exploiting the capability of the platform to perform a multi-modelling simulation. In conclusion, the simplified scenario composition and implementation procedures demonstrate the usability of the co-simulation platform, where different disciplinary fields and tools can be integrated by sharing common knowledge to build even more complex energy scenarios.

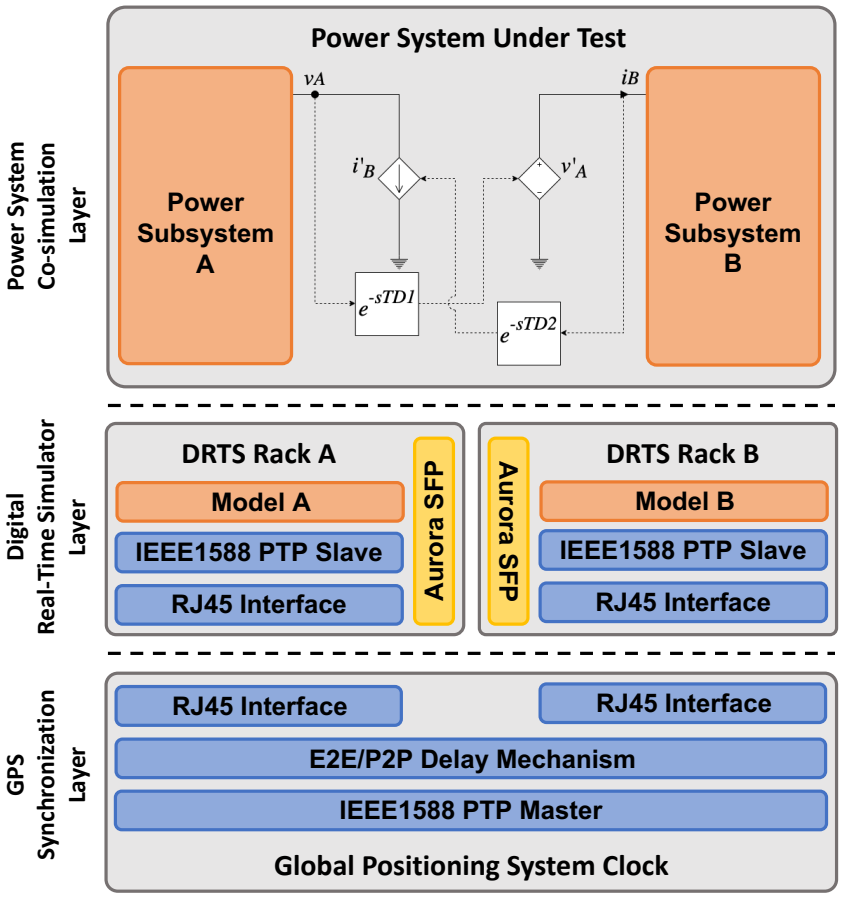


Fig. 5.9 The Digital Real-Time Co-simulation Infrastructure and its three main architectural layers.

5.2 Digital Real-Time Co-simulation

The Digital Real-Time Co-simulation Infrastructure lays the foundations of the hybrid co-simulation infrastructure to allow a point-to-point interconnection of different commercial DRTS (e.g. OPAL-RT and RTDS Technologies) by exploiting Aurora 8B/10B, a serial protocol to exchange data by employing an optical fiber medium, and IEEE1588 PTP, that is commonly used to synchronize internal reference clocks throughout a computer network. Aurora 8B/10B reduces the effects of uncertainty brought about by communication latency by adhering to real-time constraints in a co-simulated environment. IEEE1588 PTP protocol instead permits to align co-simulation execution and logging of synchronized results coming from the different DRTS.

5.2.1 Digital Real-Time Co-simulation Infrastructure

The architectural description of the proposed infrastructure is presented in Figure 5.9 and consists of three main layers: *i)* the *GPS Synchronization Layer*, *ii)* the *Digital Real-Time Simulator (DRTS) Layer*, and *iii)* the *Power System Co-simulation Layer*. The rest of this section presents every single layer and its main components and the laboratory setups that are used to test the infrastructure functionalities.

GPS Synchronization Layer

This layer ensures the overall infrastructure time synchronization by exploiting a GPS clock. In fact, the GPS is a global navigation satellite system that not only provides geolocation but also time information and synchronization by broadcasting a time reference clock aligned to the GPS atomic clock technology that resides in each GPS satellite. The GPS clock is a piece of equipment that receives the GPS signal and aligns its internal reference clock to the atomic time without needing a local atomic clock, ensuring a global synchronization of the laboratory with other laboratories worldwide. Once synchronized with the GPS, this equipment offers different local synchronization protocols, such as IRIG-B, 1PPS, and the IEEE1588 Precision Time Protocol (PTP).

Our infrastructure exploits the IEEE1588 PTP to synchronize both internal reference clocks of the interconnected DRTS. By exploiting the PTP stack functionalities, the GPS clock sets its operation in master mode to control other slaves by broadcasting PTP packets on a LAN. The end-user could choose among a set of standard PTP profiles (e.g. Default profile) that serve different final uses. Moreover, the PTP stack offers two different delay mechanisms, namely, *i)* the End-to-End (E2E) and *ii)* the Peer-to-Peer (P2P), that provides PTP functionalities to the two homonymous types of transparent clocks. The E2E mechanism allows the calculation of the overall latency among the master and slave nodes ensuring the fastest synchronization with a low precision. Moreover, this delay mechanism ensures the most interoperable version of the PTP standard, reducing the limitations on hardware that can be used in the PTP network. The P2P delay mechanism instead allows a fine-grained and precise latency calculation of each hop of the network path between the master and the slave node. However, it reduces the interoperability of different hardware since they must implement the overall PTP standard protocol stack.

Furthermore, PTP offers two different protocols to allow communication among master and slave nodes: i) Layer 2 (i.e. IEEE802.3 or Ethernet) that ensures a fast synchronization in the same network segment, and iii) Layer 3 (i.e. IP) that allows the time synchronization of wider networks. Depending on the LAN configuration of the laboratory, the end-user could choose among the two layers. Apart from this configuration, the GPS clock is interconnected to the LAN using two RJ45 Ethernet interfaces, one serving the E2E, and the other the P2P delay mechanism.

DRTS Layer

This is the core layer of the proposed infrastructure. It allows the interconnection among two commercial DRTS exploiting a bi-directional point-to-point physical interconnection. Our infrastructure exploits Aurora 8B/10B, the fastest protocol on board of commercial DRTSs. Aurora is a link-layer protocol that exchanges serial streams through a point-to-point standard full-duplex multi-mode optical fiber link ensuring a high bandwidth from 2 Gbps to 5 Gbps to exchange data with a lightweight packet overhead. Aurora datagram is completely configured by the end-user as a variable sequence of 32-bit integer and float. This protocol ensures the lowest latency of communication between the set of protocols that are commonly used for DRTS interconnection (e.g. Ethernet). To enable the Aurora protocol, each DRTS must occupy one of the available Aurora Small-form Factor Pluggable (SFP) ports for the data exchange.

Each DRTS is also equipped with a IEEE1588 PTP synchronization board since the time synchronization among the DRTSs involved in the proposed infrastructure is ensured by the IEEE1588 PTP. PTP boards are set as slave-only PTP nodes. Moreover, they are interconnected with the GPS synchronization layer with an RJ45 Ethernet cable to receive PTP packets coming from the PTP master node (i.e. GPS clock) and align the DRTS internal reference clock to the reference master clock. Finally, each interconnected DRTS implements its own logic to fulfil the real-time simulation of the assigned power subsystem by including simulation blocks that enables *i)* the Aurora link and *ii)* the PTP synchronization in the compiled models.

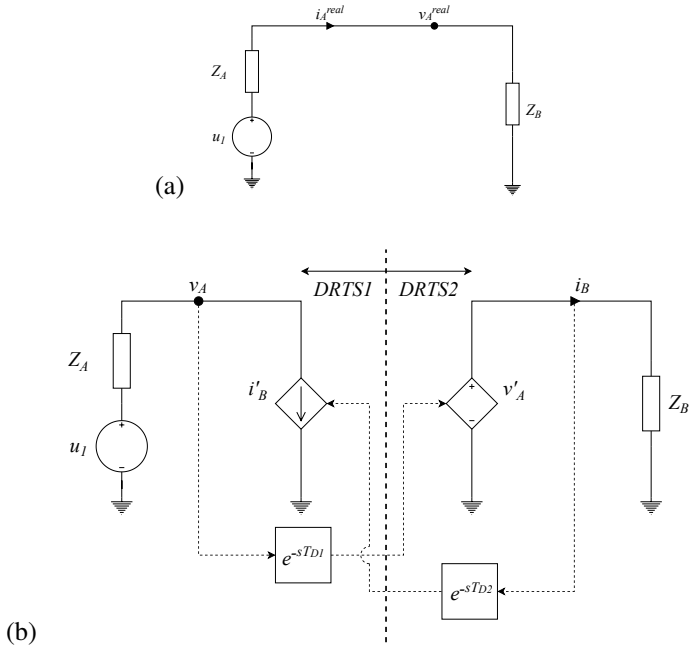


Fig. 5.10 Monolithic electric circuit (a) composed by an AC voltage source u_1 and two impedances Z_A and Z_B and the application of ITM IA (b)

Power System Co-simulation Layer

This layers enable the logical co-simulation of a power system by splitting the PSUT through the Ideal Transformer Method (ITM) Interface Algorithm (IA), commonly used in PHIL application to interconnect a physical piece of equipment, so-called DUT, to the simulated ROS. This IA is the simplest way to set up a PHIL system and it has been chosen to simply and efficiently split a PSUT into a source and load power subsystem modelled inside each DRTS. The ITM IA described in Figure 5.10b exploits a controlled voltage generator in the load part of the power subsystem that reproduces the voltage v_A measured in the source part (i.e. v'_A), and a current generator in the source part of the power subsystem to reproduce the current i_B measured in the load part (i.e. i'_B). Moreover, it applies a latency that is proportional to the latency experimented by the exchanged variable from DRTS rack 1 and rack 2 to take effect on the load part of the circuit (i.e. T_{D_1}) and vice versa (i.e. T_{D_2}).

In Figure 5.11, the equivalent block diagram of the ITM circuit could lead us to its frequency-domain stability analysis. Exploiting the ITM open-loop function described by Equation 5.1, the Nyquist diagram is calculated for $Z_A = 50\ \Omega$ and different values of Z_B, namely $50, 100, 200, 300, 400$, and $500\,\Omega$. Following the Nyquist

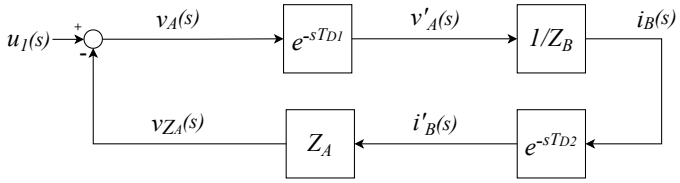


Fig. 5.11 Equivalent Block Diagram of the ITM IA

criterion, the Nyquist diagram of the open-loop function of the ITM system must not encircle the critical point $(-1,0)$ to ensure stability. As depicted in Figure 5.12, the ratio Z_A/Z_B must be minor than 1 to ensure the Nyquist criterion. Also if the stability is ensured by this criterion, a large latency of the paths T_{D_1} and T_{D_2} could provoke nonlinearities (i.e. phase shift) that impact both frequency-domain and time-domain accuracy of the overall system.

$$G_{ol} = \frac{Z_A}{Z_B} e^{-s(T_{D_1}+T_{D_2})} \tag{5.1}$$

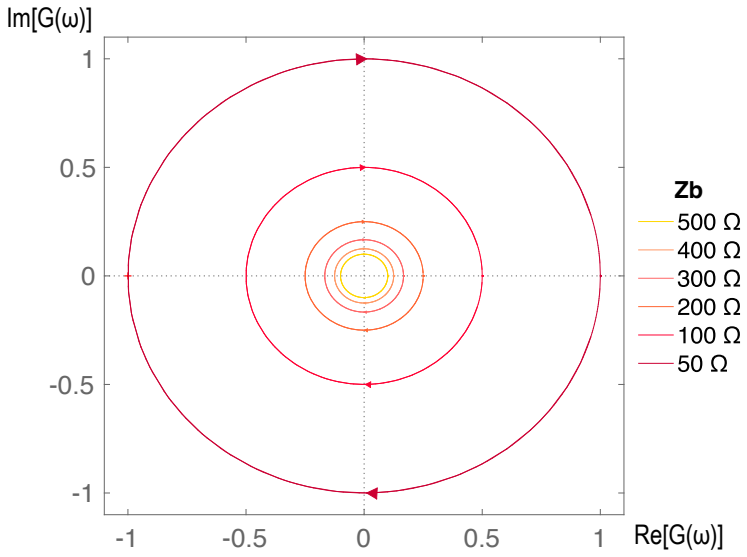


Fig. 5.12 Nyquist diagrams of the open-loop transfer function G_{ol}

Laboratory Setups

The proposed laboratory setups reduce significantly the latency between two DRTS chosen among the available commercial solutions for electric grid analysis, i.e.

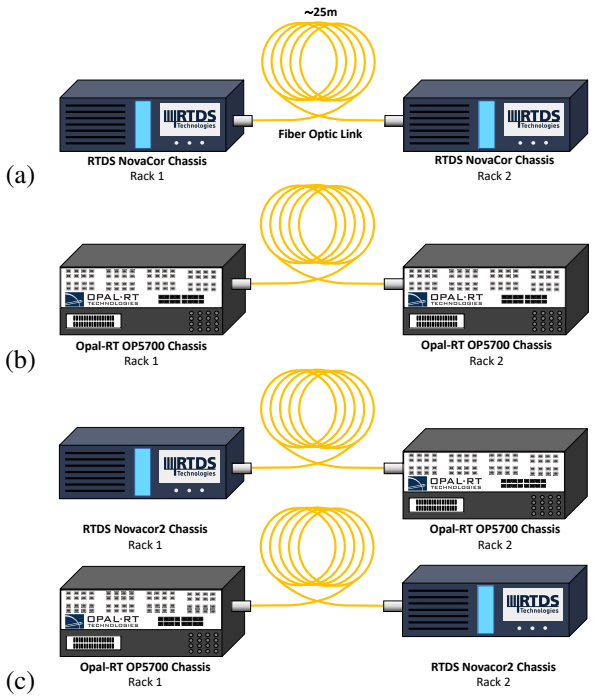


Fig. 5.13 Different configuration of the Digital Real-Time Co-simulation Layer: (a) RTDS-RTDS, (b) OPAL-OPAL, and (c) RTDS-OPAL in both directions.

OPAL-RT OP5700 (OPAL) and RTDS Technologies NovaCor (RTDS). Following the architectural description in Figure 5.9, the GPS Synchronization Layer is managed by the Meinberg microSync HR102HQ GPS clock with an IEEE 1588-2008 v2 Default Layer 2 profile. This GPS clock offers four configurable Ethernet interfaces. Since RTDS and OPAL synchronization boards implement two different Transparent Clock (TC), port 2 have been configured to manage End-to-End (E2E) TC and port 3 instead to manage Peer-to-Peer (P2P) TC. Obviously, the Meinberg microSync HR102HQ GPS clock is set as the master node of the IEEE1588 PTP stack, and the two DRTS synchronization boards racks are set as slave mode to ensure a proper setup of the synchronization with the reference master clock.

RTDS racks must be provided with the GTSYNC card capability to manage the interaction with the Meinberg GPS clock. GTSYNC is a peripheral board interconnected with the core rack using an optical fiber link that allows different kinds of synchronisation protocols (e.g. IEEE1588 PTP, 1PPS, IRIG-B) both in slave and master mode. Moreover, the GTSYNC only accept P2P TC configuration of the IEEE1588 PTP stack. So, both RTDS racks' GTSYNC cards are connected with

a RJ45 Ethernet cable to port 3 of the Meinberg GPS clock exploiting an Ethernet switch. To enable the GTSYNC capability in the RTDS, GTSYNC block must be added in the RSCAD draft of the simulated use case. OPAL instead provides the Oregano card that communicate with the core rack through PCIe bus. Like the GTSYNC, the Oregano card allows the same set of synchronization protocols. However, the TC configuration of the IEEE1588 PTP stack must be E2E. So, both OPAL racks' Oregano cards are connected with a RJ45 Ethernet cable to port 2 of the GPS clock using the Oregano Syn1588 Ethernet switch to reduce latency of the PTP E2E packets. The time regulation schema is negligible since both DRTSs evolve their simulation with an independent time-regulating schema that ensures the correct event ordering respecting the proper real-time constraint. Since the proposed circuit to test the interconnection is a simple source-load circuit, the source part will trigger the simulation starting time of the load part. Results from DRTS racks are indeed aligned considering the internal clock time as a reference since the IEEE1588 PTP stack ensures the correct time synchronisation schema.

Since each commercial DRTS solution implements its own numerical solver and implementation of the Aurora 8B/10B, different configurations of the DRTS layer are proposed in Figure 5.13 to take into analysis each possible DRTS combination. Figure 5.13a present two RTDS racks that have been coupled with a 25 meters optical fiber link exploiting the Aurora protocol. Figure 5.13b instead presents two OPAL racks interconnected by the same optical fiber link exploiting Aurora protocol. Finally, an RTDS rack and an OPAL rack have been coupled together as depicted in Figure 5.13c. This interconnection has been analysed in both directions, considering rack 1 as the generation source and rack 2 as the load of the proposed power system co-simulation scenario.

Figure 5.14 presents the sequence operation diagrams of a time step for both RTDS and OPAL are presented to highlight the management of the Aurora Read (RD) and Write (WR) operations. In Figure 5.14a, RTDS executes after each time step starting time an instantaneous WR operation followed by a RD. At the end of the RD, received variables are exchanged with the Control Signal Core that runs its operations in parallel with the Power System Component Cores. At the end of the time step, Control Signal Core and Power System Component Cores exchange both the control signal variable and the network variables (i.e. voltages and currents). The OPAL sequence operation diagram instead is presented in Figure 5.14b. At the very begging of each time step, the FPGA exchanges data received by the Aurora

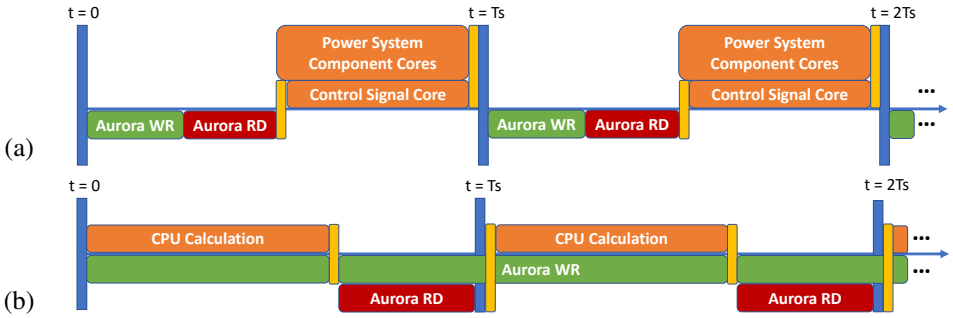


Fig. 5.14 Sequence operation diagrams of the FPGA Aurora Read (red blocks) and Write (green blocks) implementations in RTDS NovaCor (a), and OPAL-RT OP5700 (b), highlighting the data exchange between the FPGA and CPU operations (orange blocks).

RD operation in the previous step with the CPU. Then, the CPU Calculation of the model is executed. After this operation, data are moved from the CPU to the FPGA. For each time step, WR operation is continuously transmitting data each 2.5 µs and updates the values exchanged once the CPU ends the operation of moving data to the FPGA. During the idle time at the end of this exchange, the FPGA also executes an Aurora RD operation that instead terminates at the end of the time step. The Aurora WR operation continues along the next time step by sending the same data received in the previous time step until new data comes at the end of each CPU Calculation operation. This hack implemented in the FPGA bitstream of the OPAL enables faster data transmission in hybrid configuration (see Figure 5.13c).

Aurora could be enabled in RTDS chassis through RSCAD by using the *_rtds_aurora* block, so-called *Aurora block*. This block permits to define the selected SFP transceiver port, the processor number, a priority level of computation, the frame definition (i.e. exchanged environment variables with a maximum width of 128), and the sequence number blocking property of the Aurora link. In OPAL systems instead, the Aurora protocol is enabled through RT-LAB by selecting the proper SFP communication block that defines the FPGA DataIn and DataOut port numbers and enables a variable width of the exchanged signals with a maximum of 255 doubles exchanged. The Aurora link configuration (e.g. SFP transceiver port) instead is defined in the FPGA bitstream, which is uploaded during the scenario loading operations.

Finally, the Power System Co-simulation Layer implements the simple ITM circuit presented in Figure 5.10b by splitting the monolithic circuit (Figure 5.10a)

into a source circuit A and a load circuit B. This simple power system co-simulation scenario allows a precise calculation of the latency among the different configurations of the DRTS layer, and the time-domain and frequency-domain accuracy of the co-simulated results by comparing them with the standalone monolithic results.

5.2.2 Applications and Results

This section presents the experimental results of the ITM IA applied to the proposed Digital Real-Time Co-simulation Infrastructure. The ITM circuit in Figure 5.10b has been implemented in both DRTS (i.e. RTDS NovaCor and OPAL-RT OP5700) by exploiting RSCAD and RT-LAB software. In source part *A* of Figure 5.10b, u_1 is a monophase voltage source with a voltage amplitude of 100 kV peak and a nominal frequency of 50 Hz. The pure resistive impedance Z_A is set to 50 Ω. v_A is retrieved employing a metering point to send its values each time step to load part *B* through the Aurora link. The load part *B* of Figure 5.10b instead receive via the Aurora link the voltage signal v_A that forces the controlled voltage source v'_A to generate a delayed v_A signal. Z_B is set to two different values *i)* to test the ITM near the instability region, and *ii)* to present a stable version of the ITM IA application, namely 50.5 Ω and 500 Ω. To return feedback to the source part *A*, a metering point retrieves i_B current to then send it back through the Aurora link. This value is received by source part *A* and imposed through the controlled current source i'_B. The ITM IA circuit has been tested for all the infrastructure configurations, namely, *i)* RTDS-RTDS, *ii)* OPAL-OPAL, *iii)* RTDS-OPAL, and *iv)* OPAL-RTDS (see Figure 5.13). All the configurations exploit a 25-meter long standard full-duplex multi-mode optical fiber cable as physical media to interconnect both racks.

In the next sections, we discuss the experimental results on: *i)* the *ITM IA Latency Calculation* that describes the time step contributions to the overall latency experimented by the ITM IA circuit for each of the Digital Real-Time Simulation layer configurations, *ii)* the *ITM IA Time-domain Accuracy Analysis* that compares the standalone voltages signals with the co-simulated ITM IA case in the stable region ($Z_B = 500\Omega$) and near the instability region ($Z_B = 50.5\Omega$) for each configuration of the infrastructure, and *iii)* the *ITM IA Frequency-domain Accuracy Analysis* that, similarly to the time-domain case, compares the standalone frequential contents with the co-simulated solution ones for both regions and all infrastructure configurations. For each section, the time step duration T_{Sim} has been changed from 20 µs to 500 µs

to highlight possible dependencies from the time step duration that could influence the overall latency.

ITM IA Latency Calculation

For each infrastructure configuration, the overall latency experimented by the ITM IA circuit solution will be described using its sequence operation diagrams in Figure 5.15. More in-depth, these sequence diagrams describe each single time step contribution highlighting internal operations executed by each DRTS and the data exchange among the interconnected racks. Depending on the infrastructure configuration, these operations will contribute or not to the overall latency represented by $T_{D_1} + T_{D_2}$ in Equation 5.1, generating phase shifts that will impact both time-domain and frequency-domain accuracy of the ITM IA circuit solution.

RTDS-RTDS By exploiting RSCAD software libraries, Aurora blocks are set with the sequence number blocking configuration activated on port 24 for both RTDS racks. The overall latency experimented by the ITM IA application is $5T_{Sim}$ for all T_{Sim} values. The contributions are depicted in the sequence operation diagram in Figure 5.15a. From $t = 0$ to $t = T_{Sim}$, the Power System Component Core (PSCC) of RTDS rack 1 calculates v_A and passes its value to the Control System Core (CSC). From $t = T_{Sim}$ to $t = 2T_{Sim}$, rack 1 retrieves v_A value from the CSC and send it through the Aurora link by executing a write operation (WR). In the same time interval, RTDS rack 2 receives v_A value from the Aurora link with a read operation (RD) and gives it to the CSC that is in charge to send its value to the PSCC at the end of the time interval. From $t = 2T_{Sim}$ to $t = 4T_{Sim}$, rack 2 calculates the current i_B by applying v_A voltage to the controlled voltage source v'_A and, finally, moves i_B from the PSCC to the CSC. These operations take $2T_{Sim}$. From $t = 4T_{Sim}$ to $t = 5T_{Sim}$, rack 2 retrieves the i_B value from the CSC and executes an Aurora WR operation. In the same time interval, rack 1 receives its value by executing an Aurora RD operation and passes its value to the CSC. To conclude, the effect of i_B on v_A calculation requires another T_{Sim} to take effect on source part A of the circuit in Figure 5.10b. The ITM IA latency is calculated from the first v_A calculation in rack 1 at $t = T_{Sim}$ to the effect of i_B to v_A calculation at $t = 6T_{Sim}$, resulting correctly in $5T_{Sim}$.

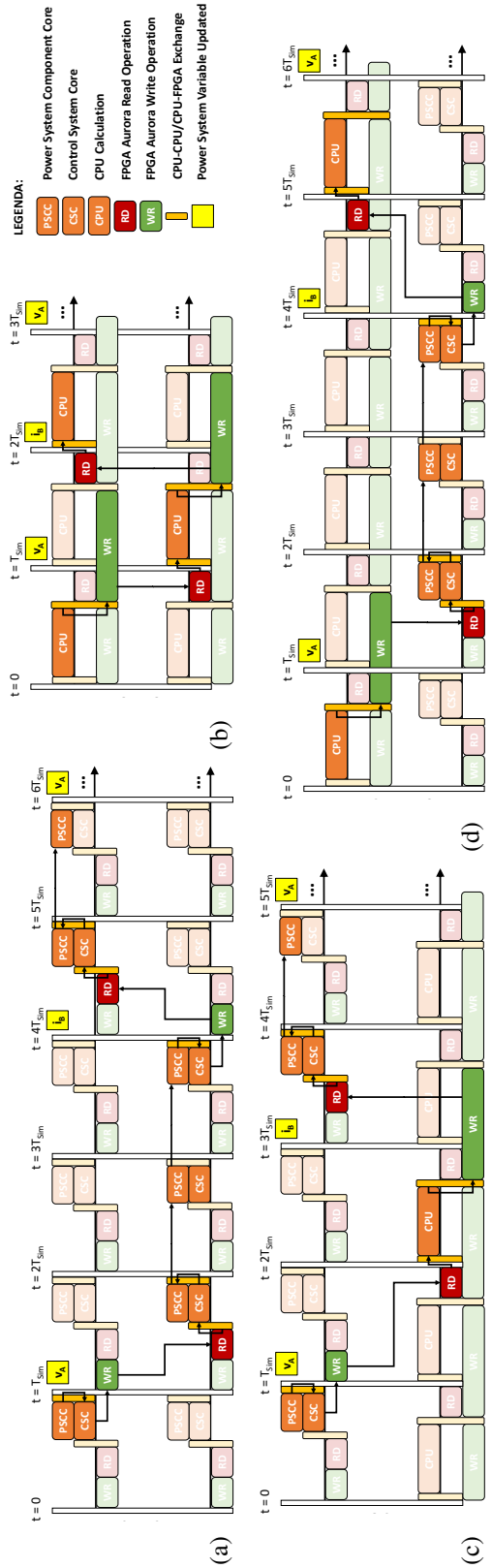


Fig. 5.15 Sequence Operation Diagrams for different Digital Real-Time Co-simulation Infrastructure configuration, namely, RTDS-RTDS (a), OPAL-OPAL (b), RTDS-OPAL (c), and OPAL-RTDS (d).

OPAL-OPAL By exploiting RT-LAB software libraries, the SFP block has been integrated into both models of the ITM circuit. SFP blocks are set on port SFP00 for both OPAL racks. The calculated latency results $2T_{Sim}$ for all T_{Sim} values and its contributions are depicted in Figure 5.15b. From $t = 0$ to $t = T_{Sim}$, the Control Processing Unit (CPU) of the OPAL rack 1 calculates v_A and provides its value to the FPGA that executes the Aurora WR operation. In the same time interval, the FPGA of the OPAL rack 2 receives the v_A value by executing the Aurora RD operation. Right at the beginning of the time interval between $t = T_{Sim}$ and $t = 2T_{Sim}$, rack 2 moves v_A value from the FPGA to the CPU. Subsequently, the CPU of rack 2 executes the calculation to retrieve the current i_B, moves its value to the FPGA, and sends it through the Aurora link with an Aurora WR operation. To conclude this time interval, rack 1 will receive the i_B value and will store it in the FPGA by executing an Aurora RD operation. In the last time interval from $t = 2T_{Sim}$ to $t = 3T_{Sim}$, i_B moves from the FPGA to the CPU to allow the calculation of the effect on v_A on the source circuit. v_A value is updated at the end of this time interval as a result of the circuit numerical solution. So, the ITM IA latency is calculated from the first v_A calculation in rack 1 at $t = T_{Sim}$ to the effect of i_B to v_A calculation at $t = 3T_{Sim}$, resulting correctly in $2T_{Sim}$.

RTDS-OPAL The source circuit of Figure 5.10b in the RSCAD draft and load counterpart in the RT-LAB model of the two previous cases are used in this first hybrid configuration. The physical interconnection of the optical fiber link has been changed from port 23 of the RTDS rack 1 to port SFP01 of the OPAL rack 2. The overall latency is $4T_{Sim}$ for all T_{Sim} values. The contributions are presented in Figure 5.15c. From $t = 0$ to $t = 2T_{Sim}$, RTDS rack 1 executes same operations described in the previous Paragraph 5.2.2, presenting as a result v_A at $t = T_{Sim}$. OPAL rack 2 receive v_A value by executing an Aurora RD operation at the end of the time interval from $t = T_{Sim}$ to $t = 2T_{Sim}$. At the very beginning of the time interval from $t = 2T_{Sim}$ to $t = 3T_{Sim}$, OPAL rack 2 moves v_A value from the FPGA to the CPU and, then, executes i_B calculation. In the same time interval, the current i_B is moved from the CPU to the FPGA to finally execute the Aurora WR operation. Aurora WR lasts till the next time interval from $t = 3T_{Sim}$ to $t = 4T_{Sim}$, where RTDS rack 1 reads i_B current and passes to the CSC its value, concluding the interval by passthrough i_B to the PSCC. Finally, RTDS rack 1 calculates v_A in the last time interval from $t = 4T_{Sim}$ to $t = 5T_{Sim}$. The ITM IA latency results $4T_{Sim}$ by considering from the

first v_A calculation in RTDS rack 1 at $t = T_{Sim}$ to the effect of i_B to v_A calculation at $t = 5T_{Sim}$.

OPAL-RTDS The source circuit in the RT-LAB draft and load circuit in the RSCAD model of the first two cases are used in this second hybrid configuration. The physical interconnection of the optical fiber link has been changed from port SFP00 of OPAL rack 1 to port 24 of RTDS rack 2. The ITM IA latency results $5T_{Sim}$ for all T_{Sim} values. As in previous cases, contributions are described by presenting the configuration's sequence operation diagram in Figure 5.15c. From $t = 0$ to $t = T_{Sim}$, OPAL rack 1 executes the same operations described in the previous Paragraph 5.2.2. However, RTDS rack 2 reads v_A value from the Aurora link in the next time interval from $t = T_{Sim}$ to $t = 2T_{Sim}$. From $t = T_{Sim}$ to $t = 5T_{Sim}$, RTDS rack 2 executes the same operations of the previous Paragraph 5.2.2. So, OPAL rack 1 retrieves the i_B value at the end of the time interval from $t = 4T_{Sim}$ to $t = 5T_{Sim}$. In the last interval, the CPU in OPAL rack 1 receives from the FPGA i_B value, calculates the v_A effect, and, finally, presents its value as a result. The overall experimented latency is $5T_{Sim}$, considering from the first v_A calculation in OPAL rack 1 to the effect of i_B to v_A calculation at $t = 6T_{Sim}$.

ITM IA Time-domain Accuracy Analysis

The time-domain analysis demonstrates that our infrastructure obtains good accuracy of the co-simulated solution. We compared the results of our distributed co-simulation system with a monolithic solution that runs the standalone circuit in Figure 5.10a. The results presented in this subsection are obtained only for a worst-case scenario, that is when T_{Sim} is set to 500 µs. The worst-case scenario is considered as a limit case since normal EMT analysis for electric grid scenario exploits a time step duration of around 50 µs. The monolithic electric circuit in Figure 5.10a has been run simultaneously to the ITM IA case to retrieve the standalone voltage and current, namely v_A^{real} and i_A^{real}. The results in Figure 5.16 are presented only for voltages v_A^{real} (blu line), v_A (green line), and v_B (orange line) to avoid repetition, as the power factor of a purely resistive circuit is 1 and currents and voltages are in phase. Voltages v_A^{real}, v_A, and v_B have been analysed for the two Z_B values that represent the stable ITM case (i.e. $Z_B = 500\,\Omega$) and near the instability region of the ITM circuit (i.e. $Z_B = 50.5\,\Omega$). Finally, the instability transient generated by case

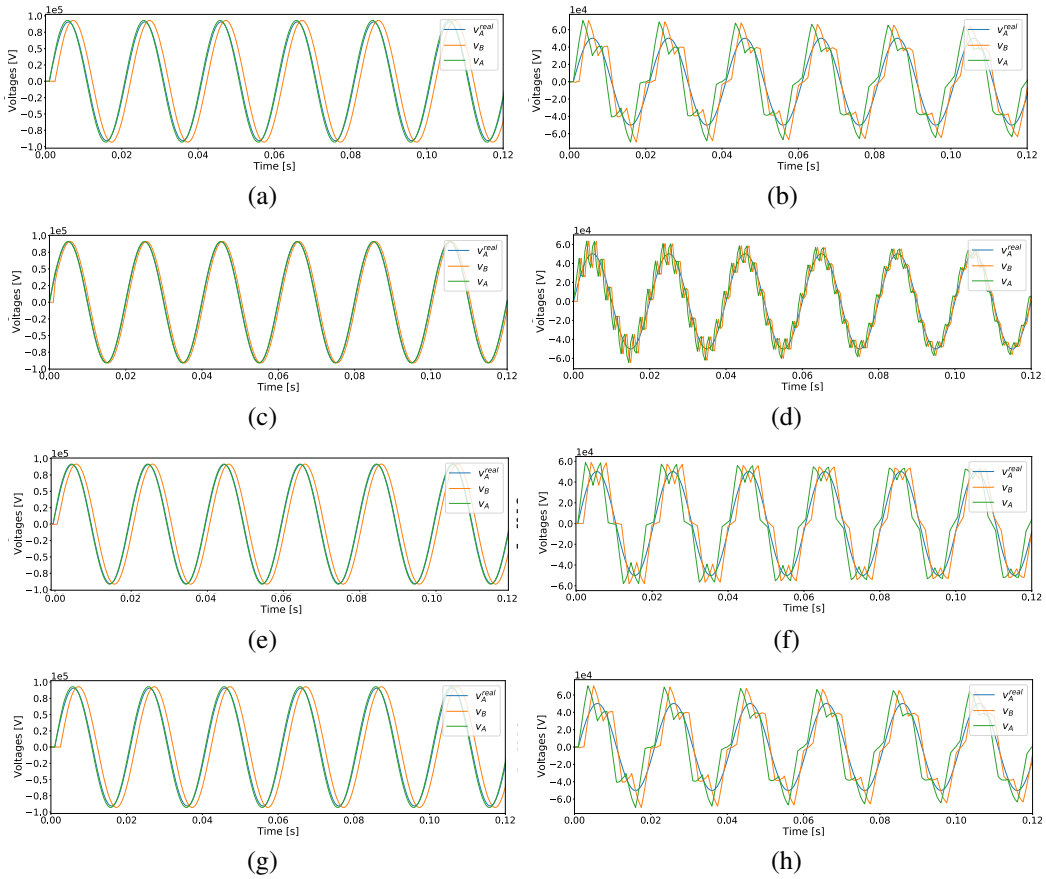


Fig. 5.16 Voltages time plots to compare time-domain accuracy of the monolithic circuit (*blue*) and ITM IA application (*green* and *orange*) for $T_s = 500\,\mu s$ in the stability region (a,c,e,g), and near the instability region (b,d,f,h) for different Digital Real-Time Co-simulation Infrastructure configurations, namely, RTDS-RTDS (a,b), OPAL-OPAL (c,d), RTDS-OPAL (e,f), and OPAL-RTDS (g,h).

$Z_B = 50.5\,\Omega$ is presented in Figure 5.17. The results demonstrate that applying a T_{Sim} lower than 500 µs ensures good time accuracy for both regions, reproducing with high fidelity the monolithic solution.

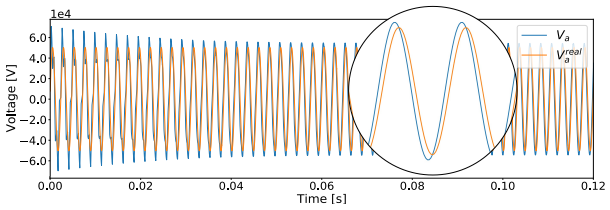
RTDS-RTDS The case $Z_B = 500\,\Omega$ is presented in Figure 5.16a. v_A overlies v_A^{real} confirming that the ITM application is capable of reproducing correctly the standalone simulation with a slight rise of 2.28% of v_A voltage peak caused by the round trip latency of the ITM circuit. v_B follows v_A, delayed of 1500 µs that is equal to $3T_{Sim}$. This is T_{D_1} latency of the ITM open-loop function G_{ol} in Equation 5.1.

This latency is described in the previous subsection 5.2.2 and is composed of the first three time step contributions.

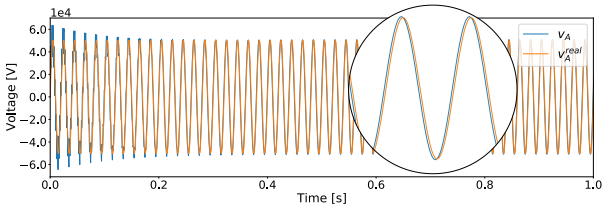
The case $Z_B = 50.5\,\Omega$ instead is presented in Figure 5.16b and presents major instabilities due to the phase shift generated by the ITM application. In particular, the distortions are generated by the round trip latency $5T_{Sim}$ equal to 2500 µs and the Z_A/Z_B ratio equal to $0.\overline{9900}$, near the instability region of the ITM open-loop function G_{ol} in Equation 5.1. v_A initial peak exceed 40.72% compared to v_A^{real}. v_B follows the v_A trend with a latency of 1500 µs. Similarly to the stable ITM case, it results correctly in $3T_{Sim}$. In Figure 5.17a, the distortion transient lasts 0.4 s stabilising the result with a 7.92% rise compared to the voltage rise of the case $Z_B = 500\,\Omega$. Finally, v_A presents a non-linear distortion of the sine due to the ITM application near the instability region.

OPAL-OPAL The case $Z_B = 500\,\Omega$ is presented in Figure 5.16c. v_A overlies v_A^{real} with an insignificant rise of 0.37% compared to the v_A voltage peak caused by the smallest round trip latency between the different infrastructure configurations equal to $2T_{Sim}$. v_B follows v_A, delayed of 500 µs that is equal to $1T_{Sim}$. This latency is described in Section 5.2.2 and is composed of the first time step contribution.The case $Z_B = 50.5\,\Omega$ instead is presented in Figure 5.16d and still presents major instabilities. In this case, the distortions are denser than in the previous case in Section 5.2.2 due to the higher frequency of the phase shift generated by the round trip latency $2T_{Sim}$ and the Z_A/Z_B ratio. v_A initial peak exceeds 26.56% compared to v_A^{real}. v_B follows the v_A trend with a latency of 500 µs. Similarly to the stable ITM case, it results correctly in $1T_{Sim}$. In Figure 5.17b, the distortion transient lasts 0.16 s stabilising the result with an 1.25% rise compared to v_A^{real}. To conclude, v_A does not present particular non-linear distortion.

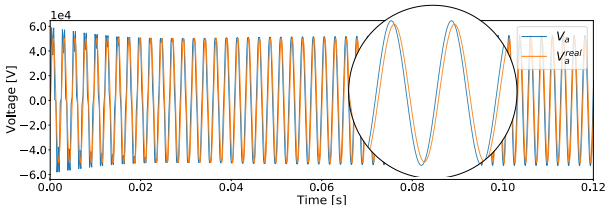
RTDS-OPAL The case $Z_B = 500\,\Omega$ is presented in Figure 5.16e. v_A follows v_A^{real} with a 0.83% rise of the v_A voltage peak. v_B follows v_A, delayed of 1000 µs that is equal to $2T_{Sim}$ confirming the description of the sequence operation diagram in Section 5.2.2. The case $Z_B = 500\,\Omega$ instead is presented in Figure 5.16f and presents similar instabilities to the previous cases. The distortions generated by the round trip latency $4T_{Sim}$ and the Z_A/Z_B ratio produce a v_A initial peak with a rise of 16.97% compared to v_A^{real}. v_B follows the v_A trend with a latency of 1000 µs, which results



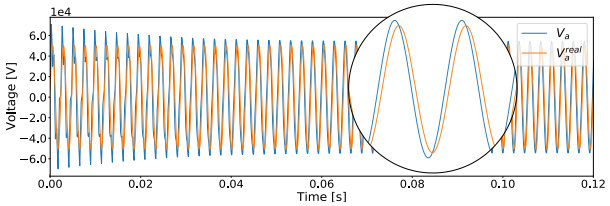
(a)



(b)



(c)



(d)

Fig. 5.17 Voltages time plots to compare the monolithic circuit (*orange*) and the transient (*blue*) when applying ITM IA for $T_s = 500\,\mu s$ near the instability region with a detail of the non linear effect on the numerical solution for different Digital Real-Time Co-simulation Infrastructure configuration, namely, RTDS-RTDS (a), OPAL-OPAL (b), RTDS-OPAL (c), and OPAL-RTDS (d).

correctly $2T_{Sim}$. In Figure 5.17c, the distortion transient lasts 0.32 s stabilising the result with a 5.14% rise compared to v_A^{real}. Lastly, v_A presents non-linear distortion smaller than case presented in Section 5.2.2 due to the lower latency experimented by the ITM IA circuit.

OPAL-RTDS The case $Z_B = 500\,\Omega$ is presented in Figure 5.16g. This case presents results similar to case in Section 5.2.2. However, v_A overlies v_A^{real} with a smaller rise equal to 1.47%. v_B follows v_A, delayed of 1500 µs that is equal to $3T_{Sim}$. The case $Z_B = 50.5\,\Omega$ instead is presented in Figure 5.16h and presents the same instabilities as the case presented in Section 5.2.2. In particular, v_A initial peak exceeds 40.72% compared to v_A^{real}. v_B follows v_A with the latency of 1500 µs, confirming the $3T_{Sim}$ latency expectation. In Figure 5.17d, the distortion transient lasts 0.4 s stabilising the result with a 7.92% rise compared to v_A^{real}. Finally, v_A presents the same non-linear distortion as the case presented in Section 5.2.2.

ITM IA Frequency-domain Accuracy Analysis

The frequency-domain analysis allows a complete understanding of the effects of the latency experimented by applying the ITM IA circuit to the proposed co-simulation infrastructure. This analysis is obtained by applying Welch's method for the Power Spectral Density (PSD) estimation applied to voltage signals v_A^{real} and v_A for both Z_B values to compare the monolithic implementation w.r.t. the co-simulated one. As depicted in Section 5.2.2, the latency experimented from the different configurations of the proposed infrastructure gives rise to non-linearity in the time-domain results generated by the phase shift caused by the ITM IA application. The highest effect is resulting near the instability region (i.e. $Z_B = 50.5\ \Omega$) for all configurations.

In fact, the phase shift time-domain effect near the instability region is similar to a triangle wave trend summed to the original voltage signal. A triangle wave can be approximated in the time-domain with additive synthesis, summing odd harmonics of the fundamental sine wave of frequency f_Δ while multiplying every other odd harmonic by -1 and multiplying the amplitude of the harmonics by 1 over the square of their mode number n as described in Equation 5.2:

$$x_{triangle}(t) = \frac{8}{\pi^2} \sum_{i=0}^{N-1} (-1)^i n^{-2} sin(2\pi f_\Delta n t) \tag{5.2}$$

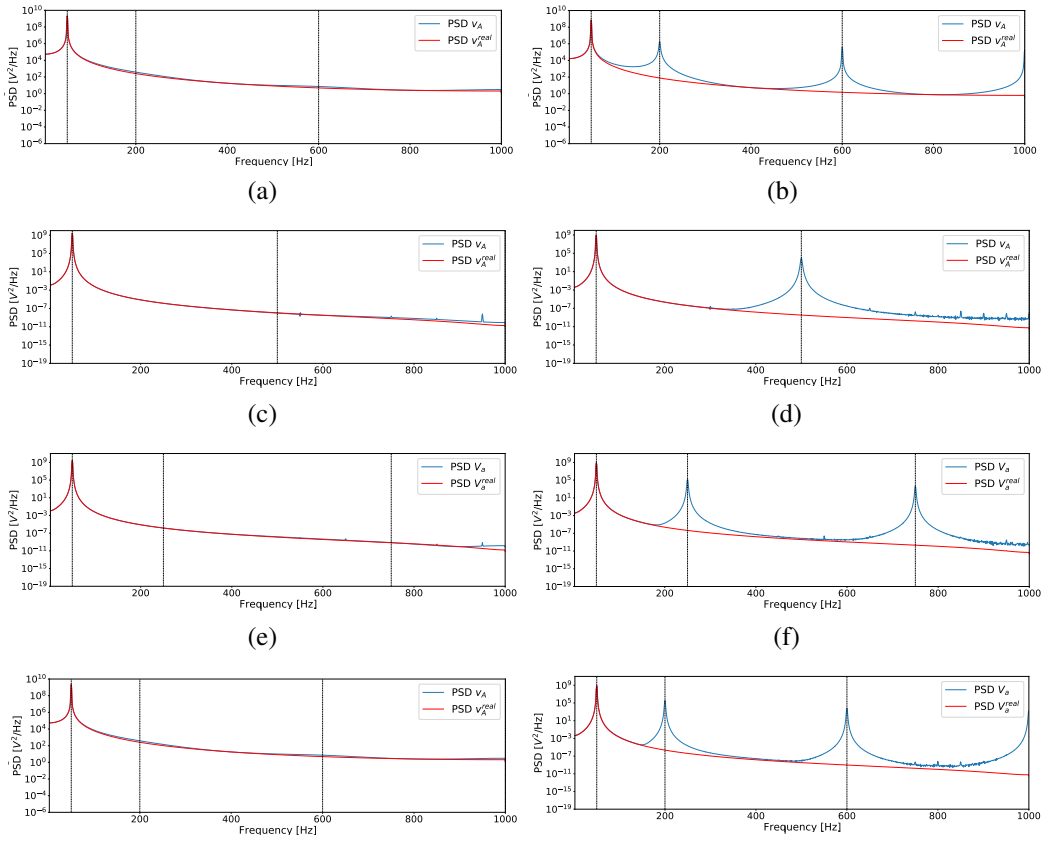


Fig. 5.18 Voltages Power Spectral Density (PSD) estimation applying Welch's method to compare frequency-domain accuracy of the monolithic circuit (*red*) and ITM IA application for $T_s = 500\,\mu s$ in the stability region (a,c,e,g), and near the instability region (b,d,f,h) for different Digital Real-Time Co-simulation Infrastructure configurations, namely, RTDS-RTDS (a,b), OPAL-OPAL (c,d), RTDS-OPAL (e,f), and OPAL-RTDS (g,h).

As for each round trip latency of the open-loop function G_{ol} in Equation 5.1 the phase shift time-domain effect changes signs, we can consider the fundamental sine wave period of the generated triangle wave T_Δ twice the round trip latency. Then, the fundamental frequency f_Δ is equal to the inverse of the period T_Δ. So, a spike in f_Δ will be always present for the case $Z_B = 50.5\ \Omega$. Following the approximation in the time-domain with additive synthesis, the other frequency components of the triangle wave will be the odd harmonics $3f_\Delta$, $5f_\Delta$, and so on. Since the resolution of Welch's method for the PSD estimation is limited by the sampling period T_{Sim} to 1 kHz, some cases will present also odd harmonics depending on the round trip latency. As matter of fact, the higher the round trip latency, the lower is f_Δ and consequently its odd harmonics.

RTDS-RTDS v_A PSD accurately overly v_A^{real} peak at $f = 50\,\text{Hz}$ (i.e. the power supply frequency) for $Z_B = 500\ \Omega$ without any other frequency disturbances, as depicted in Figure 5.18a. So, the standalone frequency content result is correctly replicated. The case $Z_B = 50.5\ \Omega$ in Figure 5.18b instead presents v_A PSD with the former peak at $f = 50\,\text{Hz}$ and three frequency peaks at $f = 200$, 600 and 1000 Hz. Since the round trip latency calculated in the previous Section 5.2.2 is $5T_{Sim}$ and following the previous assumption, T_Δ results $10T_{Sim}$ and f_Δ is exactly 200 Hz that is the fundamental sine wave component of the triangle wave. Consequently, the frequencies of the odd harmonics are 600 Hz, 1000 Hz, and so on. These three frequency components are appreciated in Figure 5.18b. Moreover, they can be noticed also in Figure 5.18a (i.e. $Z_B = 500\ \Omega$) but their effect is mitigated by the magnitude of Z_A/Z_B equal to 0.1.

OPAL-OPAL v_A PSD accurately overly v_A^{real} peak at $f = 50\,\text{Hz}$ (i.e. the power supply frequency) for $Z_B = 500\ \Omega$ in Figure 5.18c. For $Z_B = 50.5\ \Omega$, two main components result from the PSD estimation as depicted in Figure 5.18d: *i)* $f = 50\,\text{Hz}$ which is the former power supply frequency, and *ii)* $f_\Delta = 500\,\text{Hz}$ that is the fundamental sine wave of the triangle wave spectrum. The round trip latency of the ITM application for the OPAL-OPAL interconnection is $2T_{Sim}$, resulting in a $T_\Delta = 4T_{Sim}$ that confirms the PSD component of $f_\Delta = 500\,\text{Hz}$. Since the PSD is limited to 1 kHz, the odd harmonics $3f_\Delta$, $5f_\Delta$, and so on, of the triangle wave approximation cannot be appreciated.

RTDS-OPAL The stable ITM application for $Z_B = 500\ \Omega$ correctly reproduce v_A^{real} peak at $f = 50\,\text{Hz}$ in the v_A PSD represented in Figure 5.18e. For $Z_B = 50.5\ \Omega$ instead two harmonics of $f = 250\,\text{Hz}$ and $f = 750\,\text{Hz}$ can be appreciated as well as the former power supply frequency $f = 50\,\text{Hz}$, as depicted in Figure 5.18f. Following the additive synthesis hypothesis and considering the round trip latency of the RTDS-OPAL interconnection equal to $4T_{Sim}$, T_Δ results $8T_{Sim}$ and so the calculated fundamental sine frequency of the triangle wave confirms the former peak at $f = 250\,\text{Hz}$ and its first odd harmonic at $f = 750\,\text{Hz}$.

OPAL-RTDS Since for this case, the round trip latency is equal to the case in Section 5.2.2 (i.e. $5T_{Sim}$), we can assume similar observations on the frequency-domain accuracy. For $Z_B = 500\ \Omega$ in Figure 5.18g, v_A PSD closely reproduces v_A^{real} peak at $f = 50\,\text{Hz}$ without disturbances. For $Z_B = 50.5\ \Omega$, the former power supply peak is reproduced with three harmonic components at $f = 200$, 600 and 1000 Hz as depicted in Figure 5.18h. With the same assumptions of the case in Section 5.2.2, the fundamental sine frequency of the triangle wave is exactly 200 Hz and its visible odd harmonics are the former peaks at $f = 600$ and 1000 Hz.

5.3 Hybrid Co-simulation

The Hybrid Co-simulation Infrastructure makes it possible to simulate a complex MES scenario, offering MES designers a comprehensive tool to easily interconnect heterogeneous models from the module libraries in the platform in a plug-and-play manner. The infrastructure leverages a multi-model view in which the MES designer can choose among several interchangeable versions of the same simulated model, choosing from different engines (i.e. GPPL, software simulators, hardware simulators) depending on the required spatio-temporal scalability. This vision ensures the ability to simulate the fast-to-slow temporal evolution of a MES scenario in a single co-simulation platform, freeing up the potential to scale the scenario under analysis by choosing the right combination of models in a distributed infrastructure.

The infrastructure also provides the flexibility to easily extend the available module libraries and develop innovative component models following the appropriate interfaces defined by the technologies implemented in the infrastructure, unlocking simulation capabilities for large-scale MES scenarios. The designer can choose from simulation software (e.g. MATLAB Simulink, Modelica, EnergyPlus), GPPL models (e.g. Python, C++, Java), and commercial DRTS models (e.g. OPAL-RT).

Finally, the infrastructure offers the possibility of replacing simulation modules by integrating a real-world device in an HIL or PHIL manner for laboratory testing. In addition, real-world applications and services can be integrated to test their functionality in a co-simulation environment.

5.3.1 Hybrid Co-simulation Infrastructure

The infrastructure is presented in Figure 5.19, consisting of three main vertical layers: *i)* the *Data Source Layer*, *ii)* the *Co-simulation Layer*, and *iii)* the *Application Layer*. The rest of this section will describe in depth each layer.

Data Source Layer

The *Data Source Layer* contains the sources of information needed to describe a MES scenario, offering standard interfaces to access, query, and retrieve data from other layers of the platform. It includes several Database (DB) modules, which are: *i)*

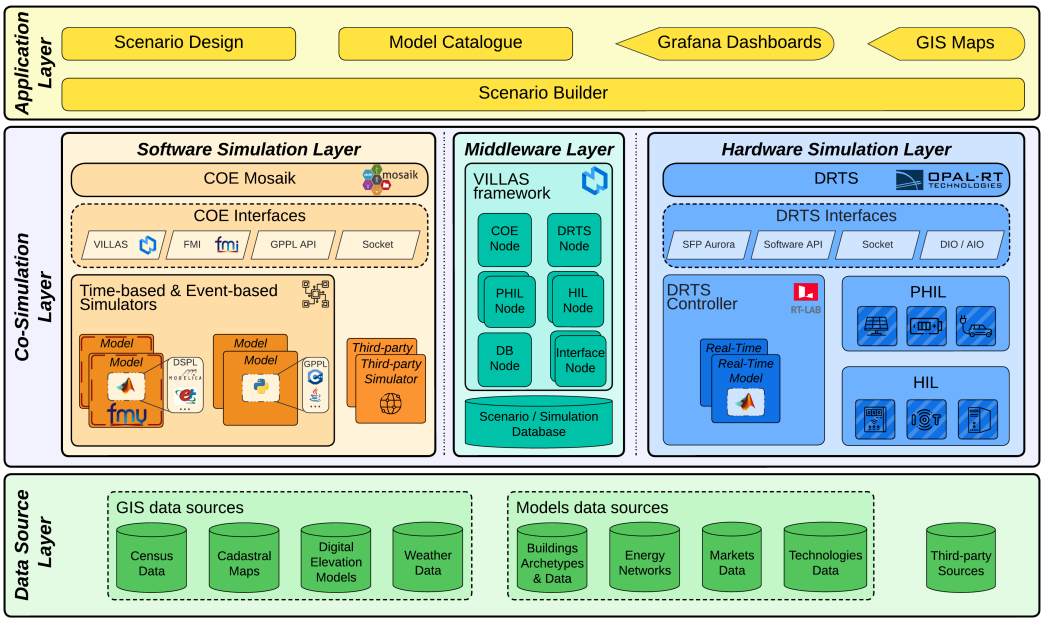


Fig. 5.19 Layered architecture schema of the Hybrid Co-simulation Infrastructure.

the *GIS data sources* to provide georeferenced data for models of the co-simulation environments that make up a specific MES scenario (e.g. *Census Data*, *Cadastral Maps*, *Digital Elevation Models*, or *Weather Data*), *ii)* the *Models data sources* to provide basic elements for model definition, such as *Building Archetypes & Data*, *Energy Networks* (e.g., district heating, power grid, distribution network), *Markets Data* and *Technologies Data* of various energy systems, and *iii)* the *Third-party Sources* to offer platform users the ability to design a custom adapter to support additional third-party DB to retrieve data from real applications.

Co-simulation Layer

The *Co-simulation Layer* is the core of the Hybrid Co-simulation Infrastructure. It sets up the co-simulation environment retrieving all the required information from the Application Layer (See Section 5.3.1). It is composed by three main horizontal layers, namely *i)* the *Middleware Layer*, *ii)* the *Software Simulation Layer*, and *iii)* the *Hardware Simulation Layer*.

Middleware Layer The *Middleware Layer* is the central component of the Co-simulation Layer that provides the communication between the soft real-time co-

simulation of the Software Simulation Layer (see Section 5.3.1) and hard real-time simulation environments of the Hardware Simulation Layer (see Section 5.3.1) exploiting a near real-time approach. It implements this communication by exploiting VILLASframework [120], a toolset for local and geographically distributed real-time co-simulations. It consists of several components called *Nodes* that allow flexible interconnections among different technological software (e.g. GPPL) and hardware (e.g. RTDS, OPAL-RT) components exploiting various communication protocols (e.g. TCP, UDP, MQTT). VILLASframework acts as an adaptation layer, exploiting a near real-time communication that follows the wall-clock time ensuring negligible communication latency among different Nodes. To ensure the proper synchronisation with the infrastructure, VILLASframework resides in a cluster node synchronised by means of IEEE1588 PTP. PTP ensures the internal clock synchronisation of each of the interconnected cluster nodes with a precision up to tens of nanoseconds. The *COE* and *DRTS Nodes* permits to exchange information among the Software and the Hardware Simulation Layers. Moreover, VILLASframework allows the interconnection of *PHIL* and *HIL Nodes* by exploiting common Internet communication protocols. The Middleware Layer includes a *Scenario/Simulation DB* based on a time-series database (i.e. InfluxDB [121]) to store information about the MES scenario and collect all results coming from the overall co-simulation environment via the *DB Node*. These results will be visualized via the Application Layer modules (see Section 5.3.1). Finally, VILLASframework is capable of supporting different *Interface Nodes* that could serve to interconnect real-world applications and services that want to feed the co-simulation environments. For instance, a real-world service for MES consumer aggregation could be interconnected to the co-simulation infrastructure to test its performances in a protected environment.

Software Simulation Layer The *Software Simulation Layer* exploits a pure software co-simulation framework based upon Mosaik [102], a Python-based framework originally designed to co-simulate Smart Grid scenarios, which is easily extensible to cope with MES domains. Its main core is the *COE Mosaik* that handles the initialisation, the data exchange management, and the time regulation and synchronisation of simulators and their model instances.

During the initialisation phase, the COE Mosaik passes to all interconnected simulators their parameters (e.g. number of model instances, time step duration, start date, end date) and model instances input/output relationships with other simulators'

model instances. This important task is managed through the Mosaik Scenario API that offers a common setup procedure calls for setting the MES scenario. Moreover, COE Mosaik manages the time regulation and synchronisation of both *Time-based and Event-based Simulators*. Time-based Simulators evolve their model instances with a constant time stepped evolution. Vice versa, Event-based Simulators wait for specific asynchronous events to trigger their model instances' internal state changes and then forward their outputs to other simulators as events.

The data exchange management instead is achieved by exploiting the *COE Interfaces* that allow forwarding the initialisation information, the time regulation and synchronisation commands, and the model instances inputs/outputs to each interconnected simulator. Moreover, the COE Interfaces allow the distribution of simulators and their models on a computer cluster, enhancing their vertical and horizontal scalability. Finally, these interfaces could enable the interconnection of *Third-party Simulator* to prevent IPR issues by exploiting the Mosaik Simulator API. Thus, a third-party company can plug in its own models in a wider MES scenario without sharing its engine.

Mosaik offers two main COE Interfaces by design: *i)* the *GPPL Interfaces* that allows the interconnection of models designed with common programming languages (i.e. Python, C++, Java) to exploit their simulation libraries (e.g. pandapower [122], pandapipes [123], odeint [124]), and *ii)* the *Socket Interfaces* that permits to interconnect distributed simulators via TCP or UDP protocol.

In our previous work [125], Mosaik have been integrated with an *FMI Interface* for FMI standard. FMI allows to encapsulate models in an FMUs. The FMU permits the coupling of the models even if they are based on different DSL and their simulation software (e.g. MATLAB Simulink, Modelica, EnergyPlus). In this paper, Mosaik has been integrated with a *VILLAS Interface* to easily exploit the Middleware Layer to exchange co-simulation results with the Hardware Simulation Layer. To that purpose, the Software Simulation Layer employs a soft real-time regulation and synchronisation that executes all models considering a common wall-clock time evolution synchronised with the Hardware Simulation Layer by means of the IEEE1588 PTP standard [126]. The cluster nodes, where the Software Simulation Layer resides, are equipped with a proper PTP board that receives the proper synchronisation packets from a GPS master clock.

Hardware Simulation Layer The *Hardware Simulation Layer* allows the interconnection of a commercial Digital Real-Time Simulator (e.g. OPAL-RT or RTDS Technologies) to the proposed co-simulation infrastructure. This layer can run specific models of MES components that require a hard real-time execution (e.g. power grid, electric vehicle charging system). Moreover, it allows the coupling of a real-time model with external Hardware-In-the-Loop (HIL) and Power Hardware-In-the-Loop (PHIL) to test and validate a real device in a protected virtual environment, avoiding huge deployment costs and the associated risk of deploying the component in a real-world environment.

It is worth noting the parallelism between the COE Mosaik and the *DRTS* hardware, which is the core of this layer. The DRTS exploits the common digital real-time capabilities of self-regulating the time evolution with small time step durations of around tens of microseconds. To ensure the proper synchronisation with the overall infrastructure, the DRTS is equipped with a IEEE1588 PTP board that receives the PTP synchronisation packets from the external GPS master clock. It offers by design different *DRTS Interfaces* to communicate with external entities. The most important interface is the *Software API* that allows to receive the compiled Real-Time Models from the *DRTS Controller* (e.g. RT-LAB or RSCAD). From the DRTS Controller application, the user can design and develop one or more *Real-Time Models* exploiting its Graphical User Interface (GUI), and compile, load and execute them on top of the DRTS hardware. The Software API is capable also to receive commands from external entities. In fact, it serves as the main interface with the Scenario Builder of the Application Layer (see Section 5.3.1) that is capable to load and run a Real-Time Model already compiled by the DRTS Controller and saved in the Model Catalogue (see Section 5.3.1).

The DRTS simulator offers other DRTS Interfaces that manages the interconnection with real devices unlocking HIL or PHIL simulation. For instance, the *Analog Input/Output (AIO)* and the *Digital Input/Output (DIO) Interfaces* forward and/or retrieve respectively analogue and digital voltage signal feeding the real-time simulation environment with real-world values. Another example is the *SFP Interfaces*, that enable fast communication protocol (i.e. Xilinx Aurora) with external hardware with a bandwidth around 2 Gbps. Finally, the DRTS offers the *Socket Interface* to exploit common transport layer protocols over Internet (i.e. TCP and UDP) and standard protocol (e.g. IEC61850, GOOSE) unlocking the control of other power equipment. This is the main interface of the infrastructure that allows communi-

cation with the Software Simulation Layer via the Middleware Layer through the VILLASframework *DRTS Node*. Finally, HIL and PHIL devices interconnected with the DRTS that are equipped with socket capabilities are also capable to communicate with the Software Simulation Layer via the Middleware Layer exploiting the VILLASframework *HIL* and *PHIL Nodes*.

Application Layer

The *Application Layer* manages the modules that describe and compose a complex MES scenario into a co-simulation environment by selecting the proper models and interconnecting them together in a user-friendly and plug-and-play fashion.

The *Model Catalogue* module acts as a library that collects all models from already deployed MES scenarios. Moreover, it gathers information about the configuration setup of each model, such as its time step duration, initial conditions, required inputs, provided outputs, and available data flows with other models. The Model Catalogue module offers to the platform users the possibility of extending the collection with new MES models by choosing one of the different possible simulation environments (i.e. simulation software, GPPL, and DRTS) with the proper co-simulation configuration required to interconnect the model to the Co-simulation Layer.

The *Scenario Design* module instead supports the user definition of a MES scenario by offering a comprehensive automated tool to interconnect models contained in the Model Catalogue module in a plug-and-play fashion. This module offers a standard YAML configuration template to set up the MES scenario by exploiting a grey-box modelling approach with minimum effort and cost [103]. The YAML tree description in Figure 5.20 extends the Pure Software Co-simulation Infrastructure YAML tree diagram of Section 5.1.1 by adding:

- the *VILLAS connection* in the Simulators Configuration that contains the specific information to set up the communication and data exchange with the *Middleware Layer* (more details in 5.3.1);
- the *Energy Networks* contains the settings about the already integrated energy network simulators that exploit the Python libraries pandapower and pandapipes [122, 123] or the DRTS Real-Time Models;

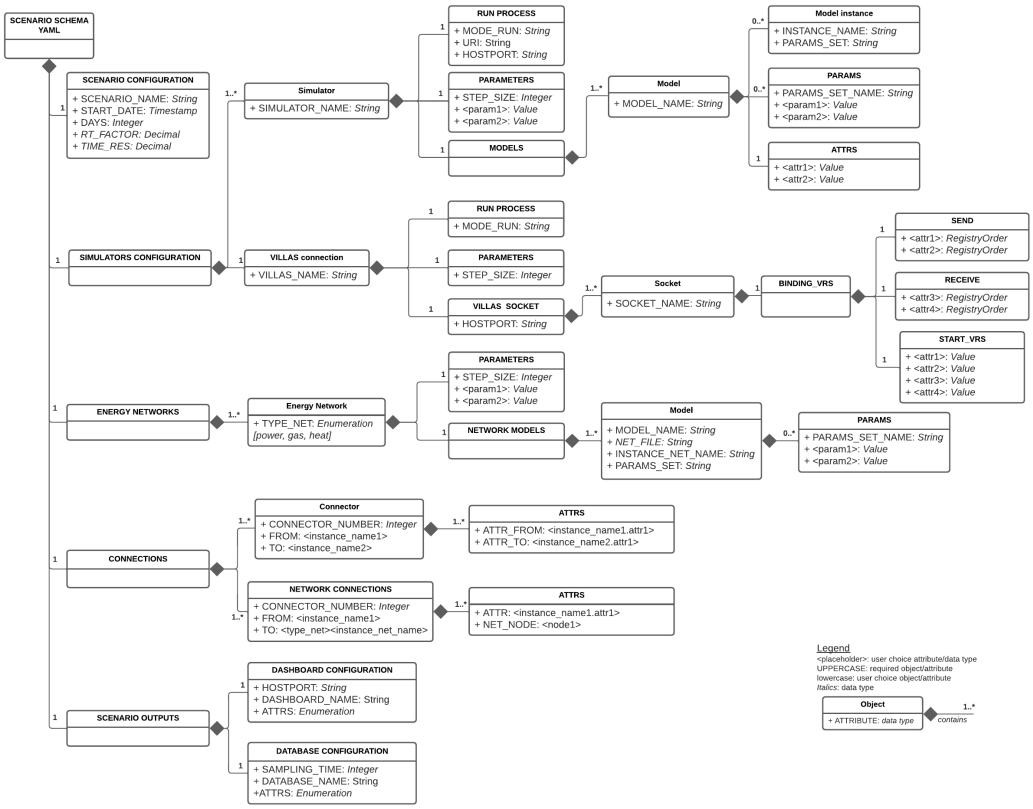


Fig. 5.20 The tree diagram from the YAML template scenario file showing the general characteristics of the scenario schema root element, with main parent elements and children elements.

- the *Network Connections* in the Connection tree that contains the information topology of the connections among energy networks exploiting a list of run-time data exchanged for each connection;

By exploiting each model configuration and setup information that resides in the Model Catalogue, the YAML template guides the platform user in designing the MES scenario, preventing the manual configuration of the interconnection of models which can be error-prone.

Finally, the *Scenario Builder* module is in charge of compiling the resulting MES scenario from the Scenario Design modules and communicates its configuration to the Co-simulation Layer, which instantiates and configures individual models to concretely execute the co-simulation environment. The Scenario Builder also validates the proposed MES scenario by exploiting simulator-specific knowledge (e.g.

model configuration and setup information) from the Model Catalogue module. This enables the Scenario Builder to identify suitable interconnection as well as to detect possible inconsistencies in the defined scenario with respect to typical input/output relationships captured by the model configuration and setup information. Moreover, the Scenario Builder set up each individual simulator software, GPPL, and DRTS for their physical interconnection and network communication to the Co-simulation Layer.

The Application Layer also includes the *Grafana Dashboard* [127] and *GIS Maps*. The *Grafana Dashboard* module presents the co-simulation results of the MES scenario under analysis by retrieving information from the Scenario/Simulation Database contained in the Middleware Layer of the Co-simulation Layer. The *GIS Maps* module instead exploits the georeferenced information from the Data Source Layer and presents them into maps to better describe co-simulation results on a spatial scale.

5.3.2 Applications and Results

This section describes the MES scenario and models exploited for testing the capabilities and performances of the proposed Hybrid Co-simulation Infrastructure. The flexibility provided by the co-simulation platform allows to use it as a virtual test-bed for complex MES use cases, as well as conducting experimental research on enabling technologies. As highlighted in the literature of recent years [6], the MES operation and planning shall be coordinated across the energy carriers, infrastructures, and consumption sectors to foster energy transition towards reliable and cost-effective energy services. In particular, the scenario proposes the provision of flexibility and effective ancillary services to the power distribution grid via the growing penetration of DERs installed on the building site (e.g., PV, electrochemical batteries) and the deployment of smart building energy management systems and control strategies combined with demand response programs [128].

In particular, the scenario proposes the evaluation of a voltage regulation ancillary service for the power grid distribution network. The service is provided directly by the Building Energy Management System (BEMS) of a building equipped with a PV and a battery system, coordinated with a Voltage Control System (VCS). Figure 5.21 shows the overall energy scenario, which consists of an area of interest of a real

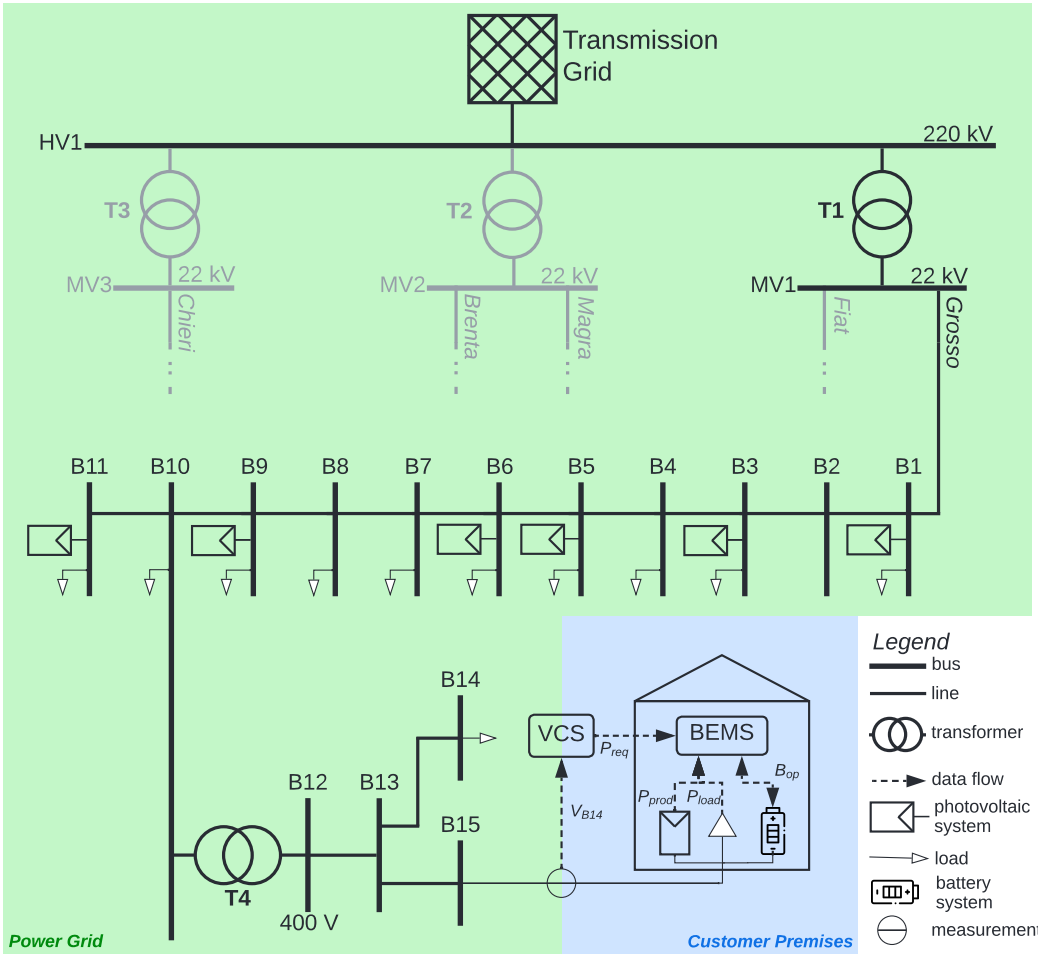


Fig. 5.21 Schema of the MV/LV power grid and the low voltage regulation system coupled with the building energy management system.

Italian MV/LV network and real demand and generation profiles obtained from an Italian DSO. The grid was taken from [129] and it consists of five main MV feeders, of which *Grosso* feeder was considered to perform the scenario simulation. The test feeder consists of 11 buses with 10 equivalent loads and 6 equivalent power injections from PV plants. These injections and withdrawals represent equivalent power system models of the external areas of the grid. The $B12$ bus extends towards the $B13$ bus the LV network where an equivalent LV load is present in bus $B14$ and the building energy system with the LV point of measure and the VCS in bus $B15$. To demonstrate the capability of the VCS and test the platform, the grid was placed under severe energy demand by reasonably increasing the loads causing a decrease of the buses' voltage towards the lower voltage limit. Indeed, the VCS measures the

voltage across the LV bus $B15$ and sends the power requests to the BEMS only if the measured voltage exceeds the tolerance of 10% around the nominal voltage value in both direction, i.e. $V_N = 0.9\ p.u.$ and $V_N = 1.1\ p.u.$, in compliance with the European standard EN 50160 CENELEC [130]. After receiving the power request, the BEMS evaluates the availability of the internal energy resources (i.e., PV production and battery capacity) based on the building load characteristics to decide if the building can provide the total amount of the power requested or a part of it and for how long. The VCS is based on a fuzzy logic PID controller that can operate at different sample times. For example, it could be deployed directly on the control system of the distribution grid, operating at very short sample times. In this scenario, the VCS is considered a cloud service over the Internet taking the real voltage measures and communicating the power requests to BEMS.

Each element constituting the energy scenario was modelled and implemented into the Hybrid Co-simulation Infrastructure. Figure 5.22 depicts the scheme of simulation blocks and their connections among them building up the scenario. In particular, the blocks represent the cyber-physical components containing the standalone models related to the customer premises and the power grid. The simulation blocks interact with the shared simulation environment linking them through different kinds of connections based on the simulator typologies, i.e., real-time hardware (blue blocks) and pure software (orange blocks) simulators are connected throughout the COE and DRTS Nodes. In addition, the blue striped block represents the HIL component, i.e., the Physical Smart Meter. To build up the energy scenario, the simulation blocks are added, parameterised, and connected through the YAML configuration files provided by the domain experts through the *Scenario Design* and finally built by the *Scenario Builder* that instances the models, composes the overall scenario automatically, and execute the co-simulation.

The following subsections describe the characteristics of the models implemented for the scenario under analysis. The various models were divided into *i)* Environment and Power Grid and *ii)* Customer Premises.

Environment and Power Grid

In this scenario, the environment can be represented by the weather module and the other simulators acting as the interface between the power grid and the customer premises, i.e., the household.

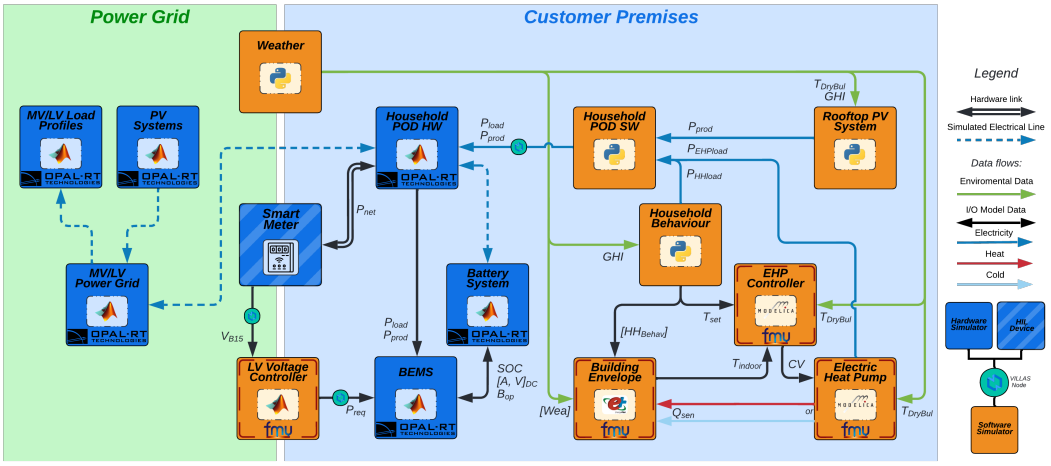


Fig. 5.22 Simulator block diagram of the energy scenario designed within the Hybrid Co-simulation Infrastructure.

Weather. Weather data are integrated from third-party data sources, such as Weather Underground [117], through an integrated Python API. The Weather module retrieves the information at the required time step by the models and distributes them in run-time through the COE Mosaik. If the minimum available time step of the weather data sources is greater than the time step required by a model, a linear interpolation is performed.

Power Grid. The DRTS simulator has been used to execute the simulation of the power grid. In particular, the OPAL-RT OP5700 is used for deploying the MV/LV power grid model. It requires the configuration of the simulation solver eMEGAsim which can be done through MATLAB Simulink. Once the model of the grid has been completed in Simulink, it needs to be uploaded on the DRTS simulator by using RT-LAB which compiles the model in order to run it with a time step duration of 50 µsec. For the scenario simulation, the aggregated MV/LV Load Profiles and PV Systems generation profiles have been used for all the buses with the only exception of the bus $B15$, which is connected with the high-detailed model of the customer premises.

Physical Smart Meter. The device used is a 3-phase smart meter prototype (a.k.a. 3-SMA) composed by a Raspberry Pi and a Data Acquisition Board that allows the device to collect data with a frequency of 3.2 KHz. In the scenario under test, the Data Acquisition Board is connected between the grid and the simulated household by means of an analogue output block in the DRTS model. So, the analogue voltage

and current measurements coming from the analogue output board are collected from the DRTS. The 3-SMA receives these values and is in charge of pre-processing the collected data each 50 msec. Such collected data are processed to obtain the voltages and currents RMS values. Moreover, the 3-SMA communicates the RMS values to the other component of the infrastructure that requires the measurement as input via the HIL Node of VILLASframework. Further details on the 3-SMA prototype can be found in [129, 131].

LV Voltage Controller. The VCS is based on fuzzy logic PID controller monitoring the voltage V_{B15} and sending the request capacity P_{req} to the BEMS if the voltage exceed the tolerance. The controller calculates the error value as the difference between the desired min/max tolerated voltage and the measured value V_{B15} and applies a correction based on proportional, integral, and derivative terms on the P_{req} that is requested to not exceed the tolerance and minimise the error over time. The model is developed in Simulink using the Fuzzy Logic and Control System Toolboxes, and exported as a FMU. In addition, Simulink allows one to freely choose the time resolution, thus providing flexibility to set the software in real-time and seeks to quickly exchange data with the real-time simulators, such as DRTS.

Customer Premises

The customer premises represent the building energy system with its physical components, such as the envelope, energy conversion technologies, and entities, such as the habitants living inside the building. They are modelled by coupling diverse simulators, described in the following sections. A more detailed description of the building energy system can be found in our previous works [103, 125].

Rooftop PV System. The Rooftop PV System is modelled by using the infrastructure presented in [118]. This infrastructure allows estimating the solar radiation and PV potential profiles in real-sky conditions with a high spatio-temporal resolution by integrating different decomposition and transposition models in literature for estimating the plane of array irradiance. It uses as inputs: *(a)* GIS data to describe building rooftops in terms of slope, orientation, possible obstacles, and shadows; *(b)* weather data provided by *Weather* block, at least the Global Horizontal Irradiance in real-sky condition *GHI* (but also the Diffuse Horizontal Irradiance and Direct Normal Irradiance if they are available) and the outdoor Dry-Bulb Temperature T_{DryBul}.

The PV simulation can be performed with the same time step as the resolution of the *GHI* data.

Household behaviour. The household behaviour was modelled by using the simulator proposed in [119]. The model uses diverse input data characterising the household, i.e., census data, weather data, *Use of Energy* and *Time of Use* surveys, to create a non-homogeneous semi-Markov model for simulating over time the household occupancy, type and duration of activities performed by household's inhabitants, which is associated to specific usage of electric appliances and lights. In the end, the Household behaviour parses the number of occupants in a zone and their interactions with lights and appliances, providing aggregated loads, appliances and lights loads, and schedules, which are given as input vector (HH_{Behav}) to the *Building Envelope*. The Household behaviour simulator simulates with a resolution of 10 minutes, but can also provide appliances and aggregate load profiles with a time step from one second to ad lib.

Electric Heat Pump and EHP Controller. The air-to-water EHP was modelled using the open-source modelling and simulation environment OpenModelica. The EHP model was exported as an FMU for co-simulation, as shown in Figure 5.22. In addition, the EHP model was interfaced through the COE Mosaik Interface FMI adapter to communicate with the platform effectively. The EHP FMU calculates the sensible heat gain required to maintain the set-point temperature T_{set} in the rooms. The EHP requires as input the control signal for the Control Valve actuator (*CV*), which is provided by the *EHP Controller*, the T_{set} selected or scheduled by the household, the measured indoor temperature T_{indoor} and the Dry-Bulb Temperature T_{DryBul}. The measured variable T_{indoor} supplied by the *Building Envelope* block is controlled to maintain the desired set-point T_{set} by using the *EHP Controller* that is a PID controller acting on the heating system water mass flow rate through regulation of the *CV*. It is worth noting that thanks to the platform's flexibility, it is possible to replace the control system with more advanced control systems in a plug-and-play fashion.

Building Envelope. The Building Envelope was modelled in EnergyPlus, a well-known open-source software for detailed energy modelling of buildings that allows calculation of building energy use, such as heating and cooling loads, disaggregated energy end-use, and many other building-related features. In addition, EnergyPlus allows exporting the building model from the IDF file as FMU for co-simulation via

the Python package *EnergyPlusToFMU*. Inputs and outputs of the *Building Envelope* FMU are managed by EnergyPlus via External Interface objects located in the IDF file. In particular, the exchanged variables are the vector of the Household behaviour variables (HH_{Behav}), the heat gain Q_{sen} provided by EHP, the indoor temperature T_{indoor} and set-point temperature T_{set}, and the vector of weather data *Wea* required by EnergyPlus. EnergyPlus can perform simulations with a minimum time step of 1 min up to 1 h.

Battery System. The electrical energy storage system was modelled in MATLAB Simulink using the Simscale Library and embedded into the DRTS to couple it with the power distribution network in real-time, in order to describe the fast dynamics correctly. The model simulates the dynamic behaviour of a Li-Ion battery, and it can be fully parameterised using a commercial battery datasheet. The battery model was electrically connected on the *behind-the-meter* Point of Delivery (POD) of the household. The main state variables of the battery (i.e., *SOC*, *A* and *V* in Direct Current (DC)) are sent to BEMS that manages the energy fluxes in the building though the battery signal operations B_{op} based on the requests of the VCS. The battery model is set to simulate the charge and discharge of the battery with a resolution of 10 minutes

Household POD HW. The Household POD represents the physical point of withdrawal and/or injection of electricity into the distribution network. It was modelled in Simulink as a PQ Load element and embedded into the DRTS. It is physically connected through a hardware link with the Smart Meter in order to exchange the power measures and electrically connected through real-time simulated electrical lines with the behind-the-meter battery system and the front-of-the-meter power distribution grid. In particular, the Household POD receives the power-related data signals from the pure software simulators of the building through COE Node of VILLASframework and translates them into real-time simulated AC voltage and current. Moreover, It sends the data of power withdrawal or injection to the BEMS.

Building Energy Management System. The BEMS is a ruled-based control algorithm that manages the charge and discharge of the battery under the depth of discharge limit, prioritising self-consumption and, eventually, charging the battery only from the surplus of PV production. However, when the BEMS receives the requests for upward or downward capacity from the VCS, it verifies if there is

space for providing the requested flexibility or not by directly controlling the battery parameters for further injection/withdrawal or the PV system only for withdrawal.

Household POD SW. The virtual Household POD provides the data interface between the hardware Household POD and the pure software simulators of the building. It collects the PV electrical generation (P_{prod}), the aggregated household electrical load (P_{HHload}), the EHP electrical consumption ($P_{EHPload}$), and main state variables of the battery (*SOC*, *A* and *V* in DC). It was used as a data collector manager returning the simulation results either in run-time or at the end of the simulation. The smart meter simulator can perform the simulation with whatever time step resolution without any limitation.

Experimental Results

The scenario presented in Section 5.3.2 was simulated in order to test the functionalities of the presented Hybrid Co-simulation Infrastructure. The following paragraphs describe respectively: *i)* the Software, Hardware, and Network Setup, *ii)* the Scenario Setup, *iii)* the Scenario Results, and *iv)* the Co-simulation Latencies.

Software, Hardware, and Network Setup The Co-simulation Layer is implemented in the Energy Center in our university campus, where each physical hardware entity in the proposed infrastructure (e.g. servers and DRTS) is interconnected via a 10 Gbps Ethernet switch minimizing the latency in data exchange over the Local Area Network (LAN).

The Software Simulation Layer in Figure 5.19 involves a master node with 2 Intel Xeon Gold 6238R CPU@2.20 GHz with 192GB DDR4@3200MHz RAM and four nodes Intel Xeon E3-1245v5 CPU@3.50 GHz with 32 GB DDR4@2133 MHz RAM. The master node hosts the COE Mosaik with its Interfaces and all the Middleware Layer entities (i.e. VILLASframework and the Scenario/Simulation Database). The four nodes instead host all the software modules in the energy scenario of Figure 5.22. *Node A* runs the Household Behaviour, the Rooftop PV System, the Household POD SW, and the Weather modules that are software simulators in Python. *Node B* hosts the Building Envelope FMU modelled with EnergyPlus. *Node C* executes the LV Voltage Controller FMU developed in MATLAB Simulink. Finally, *Node D* runs the Electric Heat Pump and the EHP Controller FMUs as OpenModelica models. These

models communicates with the COE Mosaik by implementing the FMI and GPPL API Interfaces.

All nodes have a Fedora 33 Real-Time Kernel-based operating system that is designed to maintain low latency, constant response time, and determinism of a specific application process. Finally, the Hardware Simulation Layer involves an OPAL-RT OP5700 (i.e. a DRTS) combining a Xilinx Virtex-7 FPGA with **3** over 16 Intel Xeon E5 activated processing cores to meet the requirements for the most demanding HIL and PHIL real-time applications with analogue and digital input/output expansion boards. The OPAL-RT OP5700 hosts the following modules of Figure 5.22: *i)* MV/LV Power Grid, *ii)* MV/LV Load Profiles, *iii)* Photovoltaic System, *iv)* Household POD, *v)* Battery System, and *vi)* BEMS.

To ensure the time synchronization of all simulation nodes with the OPAL-RT OP5700, the platform exploits the IEEE1588 PTP standard. The OP5700 and the nodes are all equipped with the Oregano Syn1588 PTP synchronization board to achieve the overall time synchronization of the proposed infrastructure by exploiting an external GPS clock. To reach lower latencies and a correct handling of the PTP packets, another private LAN was implemented by means of an Oregano Syn1588 switch that routes the PTP packets via hardware acceleration. The GPS clock and the Oregano cards are all interconnected to the Oregano switch via Ethernet cables.

The Hardware Layer allows to perform HIL and PHIL testing. In particular, the scenario under test includes a 3-SMA smart-meter prototype derived from our previous work [131]. The Data Acquisition Board of the 3-SMA meter is connected to the analogue output of the OPAL-RT OP5700. This board is in charge of collecting the voltage and current values generated during the simulation as if it was physically connected to the LV bus of the electric network. Measurements are continuously transferred to the computational unit of the meter which evaluates the RMS and the phase of those measurements and communicates such results to the HIL Node of VILLASframework, and thus to the other entities in the infrastructure needing these data.

All entities of the Application Layer stand in a node Intel Xeon E3-1245v5 CPU@3.50 GHz with 32 GB DDR4@2133 MHz RAM and all the Data Source Layer instead on a node Intel Xeon Silver 4114 CPU@2.20 GHz with 16 GB DDR4@3200 MHz RAM with 8x600 GB SAS 10k HD for database management

and write/read operation acceleration. These nodes do not require proper time synchronization with the Co-simulation Layer.

Scenario Setup The standardised YAML configuration files were filled through the *Scenario Design* with all data and parameters required to set up the co-simulation environment, simulators, and their models, as well as connections among them as depicted in Figure 5.22. The relevant model settings are described in the following.

The models related to the customer premises were parametrised and set as in [125]. The time-steps were set considering the capability of the solvers and computational effort, as well as the needs for the real-time hardware coupling: EnergyPlus Building Envelope 10 *min*, Modelica Electric Heat Pump 5 *min*, Rooftop PV System 15 *min*, Household Behaviour 10 *min*, Matlab Simulink LV Voltage Controller 100 ms, and Household POD SW 100 ms. In particular, these last two were set with smaller time steps as they shall be coupled with the hardware simulators (see Figure 5.22). The *Weather* module provides data to each simulation engine at the requested time step. All the models implemented in OPAL-RT (i.e., the MV/LV Power Grid, the MV/LV Load Profiles, the PV Systems, the Battery System, the BEMS, and the Household POD HW) have been compiled to run with a time step duration of 50 µsec. Moreover, it was assumed that for all the connection points of the grid the power factor is maintained above 0.95. The Battery System is composed of two battery packs in series with a rated capacity of 60 Ah and a nominal voltage of 200 V. The parameters of the discharge characteristics were derived from the built-in Li-Ion battery Simulink model.

In the end, the *Scenario Builder* module parses the YAML configuration files to retrieve all the required information to perform the scenario simulation and automatically distributes them to the Co-Simulation Layer exploiting the API of COE Mosaik, DRTS OPAL-RT, and middleware VILLASframework.

Scenario Results The scenario was simulated by exploiting the real-time capability of the Hybrid Co-simulation Infrastructure for two consecutive days during the thermal season by considering the use cases without voltage control, called BAU, and with the activated voltage control system, called VRS. The main results of the simulation are depicted in Figure 5.23 (BAU on the left and VRS on the right). It shows the main characterising variables for comparing the two use cases bringing

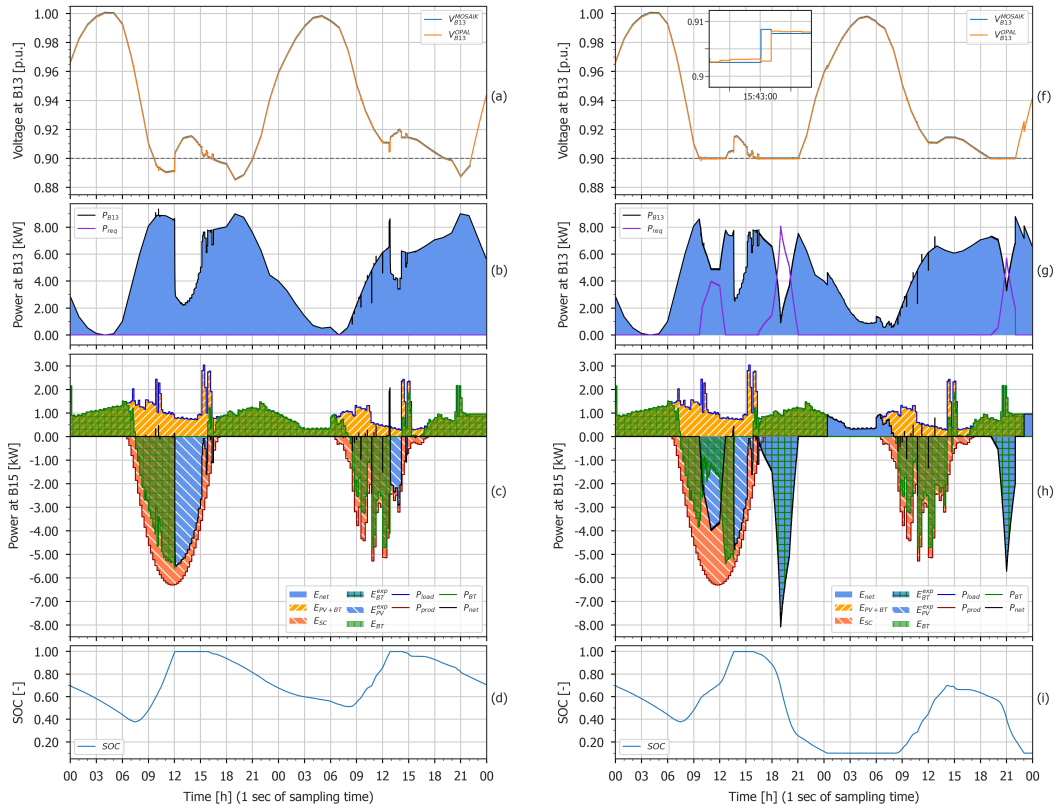


Fig. 5.23 Simulation results of two consecutive days during the thermal season by considering the use cases without voltage control Business As Usual (BAU) (*left*), and with activated voltage control system Voltage Regulation Service (VRS) (*right*). In particular, plots (c) and (h) show the power and energy exchanges at bus B15 measured on the customer premises, i.e, related to the battery (P_{BT}, E_{BT} and E_{BT}^{exp}), PV production (P_{prod} and E_{PV}^{exp}), consumption (P_{load}), net values (P_{net} and E_{net}) and self-consumption (E_{PV+BT} and E_{SC}).

out the benefits of the control strategies, as well as the delays and noises that derive from the coupling of pure software and hardware simulators. In particular, the Figure 5.23a*(a)* and *(f)* show the RMS voltage in p.u. measured at bus $B13$. To highlight the round trip time latency due to the HIL and the co-simulation application, the voltage measure was collected directly from the Real-Time Model V_{B13}^{OPAL} and indirectly from the Smart Meter V_{B13}^{MOSAIK}, both in OPAL-RT. Figure 5.23*(b)* and *(g)* show the power measured at bus $B13$, P_{B13}, and the power request, P_{req}, for voltage control, both collected from OPAL-RT. Figure 5.23*(c)* and *(h)* show the power and related energy exchanges at bus $B15$ measured on the customer premises (see the figure caption for more details), and the battery State Of Charge (SOC) in Figure 5.23*(d)* and *(i)*. In particular, the load P_{load} and production P_{prod} data were collected from Mosaik, while the data related to the battery and the net power and energy exchanges were collected from OPAL-RT. Overall, the data were collected with a sampling time of one second. However, it is possible to request information from each module exposing inputs/outputs and parameters or from the simulation environment directly at any sampling time.

By comparing the voltage curves in Figure 5.23*(a)* and *(f)*, the VCS together with the BEMS succeed on maintaining the voltage above the tolerance limit of 0.9 $p.u.$ during the whole period of simulation, as opposed to BAU use case in which the voltage falls under the limit due to the high network withdrawals, especially during the peak hours as depicted in Figure 5.23*(b)*. Considering the VRS use case, the Figure 5.23*(g)* and *(h)* clarify how VCS and BEMS act to control the voltage on the distribution grid. The first capacity request P_{req} occurs from 09:30 til 12:30 of the first simulated day amounting to approximately 4 kW at the peak. It can be seen that the request is immediately full-fill by the PV production E_{PV}^{exp} of the building by injecting directly into the grid rather than charging the battery, as highlighted by the reduction of the stored energy E_{BT} in the request period. By analysing the second grid request of the day, from 16:00 to 21:00, of about 8 kW at the peak, the capacity is provided directly from the battery that is discharging to cover the household demand E_{BT} as well as the external request E_{BT}^{exp}. It is important to note that, unlike the BAU use case, the PV production and battery capacity are exploited when needed and if possible by the VCS causing probable inconveniences for the household. For example, as shown in Figure 5.23*(h)* and *(i)* as opposed to Figure 5.23*(c)* and *(d)*, the battery results completely discharged at about 00:00 and it is not able to cover the household demand til 09:00 of the day after (the SOC reaches the discharge limit of

10 %), naturally obliging withdrawal from the grid during this period. This is a cost for the family that must be adequately covered and remunerated in order to sustain the network ancillary service. Moreover, this withdrawal, which happens in VRS use case compared to BAU, does not affect the ancillary service because it occurs at night when the demand for electricity in the grid is low.

In the end, the voltage regulation service, provided by the coordinated action of the LV VCS with the BEMS, succeeds to limit the voltage drop and not letting it exceed the tolerated limits, as highlighted in Figure 5.24 by comparing the BAU scenario with respect to the VRS one. In particular, the figure shows the area encompassing the voltage from bus $B1$ in the MV grid towards the LV grid at the final bus $B15$ in both scenarios. It can be observed that in the VRS scenario, the voltage is always maintained above 0.9 p.u., while in the BAU scenario the voltage is not controlled and it drops up to 0.885 p.u.. Clearly, the voltage in MV bus $B1$ is not influenced by the LV regulation. In addition, it can be noted that the LV regulation changes the use of the grid also at times when control does not take place, e.g. from 12:00 to 14:00 of the first day and from 13:00 to about 15:00, in which the voltage is lower than the BAU scenario, but still above the tolerated limits.

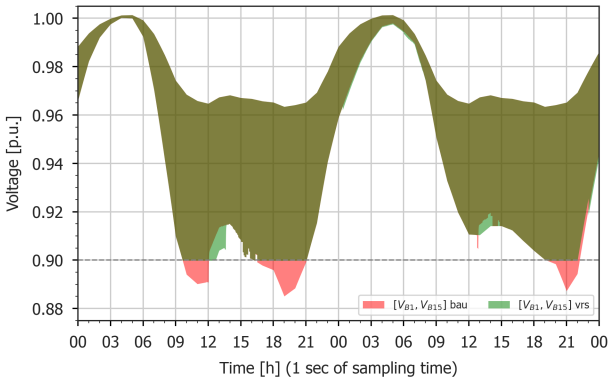


Fig. 5.24 Area encompassing the range of voltages buses in the whole MV/LV grid considering both BAU (red) and VRS (green) scenarios.

Co-simulation Latencies The complex interactions among the Co-simulation Layer entities could cause latencies and relative inaccuracies mainly due to two main factors: *i)* the SoS hierarchy of the Software Simulator Layer modules interconnected,

and *ii)* the interconnection among the Hardware Simulation Layer and the Software Simulation Layer via the Middleware Layer.

In the Software Simulation Layer, the co-simulation application decomposes the overall MES system in its fundamental components and equipment models. This SoS vision is then applied to the COE Mosaik and imposes the same hierarchy between the simulator inputs and outputs, causing latencies multiple of the smallest time step duration of the modules interconnected by the COE Mosaik in the propagation of the effects related to a particular simulator, causing inaccuracies in the overall simulation results. For instance, the Rooftop PV System module is receiving information about the GHI from the Weather module as input, which affects its output in the next time step. So the latency is exactly 1 time step duration, i.e. 100 msec. Then, the Rooftop PV System simulator is generating outputs that are communicated as input to other simulators. For instance, the power production of the PV system is forwarded to the Household POD SW module and impact its output in the next time step of co-simulation, causing two time step of latencies for being affected by the Weather module output. When the output value fluctuations of the simulator are negligible, the effect of this latency is unnoticeable. When the values are strongly fluctuating instead, the effect becomes noticeable and could cause serious inaccuracies. Moreover, this is further exacerbated when the simulators present different time step duration. In Figure 5.23*(c)* and 5.23*(g)* for instance, the power production of the PV system in the second day between 9:00 and 15:00 strongly fluctuates due to the cloudy day data received in input by the Weather module. This causes inaccuracies in the fluctuations of the Household POD SW calculation due to the latency of the above-mentioned SoS hierarchy.

Another cause of latency is caused by the interconnection among the Hardware and Software Simulation Layers via the Middleware Layer and is predominant in the case of HIL and PHIL application [126]. To calculate this effect during the co-simulation, the results of the RMS voltage have been measured directly inside the Real-Time Model and indirectly from the COE Mosaik. In Figure 5.23*(f)*, the direct and indirect measurements of the RMS voltage $B13$ in p.u. are depicted in the zoomed plot. The direct measure V_{B13}^{OPAL} is calculated inside the Real-Time Model and saved in a MATLAB file. It is worth noting that V_{B13}^{OPAL} anticipates the indirect measure V_{B13}^{MOSAIK}. In fact, V_{B13}^{MOSAIK} is experiencing a round trip time latency generated by the following operations:

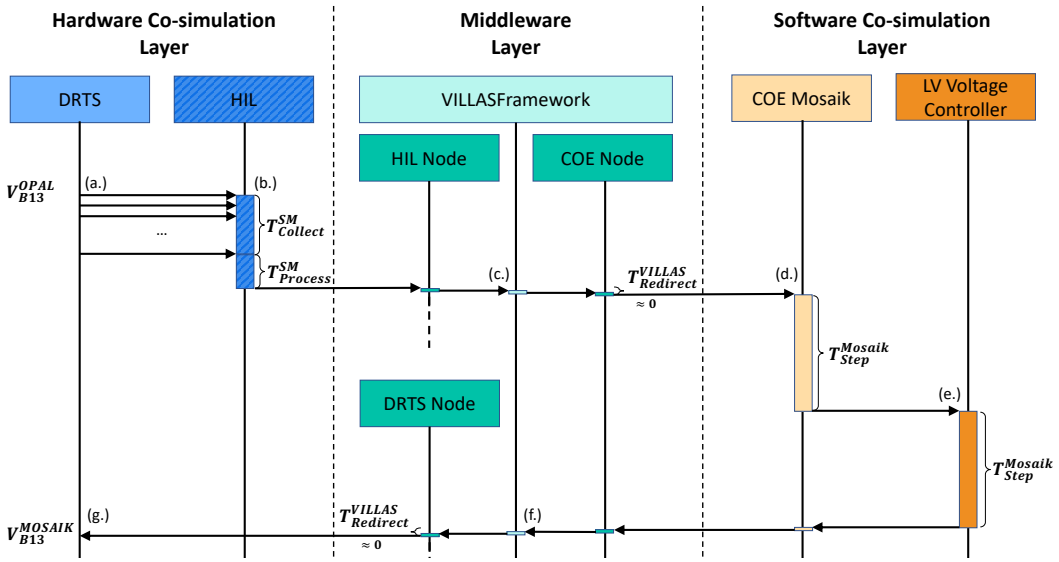


Fig. 5.25 Sequence diagram of the direct and indirect measurement operations to calculate the round trip time latency of V_{B13}^{MOSAIK}.

a. the 3-phase voltage signals of bus $B15$ are calculated in the Real-Time Model for each time step (i.e. 50 µsec) and reproduced as three ± 5 V signals in the analogue outputs of the OPAL-RT OP5700;

b. 3-SMA, the HIL Smart Meter, collects the ± 5 V signals, calculates each 50 msec the 3-phase RMS voltage values, and sends them to the HIL Node of VILLASframework via UDP in a packet;

c. VILLASframework receives this packet and redirects it to the COE Node via UDP;

d. the COE Mosaik receives the packet and sends it in the next time step (i.e. after 100 msec) to the LV Voltage Controller via TCP that decodifies it to then use the 3-phase RMS voltage values to generate the P_{req} battery request;

e. the LV Voltage Controller codifies the packet and sends it back to the COE Mosaik via TCP in the next COE Mosaik time step (i.e. after 100 msec). The Mosaik COE redirects it to the COE Node in VILLASframework via UDP;

f. VILLASframework receives the packet, translates it to the OPAL-RT data format, and redirect it to the DRTS Node;

g. Finally, the Socket Interface of the OPAL-RT OP5700 DRTS receives the packet and saves the 3-phase RMS voltages in a MATLAB file.

These operations are depicted in Figure 5.25 with their respective timings. It is worth noting that the communication latencies among the entities in the wired network (i.e. the black arrows) are negligible with respect to operations' latencies. The resulting overall round trip time latency is expressed by the following Equation:

$$RTT = T^{SM}_{Collect} + T^{SM}_{Process} + T^{VILLAS}_{Redirect} + 2 * T^{Mosaik}_{Step} + T^{VILLAS}_{Redirect} \tag{5.3}$$

where RTT is the overall round trip time; $T^{SM}_{Collect}$ is the 3-SMA collection duration of the ± 5 V signals which is a constant latency of 50 msec; $T^{SM}_{Process}$ is the 3-SMA processing time due to the RMS calculation and codification of the Google protobuffer which is a variable latency with an average value of 37.5 msec; $T^{VILLAS}_{Redirect}$ is the time spent by VILLASframework to receive a message from an input Node, process it, and redirect to the right output Node which is a variable latency with a negligible average value if 0.754 µsec; and T^{Mosaik}_{Step} is the COE Mosaik time step duration which is a constant latency of 100 msec. The overall RTT results in a variable latency with an average value of 287.5 msec, which can be neglected as compared with the communication requirements for ancillary services (e.g., voltage regulation) in a realistic scenario as demonstrated in [132].

Chapter 6

Distributed Event-Driven Platform

As demonstrated in the previous chapters, it is fundamental to exploit co-simulation environments to assess and test the feasibility of innovative services and functionalities for MES. However, the co-simulation infrastructures presented in Chapter 5 cannot switch from a simulative environment to real-world deployment of a particular application under test, limiting its Technological Readiness Level (TRL). This task is fundamental to reduce the TRL evolution of a proper component, piece of equipment, function, or service in MES context.

To cope with this issue, the Distributed Event-Driven Infrastructure has been designed to simulate and assess general purpose services for a smart management of MES to then allow their switching to a real-world deployment by exploiting an event-driven approach that exploits common communication protocols that are used in the field of MES (e.g. IEC 61850, GOOSE, PMU). In fact, it exploits a CPES description of the MES scenario that permits to include these cyber aspects in the co-simulation environments to then switch to real-world implementation of the simulated communication protocols. Figure 6.1 shows its architecture that follows the interoperability and inter-dependency among layers depicted in GAMES in Chapter 4. Its modularity takes advantage of the microservices software design pattern, which consists on developing software *as a suite of small services, each running in its own process and communicating with lightweight mechanisms* [133]. This increases flexibility and maintainability because services are *small, highly decoupled and focus on doing a small task* [134]. The platform implements IoT communication paradigms and technologies to allow a fast bidirectional communica-

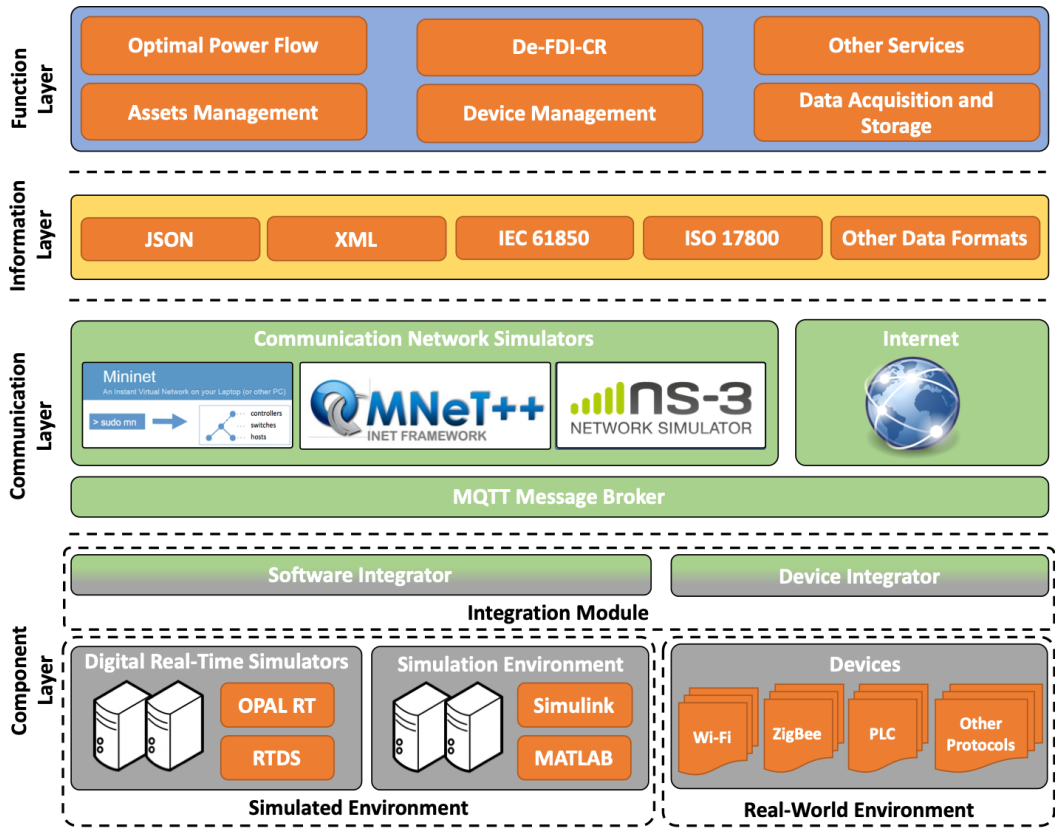


Fig. 6.1 Scheme of the proposed SGAM-Based Co-Simulation platform.

tion among its entities, either hardware or software. In particular, it exploits the two main communication paradigms: i) *publish/subscribe* [135] through MQTT protocol [136] and ii) *request/response* offering REpresentational State Transfer (REST) web services [137]. Publish/subscribe allows asynchronous communication that complements request/response. Furthermore, publish/subscribe enables the development of scalable loosely-coupled event-driven systems, services and applications that can react in (near-) real-time to certain events [138]. All these features ease developers in creating new simulation scenarios in a plug-and-play fashion also integrating third-party software. Moreover, the distributed event-driven infrastructure solution provides support to migrate from a fully simulated environment to the real-word. The rest of this section describes all the modules for each layer in the platform.

6.1 Distributed Event-Driven Infrastructure

Following the GAMES approach, the Distributed Event-Driven Infrastructure is composed by four main layers: *i)* the Component Layer, *ii)* the Communication Layer, *iii)* the Information Layer, and *iv)* the Function Layer. In the following paragraphs each layer has been described.

6.1.1 Component Layer

The **Component Layer** represents all the interacting entities, either hardware or software, needed to build a service operating in a MES scenario following a multi-model approach. It is worth noting that the three Component Layers presented in GAMES are omitted from Figure 6.1 to avoid repetitions and for readability of the architecture. As shown in Figure 6.1, it is made up of three macroblocks: i) *Simulated Environment*, ii) *Real-world Environment* and iii) *Integration Module*.

On the one hand, the *Simulated Environment* consists of both digital real-time simulators (e.g. OPAL-RT and RTDS Technologies) and different software simulation frameworks (e.g. MATLAB Simulink), where models are executed. Thanks to real-time capabilities provided by this macro-block, no significant latency is introduced to simulations. These models and components could be inherited by the co-simulation infrastructure presented in Chapter 5. On the other hand, the *Real-world Environment* consists of heterogeneous hardware devices that exploit different protocols, either wireless or wired, to communicate (e.g. Wi-Fi, ZigBee, PLC).

The components of these two macroblocks are integrated into our co-simulation platform by exploiting the *Integration Module*. To allow bidirectional communication, it acts as a bridge between the underlying technologies and the rest of the platform following a methodology in [139]. In this perspective, each component in the Simulated or in the Real-world Environment needs its own *Software-* or *Device-Integrator*, respectively. Once the corresponding integrator receives new data from its low-level technology, this data is translated into a common data format and sent to the rest of the platform by exploiting either MQTT or REST, and vice-versa.

In a nutshell, the *Component Layer* provides features to enable a modular multi-model approach, where the different models, running in their own simulation frame-

works, are combined together to build MES scenarios. Integration of DRTS allows simulations that realistically reproduce electromagnetic transient in power grid distribution and transmission networks enabling also HIL. Finally, devices deployed in the real world can be integrated into our platform to feed and asses novel services with real data.

6.1.2 Communication Layer

Future MES will be equipped with several internet-connected devices that will communicate, even with each other, to provide a service [140]. Thus, evaluating the impact of data transmission in existing communication networks is of paramount importance to address service requirements even in terms of data transmission latency [67]. In this view, the **Communication Layer** helps in connecting the different entities, either hardware or software, in our co-simulation platform.

As shown in Figure 6.1, the first module is the *MQTT Message Broker*, which is the main actor to allow a bidirectional and asynchronous communication based on the MQTT protocol. Thus, it routes data from publishers to subscribers.

This layer optionally provides different simulators to realistically emulate communication networks. Thus, developers can optionally switch from the real-word MAN to a virtual MAN to study their simulation scenarios also from the data transmission viewpoint, i.e. transmission latency, bandwidth, low-level protocols, and access media (e.g. fiber optic, LTE, NBIoT, and WiMax). Following a methodology in [141, 142], we embedded three main *Communication Network Simulators* that can be alternatively chosen by developers according to their assessments: i) *Mininet* [143], ii) *Omnet++* [144] and iii) *ns-3* [145]. Mininet is a network emulator for building virtual networks consisting of hosts, switches, controllers, and links exploiting both real kernel Linux and real network stack. It is recommended to evaluate end-to-end time latency and the maximum bandwidth of communicating devices. Whilst, Omnet++ and ns-3 are two frameworks for in-depth simulations of different physical layers and protocols in communication networks. They also allow assessments of the overall IP suite. Both Omnet++ and ns-3 are suitable for performing stack message exchange analysis to design innovative protocols. These simulators implement real-time software schedulers, hence also this layer does not introduce significant latency to whole simulation scenarios. In a test-bed co-simulation

environment, the selected communication network simulator can be deployed in a server working as a virtual router where the virtual MAN runs. Thus, all the communication traffic generated by the entities in the platform is routed through this virtual router to analyse delays, congestion, and packet losses. This is valid also for communication flows either based on MQTT or REST.

6.1.3 Information Layer

The **Information Layer** in Figure 6.1 defines data formats and information models for the data exchanged between the entities, either hardware or software, in our co-simulation platform. By default, it proposes open-standard formats that are completely language-independent, such as JavaScript Object Notation (JSON) and eXtensible Markup Language (XML). Both data formats are widely used in distributed software architectures and web applications. In particular, JSON is becoming a standard for data exchange, even among IoT devices, because it is i) lightweight, ii) easy to understand, manipulate and generate, and iii) self-describing and human-readable. Nonetheless, our platform is open in integrating other data formats according to the requirements of the different simulation scenarios.

Regarding the information models, our solution is compliant with both IEC 61850 [146] and ISO 17800 [147] which are the two main standards for making communication between IED and control centers. The former is used for information exchange between the electrical substations and IED. Whilst, the latter is used for data exchange between control systems and end-use IED.

6.1.4 Function Layer

The **Function Layer** in Figure 6.1 includes the main functionality that is commonly used by the other entities in the co-simulation platform, i.e. i) *Asset Management* to manage information related to MES equipment and their geo-referenced locations; ii) *Device Management* to manage interactions between devices and other entities in the platform (e.g. services); and iii) *Data Acquisition and Storage* to store real and/or simulated data coming from other entities in the platform into scalable non-relational database (e.g. MongoDB). This layer defines also the functionality that a general-purpose service provides for smart management of electrical distribution

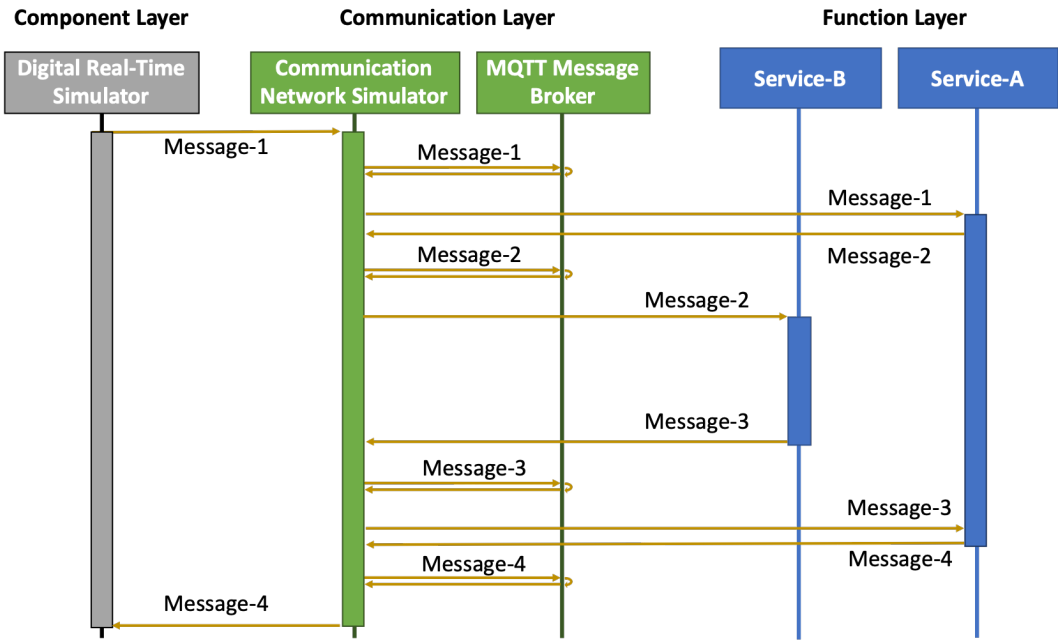


Fig. 6.2 Sequence Diagram of a generic smart grid scenario.

network, such as i) *Optimal Power Flow* and ii) *Decentralised Fault Detection and Isolation, and Centralised Restoration*, a.k.a. De-FDI-CR. The selected scenario to test the distributed event-driven infrastructure is focused on services to make a smart grid (i.e. one of the most challenging MES components) support self-healing and resilience to faults. However, thanks to its modularity, the platform is flexible in defining and including adding new services that can either play concurrently or be replaced in a plug-and-play fashion to add other MES aspects.

6.1.5 Sequence diagram of a generic smart grid scenario

This section describes the communication flow in our platform for a generic MES scenario to be co-simulated. Figure 6.2 reports a sequence diagram with the interaction among the possible actors in the co-simulation, each of them running a different model.

In this example, we suppose that the actors are running in four different servers and a DRTS that are connected in a real dedicated local area network designed to minimise network latency (e.g. by exploiting gigabit Ethernet technology).

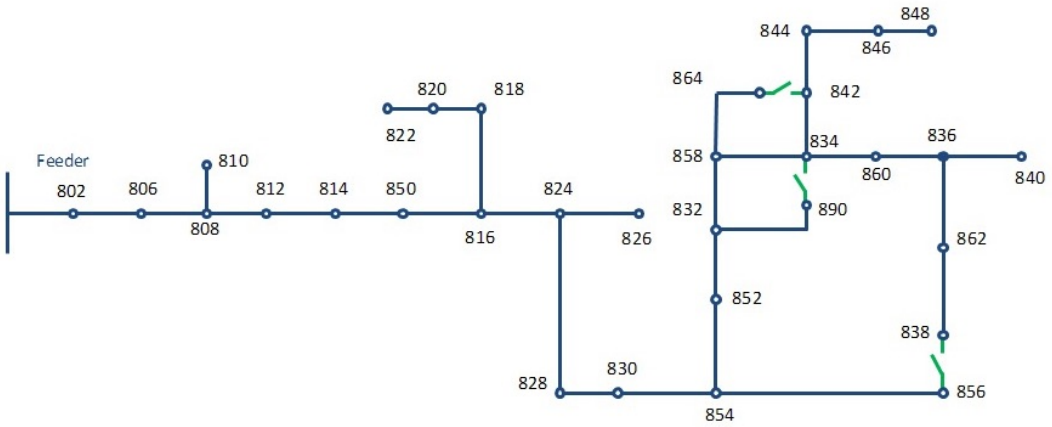


Fig. 6.3 IEEE 34-node radial network with three normally open switches.

As depicted in Figure 6.2, DRTS runs a model reproducing the behaviour of a smart grid distribution network that exchanges information with the other actors in the system by exploiting the MQTT protocol. When DRTS publishes the *Message-1* with, for instance, some alerts or measurements to MQTT Message Broker, this message is routed through the virtual MAN simulated in one of the integrated communication network simulators to realistically assess the delay in the transmission till reaching the subscribed Service-A. To fulfil its computation, Service-A invokes Service-B by sending the *Message-2* again via MQTT routed through the virtual MAN. After a post-process is performed by Service-B, it replays to Service-A sending the resulting *Message-3*. Finally, Service-A ends its computation and sends the *Message-4* with, for example, an actuation command to DRTS.

6.2 Application and Results

This section presents the experimental results of applying our developed Fault Detection Isolation and Restoration (FDIR) scheme on both the IEEE 34-node radial network [148] and a portion of an urban distribution grid, respectively.

6.2.1 IEEE 34-Node Case Study

Figure 6.3 shows a single-line diagram of the IEEE 34-node system. In order to examine different reconfiguration options, we added three Normally OPen (NOP)

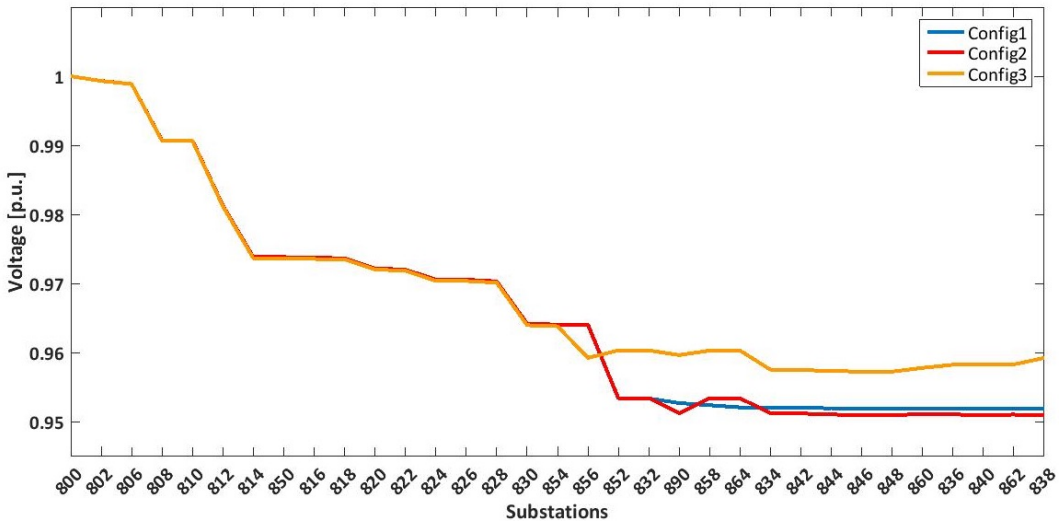


Fig. 6.4 Voltage profiles corresponding to different reconfiguration options - IEEE 34-Node Grid.

switches between nodes 864 and 842, 890 and 834, and 856 and 838. The model of this case is implemented in MATLAB Simulink to be loaded to OPAL-RT real-time simulator for running a real-time simulation. A single-phase to ground fault is triggered between nodes 858 and 834.

After detecting the fault and isolating the branch between nodes 858 and 842, FDIR central agent should decide which NOP to close to not only restore the rest of the network but also minimise the power loss and achieve the best possible voltage profile. Figure 6.4 plots different voltage profiles corresponding to different configurations. *Config1*, *Config2*, and *Config3* are configurations of the network after closing switches of 864-842, 890-834, and 856-838, respectively.

Figure 6.5 compares different power flows over lines considering different configuration options. Results plotted in Figure 6.4 and Figure 6.5 shows that *Config3* has the best system performance comparing to the other options.

To quantify this difference, we calculated the Voltage Deviation Index (VDI) defined as the sum of the square of voltage deviation at each node (Equation 6.1). The voltage deviation index for *Config1*, *Config2*, and *Config3* are $0.0455pu$, $0.0463pu$, and $0.0373pu$, respectively. From power flow results, the total power loss of the system is the least with *Config3*. The total power loss for the three configurations as *Config1*, *Config2*, and *Config3*, is obtained $5011.6kW$, $4921.5kW$, and $4813.5kW$,

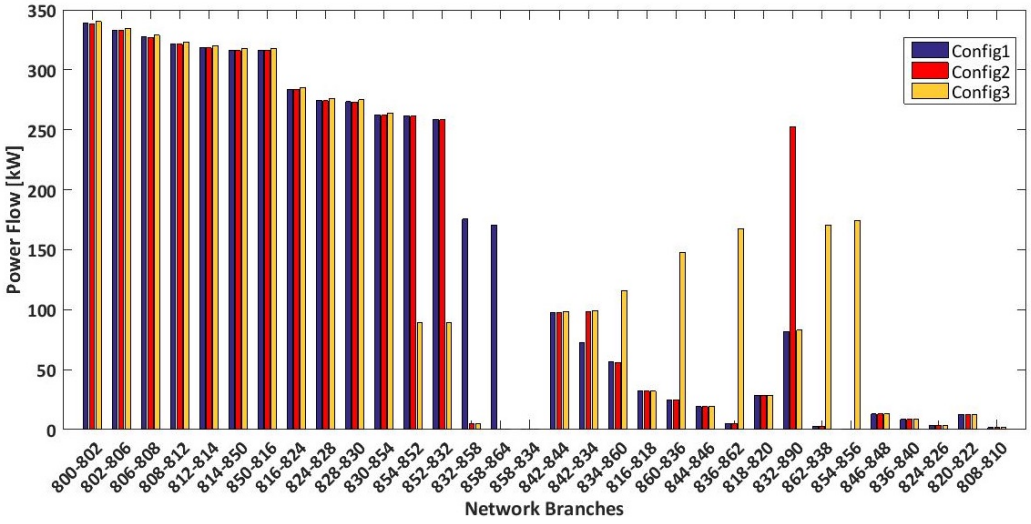


Fig. 6.5 Comparing power flow of the lines after three different reconfiguration cases - IEEE 34-Node Grid.

respectively. According to the results, FDIR finally choose *Config3* to restore the system after fault.

$$VDI = \sum_{i=1}^{N} |V_n - V_i|^2 \tag{6.1}$$

6.2.2 Three-feeder realistic case study in Turin

This section presents the experimental results of the proposed FDIR schema applied to the three-feeder network shown in Figure 6.6. This also demonstrates how our co-simulation platform would support an interoperability analysis of a use case following the GAMES.

Once the fault is triggered, an Inverse Definite Minimum Time over-current relay (IDMT) at MV3 detects the violation and triggers a sectionalizer to disconnect the supply of the rest of the feeder from MV3. Figure 6.7 presents voltage and current signals captured by the relay IED next to the fault location (i.e. at MV3). In our case, all relays are tuned with the same settings of IDMT. Hence in case of neglecting delay of waveform propagation along the line, the occurrence of a fault anywhere on the feeder would trigger all upstream relays. This is intentionally set in order to stress FDIR reaction for more peer-to-peer command exchange. The reaction is

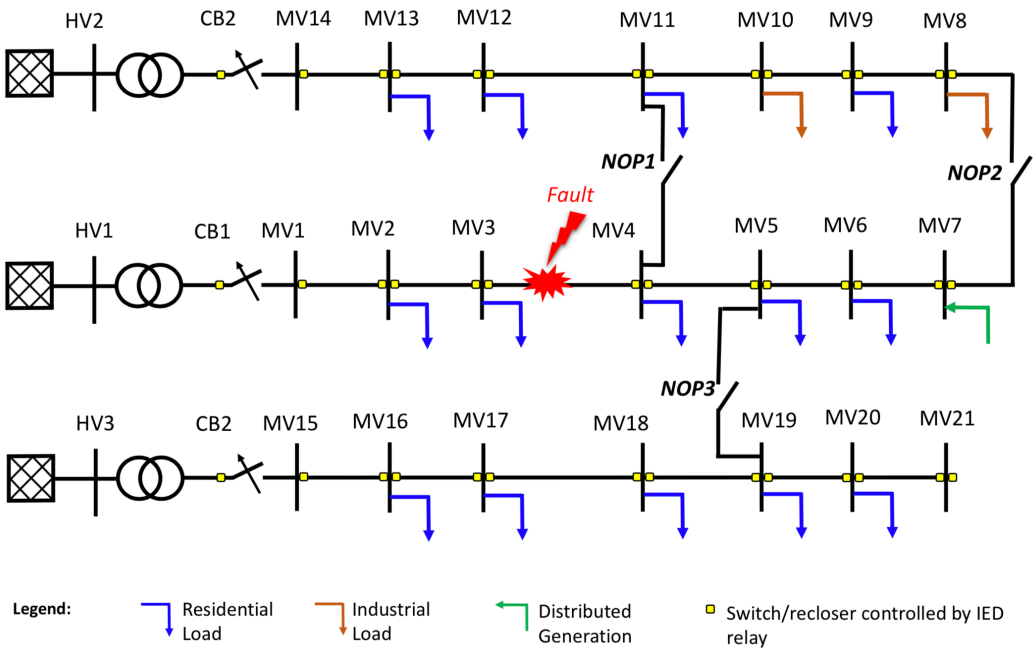


Fig. 6.6 Grid topology for co-simulation tests of De-FDI-CR.

the chain of re-closing commands starting from MV3 to the beginning of the feeder. Re-closing all open switches of the upstream side is a part of supply restoration to re-energise all the loads of the feeder before the location of the fault.

Prior to restoration, the decentralised logic isolates the section between MV3 and MV4. This is done by sending a command from MV3 to MV4 right after detecting the fault.

After isolating the fault, MV3 initiates the chain of peer-to-peer command pass as discussed above. At the same time, MV3 also sends an alarm message to the centralised logic of De-FDI-CR which contains the hardware ID of the IED installed at MV3. This launches the algorithm to create three possible network configurations based on the status of NOP1, NOP2, and NOP3; and select the best option to restore the rest of feeder. The three configurations formed by closing NOP1, NOP2 and NOP3 are named *Config1*, *Config2*, and *Config3*, respectively.

The affected feeder has a distributed generator with $29.05kW$ and $3.1kVAr$ production at the time of the study. All the load and generation values are collected by meter IEDs (i.e. smart meters) installed at the substations. The data is reported every

15 minutes through the communication system. In our case, meter and relay IEDs are the same devices with the three features as well as event recording.

Considering the distributed generator as an uncontrollable generation, an Optimal Power Flow (OPF) is called for the three configurations. We use the standard AC OPF that consists of 18 vectors of voltage angles and magnitudes and 2 vectors of generator real and reactive power injections. Since there are only two controllable generators that belong to the same network operator, the objective would be loss minimisation rather than cost minimisation. The equality constraints are simply the full set of 36 nonlinear real and reactive power balance equations as there are 18 involved nodes. The inequality constraints consist of two sets of 17 branch flow limits as non-linear functions of the bus voltage angles and magnitudes. The variable limits include an equality constraint on any reference bus angle and upper and lower limits on all bus voltage magnitudes and real and reactive generator injections. The limits for the node voltage magnitudes are set between $0.95pu$ and $1.05pu$.

The central logic algorithm is written in Python and it calls MATLAB MATPOWER to execute OPF. Changing the configurations and running OPF for the three cases, resulted in three different sets of power flow of the lines (Figure 6.8). We obtained $8.20kW$, $9.16kW$, and $7.48kW$ power loss for *Config1*, *Config2*, and *Config3*, respectively.

Figure 6.9 is depicted to represent the voltage profile of the three cases; as it is shown in this figure, *Config3* has a much better voltage profile compared to *Config1* and *Config2* in most of the substations.

As for the IEEE 34-node radial network, we calculated the VDI following the Equation 6.1. The voltage deviation obtained is given by the difference between the calculated voltage and the rated voltage equal to $1pu$ or $22volts$. Voltage deviation index for *Config1*, *Config2*, and *Config3* are $0.0012pu$, $0.0021pu$, and $0.0008pu$, respectively.

Since, no violations or reverse power flow are observed in the three cases, all are acceptable choices, however, ranking them based on either power losses or voltage deviation index makes CR send an actuation command to NOP3 to reconfigure the network as *Config3* shown in Figure 6.11.

Figure 6.10 reports the results on communication latency when a fault is located and restored at different congestion rates, neglecting the computation time of

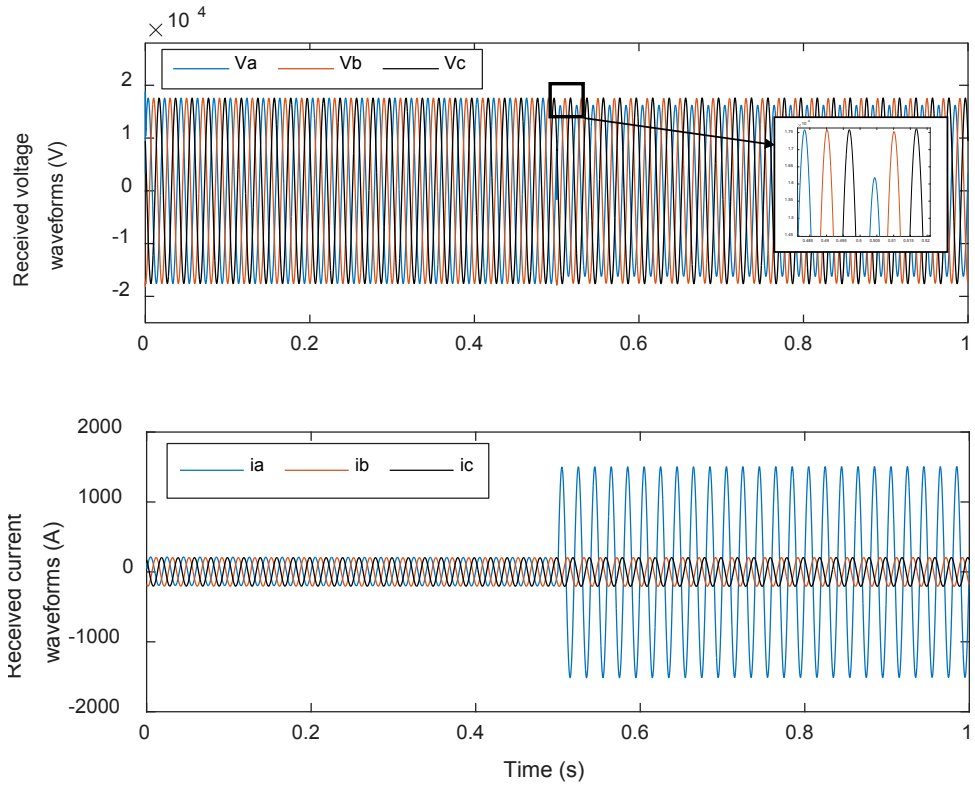


Fig. 6.7 Retrieved waveforms for the single-line to ground fault.

De-FDI-CR service. The computation time of the algorithm did not exceed a few milliseconds (10*ms* to 20*ms*) which is negligible in our case. This is the time needed to send an alarm from smart meters to De-FDI-CR and to send back an actuation command from De-FDI-CR to NOPs via MQTT across the MAN. Both alarms and commands are sent as messages of about 85 bytes each, compliant with the JSON data format.

As shown in Figure 6.10, our analysis started by considering first the MAN with a congestion rate of 1% where routers are almost completely unloaded. Thus, routers process incoming packets almost immediately. In this scenario, the median value of the communication time needed to report and restore a fault is about 40*ms*.

Increasing the overall MAN congestion up to 50% by increasing the background traffic, the median value for the communication latency raised almost constantly without exceeding the 100*ms*. This is due to the impact of the generated background traffic on routers. Indeed, the time spent by a packet in a router before being processed increases, and this creates queues on the routers, as expected.

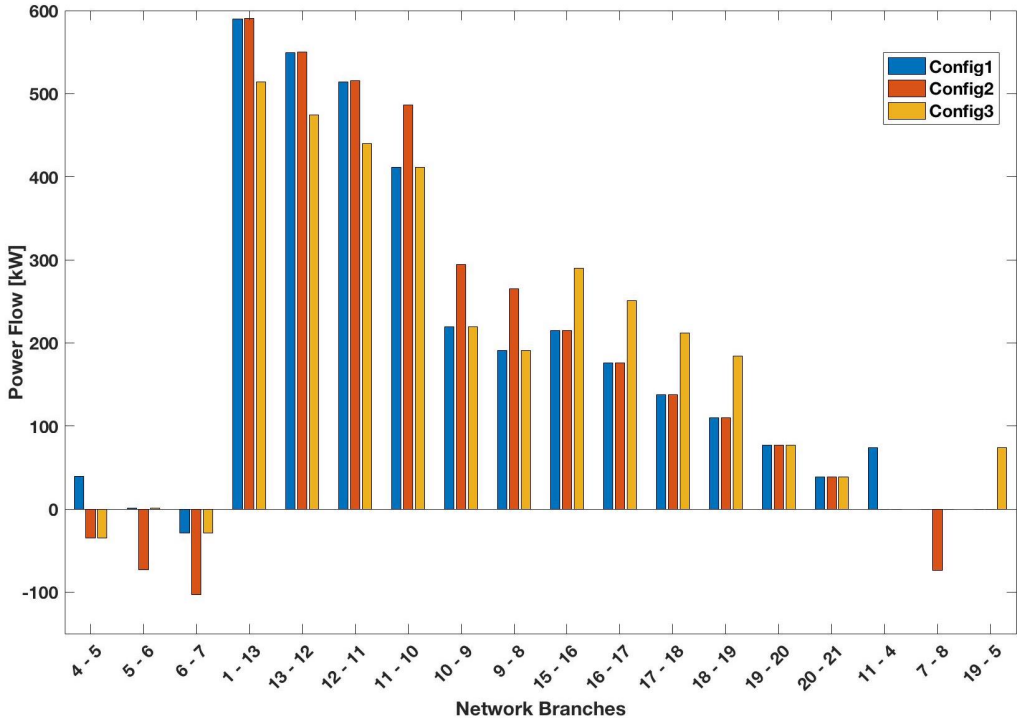


Fig. 6.8 Comparing power-flow of the lines in the three configurations.

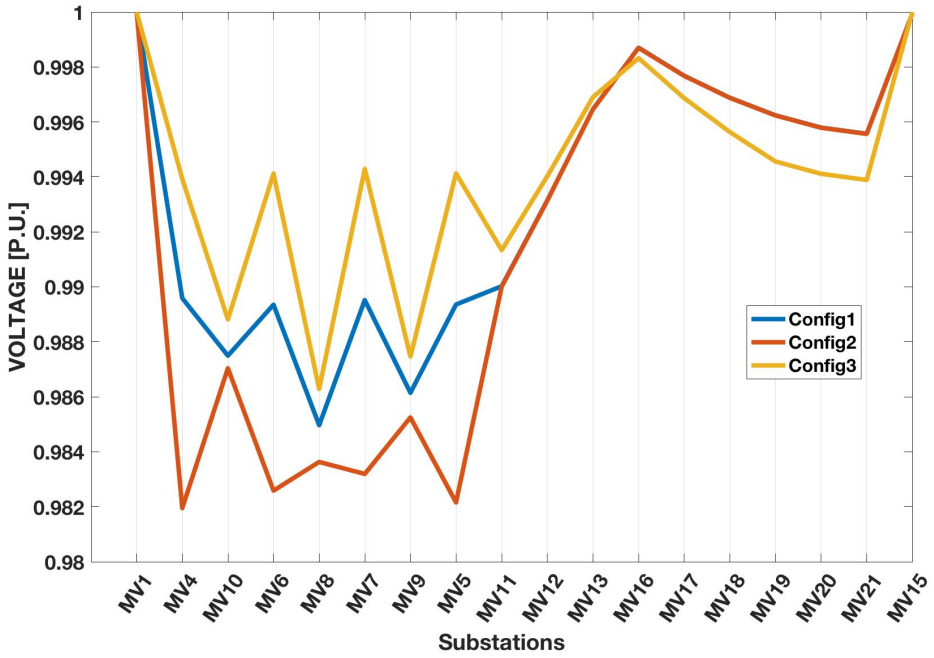


Fig. 6.9 Comparing voltage profiles in the three configurations.

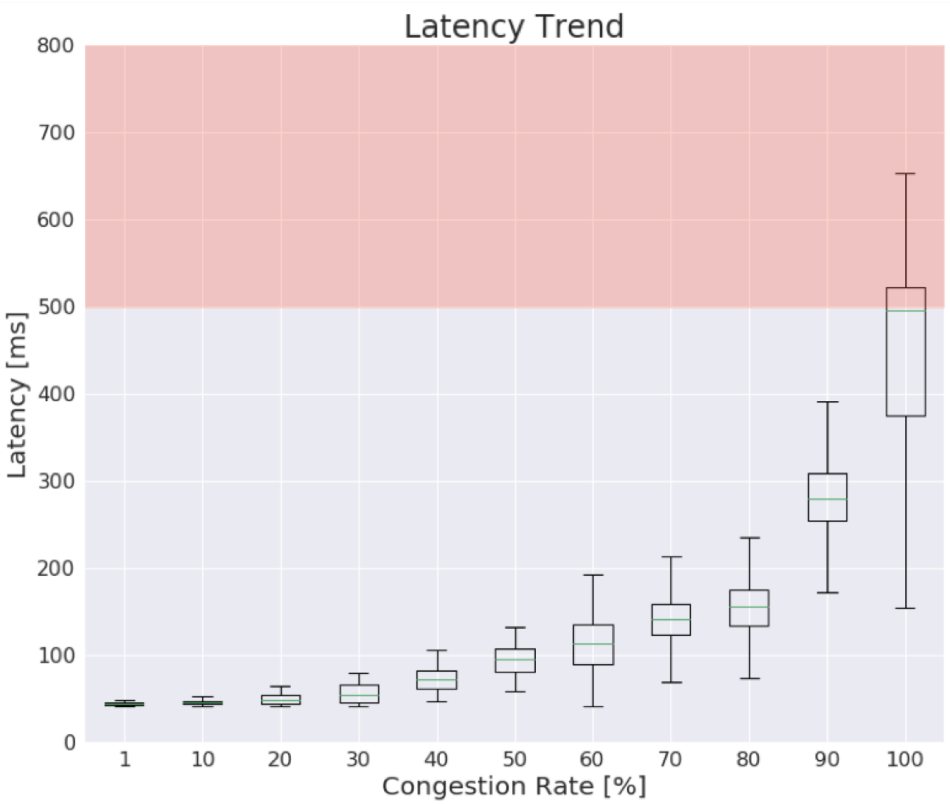


Fig. 6.10 Communication latency at different congestion rates.

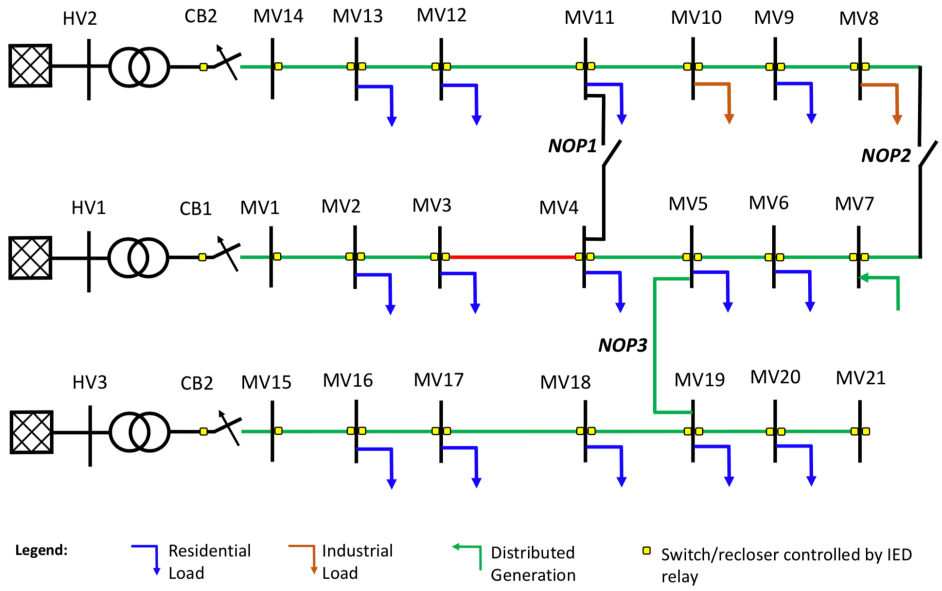


Fig. 6.11 Final reconfiguration of the network for restoration.

Performance Requirements	Performance Classes	Values	Example of services
Transfer Time	TT0	>1000 ms	Files, events, log contents, SCADA
	TT1	1000 ms	Events, alarms
	TT2	500 ms	Operator commands
	TT3	100 ms	Slow automation interactions
	TT4	20 ms	Fast automation interactions
	TT5	10 ms	Releases, status changes
	TT6	3 ms	Trips, blockings

Table 6.1 Communication requirements and performance classes for power systems defined by IEC61850

Growing the congestion from 60% to 80%, the median value for the communication latency slightly increased from almost 110*ms* to 160*ms*. Finally, with MAN congestion equal to 90% and 100% the median time delay is about 280*ms* and slightly lower than 500*ms*, respectively.

The IEC 61850 standard [67] defines the communication requirements to be addressed in the power distribution networks. Thanks to our distributed multi-model co-simulation platform, we are able to evaluate these communication requirements for a specific service (i.e. De-FDI-CR) in a realistic simulation scenario. Indeed, referring to the performance classes in Table 6.1, defined by the standard, our experimental results on time delay satisfy the requirements of classes *TT0*, *TT1* and *TT2*. This is also confirmed when the MAN congestion rate is 100%, which is a very critical situation in a communication network. However, in this very worst scenario, only a few packets exceeded the 500*ms* reaching a max time delay of about 650*ms* (see the red portion in Figure 6.10). It is worth noting that Internet providers and network managers try to avoid this critical situation that, for long periods, could lead to the collapse of the MAN itself.

Chapter 7

Scalability Comparison of Co-simulation Frameworks

The most complicated and debated issue in co-simulation applications is scalability which is defined as the property of a co-simulation framework to handle an increasing amount of heterogeneous Simulators and their model instances, considering the composite relationships that interconnect them together to run a large-scale complex system, such as a Multi-Energy System (MES). From an Information and Communication Technology (ICT) perspective, scalability is measured typically with three indicators known as scalability dimensions: *i)* size, *ii)* geographical, and *iii)* administrative scalability. Size scalability represents the issues in growing the dimension of the co-simulated system and what are the possible solutions to manage the high number of Simulators and Model Instances to run a huge complex Scenario and its orchestration. Geographical scalability, on the other hand, is the representation of the complexity of managing an increasing number of geographically distributed computational nodes (e.g., different laboratories) to implement a co-simulation Scenario. Finally, administrative scalability represents the difficulties in managing a co-simulation framework when dealing with increasing both previous scalability dimensions, thus the engineering effort required to avoid the complex setup of the co-simulation framework, the orchestrator, the distribution of Simulators and their Model Instances among network nodes, and their interconnections.

The two main scaling directions of a co-simulation framework are *i)* vertical scaling and *ii)* horizontal scaling, as illustrated in Figure 7.1. Vertical scaling takes

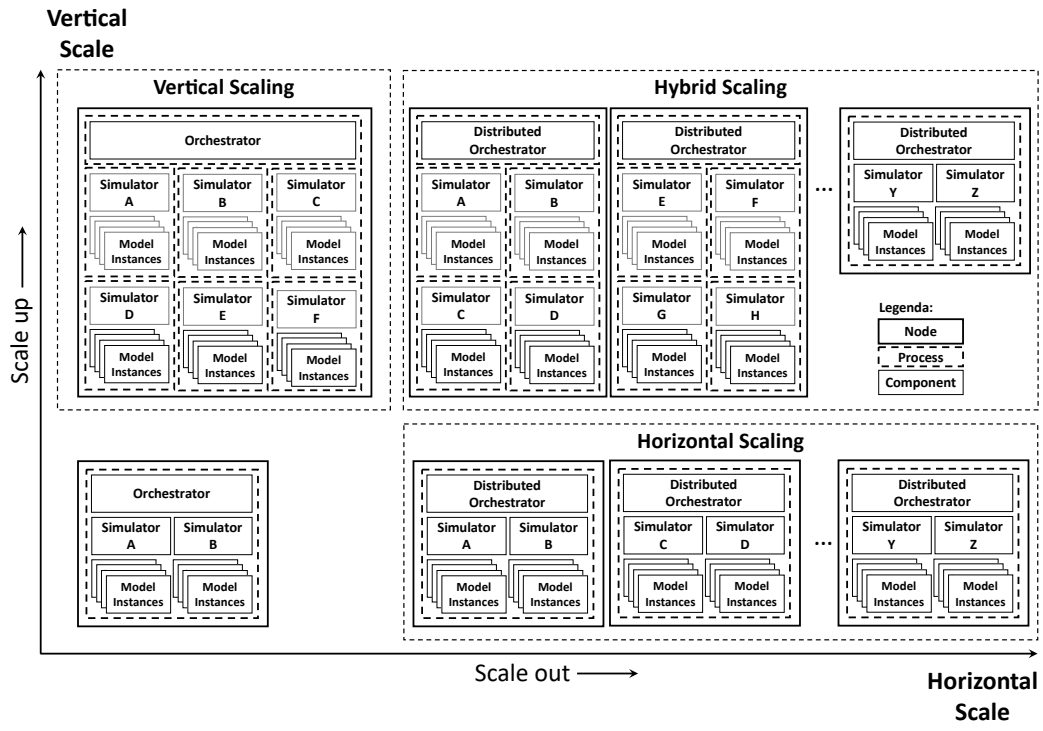


Fig. 7.1 Vertical and Horizontal Scaling

advantage of the parallel capabilities of a single node to distribute the co-simulated Scenario across multiple processes, each of which runs a certain Simulator. Depending on the Simulators, their Model Instances, and their relationships defined by the Scenario, vertical scaling could be applied with different methods and strategies. It is worth noting that this scaling direction commonly results in limited scaling of the size of the complex system. Conversely, horizontal scaling exploits the distribution of the co-simulation Scenario over multiple network nodes, joining them by means of telecommunication protocols. In this view, different Simulators are distributed over different network nodes that manage their Model Instances. Also in this case, there are different solutions depending on the relationships between the Model Instances of each involved Simulator. This approach requires a distributed co-simulation Orchestrator that can act as a load balancer that distributes tasks and manages data exchange and synchronization of all working nodes. The above two directions of scalability are not mutually exclusive and, instead, are typically used in a hybrid configuration to improve the scalability of a co-simulation framework. Hybrid configuration is an advantage when dealing with particular simulation software and/or hardware needed to simulate a specific component of a complex system.

For instance, a Digital Real-Time Simulator (DRTS) is required in some specific MES Scenarios to perform an Electromagnetic Transient (EMT) analysis of a power grid [149]. This particular hardware acts as a vertical scaling component of the hybrid scaling vision to enable fast real-time simulation of the power grid model. Then, the DRTS will be interconnected with a distributed co-simulation environment running other MES models. This distributed configuration participates in the hybrid scaling vision of implementing horizontal scaling.

For the purpose of evaluating the scalability of the co-simulation frameworks presented in Section 2.4.2, a hybrid scaling approach has been chosen among the three possible options to assess what could be the impact of scaling up and scaling out a generic MES Scenario on a distributed cluster of nodes. The benchmark configurations described in Figure 7.2 are:

1. The *Classic Co-simulation configuration* (see Figure 7.2a) is the common configuration of co-simulation frameworks (i.e. Mosaik and HELICS) where Simulators are run by different cluster nodes handled by the Orchestrator master node that manages their data exchange and synchronization. Each Simulator node manages iteratively its M Model Instances in a single process;

2. The *Multi-process Co-simulation configuration* (see Figure 7.2c) that evolves the classic configuration by enabling a multi-process division of a Simulator node (e.g. Simulator A), replicating it in N Simulator processes (e.g. Simulator $A_1, ..., A_N$). Considering M Model Instances, each Simulator process manages M/N Model Instances;

3. The *Classic Multi-Agent System configuration* (see Figure 7.2b) is the typical configuration of the AIOMAS framework. When dealing with a small number of Agents, AIOMAS exploits the Main Container for each Agent class that is spread on one of the available cluster nodes. Each Main Container manages i) the data exchange with its fellow and its Agents through the RPC protocol and ii) the distributed synchronization through its internal Clock. Likewise Model Instances, Agents are replicated in a single process by applying a concurrent multi-threading. When dealing with a high number of M of Agents, the Main Container (e.g. Main Container A) could delegate to N spawned Containers (e.g. Container $A_1, ..., A_N$) the Agent management, resulting in M/N Agents assigned to each child Container;

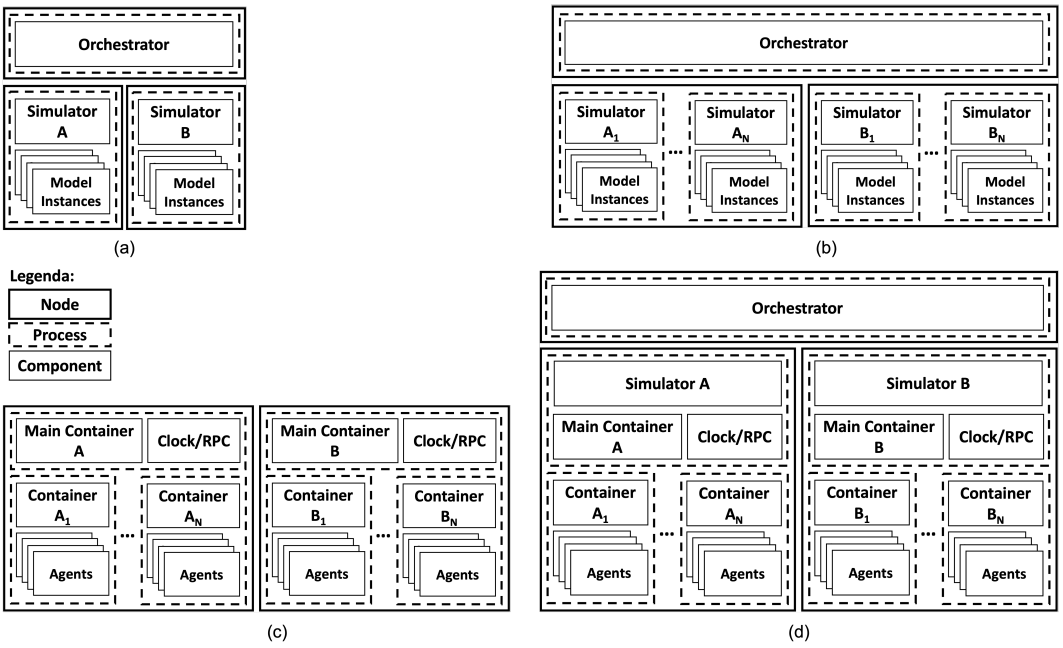


Fig. 7.2 The proposed co-simulation benchmark configurations: (a) Classic Co-simulation, (b) Classic Multi-Agent System, (c) Multi-process Co-simulation, and (d) Classic Co-simulation configuration with encapsulated multi-process Multi-Agent System.

4. The *Classic Co-simulation configuration with encapsulated multi-process Multi-Agent Systems* (see Figure 7.2d) that manages a hybrid configuration of the Classic Co-simulation configuration where each Simulator encapsulates an AIOMAS Main Containers that spawn N child Container in different sub-processes. Each of the child Containers manages M/N Agents, enhancing the scalability of a Classic Co-simulation configuration by applying the Classic Multi-Agent System one and its multi-process implementation with concurrent multi-threading.

7.1 Benchmark Key Performance Index (KPI)

The KPI was defined over a time interval of the main contributions that compose the *Total Execution Time* of a co-simulated Scenario. These processes are depicted in Figure 7.3 and are: i) the *Scenario Setup Process*, in which the co-simulation framework starts the Orchestrator, initializes Simulators with their Model Instances,

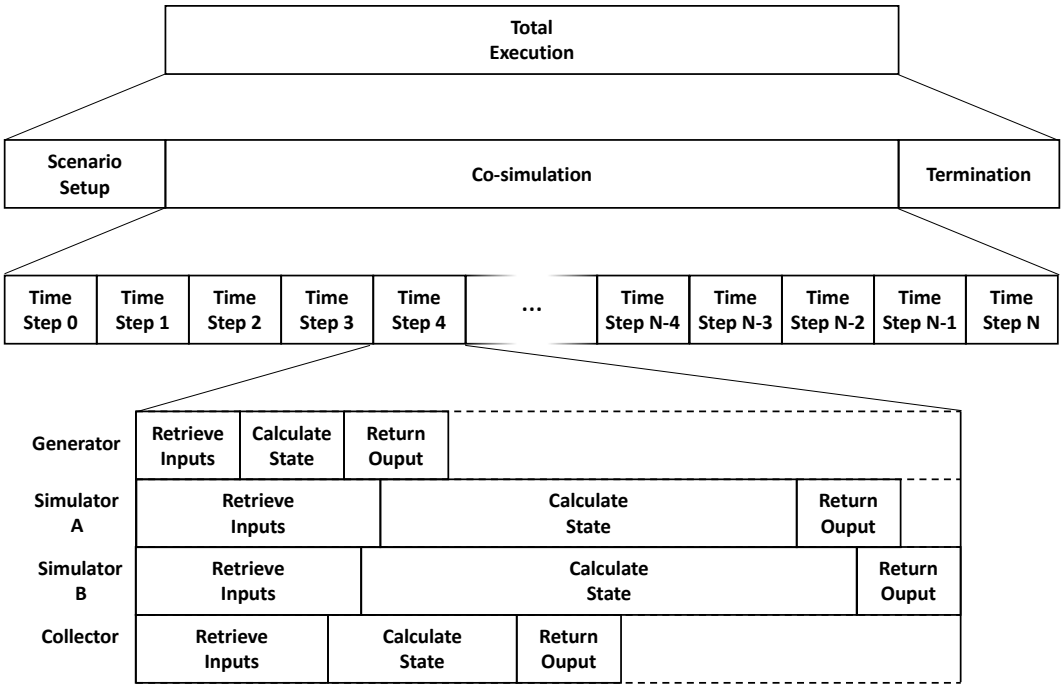


Fig. 7.3 Decomposition of the Total Execution Time in its main contribution: *i)* the Scenario Setup, *ii)* the Co-simulation, and *iii)* the Termination Processes. The former Co-simulation Process could be decomposed in its main time interval units, the Time Steps.

and, finally, links all the Model Instances to deploy the co-simulated Scenario; ii) the *Co-simulation Process* that is an iterative process in which the co-simulated Scenario evolves its state each Time Step (i.e. its fundamental unit); Finally, *iii)* The *Termination Process* in which the co-simulation framework stops the Orchestrator, releases Model Instances and terminates Simulators execution. Each Time Step is a complex routine in which each Simulator retrieves the input dependencies for its Model Instance collection, iteratively executes each Model Instance calculation updating its state, and, finally, collects Model Instance collection outputs to forward them to other Simulators. Simulators operate in parallel as depicted in Figure 7.3 where the execution of the four Simulators is highlighted for Time Step 4. This parallel execution impacts the accuracy of the co-simulated solution with a finite time step latency related to Simulator input/output dependencies that causes negligible inaccuracies of the solution with respect to a standalone simulation [150]. Each Time Step duration could vary depending on the input/output Data Exchange Management, the communication latencies, and the particular computational condition of each cluster node.

Three main time-based KPIs have been employed to evaluate the scalability of each benchmark configuration and its Scenario implementation:

- the *Setup Execution Time* which is the time interval where the co-simulation framework executes the Scenario operations;
- the *Average Time Step Duration* μ_T that is estimated as the maximum of the means of the time duration T of the S co-simulative time steps of each Simulator Sim_i involved in the co-simulation environment I;

$$\mu_T = \max_{i \in I} \mu_T^i \quad where \quad \mu_T^i = \frac{\sum_{n=1}^{S} T_n^i}{S}, \; I = \{Sim_1, ..., Sim_M\} \tag{7.1}$$

- the *Total Execution Time* that includes all the contributions of the Scenario Setup, the Co-simulation, and the Termination Processes.

7.2 Scenario Setup

The proposed benchmark to assess the scalability of the three co-simulation frameworks under analysis applies a general MES Scenario and its implementations are

described in Figure 7.5. The MES Scenario includes four typical actors of a networked urban environment. The four Simulators (or Containers in the AIOMAS perspective) are: i) the Meteo Simulator, ii) the PV Simulator, iii) the Building Simulator, and iv) the Power Grid Simulator. Figure 7.4 shows the four Simulators along with their interactions. Specifically, both the PV and the Building Simulators take weather data from the Meteo Simulator as inputs to perform their calculations. Their outputs are sent to the Power Grid Simulator which uses them as inputs to calculate the power flow of the power grid. The time resolution of the simulators is configurable and could differ from one simulator to another (e.g. Building time model duration of 1 hour and 15 min for the PV Simulators). Each Simulator is presented in the following sections.

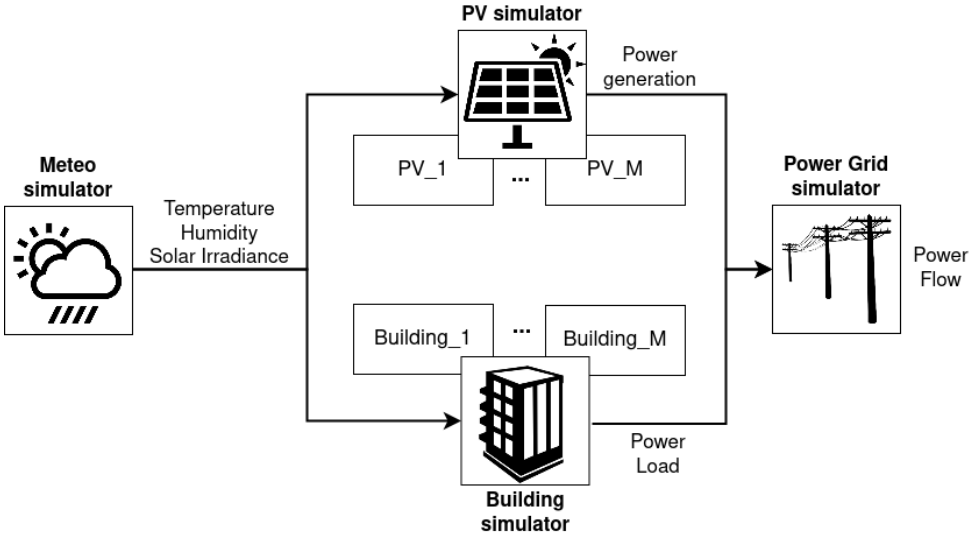


Fig. 7.4 Simulators of the analyzed physical MES Scenario

7.2.1 The Meteo Simulator

The Meteo Simulator is a stateless Simulator and acts as a weather file reader (.epw format), obtaining measurements about the weather (i.e. temperature, solar irradiance, and humidity) for each simulation time step. It forwards these outputs to both PV and Building Simulators.

7.2.2 The PV Simulator

The PV Simulator is a stateless Simulator that uses the *estimate PV energy* service of the model in [118]. It estimates the hourly energy production of PV solar panels for each simulation time step. The given model receives weather conditions and information about the surface area of each panel and uses them to perform its calculations. It also estimates the cell temperature using the so-called *NOCT* method [151] when wind speed is not available as input and, on the other hand, it uses the *Mattei* method [152] when wind speed is available. The output of this simulator is the generating power for the given time step, which is forwarded to the Power Grid Simulator.

7.2.3 The Building Simulator

The Building Simulator is a stateful Simulator that exploits the model presented in [41] to simulate a digital twin of a building equipped with a heat pump system. Within the model, the thermal behaviour of the envelope is treated with a Resistance-Capacitance model [153] and a fine-grained model of the heat pump with all subsystems (e.g., emission, distribution, and generation) is provided. The model calculates the hourly heating demand of the building considering also the heat contributions generated from human occupancy and appliances through archetypes patterns. The heat load is then converted to the heat pump power request. To conclude, the former model has four main functionalities that calculate: i) the real-time heat pump power demand, ii) the heat pump power demand forecasting, iii) the heat pump energy flexibility, and iv) the actuation commands of the heat pump control strategy. The proposed Scenario only uses the heat pump's real-time power demand as output for the Power Grid Simulator. The Building Simulator is stateful because it keeps track of the previous indoor air temperature condition when calculating the heat pump demand for a new time step. The inputs for this simulator are the weather information sent by the Meteo Simulator, while the output is the heat-pump demand load for the given time step, which is forwarded to the Power Grid Simulator.

7.2.4 The Power Grid Simulator

The Power Grid Simulator is a stateless Simulator that emulates a power grid model with different connected loads. The power grid is based on the model presented in [154] and takes advantage of the Python electrical system analysis tool pandapower [155]. This model contains a representation of the network topology and its main components (e.g. buses, lines, transformers) along with their nominal parameters. It is easily configurable and allows the integration of loads, generators, and storage. This Simulator collects the hourly data coming from the PV Simulator and the Building Simulator (i.e. the PV power generation and the Building power load) to solve the power flow analysis of the grid network.

7.2.5 Benchmark Configuration

The benchmark design will consider the general Scenario depicted above. The scalability problem is addressed by rising up the M number of Model Instances (i.e. Agents for AIOMAS framework) of the PV and Building Simulators. The different benchmark configurations will run the generic Scenario on a cluster computing environment of four nodes with a master node serving as Orchestrator when needed. The implementations of the Scenario for each configuration and its cluster deployment are described in Figure 7.5 and are:

1. The *Classic Co-simulation implementation* (see Figure 7.5a) that uses the master node as Orchestrator and the four cluster nodes, one for each of the above-mentioned Simulators. Two cluster nodes manage respectively the Meteo Simulator and the Power Grid Simulator, each one handling its single Model Instance. The remaining nodes manage respectively the PV Simulator and the Building Simulator. Finally, each Simulator iteratively runs M Model Instances in its process;

2. The *Multi-process Co-simulation implementation* (see Figure 7.5b) that uses the same implementation of the previous case (a). However, the cluster nodes, assigned to PV and Building Simulators, replicate on N processes each individual Simulator, equally distributing in each process the M Model Instances. For instance, each of the PV Simulator processes $A_1, ..., A_N$ manages M/N Model Instances;

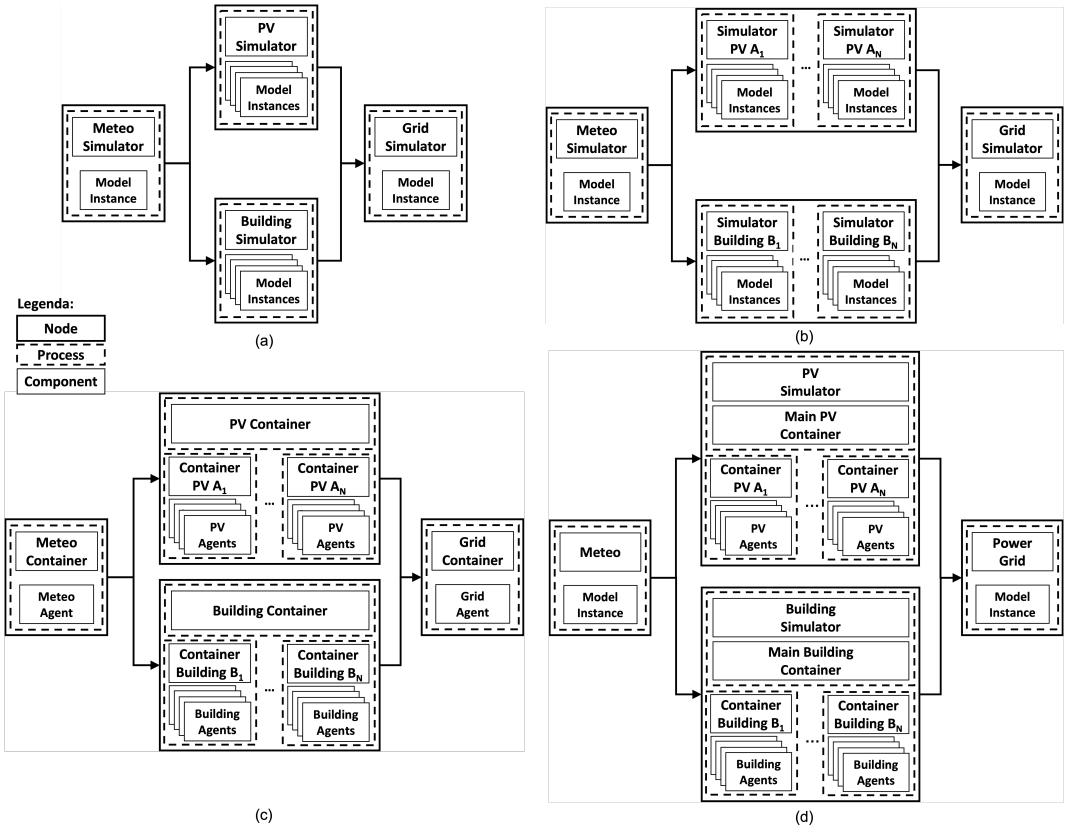


Fig. 7.5 Different implementations of the co-simulation framework benchmark configurations: (a) standalone co-simulation, (b) multi-process co-simulation, (c) standalone MAS, and (d) standalone co-simulation with encapsulated MAS.

3. The *Classic Multi-Agent System implementation* (see Figure 7.5c) that uses the four cluster nodes without the master node. Two cluster nodes implement a simple Container structure that handles respectively the Meteo Agent and the Power Grid Agent. The other two cluster nodes manage respectively the Main PV Container and Main Building Container, each one handling M Agents. Each Main Container spawns N child Containers that handle M/N Agents. For instance, the Main PV Container spawns child Containers $A_1, ..., A_N$ each one managing M/N equally distributed Agents;

4. The *Classic Co-simulation implementation with encapsulated Multi-Process Multi-Agent Systems* (see Figure 7.5d) that uses the same implementation of the Classic Co-simulation case. However, the PV and Building Simulators processes on the two cluster nodes encapsulate the Main Container class that manages the spawning of N child Containers in different sub-processes. Each

of the child Container manages M/N Agents like in the Classic Multi-Agent System implementation of Main PV/Building Containers;

Finally, the analysis was conducted on seven different co-simulation configurations. For each configuration, the M number of Model Instances (PVs and buildings) was scaled from 1, 100, 10 k, 100 k, to 1 M with the exception of configurations involving Mosaik. In fact, Mosaik has a design limitation on the maximum size of data exchanged that did not allow scaling beyond 10 k of Model Instances. The proposed configurations are: i) two configurations of the standalone co-simulation (Figure 7.5a) for HELICS and Mosaik; ii) two configurations of the multi-process co-simulation (Figure 7.5b) for HELICS multi-process and Mosaik multi-process; iii) one configuration of standalone MAS (Figure 7.5c) for AIOMAS; iv) two configurations of the standalone co-simulation with encapsulated MAS (Figure 7.5d) for HELICS-AIOMAS and Mosaik-AIOMAS.

7.3 Experimental Results

The co-simulation benchmark has been deployed in a Virtual Machines environment based upon OpenStack cluster with the specification illustrated in Table 7.1.

Node Name	Core	RAM	Storage	OS	Role
Cloud Master	16	64 Gb	128 Gb	Ubuntu server 20.04.2 LTS	Orchestrator (when needed)
Cloud 1	32	128 Gb	256 Gb	Ubuntu server 20.04.2 LTS	Building Simulators
Cloud 2	32	128 Gb	256 Gb	Ubuntu server 20.04.2 LTS	PV Simulators
Cloud 3	32	128 Gb	256 Gb	Ubuntu server 20.04.2 LTS	Meteo Simulator
Cloud 4	32	128 Gb	256 Gb	Ubuntu server 20.04.2 LTS	Power Grid Simulator

Table 7.1 Summary of the computational nodes used in the simulations

To test the scalability of the different implementations presented in Figure 7.5, the MES Scenario has been used as a baseline for the seven proposed configurations in Section Benchmark Configuration. The co-simulated environment has been run

for seven winter days with an hour resolution. Model Instances of Building and PV Simulators were scaled up. In the next sections, the time-based KPIs mentioned in Section 7.1 are presented to evaluate the comparison among the different co-simulation framework configurations.

7.3.1 Setup Execution Time

The Setup Execution Time considers the time duration of the Scenario configuration operations. In a nutshell, these operations are related to the Initialization task of a general co-simulation framework. In Figure 7.6, the setup execution time performances of the different configurations are presented. Standalone Mosaik keeps the setup time nearly constant around 7 s but it cannot handle more than 10 k instances due to its design limitations. This result highlights that standalone Mosaik is well designed for small co-simulation environments but does not scale up when considering larger environments. Mosaik multi-process instead presents the most time-consuming Initialization task among the benchmarked frameworks. In fact, Mosaik multi-process experiences a big setup time duration difference when rising from 1 instance (i.e. 1 process) to 100 instances (i.e. multiple processes), going from around 6 s to 62 s. Then, the trend flattens out because the same number of parallel processes have been launched when going from 100 to 10 k, saturating the available cores. Standalone HELICS suffers a drastic increase in the setup time from around 7 s to 66 s only when the instance number passes from 100 k to 1 M. The best performing configuration is HELICS multi-process framework which is stable at around 5 s from 1 to 100 k instances. It is worth noting that this framework drastically reduces the time increase of standalone HELICS when going from 100 k to 1 M instances by distributing the computational power for any Simulator over multiple cores, resulting in a setup time of 19 s. For the Mosaik-AIOMAS framework, results are similar to the Mosaik multi-process framework with the main difference that the major time increment occurs when going from 10 s for 100 instances to 61 s for 10 k instances. This is due to the threshold used to decide when AIOMAS starts the creation of the multi-process containers. In our case, this threshold is above 100 Model instances. This behaviour does not occur when coupling HELICS and AIOMAS because launching multi-processes and container creation is handled by HELICS with less computational overhead, following the same HELICS multi-process trend.

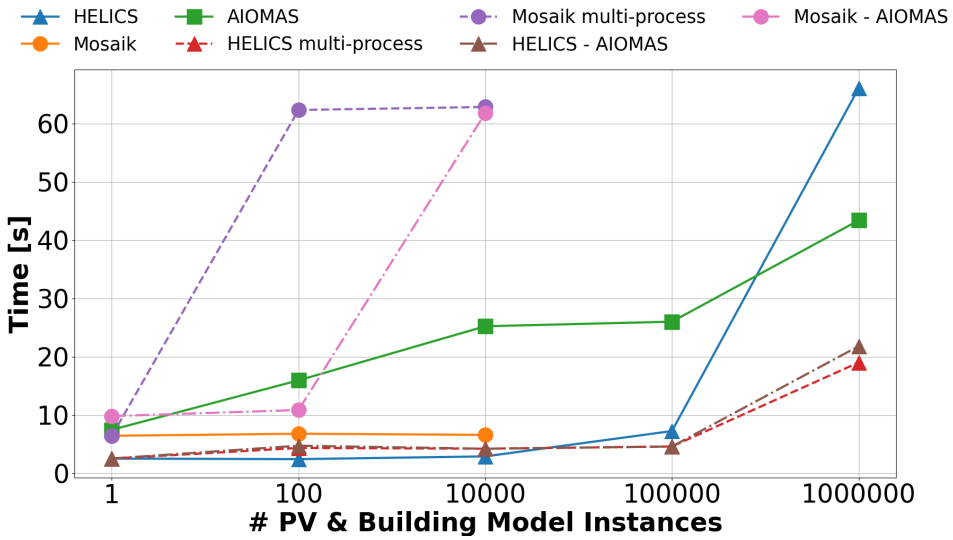


Fig. 7.6 Setup time duration of the different co-simulation frameworks and their configurations

7.3.2 Average Time Step Duration

The average time step duration reports the execution time of a complete time step that is estimated as the maximum among the meantime step duration of every single Simulator. Figure 7.7 reports this KPI for all the benchmarked co-simulation frameworks. The average time step duration differences among the co-simulation frameworks are really small up to 10 k instances, except for Mosaik-AIOMAS and AIOMAS due to the overhead introduced by the message exchange between Agents. Indeed, Mosaik-AIOMAS takes 0.78 s at 10 k instances and AIOMAS takes 0.76 s, while all the others take around 0.1 s. As already mentioned, a comparison with Mosaik is only possible below 10 k Model Instances. Up to this point, the benefits of a multi-processing approach are not significant compared to the respective standalone solutions. The limitations of both standalone and the AIOMAS frameworks are clearly visible when dealing with more than 10 k Model Instances. In fact, standalone HELICS takes about 16 times longer than its multi-process version with 1 M Model Instances. Instead, solutions incorporating AIOMAS even though they perform better than pure HELICS are outperformed by HELICS multi-process. Overall, the configuration with multi-process HELICS performs the best, particularly with paramount performance on the largest scaling Scenario.

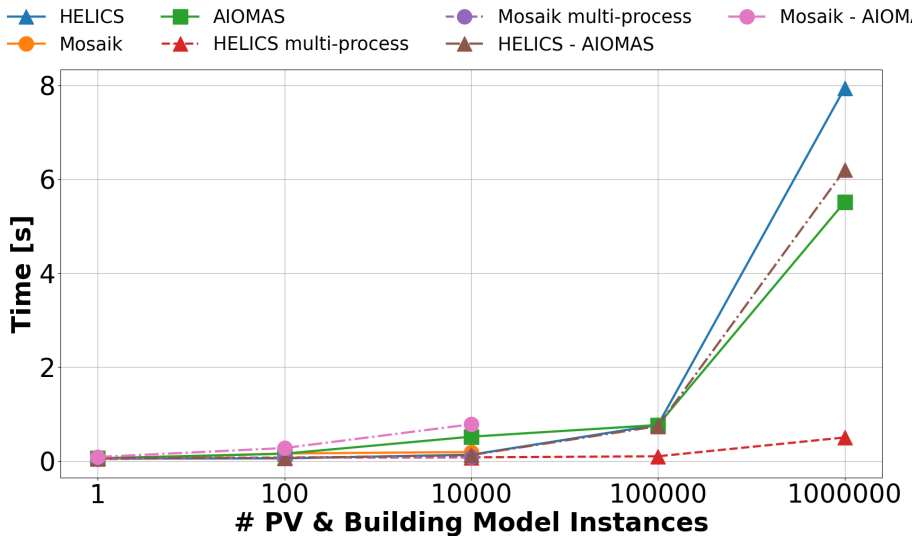


Fig. 7.7 Average time step duration of the different co-simulation frameworks and their configurations

7.3.3 Total Execution Time

Total Execution Time represents the time duration of the entire co-simulation process from Initialization to the end of all the tasks. Figure 7.8 presents trend similarities with the average time step duration KPI in Figure 7.7. In fact, this KPI is a composition of the setup execution time and the sum of the duration of the time steps required to fulfil the co-simulation of the Scenario. When dealing with short simulations, the effect of the setup execution time on the total execution time is larger. Instead, with longer simulations, this effect could be negligible. The KPI trends of the total execution time in Figure 7.8 consolidate the performance considerations made in Section 7.3.2, showing the HELICS multi-process as the best performing framework. Looking at the values at 10 k Model Instances, it is possible to see that Mosaik, Mosaik-AIOMAS, Mosaik multi-process, and AIOMAS diverges about 100 s with respect to other configurations. This is due to the higher setup execution time of these configurations and greater variability of the average time step execution.

7.3.4 Qualitative comparison

Table 7.2 presents a qualitative comparison among the benchmarked co-simulation frameworks. This comparison is done considering only the multi-process version

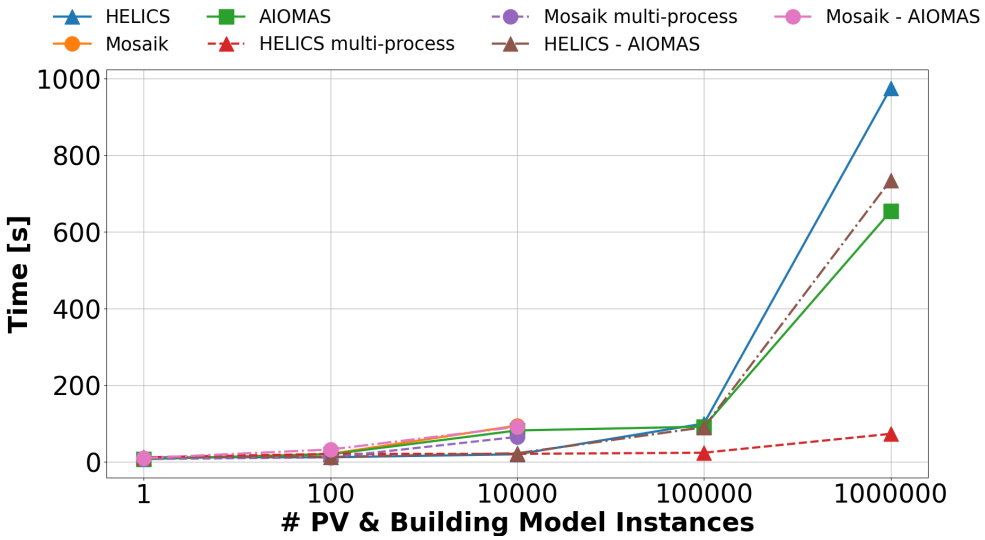


Fig. 7.8 Total execution time of the proposed Scenario for the different co-simulation frameworks and their configurations

of HELICS and Mosaik frameworks because their standalone versions, as well as the hybrid configurations (HELICS-AIOMAS and Mosaik-AIOMAS), do not bring any KPI improvements. The qualitative comparison among the benchmarked co-simulation frameworks presented in Table 7.2 points out different characteristic of the frameworks that can help in choosing the right solution depending on the Scenario under analysis. Concerning the Scenario component (see Scenario in Table 7.2), a first comparison could be made on how the configuration is performed and what effort is required to fulfil the Scenario implementation. Mosaik needs a Python script in which the end user must start the Simulators, instantiate their Models and link them through connectors. This process requires low effort since it is well documented and standardized. HELICS instead uses a JSON file to set up all the required configurations and, thus, has a really low implementation complexity. However, this process could suffer from errors in deploying the correct connection among Model Instances. Finally, AIOMAS requires more effort in the initialization due to the absence of predefined Scenario standards. Linking and instantiating the Models can occur within the Agent definitions or in separate scripts, and is completely up to the end user design. This results in a more complex but freer implementation process.

Simulation management, which is the primary responsibility of the Orchestrator, has different characteristics among the three frameworks (see Orchestrator in

Table 7.2); specifically with respect to communications and synchronization. Mosaik handles synchronization via its Scheduler, limiting or making cumbersome custom intra-step operations. Vice versa, HELICS offers an RTI to manage time regulation and synchronization, allowing a greater freedom of development than the Mosaik Scheduler. In contrast, AIOMAS does not offer a proper Orchestrator and time regulation and synchronization task is managed in a distributed clock implementation, which means that the end user must take into account this task in a programmatic way encapsulating time management into Agents. The communication approach is based on Request/Response for both Mosaik and AIOMAS that exploit respectively TCP and RPC over TCP for communication purposes. HELICS instead uses a Publish/Subscribe approach that could enhance the performance but can lead to wasting resources in data polling requests. To conclude, the three frameworks implement both event-based and time-based simulation paradigms.

Another important aspect is Simulator Integration (see Simulator Integration in Table 7.2). While HELICS and Mosaik allow the integration of a wide range of Simulators based on different programming languages and simulation software, AIOMAS has no particular API for integrating other programming languages than Python and, if necessary, this must be implemented ad-hoc by the end user. Therefore, the integration of the Simulators in Mosaik and HELICS results in a low-effort task. AIOMAS instead is more complex.

Looking at Scalability (see Scalability in Table 7.2), the three frameworks can deploy Simulators or Containers horizontally by distributing their management on different cluster nodes. However, they differ in the possible implementation of vertical scaling. For instance, Mosaik allows multi-processing but its implementation is manual. HELICS multi-processing instead is handled automatically by the co-simulation framework. AIOMAS provides an automatic concurrent multi-threading of Containers. Conversely, AIOMAS multi-processing requires manual implementation. Finally, the above features and proposed results allow for a comparison of the overall scalability performance. Mosaik performed the worst scalability mainly due to the intrinsic limitation that does not allow to scale up beyond 10 k Model Instances. AIOMAS performed medium scalability limited by the overhead increment when dealing with a rise of Agents. HELICS returned the best scalability, showing that its multi-process management of Simulators is capable of reaching 1 M instances with paramount performances.

Co-simulation Frameworks		Mosaik	HELICS	AIOMAS
Scenario	**Configuration**	Programmatic (scripting)	JSON	Programmatic (Agent-based/ scripting)
	Complexity	Low	Low	High
Orchestrator	**Synchronization**	Scheduler (SimPy)	Scheduler (RTI)	Custom (Distributed clocks)
	Communication paradigm	Request/ Response	Publish/ Subscribe	Request/ Response
	Data Exchange	TCP	ZeroMQ	TCP/RPC
	Tipology	Time-stepped/ Event-based	Time-stepped/ Event-based	Time-stepped/ Event-based
Simulator Integration	**Programming languages / Simulator Software**	Python MATLAB Java C++	Python MATLAB Java C++ Nim	Python
	Integration Complexity	Low	Low	High
Scalability	**Horizontal**	Distributed	Distributed	Distributed
	Vertical	Multi-process (Manual)	Multi-process (Automatic)	Concurrent Multi-threading (Automatic) Multi-processing (Manual)
	Performance	Low	High	Medium

Table 7.2 Qualitative Comparison among Mosaik, HELICS, and AIOMAS

It is worth noting that Mosaik demonstrated to be a valuable solution when dealing with small to medium scale MES scenarios and has been chosen in the implementation of the Pure Software Co-simulation Infrastructure and the Hybrid Co-simulation Infrastructure. Its simplicity in developing a MES scenario results in a low complexity that could help a MES designer in rapid deployment of a MES operational and planning analysis. Moreover, the Mosaik COE capabilities are comparable to the HELICS COE when dealing with time-stepped and event-based simulators. Furthermore, the simulator integration allows the fast interconnection of GPPL and simulator software with low complexity. However, scalability is limited when dealing with large-scale scenarios. HELICS has been recently proposed as a viable competitor of Mosaik by the research community. So, the future implementation of the proposed infrastructures will be upgraded to HELICS COE which will enhance their horizontal and vertical scaling.

Chapter 8

Conclusion

In this dissertation, GAMES is presented as a general-purpose architectural model tools for MES engineering application. The architectural model deals with different challenges identified in the field of MES planning, development and testing. GAMES is divided into a hierarchical infrastructure that follows *"from the black-box to the white-box"* approach. Firstly, it integrates an MBSE architectural model for MES use case description extending SGAM. Then, it allows the systemic description of each model permitting a grey-box description of each component involved in the MES use case through SysML. Finally, GAMES will allow to translate the UML/SysML descriptors into PSM and compiled into DSL code to simulate software components and connect specific hardware into the simulation loop.

Following GAMES architecture, three configurations of the Distributed Multi-Modelling Co-simulation Infrastructure have been proposed to face the execution of MES use case co-simulation:

- The Pure Software Co-simulation Infrastructure is capable to perform multi-modelling scenario simulation of MES, evaluating its energy performance by analysing complex dynamics, operations, and control strategies. The platform permits the integration of several and heterogeneous simulators by coupling Mosaik framework and the FMI standard in a shared and distributed environment. Indeed, the platform applies distributed computing to parallelise simulators' execution of different models. This choice enhances the flexibility and scalability of the overall co-simulation platform, extending feasible interconnections between models and simulators. Moreover, the scenario-making

process was simplified and centralised by adopting a single standardised configuration file managed by the *Scenario Builder* module. In this way, the platform permits to easily plug-and-play one or more models with predefined settings patterns. A demonstration example of a building energy system was designed, and a test-case scenario was presented to test the functionalities, capability, and usability of the co-simulation platform. From the results, it is possible to observe how the platform can co-simulate different aspects of a building with a high level of detail and complexity without losing the flexibility and efficiency of the integrated subsystems solvers.

- The Digital Real-Time Co-simulation Infrastructure that is capable to interconnect different commercial DRTS by means of Aurora 8B/10B communication protocol. The infrastructure reduces the communication latency experienced by the data exchange among DRTS, allowing to run EMT analysis of a SoS co-simulated power system scenario. Moreover, the infrastructure offers IEEE1588 PTP standard as a synchronization method to avoid misalignment of the real-time executions, permitting to compare results coming from every single DRTS for logging and post-processing purposes. The infrastructure proposes the PHIL ITM IA as the most simple method to split the power system scenario into sub-models each of them runs by the interconnected DRTS, following a SoS approach. Similar to a PHIL set-up, ITM IA ensures stability and accuracy of the numerical solution of a PSUT within the following constraint: $Z_A/Z_B << 1$. Furthermore, using the Aurora protocol for communication helps reduce the latency effect on the ITM IA and therefore improves its stability and accuracy. The communication latency is calculated by exploiting the ICT infrastructure of DRTSs under analysis, highlighting the internal data exchange among CPU and FPGA to correctly manage Aurora 8B/10B communication protocol. The simple PSUT has been run in a worst-case scenario with a simulation time step of 500 µs to assess the time-domain accuracy of the solution in both stability and near the instability regions of the ITM IA. The proposed infrastructure ensures in both cases an acceptable accuracy in reproducing the behaviour of the monolithic electric circuit. As EMT analysis commonly uses smaller time steps, around 50 µs, we assume that our infrastructure allows the EMT analysis of a larger scenario, enhancing the scalability of PSUT. It is worth noting that a smaller time step also allows

for a relaxation of the constraint of the impedance ratio, making it possible to operate with $Z_A/Z_B \approx 1$.

- The Hybrid Co-simulation Infrastructure joins the previous two configuration and allows the interconnection among multi-model simulation software based on DSL (e.g. MATLAB Simulink, Modelica, EnergyPlus), General-Purpose Programming Language (GPPL) (e.g. Python, C++, Java), and commercial Digital Real-Time Simulator (DRTS) (e.g. OPAL-RT). The soft (i.e. simulation software, GPPL) and hard (i.e. DRTS) real-time environments communication is ensured by a near real-time middleware that exploits VILLAS-framework [156] demonstrating the correct data exchange among simulation models and entities with negligible latencies. By employing this disruptive strategy, the infrastructure ensures the correct wall-clock time evolution of the co-simulated MES scenario and respects the real-time constraint of the interconnected DRTS that permits to include fast time-stepped simulation (e.g. power grid model) into the MES scenario. Moreover, the DRTS capabilities enable the HIL and PHIL testing of real-world hardware, creating a powerful testbed for innovative power grid technologies and components. The flexibility provided by the proposed co-simulation platform permits to use it as a virtual test bed for complex MES, as well as conducting experimental research on enabling technologies.

Then, this dissertation proposes the Distributed Event-Driven Platform to demonstrate how this infrastructure could be used to map and simulate new strategies and algorithms with a GAMES-based approach, to then scale up to real-world applications. The integration of DRTS enables the possibility of developing HIL and PHIL setups for testing physical devices and power equipment while integrating new algorithms and strategies. Moreover, the integration of communication network simulators in the platform enables the possibility to test and develop use cases by taking into account the performance of the communication networks and protocols. To conclude, this platform allows the deployment of a test bed scheme for innovative MES services and functionalities in a real-world system by simply switching from the co-simulated environment to the Internet.

Finally, this dissertation compares and tests the scalability of two popular co-simulation orchestrators, namely Mosaik and HELICS. The MAS frameworks AIOMAS have been included to test its scalability as a co-simulation framework.

The comparison is developed using a simple co-simulations MES scenario. The results show that Mosaik has a limitation in the number of connected entities that have stopped at ten thousand instances. However, it presents low complexity in the MES use case design for small to medium scenarios and the integration of simulators from GPPL to software simulators. So, it has been chosen as the principal COE of the Pure Software Co-simulation Infrastructure and its extension towards the Hybrid Co-simulation Infrastructure. On the other hand, HELICS has shown the capability of scaling up to millions of simulator instances with paramount performance. AIOMAS has demonstrated the same scaling capabilities of HELICS w.r.t to the number of instances. However, HELICS outperforms AIOMAS in terms of simulation time if the simulators are executed in multi-process. Based on this scalability assessment, HELICS will be proposed as the future COE implementation of the proposed infrastructures to push forward a multi-purpose, multi-model, multi-time, and multi-scale co-simulation framework for MES analysis.

The proposed co-simulation platform and its variants aim to intelligently optimize the planning and operation of an MES and test new advanced solutions. For example, it is possible to simulate the abundance of different buildings, such as the scenario presented in Section 5.1.2, by considering the dynamic behaviour of energy networks simulated by a DRTS. These kinds of scenarios can be simulated to analyze and optimize MES use cases, to improve decision making, and to provide appropriate energy policies and interventions at the urban scale. In addition, an agent-based approach can be implemented to model the entities and actors that make up the complex socioeconomic environment, such as distributors, aggregators, and energy communities, who govern their physical systems by interacting with each other. The proposed infrastructure is a disruptive tool that will have a broad impact for both academic research and energy stakeholders, providing a common playing field where they can share their knowledge and evaluate new technologies and solutions for the widespread application of the MES vision. This is possible, and indeed extremely beneficial, to further the energy transition and achieve the ambitious goals of the Glasgow [3] agreement set by the United Nations, reducing our energy footprint to save our planet.

References

[1] UN DESA. World urbanization prospects: The 2018 revision. 2019.

[2] United Nations. Energy, UN-Habitat. https://unhabitat.org/urban-themes/energy/.

[3] Glasgow climate pact.

[4] European Commission. Communication from the commission to the European Parliament, the Council, the European Economic and Social Committee and the Committee of the Regions. Powering a climate-neutral economy: An EU Strategy for Energy System Integration, COM/2020/299 final. https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=COM:2020:299:FIN, 2020.

[5] European Commission. Clean energy for all Europeans. *Luxemburg: Publication Office of the European Union*, 2019.

[6] P. Mancarella. MES (Multi-Energy Systems): An overview of concepts and evaluation models. *Energy*, 65:1–17, 2014.

[7] Carlo Cambini, Raffaele Congiu, Tooraj Jamasb, Manuel Llorca, and Golnoush Soroush. Energy systems integration: implications for public policy. *Energy policy*, 143:111609, 2020.

[8] Laura Abrardi. Behavioral barriers and the energy efficiency gap: a survey of the literature. *Journal of Industrial and Business Economics*, 46(1):25–43, 2019.

[9] Reinhard Haberfellner, Olivier De Weck, Ernst Fricke, and Siegfried Vössner. System Thinking. In *Systems Engineering: Fundamentals and Application*, chapter 1, pages 3–26. Birkhäuser, 2019.

[10] Adam M. Ross, Donna H. Rhodes, and Daniel E. Hastings. Defining changeability: Reconciling flexibility, adaptability, scalability, modifiability, and robustness for maintaining system lifecycle value. *Systems Engineering*, 11(3):246–262, 2008.

[11] INCOSE. Systems Engineering Vision 2020. http://www.incose.org/ProductsPubs/pdf/SEVision2020_20071003_v2_03.pdf.

[12] Steve Cook, Conrad Bock, Pete Rivett, Tom Rutt, Ed Seidewitz, Bran Selic, and Doug Tolbert. Unified modeling language (UML) version 2.5.1. Standard, Object Management Group (OMG), 12.

[13] ISO/IEC 19505-2:2017 Information technology — Object Management Group Unified Modeling Language (OMG UML). https://www.iso.org/standard/52854.html.

[14] System modeling language (SysML) version 1.6. Standard, Object Management Group (OMG), 12.

[15] ISO/IEC 19514:2017 Information technology — Object management group systems modeling language (OMG SysML). https://www.iso.org/standard/65231.html.

[16] ISO/IEC/IEEE 42010:2011 Systems and software engineering — Architecture description. https://www.iso.org/standard/50508.html.

[17] J Bruinenberg, L Colton, E Darmois, J Dorn, J Doyle, O Elloumi, H Englert, R Forbes, J Heiles, P Hermans, et al. CEN CENELEC ETSI Smart Grid Coordination Group on Smart Grid Reference Architecture. *CEN CENELEC ETSI Technical Report*, pages 98–107, 2012.

[18] ISO/IEC 27000 family - Information security management systems. https://www.iso.org/isoiec-27001-information-security.html.

[19] David Wollman, Chris Greer, Dean Prochaska, Paul Boynton, Feffrey Mazer, Cuong Nguyen, Gerald Fitzpatrick, Thomas Nelson, Galen Koepke, Allen Hefner, Victoria Pillitteri, Tanya Brewer, Nada Golmie, David Su, Allan Eustis, David Holmberg, and Steve Bushby. NIST Framework and Roadmap for Smart Grid Interoperability Standards, Release 3.0. 2014.

[20] IEC Central Secretary. Life-cycle management for systems and products used in industrial-process measurement, control and automation. Standard IEC TR 26890, International Electrotechnical Commission, 2013.

[21] IEC Central Secretary. Enterprise-control system integration. Standard IEC TR 62264, International Electrotechnical Commission, 2013.

[22] IEC Central Secretary. Batch control. Standard IEC TR 61512, International Electrotechnical Commission, 1997.

[23] Steven Widergren, Dave Hardin, Ron Ambrosio, R. Drummond, Erich Gunther, Grant Gilchrist, and David Cohen. GridWise Interoperability Context-Setting Framework. 2008.

[24] Mathias Uslar, Sebastian Rohjans, Christian Neureiter, Filip Andrén, Jorge Velasquez, Cornelius Steinbrink, Venizelos Efthymiou, Gianluigi Migliavacca, Seppo Horsmanheimo, Helfried Brunner, and Thomas Strasser. Applying the smart grid architecture model for designing and validating system-of-systems

in the power and energy domain: A european perspective. *Energies*, 12:258, 2019.

[25] E. Mocanu, K. O. Aduda, P. H. Nguyen, G. Boxem, W. Zeiler, M. Gibescu, and W. L. Kling. Optimizing the energy exchange between the smart grid and building systems. In *In Proc. of: 49th UPEC*, pages 1–6, 2014.

[26] L.A. Hurtado, P.H. Nguyen, and W.L. Kling. Smart grid and smart building inter-operation using agent-based particle swarm optimization. *Sustainable Energy, Grids and Networks*, 2:32 – 40, 2015.

[27] G. Schuh, J. Fluhr, M. Birkmeier, and M. Sund. Information system architecture for the interaction of electric vehicles with the power grid. In *In Proc. of: 10th ICNSC*, pages 821–825, 2013.

[28] Dewan Shafiullah, Trung Vo, Phuong Nguyen, and A.J.M. Pemen. Different smart grid frameworks in context of smart neighborhood: A review. pages 1–6, 08 2017.

[29] Peter Palensky, Arjen A Van Der Meer, Claudio David Lopez, Arun Joseph, and Kaikai Pan. Cosimulation of intelligent power systems: Fundamentals, software architecture, numerics, and coupling. *IEEE Industrial Electronics Magazine*, 11(1):34–50, 2017.

[30] Jianhua Zhang, Jeff Daily, Ryan A. Mast, Bryan Palmintier, Dheepak Krishnamurthy, Tarek Elgindy, Anthony Florita, and Bri-Mathias Hodge. Development of helics-based high-performance cyber-physical co-simulation framework for distributed energy resources applications. In *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pages 1–5, 2020.

[31] Cláudio Gomes, Casper Thule, David Broman, Peter Gorm Larsen, and Hans Vangheluwe. Co-simulation: a survey. *ACM Computing Surveys (CSUR)*, 51(3):1–33, 2018.

[32] Gerald Schweiger, Cláudio Gomes, Georg Engel, Irene Hafner, J Schoeggl, Alfred Posch, and T Nouidui. An empirical survey on co-simulation: Promising standards, challenges and research needs. *Simulation modelling practice and theory*, 95:148–163, 2019.

[33] IEEE 1516-2010 - IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)– Framework and Rules.

[34] Ieee standard for modeling and simulation (m amp;amp;s) high level architecture (hla)– framework and rules - redline. *IEEE Std 1516-2010 (Revision of IEEE Std 1516-2000) - Redline*, pages 1–38, Aug 2010.

[35] The Portico Project. Portico RTI. http://www.porticoproject.org/comingsoon/.

[36] Pitch Technologies. Pitch RTI. http://pitchtechnologies.com/products/prti/.

[37] VT MÄK Solution. MAK RTI. https://www.mak.com/products/link/mak-rti.

[38] T. Blochwitz, M. Otter, M. Arnold, C. Bausch, C. Clauß, H. Elmqvist, A. Junghanns, J. Mauss, M. Monteiro, T. Neidhold, D. Neumerkel, H. Olsson, J. v. Peetz, S. Wolf, Atego Systems Gmbh, Qtronic Berlin, Fraunhofer Scai, and St. Augustin. The functional mockup interface for tool independent exchange of simulation models. In *In Proceedings of the 8th International Modelica Conference*, 2011.

[39] Manisa Pipattanasomporn, Hassan Feroze, and Saifur Rahman. Multi-agent systems in a distributed smart grid: Design and implementation. In *2009 IEEE/PES Power Systems Conference and Exposition*, pages 1–8. IEEE, 2009.

[40] Robin Roche, Benjamin Blunier, Abdellatif Miraoui, Vincent Hilaire, and Abder Koukam. Multi-agent systems for grid energy management: A short review. In *IECON 2010-36th Annual Conference on IEEE Industrial Electronics Society*, pages 3341–3346. IEEE, 2010.

[41] Pietro Rando Mazzarino, Claudia De Vizia, Enrico Macii, Edoardo Patti, and Lorenzo Bottaccioli. An agent-based framework for smart grid balancing exploiting thermal flexibility of residential buildings. In *2021 IEEE International Conference on Environment and Electrical Engineering and 2021 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I CPS Europe)*, pages 1–6, 2021.

[42] HSVS Kumar Nunna and Suryanarayana Doolla. Multiagent-based distributed-energy-resource management for intelligent microgrids. *IEEE Transactions on Industrial Electronics*, 60(4):1678–1687, 2012.

[43] Tobias Jung, Payal Shah, and Michael Weyrich. Dynamic co-simulation of internet-of-things-components using a multi-agent-system. *Procedia CIRP*, 72:874–879, 2018.

[44] Ieee standard for modeling and simulation (m amp;s) high level architecture (hla)– object model template (omt) specification. *IEEE Std 1516.2-2010 (Revision of IEEE Std 1516.2-2000)*, pages 1–110, 2010.

[45] Ieee standard for modeling and simulation (m amp;s) high level architecture (hla)– object model template (omt) specification - redline. *IEEE Std 1516.2-2010 (Revision of IEEE Std 1516.2-2000) - Redline*, pages 1–112, 2010.

[46] Patrick Th Eugster, Pascal A Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM computing surveys (CSUR)*, 35(2):114–131, 2003.

[47] Hans-Kristian Ringkjøb, Peter M Haugan, and Ida Marie Solbrekke. A review of modelling tools for energy and electricity systems with large shares of variable renewables. *Renewable and Sustainable Energy Reviews*, 96:440–459, 2018.

[48] Mike Vogt, Frank Marten, and Martin Braun. A survey and statistical analysis of smart grid co-simulations. *Applied energy*, 222:67–78, 2018.

[49] Lorenzo Bottaccioli, Abouzar Estebsari, Enrico Pons, Ettore Bompard, Enrico Macii, Edoardo Patti, and Andrea Acquaviva. A flexible distributed infrastructure for real-time co-simulations in smart grids. *IEEE Transactions on Industrial Informatics*, 13(6):3265–3274, 2017.

[50] Daniele Salvatore Schiera, Francesco Demetrio Minuto, Lorenzo Bottaccioli, Romano Borchiellini, and Andrea Lanzini. Analysis of rooftop photovoltaics diffusion in energy community buildings by a novel gis-and agent-based modeling co-simulation platform. *IEEE Access*, 7:93404–93432, 2019.

[51] Vincent Reinbold, Christina Protopapadaki, Jean-Philippe Tavella, and Dirk Saelens. Assessing scalability of a low-voltage distribution grid co-simulation through functional mock-up interface. *Journal of Building Performance Simulation*, pages 1–13, 2019.

[52] Cornelius Steinbrink, Marita Blank-Babazadeh, André El-Ama, Stefanie Holly, Bengt Lüers, Marvin Nebel-Wenner, Rebeca P Ramírez Acosta, Thomas Raub, Jan Sören Schwarz, Sanja Stark, et al. Cpes testing with mosaik: Co-simulation planning, execution and analysis. *Applied Sciences*, 9(5):923, 2019.

[53] Alok Kumar Bharati and Venkataramana Ajjarapu. SMTD co-simulation framework with HELICS for future-grid analysis and synthetic measurement-data generation. *IEEE Transactions on Industry Applications*, pages 1–1, 2021.

[54] H. W. Dommel. Digital computer solution of electromagnetic transients in single-and multiphase networks. *IEEE Transactions on Power Apparatus and Systems*, PAS-88(4):388–399, 1969.

[55] Christian Dufour, Simone Araújo, and Jean Bélanger. A survey of smart grid research and development involving real-time simulation technology. In *2013 IEEE PES Conference on Innovative Smart Grid Technologies (ISGT Latin America)*, pages 1–8, 2013.

[56] J. Mahseredjian, V. Dinavahi, and J. A. Martinez. Simulation tools for electromagnetic transients in power systems: Overview and challenges. *IEEE Transactions on Power Delivery*, 24(3):1657–1669, 2009.

[57] Kati Sidwall and Paul Forsyth. Advancements in real-time simulation for the validation of grid modernization technologies. *Energies*, 13(16), 2020.

[58] P. G. McLaren, R. Kuffel, R. Wierckx, J. Giesbrecht, and L. Arendt. A real time digital simulator for testing relays. *IEEE Transactions on Power Delivery*, 7(1):207–213, 1992.

[59] C. Dufour, Hoang Le-Huy, J. C. Soumagne, and A. El Hakimi. Real-time simulation of power transmission lines using marti model with optimal fitting on dual-dsp card. *IEEE Transactions on Power Delivery*, 11(1):412–419, 1996.

[60] Xuegong Wang, D. A. Woodford, R. Kuffel, and R. Wierckx. A real-time transmission line model for a digital tna. *IEEE Transactions on Power Delivery*, 11(2):1092–1097, 1996.

[61] O. Devaux, L. Levacher, and O. Huet. An advanced and powerful real-time digital transient network analyser. *IEEE Transactions on Power Delivery*, 13(2):421–426, 1998.

[62] J. A. Hollman and J. R. Marti. Real time network simulation with pc-cluster. *IEEE Transactions on Power Systems*, 18(2):563–569, 2003.

[63] Lok-Fu Pak, M. O. Faruque, Xin Nie, and V. Dinavahi. A versatile cluster-based real-time digital simulator for power engineering research. *IEEE Transactions on Power Systems*, 21(2):455–465, 2006.

[64] Y. Chen and V. Dinavahi. Fpga-based real-time emtp. *IEEE Transactions on Power Delivery*, 24(2):892–902, 2009.

[65] T. Hatakeyama, A. Riccobono, and A. Monti. Stability and accuracy analysis of power hardware in the loop system with different interface algorithms. In *2016 IEEE 17th Workshop on Control and Modeling for Power Electronics (COMPEL)*, pages 1–8, 2016.

[66] Ieee standard for synchrophasor data transfer for power systems. *IEEE Std C37.118.2-2011 (Revision of IEEE Std C37.118-2005)*, pages 1–53, 2011.

[67] IEC61850-5 Edition. Communication networks and systems in substations–part 5: Communication requirements for functions and device models.

[68] M. L. Crow and J. G. Chen. The multirate method for simulation of power system dynamics. *IEEE Transactions on Power Systems*, 9(3):1684–1690, 1994.

[69] Juan Montoya, Ron Brandl, Keerthi Vishwanath, Jay Johnson, Rachid Darbali-Zamora, Adam Summers, Jun Hashimoto, Hiroshi Kikusato, Taha Selim Ustun, Nayeem Ninad, Estefan Apablaza-Arancibia, Jean-Philippe Bérard, Maxime Rivard, Syed Qaseem Ali, Artjoms Obushevs, Kai Heussen, Rad Stanev, Efren Guillo-Sansano, Mazheruddin H. Syed, Graeme Burt, Changhee Cho, Hyeong-Jun Yoo, Chandra Prakash Awasthi, Kumud Wadhwa, and Roland Bründlinger. Advanced laboratory testing methods using real-time simulation and hardware-in-the-loop techniques: A survey of smart grid international research facility network activities. *Energies*, 13(12):1–38, 2020.

[70] E. Bompard, A. Monti, A. Tenconi, A. Estebsari, T. Huang, E. Pons, M. Stevic, S. Vaschetto, and S. Vogel. A multi-site real-time co-simulation platform for the testing of control strategies of distributed storage and v2g in distribution networks. In *2016 18th European Conference on Power Electronics and Applications (EPE'16 ECCE Europe)*, pages 1–9, 2016.

[71] Ettore Bompard, Sergio Bruno, Stefano Frittoli, Giovanni Giannoccaro, Massimo La Scala, Andrea Mazza, Enrico Pons, and Carmine Rodio. Remote phil distributed co-simulation lab for tso-dso-customer coordination studies. In *2020 AEIT International Annual Conference (AEIT)*, pages 1–6, 2020.

[72] Antonello Monti, Marija Stevic, Steffen Vogel, Rik W. De Doncker, Ettore Bompard, Abouzar Estebsari, Francesco Profumo, Rob Hovsapian, Manish Mohanpurkar, Jack David Flicker, Vahan Gevorgian, Siddharth Suryanarayanan, Anurag K. Srivastava, and Andrea Benigni. A global real-time superlab: Enabling high penetration of power electronics in the electric grid. *IEEE Power Electronics Magazine*, 5(3):35–44, 2018.

[73] Marija Stevic, Abouzar Estebsari, Steffen Vogel, Enrico Pons, Ettore Bompard, Marcelo Masera, and Antonello Monti. Multi-site european framework for real-time co-simulation of power systems. *IET Generation, Transmission & Distribution*, 11(17):4126–4135, 2017.

[74] W. Ren, M. Steurer, and T. L. Baldwin. Improve the stability and the accuracy of power hardware-in-the-loop simulation by selecting appropriate interface algorithms. *IEEE Transactions on Industry Applications*, 44(4):1286–1294, 2008.

[75] G. Lauss, F. Lehfuß, A. Viehweider, and T. Strasser. Power hardware in the loop simulation with feedback current filtering for electric systems. In *IECON 2011 - 37th Annual Conference of the IEEE Industrial Electronics Society*, pages 3725–3730, 2011.

[76] P. C. Kotsampopoulos, F. Lehfuss, G. F. Lauss, B. Bletterie, and N. D. Hatziargyriou. The limitations of digital simulation and the advantages of phil testing in studying distributed generation provision of ancillary services. *IEEE Transactions on Industrial Electronics*, 62(9):5502–5515, 2015.

[77] F. Lehfuß, G. Lauss, and T. Strasser. Implementation of a multi-rating interface for power-hardware-in-the-loop simulations. In *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, pages 4777–4782, 2012.

[78] S. Paran and C. S. Edrington. Improved power hardware in the loop interface methods via impedance matching. In *2013 IEEE Electric Ship Technologies Symposium (ESTS)*, pages 342–346, 2013.

[79] J. Siegers and E. Santi. Improved power hardware-in-the-loop interface algorithm using wideband system identification. In *2014 IEEE Applied Power Electronics Conference and Exposition - APEC 2014*, pages 1198–1204, 2014.

[80] Ieee standard for a precision clock synchronization protocol for networked measurement and control systems. *IEEE Std 1588-2019 (Revision ofIEEE Std 1588-2008)*, pages 1–499, 2020.

[81] Steven M. Blair, Mazheruddin H. Syed, Andrew J. Roscoe, Graeme M. Burt, and Jean-Pierre Braun. Measurement and analysis of pmu reporting latency for smart grid protection and control applications. *IEEE Access*, 7:48689–48698, 2019.

[82] Hao Bai, Xueyong Tang, Shuhui Pan, Julong Chen, Changcheng Zhou, Pu Deng, Zhiyong Yuan, and Qingsheng Li. A real-time co-simulation of pv power generation system using transmission line model interface. *Energy Reports*, 8:196–204, 2022. 2021 The 8th International Conference on Power and Energy Systems Engineering.

[83] Renzo Fabián Espinoza, Guilherme Justino, Rodrigo B. Otto, and Rodrigo Ramos. Real-time rms-emt co-simulation and its application in hil testing of protective relays. *Electric Power Systems Research*, 197:107326, 2021.

[84] Chunbo Tian, Haoran Zhao, Junchao Diao, and Bing Li. Electromagnetic and electromechanical hybrid simulation based on mosaik framework. In *2020 IEEE/IAS Industrial and Commercial Power System Asia (I CPS Asia)*, pages 887–891, 2020.

[85] Carlos Queiroz, Abdun Mahmood, and Zahir Tari. Scadasim—a framework for building scada simulations. *IEEE Transactions on Smart Grid*, 2(4):589–597, 2011.

[86] A.A. van der Meer, P. Palensky, K. Heussen, D.E. Morales Bondy, O. Gehrke, C. Steinbrinki, M. Blanki, S. Lehnhoff, E. Widl, C. Moyo, T.I. Strasser, V.H. Nguyen, N. Akroud, M.H. Syed, A. Emhemed, S. Rohjans, R. Brandl, and A.M. Khavari. Cyber-physical energy systems modeling, test specification, and co-simulation based testing. In *2017 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*, pages 1–9, 2017.

[87] Jay Johnson, Ifeoma Onunkwo, Patricia Cordeiro, Brian J. Wright, Nicholas Jacobs, and Christine Lai. Assessing der network cybersecurity defences in a power-communication co-simulation environment. *IET Cyber-Physical Systems: Theory & Applications*, 5(3):274–282, 2020.

[88] Kenneth Hopkinson, Xiaoru Wang, Renan Giovanini, James Thorp, Kenneth Birman, and Denis Coury. Epochs: a platform for agent-based electric power and communication simulation built from commercial off-the-shelf components. *IEEE Transactions on Power Systems*, 21(2):548–558, 2006.

[89] The Network Simulator - ns-2. https://www.isi.edu/nsnam/ns/.

[90] H. Georg, S. C. Müller, C. Rehtanz, and C. Wietfeld. Analyzing cyber-physical energy systems:the inspire cosimulation of power and ict systems using hla. *IEEE Transactions on Industrial Informatics*, 10(4):2364–2373, Nov 2014.

[91] OPNET Network Simulator. https://opnetprojects.com/opnet-network-simulator/.

[92] C. Shum, W. Lau, T. Mao, H. S. Chung, K. Tsang, N. C. Tse, and L. L. Lai. Co-simulation of distributed smart grid software using direct-execution simulation. *IEEE Access*, 6:20531–20544, 2018.

[93] Hua Lin, Santhosh S Veda, Sandeep S Shukla, Lamine Mili, and James Thorp. Geco: Global event-driven co-simulation framework for interconnected power system and communication network. *IEEE Transactions on Smart Grid*, 3(3):1444–1456, 2012.

[94] Chia-han Yang, Gulnara Zhabelova, Chen-Wei Yang, and Valeriy Vyatkin. Cosimulation environment for event-driven distributed controls of smart grid. *IEEE Trans. on Industrial Informatics*, 9(3):1423–1435, 2013.

[95] V. H. Nguyen, Y. Besanger, Q. T. Tran, C. Boudinnet, T. L. Nguyen, R. Brandl, and T. I. Strasser. Using power-hardware-in-the-loop experiments together with co-simulation for the holistic validation of cyber-physical energy systems. In *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, Sept 2017.

[96] D. Bian, M. Kuzlu, M. Pipattanasomporn, S. Rahman, and Y. Wu. Real-time co-simulation platform using opal-rt and opnet for analyzing smart grid performance. In *2015 IEEE Power Energy Society General Meeting*, pages 1–5, July 2015.

[97] V. Venkataramanan, A. Srivastava, and A. Hahn. Real-time co-simulation testbed for microgrid cyber-physical analysis. In *2016 Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*, pages 1–6, April 2016.

[98] Hwantae Kim, Kangho Kim, Seongjoon Park, Hyunsoon Kim, and Hwangnam Kim. Cosimulating communication networks and electrical system for performance evaluation in smart grid. *Applied Sciences*, 8(1):85, 2018.

[99] Markus Mirz, Lukas Razik, Jan Dinkelbach, Halil Alper Tokel, Gholamreza Alirezaei, Rudolf Mathar, and Antonello Monti. A cosimulation architecture for power system, communication, and market in the smart grid. *Complexity*, 2018, 2018.

[100] Steffen Schütte, Stefan Scherfke, and Martin Tröschel. Mosaik: A framework for modular simulation of active components in smart grids. In *Proc. of SGMS11*, pages 55–60. IEEE, 2011.

[101] Modelica Association Project. Functional Mockup Interface. https://fmi-standard.org/.

[102] OFFIS - Institute for Information Technology. Mosaik. https://mosaik.offis.de/.

[103] D. S. Schiera, L. Barbierato, A. Lanzini, R. Borchiellini, E. Pons, E. F. Bompard, E. Patti, E. Macii, and L. Bottaccioli. A distributed platform for multi-modelling co-simulations of smart building energy behaviour. In *2020 IEEE International Conference on Environment and Electrical Engineering and 2020 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I CPS Europe)*, pages 1–6, 2020.

[104] Jacob Chapman, Peer-olaf Siebers, and Darren Robinson. Multi-agent stochastic simulation of occupants for building simulation. In *In Proc. of: Building Simualtion 2017 IBPSA*, 2017.

[105] Daren Thomas, Clayton Miller, Jérôme Kämpf, and Arno Schlueter. Multi-scale co-simulation of energyplus and citysim models derived from a building information model. In *In Proc. Of: IBPSA Bausim 2014*, pages 469–476, 2014.

[106] Younghoon Kwak, Jung-Ho Huh, and Cheolyong Jang. Development of a model predictive control framework through real-time building energy management system data. *Applied Energy*, 155:1–13, 2015.

[107] Andreas Nicolai and Anne Paepcke. Co-simulation between detailed building energy performance simulation and modelica hvac component models. In *In Proc. of: 12th International Modelica Conference, Prague, Czech Republic, May 15-17, 2017*, number 132, pages 63–72. Linköping University Electronic Press, 2017.

[108] Kunpeng Wang, Peer-Olaf Siebers, and Darren Robinson. Towards generalized co-simulation of urban energy systems. *Procedia engineering*, 198:366–374, 2017.

[109] Mengda Jia and Ravi Srinivasan. Building performance evaluation using coupled simulation of energyplus™ and an occupant behavior model. *Sustainability*, 12(10):4086, 2020.

[110] Abbass Raad, Vincent Reinbold, Benoit Delinchant, and Frédéric Wurtz. Energy building co-simulation based on the wrm algorithm for efficient simulation over fmu components of web service. *Proceedings of the BS*, 15, 2015.

[111] Tianzhen Hong, Hongsan Sun, Yixing Chen, Sarah C Taylor-Lange, and Da Yan. An occupant behavior modeling tool for co-simulation. *Energy and Buildings*, 117:272–281, 2016.

[112] European Parliament. Directive 2010/31/EU of the European Parliament and of the Council of 19 May 2010 on the energy performance of buildings, 2010.

[113] Michael Wetter. Co-simulation of building energy and control systems with the building controls virtual test bed. *Journal of Building Performance Simulation*, 4(3):185–203, 2011.

[114] E.ON Energy Research Center. VILLASframework. https://villas.fein-aachen.org.

[115] L. Barbierato, D. S. Schiera, E. Patti, E. Macii, E. Pons, E. F. Bompard, A. Lanzini, R. Borchiellini, and L. Bottaccioli. Games: A general-purpose architectural model for multi-energy system engineering applications. In *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 1405–1410, 2020.

[116] Dassault Systèmes. FMPy Python Library. https://github.com/CATIA-Systems/FMPy.

[117] Weather Underground. Weather Forecast & Reports, Long Range & Local. https://www.wunderground.com.

[118] Lorenzo Bottaccioli, Edoardo Patti, Enrico Macii, and Andrea Acquaviva. Gis-based software infrastructure to model pv generation in fine-grained spatio-temporal domain. *IEEE Systems Journal*, 12(3):2832–2841, 2017.

[119] Lorenzo Bottaccioli, Santa Di Cataldo, Andrea Acquaviva, and Edoardo Patti. Realistic multi-scale modeling of household electricity behaviors. *IEEE Access*, 7:2467–2489, 2018.

[120] Antonello Monti, Marija Stevic, Steffen Vogel, Rik W. De Doncker, Ettore Bompard, Abouzar Estebsari, Francesco Profumo, Rob Hovsapian, Manish Mohanpurkar, Jack David Flicker, Vahan Gevorgian, Siddharth Suryanarayanan, Anurag K. Srivastava, and Andrea Benigni. A global real-time superlab: Enabling high penetration of power electronics in the electric grid. *IEEE Power Electronics Magazine*, 5(3):35–44, 2018.

[121] InfluxData. InfluxDB. https://www.influxdata.com/products/influxdb-overview/.

[122] L. Thurner, A. Scheidler, F. Schafer, J. H. Menke, J. Dollichon, F. Meier, S. Meinecke, and M. Braun. pandapower - an open source python tool for convenient modeling, analysis and optimization of electric power systems. *IEEE Transactions on Power Systems*, 2018.

[123] Daniel Lohmeier, Dennis Cronbach, Simon Ruben Drauz, Martin Braun, and Tanja Manuela Kneiske. Pandapipes: An open-source piping grid calculation package for multi-energy grid simulations. *Sustainability*, 12(23), 2020.

[124] Karsten Ahnert and Mario Mulansky. Odeint–solving ordinary differential equations in c++. In *AIP Conference Proceedings*, volume 1389, pages 1586–1589. American Institute of Physics, 2011.

[125] Daniele Salvatore Schiera, Luca Barbierato, Andrea Lanzini, Romano Borchiellini, Enrico Pons, Ettore Bompard, Edoardo Patti, Enrico Macii, and Lorenzo Bottaccioli. A distributed multimodel platform to cosimulate multienergy systems in smart buildings. *IEEE Transactions on Industry Applications*, 57(5):4428–4440, 2021.

[126] Luca Barbierato, Enrico Pons, Andrea Mazza, Ettore Francesco Bompard, Vetrievel Subramaniam Rajkumar, Peter Palensky, Enrico Macii, Lorenzo Bottaccioli, and Edoardo Patti. Stability and accuracy analysis of a distributed digital real-time cosimulation infrastructure. *IEEE Transactions on Industry Applications*, 58(3):3193–3204, 2022.

[127] Grafana Labs. Grafana. https://grafana.com.

[128] Morteza Vahid-Ghavidel, Mohammad Sadegh Javadi, Matthew Gough, Sergio F Santos, Miadreza Shafie-Khah, and Joao PS Catalao. Demand response programs in multi-energy systems: A review. *Energies*, 13(17):4332, 2020.

[129] Abouzar Estebsari, Matteo Orlando, Enrico Pons, Andrea Acquaviva, and Edoardo Patti. A novel internet-of-things infrastructure to support self-healing distribution systems. In *2018 International Conference on Smart Energy Systems and Technologies (SEST)*, pages 1–6. IEEE, 2018.

[130] CEN CENELEC. CEN CENELEC EN 50160 Voltage characteristics of electricity supplied by public distribution systems. https://www.se.com/ww/library/SCHNEIDER_ELECTRIC/SE_LOCAL/APS/204836_1312/DraftStandard0026rev2-DraftEN501602005-05.pdf.

[131] Matteo Orlando, Abouzar Estebsari, Enrico Pons, Marco Pau, Stefano Quer, Massimo Poncino, Lorenzo Bottaccioli, and Edoardo Patti. A smart meter infrastructure for smart grid iot applications. *IEEE Internet of Things Journal*, 2021.

[132] Luca Barbierato, Abouzar Estebsari, Lorenzo Bottaccioli, Enrico Macii, and Edoardo Patti. A distributed multimodel cosimulation platform to assess general purpose services in smart grids. *IEEE Transactions on Industry Applications*, 56(5):5613–5624, 2020.

[133] M. Fowler and J. Lewis. Microservices. http://martinfowler.com/articles/microservices.html, 2014.

[134] S. Newman. *Building Microservices*. O'Reilly Media, Inc., 2015.

[135] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM CSUR*, June 2003.

[136] Message Queue Telemetry Transport (MQTT). http://mqtt.org.

[137] Roy T Fielding and Richard N Taylor. Principled design of the modern web architecture. *ACM Transactions on Internet Technology*, 2(2):115–150, 2002.

[138] Edoardo Patti, Angeliki Lydia Antonia Syrri, Marco Jahn, Pierluigi Mancarella, Andrea Acquaviva, and Enrico Macii. Distributed software infrastructure for general purpose services in smart grid. *IEEE Transactions on Smart Grid*, 7(2):1156–1163, 2016.

[139] Edoardo Patti, Andrea Acquaviva, and Enrico Macii. Enable sensor networks interoperability in smart public spaces through a service oriented approach. In *Advances in Sensors and Interfaces (IWASI), 2013 5th IEEE International Workshop on*, pages 2–7. IEEE, 2013.

[140] Emilio Ancillotti, Raffaele Bruno, and Marco Conti. The role of communication systems in smart grids: Architectures, technical solutions and research challenges. *Computer Communications*, 36(17-18):1665–1697, 2013.

[141] Christoph P Mayer and Thomas Gamer. Integrating real world applications into omnet++. *Institute of Telematics, University of Karlsruhe, Karlsruhe, Germany, Tech. Rep. TM-2008-2*, 2008.

[142] András Varga. Discrete event simulation system. In *Proc. of the European Simulation Multiconference (ESM'2001)*, 2001.

[143] Mininet. http://mininet.org/.

[144] OMNeT++ Discrete Event Simulator. https://www.omnetpp.org/.

[145] The Network Simulator - ns-3. https://www.nsnam.org/.

[146] Jianchao Zhang, Boon-Chong Seet, and Tek Tjing Lie. An event-based resource management framework for distributed decision-making in decentralized virtual power plants. *Energies*, 9(8):595, 2016.

[147] ISO 17800:2017 - Facility smart grid information model. https://www.iso.org/standard/71547.html.

[148] IEEE 34-node radial network. https://cmte.ieee.org/pes-testfeeders/.

[149] Luca Barbierato, Enrico Pons, Andrea Mazza, Ettore Bompard, Vetrivel Subramaniam Rajkumar, Peter Palensky, Enrico Macii, Lorenzo Bottaccioli, and Edoardo Patti. Stability and accuracy analysis of a distributed digital real-time co-simulation infrastructure. *IEEE Transactions on Industry Applications*, pages 1–1, 2022.

[150] Cornelius Steinbrink, Arjen A van der Meer, Milos Cvetkovic, Davood Babazadeh, Sebastian Rohjans, Peter Palensky, and Sebastian Lehnhoff. Smart grid co-simulation with mosaik and hla: a comparison study. *Computer Science-Research and Development*, 33(1):135–143, 2018.

[151] Fouzia Brihmat and Said Mekhtoub. Pv cell temperature/pv power output relationships homer methodology calculation. In *Conférence Internationale des Energies Renouvelables" CIER'13"/International Journal of Scientific Research & Engineering Technology*, volume 1, pages 0–0. International Publisher &C. O, 2014.

[152] Michel Mattei, Gilles Notton, Christian Cristofari, Marc Muselli, and Philippe Poggi. Calculation of the polycrystalline pv module temperature using a simple method of energy balance. *Renewable energy*, 31(4):553–567, 2006.

[153] Marco Massano, Enrico Macii, Edoardo Patti, Andrea Acquaviva, and Lorenzo Bottaccioli. A grey-box model based on unscented kalman filter to estimate thermal dynamics in buildings. In *2019 IEEE International Conference on Environment and Electrical Engineering and 2019 IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe)*, pages 1–6. IEEE, 2019.

[154] Abouzar Estebsari, Pietro Rando Mazzarino, Lorenzo Bottaccioli, and Edoardo Patti. Iot-enabled real-time management of smart grids with demand response aggregators. *IEEE Transactions on Industry Applications*, 58(1):102–112, 2021.

[155] L. Thurner, A. Scheidler, F. Schäfer, J. Menke, J. Dollichon, F. Meier, S. Meinecke, and M. Braun. pandapower — an open-source python tool for convenient modeling, analysis, and optimization of electric power systems. *IEEE Transactions on Power Systems*, 33(6):6510–6521, Nov 2018.

[156] Steffen Vogel, Vetrivel Subramaniam Rajkumar, Ha Thi Nguyen, Marija Stevic, Rishabh Bhandia, Kai Heussen, Peter Palensky, and Antonello Monti. Improvements to the co-simulation interface for geographically distributed real-time simulation. In *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society*, volume 1, pages 6655–6662, 2019.

Appendix A

List of Publications

Luca Barbierato has been author/coauthor in:

- 7 over 9 accepted paper in peer reviewed international journals
- 11 over 11 accepted paper in peer reviewed international conference
- 1 accepted extended abstract in international conference
- 1 accepted article in a technical national magazine

A.1 Peer Reviewed International Journal

J.1 A Cloud-based Smart Metering Infrastructure for Distribution Grid Services and Automation / Pau, Marco; Patti, Edoardo; **Barbierato, Luca**; Estebsari, Abouzar; Pons, Enrico; Ponci, Ferdinanda; Monti, Antonello. - In: SUSTAINABLE ENERGY, GRIDS AND NETWORKS. - ISSN 2352-4677. - STAMPA. - 15(2018), pp. 14-25. [10.1016/j.segan.2017.08.001]

J.2 Design and accuracy analysis of multi-level state estimation based on smart metering infrastructure / Pau, Marco; Patti, Edoardo; **Barbierato, Luca**; Estebsari, Abouzar; Pons, Enrico; Ponci, Ferdinanda; Monti, Antonello. - In: IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT. - ISSN 0018-9456. - STAMPA. - 68:11(2019), pp. 4300-4312. [10.1109/TIM.2018.2890399]

J.3 A SGAM-Based Test Platform to Develop a Scheme for Wide Area Measurement-Free Monitoring of Smart Grids under High PV Penetration / Estebsari, Abouzar; **Barbierato, Luca**; Bahmanyar, Alireza; Bottaccioli, Lorenzo; Macii, Enrico; Patti, Edoardo. - In: ENERGIES. - ISSN 1996-1073. - 12:8(2019). [10.3390/en12081417]

J.4 A Distributed IoT Infrastructure to Test and Deploy Real-Time Demand Response in Smart Grids / **Barbierato, Luca**; Estebsari, Abouzar; Pons, Enrico; Pau, Marco; Salassa, FABIO GUIDO MARIO; Ghirardi, Marco; Patti, Edoardo. - In: IEEE INTERNET OF THINGS JOURNAL. - ISSN 2327-4662. - STAMPA. - 6:1(2019), pp. 1136-1146. [10.1109/JIOT.2018.2867511]

J.5 A Distributed Multimodel Cosimulation Platform to Assess General Purpose Services in Smart Grids / **Barbierato, Luca**; Estebsari, Abouzar; Bottaccioli, Lorenzo; Macii, Enrico; Patti, Edoardo. - In: IEEE TRANSACTIONS ON INDUSTRY APPLICATIONS. - ISSN 0093-9994. - 56:5(2020), pp. 5613-5624. [10.1109/TIA.2020.3010481]

J.6 A Distributed Multi-Model Platform to Cosimulate Multi-Energy Systems in Smart Buildings / Schiera, Daniele Salvatore; **Barbierato, Luca**; Lanzini, Andrea; Borchiellini, Romano; Pons, Enrico; Bompard, Ettore; Patti, Edoardo; Macii, Enrico; Bottaccioli, Lorenzo. - In: IEEE TRANSACTIONS ON INDUSTRY APPLICATIONS. - ISSN 0093-9994. - ELETTRONICO. - 57:5(2021), pp. 4428-4440. [10.1109/TIA.2021.3094497]

J.7 Stability and Accuracy Analysis of a Distributed Digital Real-Time Co-simulation Infrastructure / **Barbierato, Luca**; Pons, Enrico; Mazza, Andrea; Bompard, Ettore; Subramaniam Rajkumar, Vetrivel; Palensky, Peter; Macii, Enrico; Bottaccioli, Lorenzo; Patti, Edoardo. - In: IEEE TRANSACTIONS ON INDUSTRY APPLICATIONS. - ISSN 0093-9994. - STAMPA. - 58:3(2022), pp. 3193-3204. [10.1109/TIA.2022.3155459]

J.8* Exploring Limits of Distributed Real-Time Power System Simulations via System-of-Systems Co-simulation / **Barbierato, Luca**; Pons, Enrico; Bompard, Ettore; Subramaniam Rajkumar, Vetrivel; Palensky, Peter; Bottaccioli, Lorenzo; Patti, Edoardo. - In: IEEE SYSTEMS JOURNAL. - UNDER REVIEW

J.9* Coupling of Software and Hardware Simulators through a Hybrid Multi-model Co-simulation Infrastructure for Multi-Energy Systems / **Barbierato, Luca**; Schiera, Daniele Salvatore; Orlando, Matteo; Lanzini, Andrea; Pons, Enrico; Bottaccioli, Lorenzo; Patti, Edoardo. - In: IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING. - UNDER REVIEW

A.2 Peer Reviewed International Conference

IC.1 Low Voltage System State Estimation based on Smart Metering Infrastructure / Pau, Marco; Patti, Edoardo; **Barbierato, Luca**; Estebsari, Abouzar; Pons, Enrico; Ponci, Ferdinanda; Monti, Antonello. - ELETTRONICO. - (2016), pp. 1-6. Paper presented at the 2016 IEEE International Workshop on Applied Measurements for Power Systems (AMPS) held in Aachen (DE) from 28 to 30 September 2016 [10.1109/AMPS.2016.7602804].

IC.2 Fault Detection, Isolation and Restoration Test Platform Based on Smart Grid Architecture Model Using Internet-of-Things Approaches / Estebsari, Abouzar; Patti, Edoardo; **Barbierato, Luca**. - (2018), pp. 1-5. Paper presented at the 18th IEEE International Conference on Environment and Electrical Engineering (EEEIC) held in Palermo from 12 to 15 June 2018 [10.1109/EEEIC.2018.8494449].

IC.3 A Distributed Software Solution for Demand Side Management with Consumer Habits Prediction / **Barbierato, Luca**; Bottaccioli, Lorenzo; Macii, Enrico; Grasso, Ennio; Acquaviva, Andrea; Patti, Edoardo. - (2019), pp. 1-6. Paper presented at the 2019 IEEE International Conference on Environment and Electrical Engineering (EEEIC 2019) held in Genoa, Italy from 11 to 14 June 2019 [10.1109/EEEIC.2019.8783512].

IC.4 A Distributed Platform for Multi-modelling Co-simulations of Smart Building Energy Behaviour / Schiera, Daniele Salvatore; **Barbierato, Luca**; Lanzini, Andrea; Borchiellini, Romano; Pons, Enrico; Bompard, Ettore Francesco; Patti, Edoardo; Macii, Enrico; Bottaccioli, Lorenzo. - (2020). Paper presented at the IEEE International Conference on Environment and Electrical Engineering (EEEIC 2020) held in Madrid, Spain from 09 to 12 June 2019 [10.1109/EEEIC/ICPSEurope49358.2020.9160641].

IC.5 GAMES: a General-purpose Architectural model for Multi-Energy System engineering applications / **Barbierato, Luca**; Schiera, Daniele Salvatore; Patti, Edoardo; Macii, Enrico; Pons, Enrico; Bompard, Ettore Francesco; Lanzini, Andrea; Borchiellini, Romano; Bottaccioli, Lorenzo. - (2020), pp. 1405-1410. Paper presented at the 2020 IEEE International Computer Software and Applications Conference (COMPSAC) held in Madrid, Spain from 13 to 17 July 2020 [10.1109/COMPSAC48688.2020.00-59].

IC.6 Stability and Accuracy Analysis of a Real-time Co-simulation Infrastructure / **Barbierato, Luca**; Pons, Enrico; Mazza, Andrea; Bompard, Ettore Francesco; Subramaniam Rajkumar, Vetrivel; Palensky, Peter; Macii, Enrico; Bottaccioli, Lorenzo; Patti, Edoardo. - ELETTRONICO. - (2021), pp. 1-6. Paper presented at the 21st IEEE International Conference on Environmental and Electrical Engineering (EEEIC 2021) held in Bari, Italy from 7 to 10 September 2021 [10.1109/EEEIC/ICPSEurope51590.2021.9584687].

IC.7 Load Profiles Clustering and Knowledge Extraction to Assess Actual Usage of Telecommunication Sites / Eiraudo, Simone; **Barbierato, Luca**; Giannantonio, Roberta; Porta, Alessandro; Lanzini, Andrea; Borchiellini, Romano; Macii, Enrico; Patti, Edoardo; Bottaccioli, Lorenzo. - (2021), pp. 1-6. Paper presented at the 21st IEEE International Conference on Environmental and Electrical Engineering (EEEIC 2021) held in Bari, Italy from 7 to 10 September 2021 [10.1109/EEEIC/ICPSEurope51590.2021.9584633].

IC.8 Synthetic Ground Truth Generation of an Electricity Consumption Dataset for Anomaly Detection / Mascali, Lorenzo; Eiraudo, Simone; **Barbierato, Luca**; Schiera, Daniele Salvatore; Giannantonio, Roberta; Patti, Edoardo; Bottaccioli, Lorenzo; Lanzini, Andrea. - (2022), pp. 1-6. Paper accepted at the 5th International Conference on Smart Energy Systems and Technologies (SEST 2022) held in Eindhoven, The Netherlands from 5 to 7 September 2022

IC.9 A Neural Network-based Methodology for Non-Intrusive Energy Audit of Telecom Sites / Eiraudo, Simone; **Barbierato, Luca**; Giannantonio, Roberta; Patti, Edoardo; Bottaccioli, Lorenzo; Lanzini, Andrea. - (2022), pp. 1-6. Paper accepted at the 5th International Conference on Smart Energy Systems and Technologies (SEST 2022) held in Eindhoven, The Netherlands from 5 to 7 September 2022.

IC.10 Non-Intrusive Load Disaggregation of Industrial Cooling Demand with LSTM Neural Network / Eiraudo, Simone; **Barbierato, Luca**; Giannantonio, Roberta; Patti, Edoardo; Bottaccioli, Lorenzo; Lanzini, Andrea. - (2022), pp. 1-6. Paper accepted at the 22st IEEE International Conference on Environmental and Electrical Engineering (EEEIC 2022) held in Bari, Italy from 28 June to 1 July 2022.

IC.11 COMET: Co-simulation of Multi-Energy Systems for Energy Transition / **Barbierato, Luca**; Schiera, Daniele Salvatore; Scaccia, Rossano; Margara, Alessandro; Bottaccioli, Lorenzo; Patti, Edoardo. - (2022), pp. 1-6. Paper accepted at the 2022 IEEE International Computer Software and Applications Conference (COMPSAC) held in Turin, Italy from 27 June to 1 July 2022.

A.3 Others

O.1 Design and development of a co-simulation platform for Multi-Energy System analysis / **Barbierato, Luca**; Bottaccioli, Lorenzo; Macii, Enrico; Acquaviva, Andrea; Patti, Edoardo. - (2019), pp. 36-36. Extended abstract presented at the 4th Energy for Sustainability International Conference - Designing a Sustainable Future (EFS 2019) held in Turin, Italy from 24 to 26 July 2019.

O.2 Analysis of the flexibility of residential electricity demand in Italy / **Barbierato, Luca**; Bella, Valter; Bellifemine, Fabio; Gallanti, Massimo; Maggiore, Simone. - In: L'ENERGIA ELETTRICA. - ISSN 0013-7308. - 91:4(2014), pp. 105-116.