Laboratory Infrastructure Integration and Automation

A. Acosta, G. Silano, G. Paludetto, O. Gehrke, M. C. Pham, Q. T. Tran, V. Rajkumar, S. Vogel, and A. Monti

Abstract This chapter presents a software suite to support the implementation of distributed Research Infrastructure experiments. These tools provide communication interoperability and support Configuration Management. Moreover, we propose a laboratory middleware that extends these functionalities with the concept of Research Infrastructure as Code.

A. Acosta (✉) · A. Monti
RWTH Aachen University, Aachen, Germany
e-mail: andres.acosta@eonerc.rwth-aachen.de

A. Monti
e-mail: amonti@eonerc.rwth-aachen.de

G. Silano · G. Paludetto
Ricerca sul Sistema Energetico - RSE S.p.A, Milan, Italy
e-mail: giuseppe.silano@rse-web.it

G. Paludetto
e-mail: gabriele.paludetto@rse-web.it

O. Gehrke
Technical University of Denmark, Kgs. Lyngby, Denmark
e-mail: olge@dtu.dk

M. C. Pham · Q. T. Tran
French Alternative Energies and Atomic Energy Commission, Paris, France
e-mail: minh-cong.pham@cea.fr

Q. T. Tran
e-mail: quoctuan.tran@cea.fr

V. Rajkumar
Delft University of Technology, and TenneT TSO B.V, Delft, The Netherlands
e-mail: v.rajkumar@tudelft.nl

S. Vogel
OPAL-RT Germany, Nürnberg, Germany
e-mail: steffen.vogel@opal-rt.com

T. I. Strasser et al. (eds.), *European Guide to Smart Energy System Testing*, SpringerBriefs in Energy, https://doi.org/10.1007/978-3-031-99451-7_6

1 Large-Scale Integration of Laboratories

The work presented in this chapter builds on previous experience with experiments involving multiple RTIs in a variety of different configurations and across organisational boundaries [11]. This includes the real-time coupling of multiple geographically distributed physical laboratories as well as the coupling between physical laboratories and different types of simulators running in real-time [10], primarily in CHIL and PHIL configurations (see in chapter "Improved Hardware-in-the-Loop-based Testing").

One of the main insights gained in the process is that there are both technical and fundamentally non-technical obstacles to RTI integration. The former relate to the mechanics of moving data and/or signals from A to B, through firewalls and across large distances (data transport), while the latter are primarily artefacts of the meeting of different organisations, procedures and their underlying assumptions. Usually, the development focus tends to be on the former, leading to an ad-hoc, as-needed approach, which yields quick early results but soon becomes inefficient without addressing the latter.

One of the specific challenges of the CPES field is the large range of possible answers to the question "What is a Smart Energy testing laboratory?". Of special importance to RTI integration is the broad range of automation levels across RTIs and the large variety of approaches to laboratory automation. Therefore, this chapter presents several technical solutions to overcome these issues.

2 Preliminaries and Definitions

RTIs often focus on research and testing of CPES applications, feature tools and equipment for offline simulation, real-time simulation, and HIL. Moreover, communications are emulated, for instance, for testing CPES applications [1]. Consequently, RTIs often require defining tools and methods to provide interoperability, interconnect equipment from different vendors, and support multi-protocol communications. This is achieved through a combination of SCADA systems and custom solutions, e.g., those based on REST APIs. Several technological advancements led to the idea of geographically distributed interconnection of laboratories, which allows resource, more complex scenarios, and multidisciplinary collaboration. However, when the interconnection takes place over the public internet, there are challenges to keeping real-time constraints and providing cybersecurity, among others. Therefore, in RTI integration scenarios, we can distinguish between two types of communication:

- *Internal Communications*: As mentioned above, this communication takes place inside the RTI. It interconnects Digital Real-Time Simulator (DRTS) platforms, and CHIL and power equipment. Sometimes, the RTI can also feature a SCADA system and other tools for communication, such as APIs and custom scripts. Here, *adapter* refers to any module that provides interoperability between communication protocols at the local level. This concept will be refined later in Sect. 3.2.

- *Transport Communications*: Enable interconnection between geographically distributed RTIs. The underlying tools are called *Transports*. Several solutions providing the functionality of Transports have been implemented, taking into account different requirements. In the power and energy systems community, multiple efforts have resulted in tools such as HELICS [3], the Distributed Co-simulation Protocol [4], and OpSim [13]. ERIGrid 2.0 focuses on VILLASnode [6], JaNDER [7], and AIT's Lablink [9] frameworks.

As the number of devices and interconnections in a given RTI increases, the organization and configuration of laboratory equipment become more difficult. Aspects like a unified and automated naming convention for signal exchange, signal units and their scaling, and the way internal communications are implemented become more relevant as the number of participating RTIs increases in the distributed experiment [12]. This led to the idea of defining a set of tools to simplify the RTI integration process with the ability to enhance and extend the capabilities of the existing transport tools. These tools are presented below.

3 Tools for Research Infrastructure Integration

The two main types of communications involved in a multi-RI experiment (i.e., internal and transport communications) are outlined above. This section introduces three available transport tools and the universal API (uAPI) for internal communications. Moreover, the uAPI enables interoperability between transports and some of its main functionalities are shown through tests.

3.1 Laboratory Coupling Tools

The main purpose of these tools is to provide transport communications. As mentioned above, different projects and initiatives led to the implementation of multiple lab coupling tools, each of them with its advantages and disadvantages. The ERIGrid 2.0 project had the ambitious goal of demonstrating the interoperability of multiple lab coupling tools in distributed RTI experiments. In particular, three transport tools, which are described as follows, are considered:

- *VILLASnode* [2] is a flexible high-performance gateway supporting more than 20 communication protocols. VILLASnode is part of VILLASframework, a toolset to enable the interconnection of laboratory equipment. It covers a wide set of applications, ranging from local real-time co-simulations with dynamics in the range of microseconds to geographically distributed PHIL experiments.

- *JaNDER* [7] serves as a middleware solution designed to facilitate seamless data exchange between smart systems through secure, standardized APIs. Developed within the predecessor ERIGrid [10], JaNDER utilizes `HTTPS` for secure communication and a cloud-based Redis database[1] to replicate data between local smart systems and a central cloud node. Its core functionality involves mirroring infrastructure data bidirectionally—data recorded in a local database is simultaneously updated in the cloud and vice versa—ensuring interoperability across different transport services.
- *Lablink* [9] is a platform developed by AIT that provides communication between DRTS and HIL devices. The Lablink Core middleware enables communication between distributed clients, while communications are implemented using the MQTT protocol, which allows asynchronous messaging and supports secure connections. Synchronous communications are achieved using remote procedure calls. Moreover, the Lablink Core exposes interfaces that can be used by clients to interconnect hardware and simulators and by utilities to provide services on top, like synchronization.

3.2 Universal API

The uAPI was proposed as a unified protocol to implement internal communications and allow interoperability between transport platforms. Thus, RTI SCADA systems and other internal communication mechanisms can interconnect to remote RIs using any transport tool, provided that it supports uAPI. Besides, the uAPI provides harmonized functionality by defining a set of common methods to access the list of available signals, RIs, status, etc.

Being a REST API, the uAPI requires a server side and a client side. The server side is implemented as part of the transport tool, whereas the client side is implemented as an adapter module, which connects to the internal SCADA system or directly to the laboratory equipment. It is worth stressing that this concept enables a substantial increase in software reusability since the interoperability provided by the uAPI allows transparent interchange adapters between RTIs. In this way, once a new adapter is implemented to support a specific communication protocol, it becomes available immediately for the three transport tools.

The uAPI is structured according to the OpenAPI specification,[2] which aims to standardize an HTTP API. This specification outlines a set of guidelines for defining APIs, including the available endpoints, the data models employed, authentication mechanisms, and more, so that clients can understand how to interact with the API, as well as to automatically generate code and documentation. A major advantage of adhering to the OpenAPI specification is that it promotes uniformity and minimizes

[1] https://redis.io/.

[2] https://www.openapis.org/.

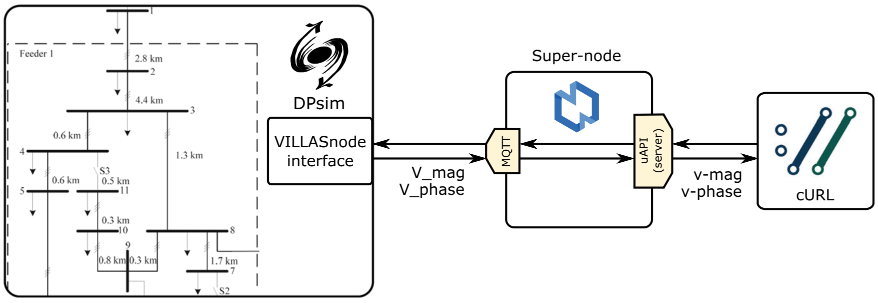

Fig. 1 Example setup for testing the uAPI.

the need for manual documentation. Offering a standardized method to describe API, it ensures that they are well-documented and easy to understand. Moreover, the OpenAPI specification facilitates the automatic generation of client libraries, significantly reducing the time developers need to invest. All documentation related to the developed uAPI can be accessed publicly through the GitHub Repository.[3]

To illustrate the uAPI's implementation, consider the setup shown in Fig. 1. Here, the open-source RTS DPsim [5] runs the CIGRE-MV benchmark. Node voltages are sent through the VILLASnode interface via MQTT to an instance of VILLAS node, which implements the server side of the uAPI. In this case, the uAPI functions using the cURL command line utility[4] can be verified. The interested reader can find an open-source implementation of this scenario in the corresponding GitHub Repository.[5]

In this case, VILLASnode runs on the local host machine. Thus, for this specific transport, the root endpoint of the uAPI corresponds to the URL http://localhost:8080/api/v2/universal/node_uapi/, where `node_uapi` is the name of the node defined in the VILLASnode configuration file. The cURL command to get a list of the available channels would be:

```
curl -v
 ↪  http://localhost:8080/api/v2/universal/node_api/channels
```

The channels are the signals that are sent from the DPsim simulator; in this case, a total of 30 signals is obtained, corresponding to the magnitudes and phases of the 15 buses found in the CIGRE-MV benchmark. An extract of the response from the uAPI server is shown in Fig. 2.

[3] https://erigrid2.github.io/JRA-3.1-api/universal-api.html.

[4] https://curl.se/.

[5] https://github.com/ERIGrid2/MOOC4.git.

Fig. 2 uAPI response for GET/channels

```
[
    {
        "id": "n1-v",
        "datatype": "float",
        "readable": true,
        "writable": false,
        "description": "Voltage N1",
        "rate": 1.0,
        "payload": "samples"
    },
    {
        "id": "n1-ph",
        "datatype": "float",
        "readable": true,
        "writable": false,
        "description": "Phase N1",
        "rate": 1.0,
        "payload": "samples"
    }
]
```

Fig. 3 uAPI response for GET/channel/sample

```
{
"timestamp": 1742461686.0433388,
"value": -0.029712912935014171,
"validity": "unknown",
"source": "unknown",
"timesource": "unknown"
}
```

To get the value of a specific channel, the endpoint would be `channel/<channel-id>/sample`, being `<channel-id>` the id of the channel identified in the previous step, as shown in the following cURL request.

```
curl -v
↪  http://localhost:8080/api/v2/universal/node_uapi/channel/n15-ph/sample
```

which results in an output similar to the one in Fig. 3.

3.3 Configuration Management

The work in ERIGrid 2.0 builds on the foundation laid by its predecessor, ERIGrid, which pioneered experiments involving multiple RTIs, known as "multi-RI experiments," in various configurations [11]. These experiments included the real-time coupling of geographically distributed physical laboratories, as well as the integration of physical laboratories with different types of simulators operating in real-time, particularly in PHIL and CHIL setups. During the demonstration phase of ERIGrid, valuable experience was gained in implementing closed-loop interconnections of RTIs across organizational boundaries. One of the key insights from this process was that many of the challenges encountered were fundamentally non-technical, arising primarily from the convergence of different organizations, procedures, and their underlying assumptions. The RTI interconnection concept was successfully demonstrated in ERIGrid, but no structured process was developed for cross-organizational experimental setups. Ad hoc adjustments were made, leading to challenges exacerbated by coordinating laboratory time between organizations. Key obstacles included the following points:

- A technical solution for signal exchange between RTIs was developed, but aligning signal interpretation across RIs required significant effort. Confusion over signal labels, units, conventions, scaling, and value ranges caused extensive debugging, repeated experiments, and delays.
- Repeating or resuming multi-RTI experiments was more complex than in a single lab, as it required recreating identical conditions across multiple labs with numerous DERs which are often reconfigured by other users, and added significant challenges.
- There are two main ways to record data in geographically distributed experiments: sending all data to a central logger in real-time or recording data locally at each site. The first approach often faces technical limits, such as bandwidth constraints, while the second requires merging logs from different sites. Beyond ensuring synchronized clocks, challenges arise if data labels and units are not standardized, mirroring the signal configuration issues described earlier.

While none of these obstacles are insurmountable or unsolvable, the lack of a systematic approach has led to significant manual, error-prone, and time-consuming efforts. These challenges are further amplified by the geographical distance between participants. Therefore, the objective of applying the Configuration Management (CM) approach in ERIGrid 2.0 is to automate the configuration of data exchange for multi-RTI experiments. This involves the following:

- *Offline Global Configuration/Description*: Defining the general data flow between the participating RTIs in an offline setting.
- *Online Automated Mapping*: Dynamically mapping local RTI signals to the appropriate data channels for the experiment, based on the global configuration.

The proposed CM approach is illustrated in Fig. 4, depicting two example RTIs conducting a joint experiment. Data exchange between these RTIs is enabled through dedicated laboratory coupling tools, such as the VILLAS Framework, JaNDER, or Lablink. The CM framework consists of two components. First, a module that splits and maps a global configuration file definition to multiple local files containing the specific configuration of a given RI participating in the distributed experiment, i.e., the CM tool. The second component is the ERIGrid 2.0 launcher, which parses this

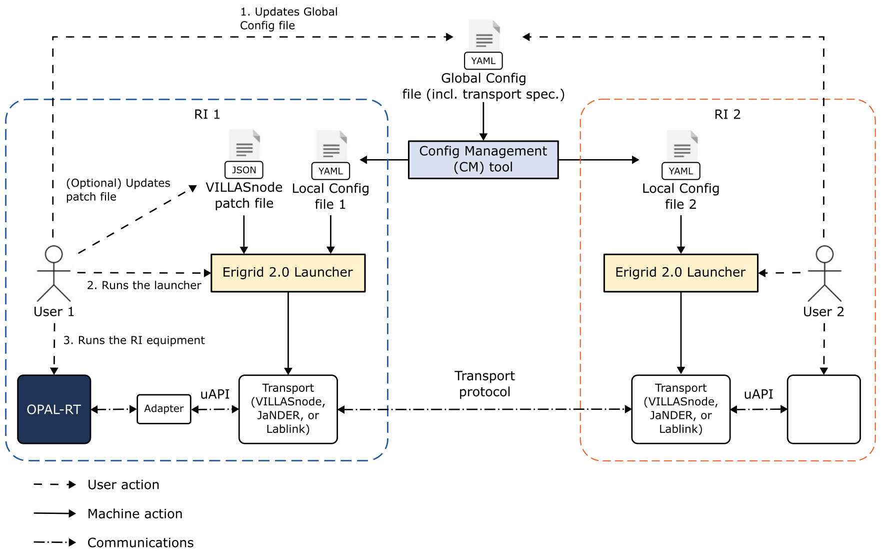

Fig. 4 Configuration management workflow and tools

local configuration file and runs the corresponding transport tool. The CM approach combines these components in a highly automated workflow comprising three steps:

1. The process starts with the definition of the shared global configuration file, where the transport communications and the input/output connections among RTIs are defined.
2. The user runs the ERIGrid 2.0 launcher, specifying the local configuration file produced by the CM tool as input.
3. Finally, the user starts the underlying TRI equipment, e.g., DRTS, grid emulators or load banks.

Although the process currently requires the user to run the CM tool, the level of automation can be increased through a Continuous Integration and Continuous Delivery/Deployment (CI/CD) task, through which the global configuration file is centrally managed using a version control system, e.g. Git. With every new commit/push of this file, a task is triggered to run the CM tool and new versions of the local configuration files are created. It is worth stressing that the approach does not disclose the way each RTI operates internally since only the configuration of the transport communications is shared. This implies that some transports like VILLASnode require an extra step to take care of configuring the internal communications. In the case of VILLASnode, the configuration is completed by adding a patch file as an input to the ERIGrid 2.0 launcher. Thus, the internal configuration of the RTIs can be kept locally in the patch file.

4 Towards a Laboratory Middleware

The tools described in the previous sections constitute an important step towards a unified approach for research infrastructure integration and automation. However, as pointed out above, there are still some configuration steps not covered by the CM tool, which require the lab operator to prepare, control (starting and stopping) and monitor the DRTS and hardware equipment. This is more problematic as the number of devices increases not only at the local level but also for distributed RTI experiments. Here it is proposed the development of a laboratory middleware that allows to manage RTI equipment and incorporates the tools described in the previous sections.

The primary purpose of the laboratory middleware is to enable seamless data exchange between Smart Systems. This middleware supports a range of systems, including physical laboratories, real-time and non-real-time simulators, and builds upon the ERIGrid 2.0 Transports. The main objectives of middleware are: (i) Allowing Smart Systems to switch effortlessly between transport tools (e.g., from JaNDER to VILLAS) without the need for multiple interfaces or adjustments to data formats and naming conventions, (ii) introducing desirable features not currently offered by existing transport modules through modular services, and (iii) centralising shared features from individual transport modules to avoid duplication. For example, a sophisticated time synchronization mechanism from one transport can be offered as a common service to benefit other transports with less advanced solutions.

The overall architecture consists of three layers, as shown in Fig. 5: (i) Infrastructure layer, (ii) platform layer, and (iii) software layer. The *infrastructure layer* is the foundational layer comprising physical lab setups, hardware devices, and SCADA systems. Through this layer, laboratory operators will be able to monitor and control multiple (distributed) devices from a central interface. This idea is supported by the fact that many DRTS and hardware devices can be controlled using APIs. The *platform layer* provides essential services, such as network emulation, time synchronization, and resource provision. Finally, the *software layer* facilitates data exchange through tools like VILLAS, JaNDER, and Lablink, with the uAPI offering a standardised set of functionalities. This layered approach ensures efficient, flexible, and unified communication across diverse Smart Systems.

An important building block of the laboratory middleware is RIasC, short for Research Infrastructure as Code. RIasC is a framework designed to accelerate distributed RTI experiments. It consists of a suite of tools and services that can be used independently or in combination to support these experiments. Infrastructure as Code (IasC) refers to the practice of managing and provisioning data centers using machine-readable definition files rather than traditional methods such as manual configuration or interactive tools. This approach manages both physical components, like bare-metal servers and virtual machines, along with their associated configurations. Typically, these definitions are stored in version control systems, and IasC can be implemented using either scripts or declarative definitions. The term is most commonly associated with promoting declarative methods for infrastruc-

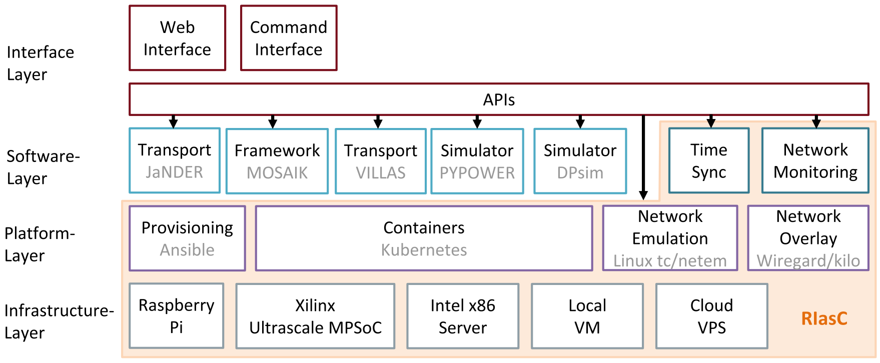

Fig. 5 Overview of the ERIGrid 2.0 laboratory middleware architecture [8]

ture management. RIasC realizes the principles of IasC by leveraging existing cloud computing technologies and applying them in research environments. RIasC offers several functionalities aimed at accelerating distributed RI experiments:

- *Provisioning of Mobile Units*: In RIasC, a mobile unit is a gateway that connects local laboratory equipment to remote laboratories. RIasC handles the setup, configuration, and maintenance of these units.
- *Deployment of containerised Software Components*: Mobile units join a Kubernetes cluster,[6] which allows for the declarative deployment of containerized applications.
- *Transparent Inter-Laboratory Overlay Network*: RIasC simplifies the process of setting up distributed experiments across multiple laboratories by providing a transparent IP overlay network that connects all participating labs.
- *Time Synchronization*: For geographically distributed coupled subsystems, RIasC provides a time synchronization service to ensure that experiments are executed with a common time base, allowing proper temporal alignment of simulation results, such as merged logs.
- *Network Emulation*: RIasC includes a network emulation service that allows researchers to define network characteristics (e.g., communication delay, packet loss, throughput) declaratively, enabling the realistic emulation of real-world network conditions.

The code base of RIasC has been released as open-source under the Apache 2.0 license. In addition, it includes a user-friendly website with useful documentation.[7]

[6] https://kubernetes.io.

[7] https://erigrid2.github.io/riasc/.

5 Final Thoughts

This chapter identified the main challenges faced by the implementation of multi-RTI experiments, with a focus on integration and automation. A suite of tools was proposed to harmonize communications at the RTI and the inter-RTI level and to facilitate configuration management. Concretely, the uAPI provides a uniform interoperability layer with useful data structures and functionalities. In addition, the CM tool enables a highly automated workflow to prepare experiments and run the transport tools at each RTI, with which RTIs can communicate. Finally, further automation necessities were identified, and a middleware architecture was presented, along with the RIasC concept.

References

1. Abdelrahman MS, Kharchouf I, Nguyen TL, Mohammed OA (2023) A hybrid physical co-simulation smart grid testbed for testing and impact analysis of cyber-attacks on power systems: framework and attack scenarios. Energies 16(23):7771. https://doi.org/10.3390/en16237771. Number: 23 Publisher: Multidisciplinary Digital Publishing Institute
2. Bach A, Monti A (2025) Remote real-time testing of physical components using communication setup automation. IEEE Access **13**:39066–39075. https://doi.org/10.1109/ACCESS.2025.3546311. Conference Name: IEEE Access
3. Hardy TD, Palmintier B, Top PL, Krishnamurthy D, Fuller JC (2024) HELICS: a co-simulation framework for scalable multi-domain modeling and analysis. IEEE Access 12:24325–24347 (2024). https://doi.org/10.1109/ACCESS.2024.3363615. Conference Name: IEEE Access
4. Krammer M, Schuch K, Kater C, Alekeish K, Blochwitz T, Materne S, Soppa A, Benedikt M (2019) Standardized integration of real-time and non-real-time systems: the distributed co-simulation protocol. In: Modelica, pp 157–009. https://2019.international.conference.modelica.org/proceedings/html/papers/Modelica2019paper1C3.pdf
5. Mirz M, Dinkelbach J, Monti A (2020) DPsim-Advancements in power electronics modelling using shifted frequency analysis and in real-time simulation capability by parallelization. Energies 13(15):3879. https://doi.org/10.3390/en13153879. Publisher: Multidisciplinary Digital Publishing Institute
6. Monti A, Stevic M, Vogel S, Doncker RW, Bompard E, Estebsari A, Profumo F, Hovsapian R, Mohanpurkar M, Flicker JD, Gevorgian V, Suryanarayanan S, Srivastava AK, Benigni A (2018) A global real-time superlab: enabling high penetration of power electronics in the electric grid. IEEE Power Electron Mag 5(3):35–44. https://doi.org/10.1109/MPEL.2018.2850698
7. Pellegrino L, Pala D, Bionda E, Rajkumar VS, Bhandia R, Syed MH, Guillo-Sansano E, Jimeno J, Merino J, Lagos D, Maniatopoulos M, Kotsampopoulos P, Akroud N, Gehrke O, Heussen K, Tran QT, Nguyen VH (2020) Laboratory coupling approach, pp 67–86. Springer International Publishing. https://doi.org/10.1007/978-3-030-42274-5_5
8. Rajkumar V, Silano G, Gehrke O, Vogel S, Widl E, Paludetto G, Rikos E, Zerihun TA, Stefanov A, Palensky P, Strasser TI (2024) Laboratory middleware for the cyber-physical integration of energy research infrastructures. In: 2024 12th workshop on modeling and simulation of cyber-physical energy systems (MSCPES), pp 1–5. https://doi.org/10.1109/MSCPES62135.2024.10542755
9. Stahleder D, Reihs D, Lehfuss F (2018) LabLink-A novel co-simulation tool for the evaluation of large scale EV penetration focusing on local energy communities. In: CIRED 2018 Ljubljana workshop on microgrids and local energy communities. AIM, Ljubljana, Slovenia. Publisher: AIM

10. Strasser T, de Jong E, Sosnina M (2020) European guide to power system testing: the ERIGrid Holistic approach for evaluating complex smart grid configurations. Springer. https://doi.org/10.1007/978-3-030-42274-5
11. Strasser TI, Andren FP, Widl E et al (2018) An integrated pan-European research infrastructure for validating smart grid systems. e & i Elektrotechnik und Informationstechnik 135(8):616–622. https://doi.org/10.1007/s00502-018-0667-7
12. Strasser TI, Moyo C, Bründlinger R, Lehnhoff S, Blank M, Palensky P, van der Meer AA, Heussen K, Gehrke O, Rodriguez JE, Merino J, Sandroni C, Verga M, Calin M, Khavari A, Sosnina M, de Jong E, Rohjans S, Kulmala A, Mäki K, Brandl R, Coffele F, Burt GM, Kotsampopoulos P, Hatziargyriou N (2017) An integrated research infrastructure for validating cyber-physical energy systems. In: Marík V, Wahlster W, Strasser T, Kadera P (eds) Industrial applications of holonic and multi-agent systems. Springer International Publishing, Cham, pp 157–170. https://doi.org/10.1007/978-3-319-64635-0_12
13. Vogt M, Marten F, Montoya J, Többermann C, Braun M (2019) A REST based co-simulation interface for distributed simulations. In: 2019 IEEE Milan PowerTech, pp 1–6. https://doi.org/10.1109/PTC.2019.8810661