# Co-Simulation Environment for Event-Driven Distributed Controls of Smart Grid

Chia-han Yang, *Non-member*, Gulnara Zhabelova, *Non-member*, Chen-Wei Yang, *Student Member, IEEE, and* Valeriy Vyatkin, *Senior Member, IEEE,*

***Abstract* – This paper proposes a co-simulation environment for "hardware in the loop" or "software in the loop" validation of distributed controls in Smart Grid. The controls are designed using model-driven engineering with the IEC 61499 Function Block architecture. These are connected with plant models, for example in MATLAB/Simulink, through communication channels such as UDP or TCP sockets. This solution enables multi-closed-loop plant-controller simulation. The communication between plant and controller is event-driven. In order to perform realistic simulation, the proposed solution takes into account computation and communication delays on the controller side in Function Blocks and compensates model time on the plant side in MATLAB model accordingly. Causality and accuracy of the method have been formally addressed. This approach has been tested and demonstrated with several Smart Grid-related examples.**

***Index Terms* - Smart Grid, Simulation, IEC 61499, Software in the loop, Hardware in the loop, MATLAB Simulink, Function Blocks**

## I. INTRODUCTION

The Smart Grid is defined as an integration of electricity and communication, so that the electric network will be "always available, live, interactive, interconnected, and tightly coupled with the communications in a complex energy and information real-time network" [1], [2]. The result will be more efficient power systems capable of better managing the growing power consumption, providing fault resilience and seamlessly integrating Distributed Energy Resources (DER), such as renewable energy sources (e.g. wind and solar) [3].

There is common understanding in the research as well as in the industrial community that traditional hierarchical automation design approaches have limited applicability in Smart Grid. Therefore, the control architecture of the Smart Grid is seen by many researchers as a heterogeneous network of controllers communicating in a peer- to- peer manner [4, 5].

Development and deployment of Smart Grid controls following this principle raises a number of challenges related to verification and validation of distributed grid intelligence (DGI). Hardware in the Loop (HiL) and Software in the Loop (SiL) simulations are often used to validate controllers in Smart Grid-related projects [6-13]. However, in order to ensure the correctness of simulation results one needs to establish a system-level model of the distributed system which would combine dynamics of the primary equipment and computational processes in distributed communicating control nodes. This is especially hard when distributed systems are considered due to asynchrony of control nodes and variable communication times.

The IEC 61499 standard [14], [15] has been established as a reference system architecture for distributed embedded and automation systems design. This standard introduces a new notion of Function Block (FB) which is an event-driven module encapsulating one or several functions. The standard aims at flexibility and re-configurability of automation systems. Smart Grid is seen as one of the promising areas of its application [16], [17]. Besides, IEC 61499 can serve as an efficient device-level executable implementation of designs based on the popular IEC 61850 standard from the power distribution domain, as proposed in [18]. The IEC 61850 standard suggests the concept of Logical Nodes (LN) for object-oriented code design in Intelligent Electronic Devices (IED). It was proposed in [13] to enhance the LN concept of IEC 61850 by adding agent-based intelligence and encapsulating the resulting Intelligent Logical Nodes into IEC 61499 function blocks.

This paper presents a co-simulation approach that relies on common communication channels (e.g. through User Datagram Protocol (UDP) or Transmission Control Protocol (TCP) sockets) to construct the closed-loop models. The controller implementation is done in Function Blocks while the plant model is implemented in a simulation environment based on time steps, e.g. MATLAB. The approach is to couple the environments into a joint simulation. The advantage of this approach is that only a little modification is required to the existing plant and control models. In particular, this paper addresses the problem of correct simulation timing in such co-simulation environment. The proposed solution has been validated in several Smart Grid related test cases with three different platforms implementing distributed controls.

The paper is structured as follows: Section II surveys the related works. Section III discusses basics of event-driven co-

John Yang is with The University of Auckland, 1142, New Zealand.(e-mail: cyan034@aucklanduni.ac.nz).

Gulnara Zhabelova is with The University of Auckland, 1142, New Zealand (e-mail: gzha046@aucklanuni.ac.nz).

Chen-Wei Yang is with The University of Auckland, 1142, New Zealand (e-mail: cyan083@aucklanduni.ac.nz).

Valeriy Vyatkin, is with the Department of Computer Science, Computer and Space Engineering, Lulea Tekniska Universitet, Sweden (e-mail: valeriy.vyatkin@ltu.se).

simulation environment computational implementation and considers differences of simulation from the real-life plant-controller interaction in order to establish the conditions of correct simulation later in the paper. Section IV presents implementation details of a correct co-simulation in the closed-loop that takes into account computation and communication times on the controller side. Results of three experiments are presented in Section V. Section VI concludes this paper with discussion of future work plans.

## II. STATE OF THE ART

Using a simulation model and environment to validate the controller design "in the loop" is a fairly common technique in industrial practice. However, a straightforward approach to that, is to run the simulation platform faster (in real-time) by increasing performance of the computing platform and ignoring the overheads introduced by the simulation environment. For example, commercial simulation systems, such as RTDS and Opal RT follow this approach.

Similar approaches in creating "in the loop" simulation environment to validate PLC code can be exemplified by RRS (Realistic Robot Simulation) [19] and CAPE (Computer Aided Production Engineering) Tools [20]. As reported in [21], closed-loop simulation can be implemented via a standardized interface such as OPC for IEC 61131-3 PLC design. OPC defines interfaces between PLCs and computers. According to [21], simulation-based PLC code verification is a part of virtual commissioning, where the control code is verified against model of the controlled process simulated in an external simulation environment. However, the paper identifies several problems with this approach that can lead to an unreliable verification results. The four major problems with the OPC interface are: the so called *free-wheeling*, communication jitter and delay, race condition and slow sampling. Free-wheeling occurs as a result of unsynchronized execution of model and controller, when one run on one side can correspond to several runs on the other. The presented solution includes synchronization of both execution cycles and tweaking the controller code in order to ensure its time-dependent functions (such as timers) still work correctly after such synchronization.

In the context of distributed automation and IEC 61499 architecture, a design and verification framework has been proposed in [22] based on the use of multi-closed-loop models. Following this framework, some model-driven approaches have been proposed and implemented to help in validation and verification of Function Block designs. This includes the Intelligent Mechatronics Components (IMCs) [23, 24] concept based on the model-view-control (MVC) design patterns, and Composite Automation Type (CAT) concept by NxtControl [25]. These model-driven designs are applied to various industrial applications for closed-loop simulation and visualisation, such as smart-grid power systems, baggage handling systems (BHS) [26], building management systems [25], etc.

Another example of distributed system validation is the work by Maturana et al. [27] demonstrating co-simulation between MATLAB and SoftPLC to validate distributed multi-agent control design. There are two research groups [16] and [12] investigating event driven control for Smart Grid using IEC 61499 and applying co-simulation of the developed control code with the power system models.

An early version of a co-simulation environment for Smart Grid distributed control modelled in IEC 61499 communicating with MATLAB power system model was presented in [16]. At that stage both environments were running concurrently without taking into account timing of the communication and computation. The difference of IEC 61499 from PLC is event-driven execution, which potentially can eliminate the problem of free-wheeling, but all other timing issues raised in [21] remain intact.

Strasser et al. [12] present a similar approach of simulating the coordinated voltage control of an Under-Load Tap Changer (ULTC) implemented as IEC 61499 control application in the 4DIAC framework and the ULTC model together with a model of the distribution network simulated in the GNU Octave/PSAT environment. This work's contribution is a co-simulation framework for Smart Grid applications based on open standard and open source software. However, the paper does not address the timing issues of the co-simulation.

Other works do not utilise IEC 61499 event-driven control model, but widely apply co-simulation of control and power system model to validate Smart Grid solutions.

Godfrey et al. in [9] analyse the impact of wireless communication on the deployment of distributed energy resources on a model of distribution circuit (feeder). The co-simulation was performed using OpenDSS discrete-event simulator and ns2 network simulator. The work is an attempt to create a realistic simulation of the power system with distributed generation and energy storage nodes. The simulation takes into account various communication network delays and studies their impact on the overall system performance. The simulation of power system in closed-loop with its control was not a focus of this work.

Similar work of co-simulation power system model with the model of communication network is performed by Liberatore and Al-Hammouri in [10] and Lin et al in [11]. Liberatore and Al-Hammouri also apply ns-2 network simulator for the models of the communication between Smart Grid distributed devices and sensors. For the power system modelling they use power grid simulator based on Modelica [10].

These papers acknowledge the impact of the communication delays on the power system dynamics. Since Smart Grid is a distributed system, integrating geographically spread generation, control devices and sensors, it heavily relies on communication network. Papers [9-11] study impact of communication network topology and communication delay on the Smart Grid dynamics.

However, neither of mentioned works addresses the timing issues raised in [21].

The compensation techniques for accuracy improvement in HiL or SiL simulation have been addressed in a number of works, in different application domains. For example, the work by Chinchul and Wootaik [28] analyses unavoidable time delay effects in a hardware-in-the-loop (HIL) simulation for automotive permanent magnet synchronous motor drive systems and proposes a compensation method for the delays incurred at complex interfaces between the model and controller. The compensation technique, however, is based on the modification of the model itself for a measured delay. Also, the method does not address the granularity of the model time in the simulation environment.

Dufour and Belanger indicate in [29] that in modelling of some basic converter circuit topologies, exhibiting fast switching dynamics, simulations with relatively large fixed time step can cause multiple switching events in a single time-step. The paper proposes a compensation technique embedded into a proprietary MATLAB solver that allows for correct real-time simulation of precision-critical hardware-in-the-loop (HIL) systems.

There are standard co-simulation interfaces such as HLA [30, 31], FMI [32] and DIS [33] developed for purpose of large scale system model integration. FMI, HLA and DIS are mainly used in automotive industry, space and defense projects. Their prior application in the power system domain is unknown to the authors.

The learning curve of the HLA, FMI and DIS is steep and considerably long.

Each simulation has to implement a communication layer, a part of the HLA or FMI framework, to be compatible and able to participate in co-simulation.

Both co-simulation frameworks require a central coordinator (in HLA - RTIG) which manages simulation and transfers data between the simulators. In FMI there is a master algorithm, which coordinates the data exchange between simulations and synchronizes all slave simulations.

The open versions of the mentioned interfaces rely on the best effort synchronization mode. Each simulation (node) provides data as fast as it can and updates own data as soon as it receives them. This method does not guarantee that at a given moment the data received by any 2 nodes from same source is consistent. However, HLA guarantees that each simulator is at the same simulation step. The communication during the simulation step is allowed. In contrast, FMI restricts the data exchange to the discreet communication points and provides synchronization of the co-simulated environments. However, description of the synchronization mechanisms is not easily available. It is understood that each master algorithm implements its own synchronization method.

The commercial implementations of these interfaces do not expose the synchronization mechanism, thus it is unknown how nodes are synchronized, therefore it is difficult to evaluate their suitability for a given co-simulation task in Smart Grid.

Hua *et al* [11] also have proposed a co-simulation framework for coupling continuous power system simulators with discrete event driven communication network and control models. The simulations are interleaved: either one of the solvers get to be executed at a given time. The paper introduces the global scheduler, which loads the simulation steps of the power system at the initialization time and orders them according to the time stamp. The event, coming from discrete event simulator of control network, is scheduled between the simulation steps according to its time stamp. When the power system simulation round is completed, the simulation is suspended till the scheduler processes the next simulation round. In this case, the issue of accumulated errors described later in the paper is still exists, when the simulation step of power system is relatively large. Since the scheduler needs to complete the simulation round and then move to the next scheduled tasks, the control event, arrived at the middle of the simulation step, will be scheduled just after it. Another issue of this approach, , is that the co-simulation framework does not consider computation time of the controller and communication delays, which affect both power system and control model dynamics and, therefore, should be accounted for.

Bankier in [34] proposes GridIQ - a simulation framework for co-simulation of agent based solutions and power system models. The GridIQ performs the role of a bridge between agents and power systems simulation. GridIQ supports JADE agent development platform and PSAT power system analysis tool. The execution of the agent network and power system analysis is interleaved. This approach is intuitive and eliminates problems occurring with concurrent execution of the control and plant models. However, in this case, the time taken by the agents to make decision is ignored and not considered by power system model dynamics. The computation time of the agent network and simulation of power system are not accounted for and are considered instantaneous. These are incorrect assumptions. Also GridIQ does not account for communication delays between agents and between agent control model and power system model.

After the short observation of the state of the art, the following conclusions can be made:

There is great need for SiL or HiL environments for distributed systems, in particular based on the new IEC 61499 standard. The event-driven execution of program components in this standard creates new challenges (but also new opportunities) for co-simulation architectures. There are several timing-related issues identified in the literature which need to be addressed in the context of proposing such a distributed co-simulation framework. The existing compensation techniques for delay and jitter require deep modification of the model or application of a dedicated solver, which are not universal.

Our work focuses on closed-loop simulation of control model and plant model. The paper proposes generic socket based co-simulation framework, where each simulator is not required to implement additional communication layer. Each simulator inserts UDP communication sockets where the data is to be received or sent. However, such co-simulation mechanisms could introduce so-called free-wheeling, communication jitter and delay, race condition and slow sampling.

Therefore, the main contribution of this paper is an attempt to understand these timing issues and formally describe the synchronization mechanism between control model and plant model, simulated in closed-loop. The paper proposes and formally describes the synchronization mechanism which compensates for delay, jitter, and simulation time step duration.

Further in the paper MATLAB/Simulink is referred to as a primary plant modelling environment, but the proposed method can be applied to other similar tool-sets.

## III. Event-driven Co-Simulation Environment

The proposed co-simulation approach uses a common communication protocols to establish data connection between FB controller(s) with existing model of plant in closed-loop as shown in Fig. 1. In this paper we consider simulation environments based on time steps, such as MATLAB [35], OpalRT [36], PowerWorldSimulator [37] and Eurostag [38]. The time step can be variable or fixed. Due to different speed of execution, one side (e.g. plant or controller) can produce data more frequently than the other. One should note that plant model works in the model time whose scale can be different from the real-time. This is also true for the controller in function blocks: its execution time during the simulation can be different from that when deployed to an embedded device.

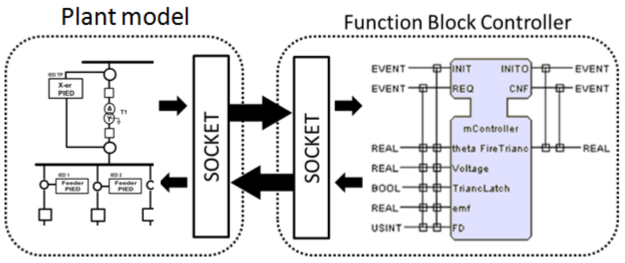


Fig. 1. Closed-loop simulation between MATLAB and FB models.

Communication sockets on the function block side are implemented using Service Interface Function Blocks (SIFB), and on the simulation tool side as custom blocks programmed in C or any other available programming languages. IEC 61499 follows event-driven control paradigm therefore the classic control loop setup in Fig. 2 (a) needs to be modified as follows. As an example of controller algorithm, suppose the controller is reading plant parameter $g(t)$ and keeping it in certain boundaries by changing control variable $c(t)$. Whenever $g(t)$ crosses a threshold boundary (event $e_1$ in Fig. 2(b)), the plant emits a message that is received by the controller ($e_2$). After a certain (computational) delay $t_c$ the controller updates $c(t)$ and notifies the plant by a message (received at $e_4$).

Plant and controller are concurrently active, so the plant state variables change following plant dynamics in the interval $[e_1, e_4]$ while the controller takes time to change the $c(t)$.

In co-simulation environment composed of concurrently running event-driven controller and model of the plant the difference in behaviour is as presented in Fig. 3. The plant model operates in discretized time intervals (steps $S_1, S_2, \dots, S_i$, of a fixed $t_d$ or variable duration) and is exchanging parameters with the controller only at the boundaries of these intervals. Therefore, notification of the threshold crossing event $e_1$ will be delayed till the end of the discrete interval $e_{1s}$ as well as the feedback notification $e_4$. As a result of this, the duration of the interval $[e_1, e_4]$ in the simulation can be substantially longer than in real life.

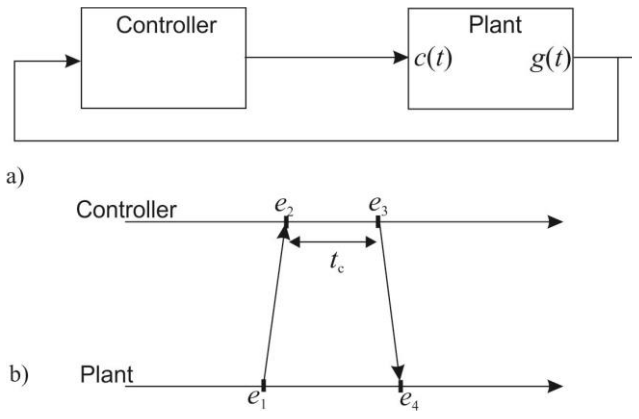


Fig. 2. a) Closed-loop plant-controller system; b) Event-driven communication between plant and controller.

A possible impact of that is illustrated in Fig. 4. There are plots of $g(t)$ and $c(t)$ for three scenarios presented: real-life ($c_r(t)$), the case when the controller's decision making delay is slightly longer than the step interval of the plant model ($c_1(t)$) and the case when it is slightly shorter than that ($c_2(t)$).

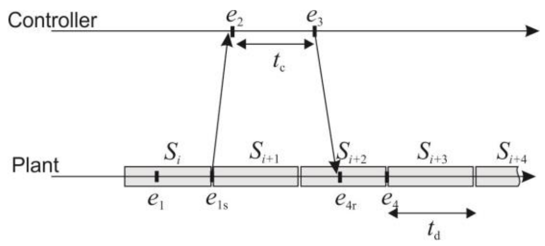


Fig. 3. Plant-Controller interaction in simulation.

As the simulation is performed using fixed or variable step sizes, the controller communication feedback will be read at the beginning of such simulation step. Moreover, the plant message $e_1$ will be sent at the end of the simulation step, even though its event $e_1$ actually happened in the middle of the step.

Thus, there is a clear impact of the controller computation time on the system results. If the computation takes a bit longer than the simulation step, the data from the controller will be only updated at the end of the simulation step, and therefore make the reaction time of the controller longer than it really is, allowing for much error to accumulate and result in greater overshoot. If the computation time is shorter than simulation step size, then the controller command will take effect at the next simulation step.

As one sees from the figure, in both simulation cases the overshot of the control parameter g(t) can be significant. This can make the simulation results much different from the real life ones which diminish the value of simulation. Besides, the simulation environment is unstable being susceptible to slight variations in the controller computation time $t_c$: the difference between simulation results when $t_c$ is just slightly below or above $t_d$ can be substantial.

Therefore, it would make sense to build a simulation environment in which plant and controller activities would be interleaved instead of being concurrent as shown in Fig. 5. Once the model of the plant notified the controller of event $e_1$ in the step $S_i$, the plant would "freeze" and resume with the step $S_{i+1}$ only after receiving the notification event $e_4$.

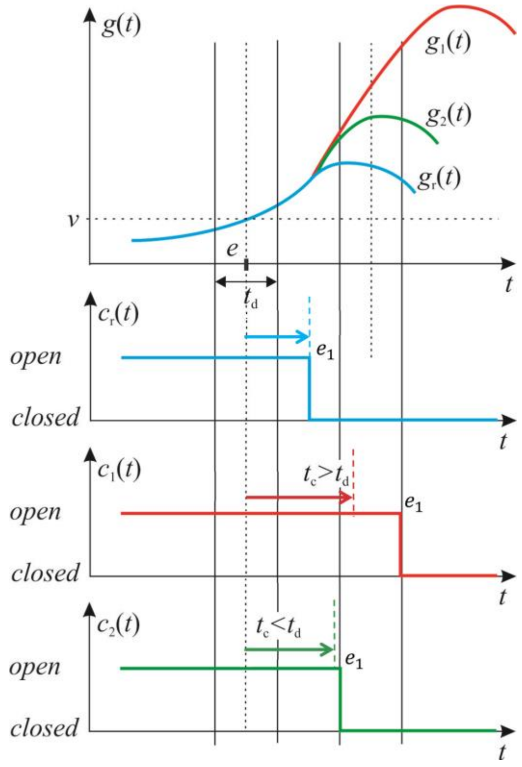


Fig. 4. Comparison of plant behavior in real-life and in simulation.

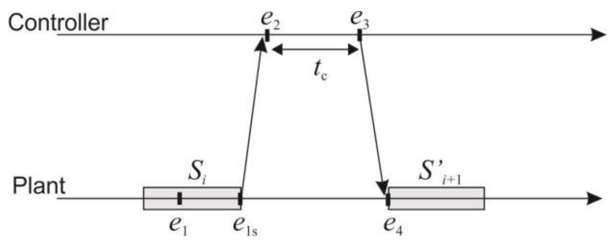


Fig. 5. Adjusted co-simulation environment.

This setup would be free of the problem discussed above, but would have another problem: the controller computation time $t_c$ would be ignored in the plant dynamics. Such a situation is also incorrect.

The proposed adjustment is to increase the model time in step $S_{i+1}$ by $t_c$ and recalculate g($t$) based on this adjusted time. Implementation of this adjustment will be considered in Section IV.

## IV. Co-Simulation Framework

Based on the observations presented in Section III, the co-simulation environment will be synthesized in the general form presented in Fig. 6. Controller and Plant blocks represent the controller execution environment in function blocks and model of the plant respectively.

$MDO_n$ and $MDI_n$ denote the data output set and data input set of the plant model respectively at nth time step, while $DI_n$ and $DO_n$ denote the data input set and data output set of the FB model at nth execution cycle. Symbols $\alpha$ and $\delta$ stand for events activating these environments respectively, while $\varepsilon$ and $\beta$ are events notifying of the end of the step or controller execution respectively. The Proxy is an intermediate agent whose role is to adjust the plant's clocks according to the following algorithm 1.

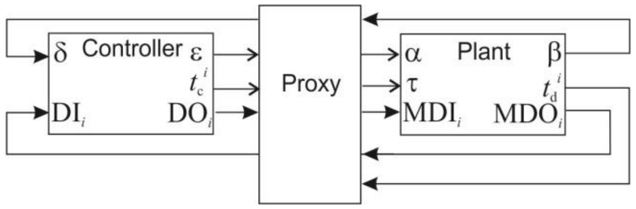


Fig. 6. Proposed co-simulation framework.

If the controller execution time ($t_c$) is less than or equal to the time step at the plant side ($t_d$), the simulation can be correctly performed without any additional consideration in the closed-loop data communication. The Proxy comes to play when $t_c > t_d$ to adjust clocks of the plant. The events sent from plant and controller to the Proxy are timestamped by the proxy using the provided information on the execution time $t_c^i$ and $t_d^i$.

**Algorithm 1**:

1. The condition $t_c \le t_d$ implies that the proxy will be setting $DI_n = MDO_n$ and $MDI_n = DO_n$ in every execution cycle.
2. However, in the case with $t_c > t_d$, the proxy has to track down the time stamp and update the data accordingly.
3. Based on these time parameters, the proxy is able to determine the status of the data communication between the two models. It stores all the data ($MDO_n$ and $DO_n$) and makes the comparison of the time parameters. Then it determines what data is to output back to plant model depending on the result of this comparison. This means even when $DI_n$ is updated with values from $MDO_n$, the MDI set values may not be updated with the new value ($DO_{n+1}$) and they are determined by the proxy.
4. If the execution time of the controller implementation is greater than the current model time step, the Function Block controller output does not take effect until the controller execution time exceeds the accumulated model time at the plant side. In other words,

$$MDI_{n+k} = \begin{cases} DO_n, & T_{controller} > \sum_{i=n}^{n+k} t_i \\ DO_{n+1}, & T_{controller} \le \sum_{i=n}^{n+k} t_i \end{cases} \quad (1)$$

This simulation set-up is particularly useful when the plant model is set to be simulated with a variable time step. The proxy compares the time parameters at every time step even when the step sizes are different at different execution cycle. If the discrete simulation or simulation with fixed time step is chosen, the proxy may be simplified as the execution time of the controller is now only compared against a fixed value. This can be set up simply in the model (either on the power system simulation tool side or Function Block side).

To summarize the properties of the introduced environment the following notation is introduced. Let E denote the set of all events in the real plant-controller system during a single run of the controller under fixed parameters, and S the set of events in the introduced co-simulation environment formed under identical conditions. Let $e_1$ and $e_2$ denote two arbitrary events in E such that $e_1$ happened before $e_2$, (denoted $e_1 \rightarrow e_2$), and $r_1$ and $r_2$ are the corresponding reaction events (generated by the real controller). Events $s(e1)$, $s(e2)$, $s(r1)$ and $s(r2)$ are their counterparts in the simulation world. Let the time interval between events $e_1$ and $e_2$ is denoted as $[e_1, e_2]$. In the simulation world, $[s(e_1),s(e_2)]$ and $[s(r_1),s(r_2)]$ are the corresponding model time interval representation.

**Lemma 1:** The co-simulation environment introduced in algorithm 1 possesses the following properties:

1) Causality preservation: the proxy does not change the order of events (i.e. $\forall\ e_1, e_2 \in \text{E}: e_1 \rightarrow e_2 \Rightarrow s(e_1) \rightarrow s(e_2)$).
2) Precision: time interval between events in the simulation is not greater than in the real world with precision $t_d$ – the maximum step duration in simulation tool: $\forall\ e_1, e_2 \in \text{E}: e_1 \rightarrow e_2: [s(e_1),s(e_2)] \leq [e_1,e_2] + t_d$.

Formal proof of the properties is omitted due to space constraints. The idea of proof for both properties comes from the constructive definition of the Proxy's functionality and uses the fact that clocks of the Plant are monotonously increasing.

The proposed co-simulation method is implemented on the example of in the function blocks and MATLAB framework as follows. Function block controller is an example of event driven environment, and MATLAB is one of the discrete step based simulation tools. The proxy is implemented as a single Function Block that cooperates with service interface FB that is sending and receiving data from the MATLAB side as shown in Fig. 7. This Function Block (named "TimeCompare") takes the controller's execution time and the sampling rate used in the simulation as the inputs, and compares them in such a way that it will update the latest received data into the controller only if the accumulated time has exceeded the indicated controller execution time (i.e. representing the controller has finished its execution in this model-time simulation).

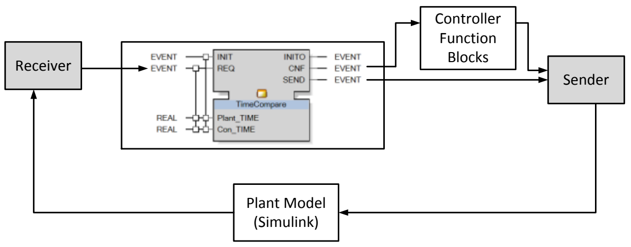


Fig. 7. Implementation of the proposed co-simulation framework.

## V. Examples of Implementation

This section presents three test cases in which the proposed co-simulation method has been applied. Each of the scenarios uses particular controller architecture. The examples illustrate the applicability of the proposed co-simulation method in different implementation environments of IEC 61499. The case, presented in subsection A, uses the time compensation mechanism, introduced in section IV, while subsections B and C present examples of distributed systems co-simulation where the timing issues were not considered. Still, those cases comply with the part 1 of the lemma 1 on causality preservation.

### A. Distributed protection example

The experiment described in this section, was conducted in the framework of FREEDM NSF project [39] where a novel protection scheme has been proposed, which is faster than conventional protection [40]. The experiment is described in more detail in [41].

The main concept is to divide the system into zones, using FID – fault isolation devices (new generation circuit breakers). Thus FID is at the borders of each section. The FREEDM protection strategy is tested at the Green Energy Hub model Fig. 8 is used as the demonstration example in the project.

As seen from Fig. 8, the protection scheme is divided into three zones plus the overall zone 0 which is a backup protection for entire system. At each zone analogue merging unit (AMU) is placed at the terminal of distribution line and the feeder of the load to measure current, digitize and transfer the sensed values to the Intelligent Fault Management (IFM) functionality of FREEDM system. Each zone has an IFM which runs the protection algorithms and incorporates DGI – distributed grid intelligence.

The primary protection used is based on the following differential scheme: if the sum of current in a zone equals zero, it indicates either there is no fault or fault is outside the zone of that IFM. In case the sum of the currents within a zone is not zero then the fault is within the zone and IFM makes decisions to trip the FIDs at the border of the faulty section. GPS time stamps are attached to each samples sent from AMU to ensure accuracy of the protection algorithm. IFM collects the sampled values from AMUs with similar time stamps and sums up these values to check if it is zero. If the sum is not zero, it holds the value and counts next coming data. If the sum is not zero for all next 10 samples, then IFM makes decision that there is a fault within the zone. IFM sends a trip signal to FIDs on the border of the zone to isolate the faulty section. In case of zero sum for any of the next 10 samples, IFM concludes that there is no fault in the zone, and resets the counter. This protection algorithm mostly relies on working of IFM, which in this case can be a computer or digital relay.

The control algorithm consists of differential and overcurrent protection. It is implemented using the iLN architecture. IEC 61850 models these functions as PDIF and PIOC Logical Nodes [42]. In the protection scheme IFM sends a trip signal to circuit breaker (CB), therefore the control system should have a CB model. According to the standard, CB is modelled as XCBR LN. Thus PDIF LN or PIOC LN issues trip signal to XCBR LN.

implemented in IEC 61499, following the rules defined in [16, 18]. For simplicity, only one zone is exemplified.

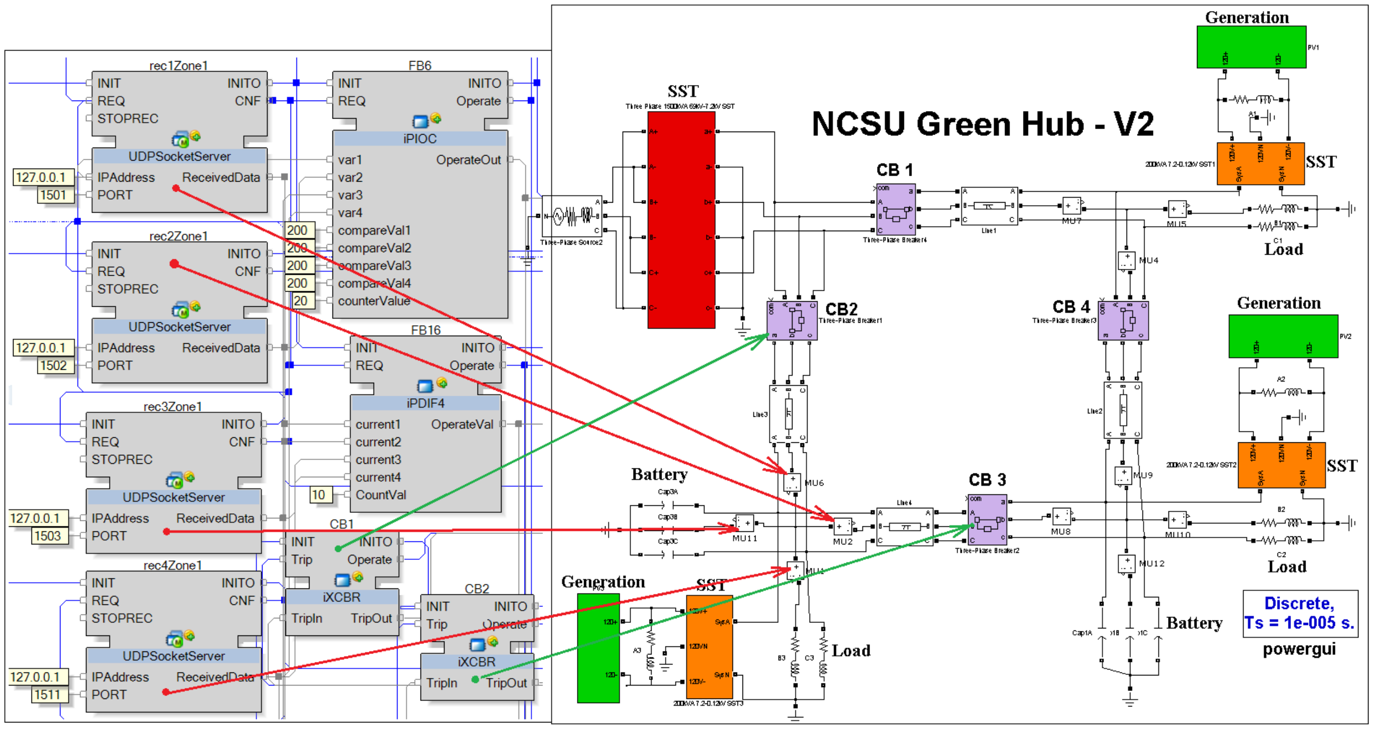


Fig. 9. Control system for one zone of the Green Hub system.

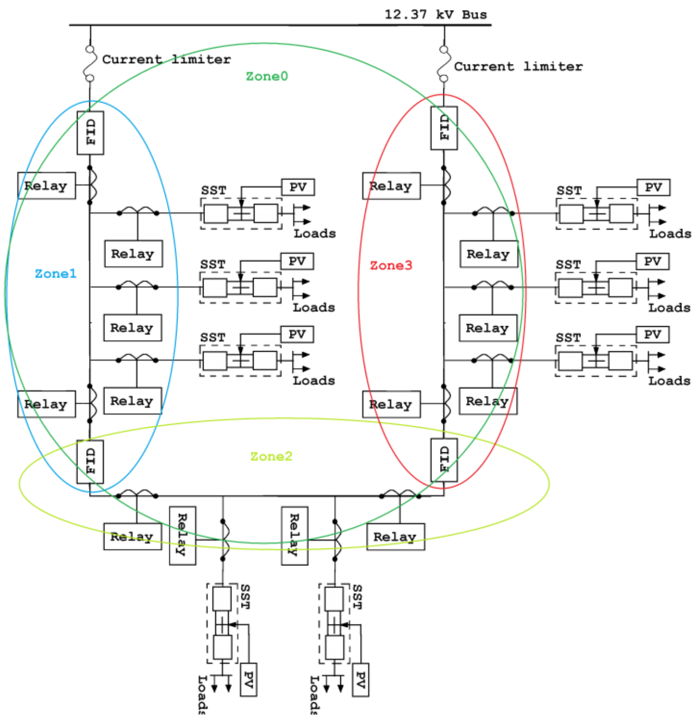


Fig. 8. Green hub loop and its protection zones.

Fig. 9 shows the Green Hub MATLAB model and the corresponding protection system mapped to IEC 61850 and Same data exchange as in the FLISR case is designed. The current measurement points model the current transformer and send current samples to the IFM system developed in IEC 61499, which in turn has XCBR iLN sending open/close commands to circuit breaker in MATLAB model.

The fault is simulated to occur in zone 3, where IFM 3 is operating. All IFM agents are constantly monitoring current within the assigned zones. IFM 3 will notice that the current is out of balance, when sum of the current samples is not equal to zero. It starts counting the number of consecutive instances where summed current is not zero. Once the number reaches pre-set value, in this case 10, the IFM sends the trip signal to XCBRs 3 and 4, which will isolate the faulty zone by tripping. Fig. 10 illustrates the current at the zone 3, where the fault has been injected. Fault is simulated at 0.149 s. IFM has isolated the fault at 0.159 s.

The other IFMs will sense the fault in the overall system and sum of the currents in the zone is no longer zero, however, since the result does not exceed the differential slope, these IFMs do not trip. Thus selectivity of the protection scheme is ensured. After isolating the faulty zone, the current within the non-faulty zones return back to steady state and normal operation. The control system consists of FBs – LN from the developed iLN library. NxtStudio has been used as IEC 61499 execution environment. There is a direct relation (mapping) between equipment and automation functions used in the Green hub system and corresponding FBs (iLNs) in the control model: circuit breaker – iXCBR, receiver of digitalized current samples – UDPSocketServer, differential protection – iPDIF and overcurrent protection – iPIOC.

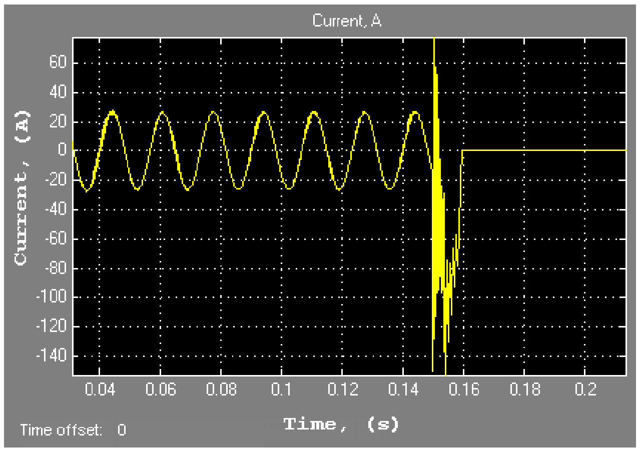


Fig. 10. Current on the load in zone 3.

From Fig. 10, the co-simulation environment without the proxy mechanism isolated the faulty section within 0.010 s. This experiment is for simulation purposes and was not designed to comply with strict timing requirements of protection schemes.

The pattern proposed in Fig. 7 is used to address the compensation for time step duration, along with communication delay and jitter.

The proxy proposed in Section IV to take into account of controller execution time has been implemented as a single Function Block that interacts with service interface FB that is sending and receiving data from the MATLAB side (see Fig. 7). This Function Block (named “TimeCompare” in this experiment) takes the controller’s execution time and the sampling rate used in the simulation as the inputs, and compares them in such a way that it will update the latest received data into the controller only if the accumulated time has exceeded the indicated controller execution time (i.e. representing the controller has finished its execution in this model-time simulation).

Fig. 11 demonstrates the effect of the computation delay. The simulation was run with two different controller execution times: 0.001 s and 0.002 s. The time taken to isolate the faulty section is 0.016 s in the first case, with controller execution time of 0.001 s. The time to trip circuit breaker is twice as long 0.033 s in the second case with controller execution time of 0.002 s. (see Fig. 11). Note, the simulation of the controller and plant model (power system) was performed on the same PC. The future work includes running code on the dedicated hardware and impact of the execution time of that hardware and communication delays can be taken into account.

Relating the experiment to Fig. 4, the g(t) function here is the current of the protection zone 3. The c(t) function is the circuit breaker position, controlled by the protection algorithm developed in IEC 61499. The c(t) =0 - means the circuit breaker is closed, and c(t) =1 - means it is open. In the case of the g(t) function in Fig. 4:

$g_r(t) = f(t) * c_r(t)$ , $c_r(t)$ in the real time example;

$g_1(t) = f(t) * c_1(t)$ , $c_1(t)$ when $t_c > t_d$;

$g_2(t) = f(t) * c_2(t)$ , $c(t)$ when $t_c<t_d$.

That is in case $t_c>t_d$, the $g_1(t)$ function will continue to evaluate its dynamics for another full simulation step. With the next simulation step the decision of the controller $c_1(t)$ will take effect and the $g_1(t)$ starts declining.

So effect of joint simulation comes down to the power system model dynamics being evaluated for another full simulation step.

As the current on the load in the zone 3 is being affected by the fault in the zone, the control algorithm is evaluating the model parameters and making decision to isolate the fault by opening the circuit breakers. If the control signal is received in the middle of the simulation step size ($g_2(t)$), this will take effect in the next simulation step. This is within the precision defined in Lemma 1, property 2 in section IV.

This directly reveals how the execution time affects the result of the simulation. Protection schemes are sensitive to time delays. Required reaction time to open/close circuit breaker is about 3 ms [42]. The longer the reaction time of the controller, the longer the feeder and the equipment on it will be exposed to the high current. This can result in cascading effect of the fault, explosion on the feeder/substation and other harmful circumstances.

This problem can only be spotted by using the time compensation scheme of the proposed co-simulation approach.

The simulation in MATLAB was conducted with the "Variable step" type, "oder45 (Dormand-Prince)" solver. The sampling time of the model is 1e-5 second, simulation type: “Discrete”.

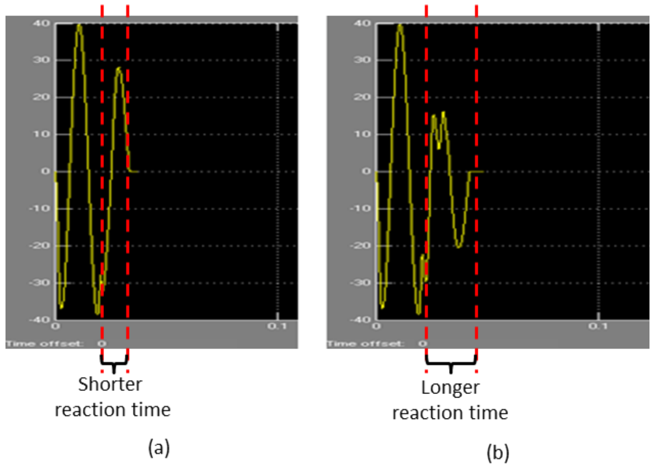


Fig. 11. The simulation output observation (a) without the proposed proxy (b) with the proposed proxy with execution time twice longer than the sampling rate.

The experiments will be extended to perform HiL co-simulation, where the distributed controls will be deployed to a network of communicating IEC 61499 compliant PLCs Beckhoff CX1020. This way real execution times and communication times can be taken into account.

### *B. Multi-agent fault location isolation and power restoration*

The experiment presented in detail in [43], deals with multi-agent implementation of fault location, isolation and service restoration (FLISR) for a model power distribution system whose structure is presented in [44].

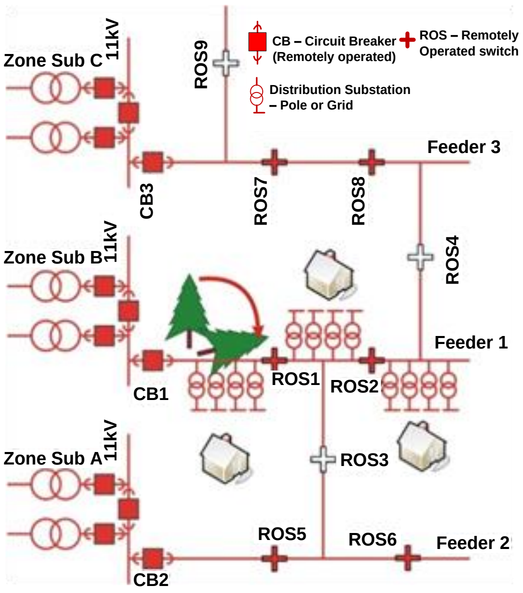


Fig. 12. Sample power distribution utility and location of the fault.

The FLISR scenario was the first work to demonstrate the possibility of co-simulation framework for distributed systems. It provided a real-time event-driven simulation in hardware-in-the-loop simulation with a control, protection device, or other equipment. The user could introduce interactively a phase-to-ground fault into the distribution system during the simulation, and monitor and validate the reaction of the distributed control code on the inserted external disturbance to the system. Thus, the FLISR scenario has proven the feasibility and advantages of the co-simulation framework.

The scenario is as follows. The distribution utility consists of three 11kV feeders supplied by three different zone substations. The 11kV feeders are shown in a simplified form, with only the backbone and ties to adjacent feeders. The scenario begins with a tree falling on the 11kV mains, causing a permanent fault on feeder F1. The feeder protection trips (opens) the circuit breaker CB1 at zone substation B. Sectionalising switches ROS1 and ROS2, being downstream of the fault location, do not register the passage of fault current. In anticipation of possible follow-up action, they remember the load currents that were flowing through them just before the fault occurred. After one attempted automatic re-closure, CB1 goes to lockout.

The distributed multi-agent control implementing this behaviour was implemented using the Intelligent Logical Node Architecture. This architecture implements the logical nodes (LN) proposed in the IEC 61850 standard by means of function blocks of the IEC 61499 standard. Logical nodes correspond to primary equipment of power distribution systems, such as circuit breakers, switches, meters, protection relays, etc. The FB implementation of a LN (called iLN – intelligent logical node) includes the information model and an autonomous agent controller (so called “intelligence”) of this LN. It is envisioned that in Smart Grid the primary equipment will be equipped with embedded controllers executing the iLNs and they will collaborate towards achieving the desired properties of the system. In order to test the resulting behaviour of the network of communicating iLNs, they need to interface real physical primary equipment (circuit breakers, switches and transformers) or a model of it. Following the proposed co-modelling approach, iLNs were connected to the corresponding system model in MATLAB. For execution of the control part the Function Blocks Development Kit (FBDK) [45] was used.

The measurement block in MATLAB, modelling current transformers transmit current samplings to the TCTR iLN in the control code. The XCBR iLN representing circuit breakers in the control code send open/close commands to the circuit breaker in the MATLAB distribution network model.

Results of the co-simulation of the FLISR scenario are presented in Fig. 13. The first graph is the control signal of the corresponding tie switch with the values: 0 – switch open, 1 – switch close. The fault occurs on section CB1, and supply should be restored on ROS1 and ROS2 sections. Fig. 13 demonstrates that all three sections of the feeder 1 had normal current before the fault. As it can be seen from the “CB1” graph, the current transformer detects the fault current of value higher than 2000A at about 3.32s and protection function trips the circuit breaker CB1, so current becomes zero at 3.352s.

After a certain delay, the RREC iLN recloses the circuit breaker at the 3.4s in case it is a temporary fault. However, the protection detects the fault again (the fault current between 3.4s and 3.44s) and trips the circuit breaker, this time RREC goes to lockout. The “CB1” plot shows that power is cut on feeder 1 at the time 3.44s. The difference between 3.4s and 3.44s is the time to get the signal processed and devices to operate.

The switches ROS1 and ROS2, having learned that their sections do not have a fault, decide to request the alternative supply: ROS1 from tie switch ROS3 and ROS2 from tie switch

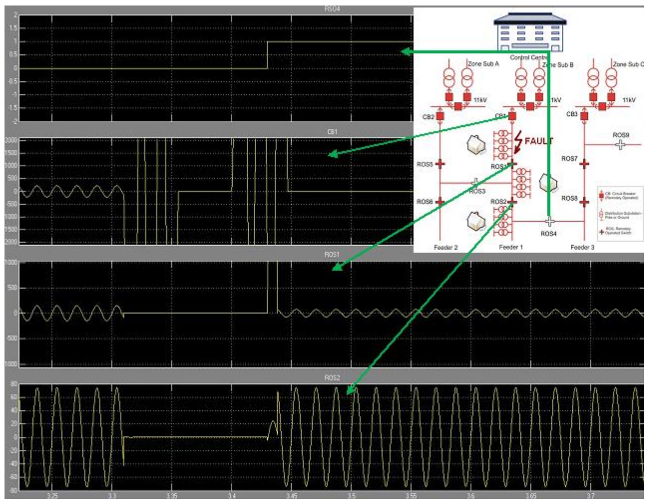


Fig. 13. Simulation results of the FLISR scenario [36].

ROS4. Thus ROS1 and ROS2 have got the supply from adjacent feeders 2 and 3 accordingly: the graphs "ROS1" and "ROS2" show that at the time around 3.48s the current values come back to normal – the power has been restored. Graph "ROS4" illustrates the behaviour of tie switch ROS4, which closes (at 3.46s the value is 1) as there is enough capacity to restore the supply for section ROS2. Graphs "ROS1" and "ROS2" demonstrate the supply restoration on the corresponding sections.

This scenario proves that the distributed control of power grid is possible. Autonomous components of power distribution system can collaborate and sustain power grid operation. The plots demonstrate that FLISR mechanism carried out by intelligent components of the system without central control intervention works: the supply has been correctly restored on the non-faulted sections of the faulted feeder regardless of the fault location.

The co-simulation environment set-up helps determine immediately if the intelligent nodes (iLNs) have been designed appropriately and behave properly to handle specified scenarios. Thanks to the capability of "real-time" event-driven simulation, user is able to test each possible case of fault position (each feeder section), and validate the developed distributed control code.

### C. *Distributed co-simulation of "52 blocking"*

The last test case presented in this paper is the "52 Blocking" application [46] to illustrate the implementation of editable logic within logical nodes. The application describes the safety operation of a circuit breaker by calculating the enable open (EnaOpn) and the enable close (EnaCls) attribute value of the CILO logical node. This example is presented in detail in [47].

The iLN architecture was applied to capture the editable logic, and ISaGRAF was used as an environment for IEC 61499 implementation. To add the editable logic to the iLN architecture, editable logic can be implemented as an additional function block next to the intelligence and the database function block within the iLN architecture.

In the "52 Blocking" application, the function block network is distributed over four devices. Fig. 14 illustrates how the IEC 61499 logical nodes can be distributed over several devices. The GGIO iLNs are distributed in device 1, the XSWI iLNs are distributed in device 2 and the XCBR and the CILO iLNs are distributed in device 3. The 4th device contains the publisher and subscriber function blocks which are used for communication in the co-simulation environment. With the system being distributed, each intelligent logical node relies on the internal intelligence within to co-ordinate the exchange of data in the distributed system.

## VI. Conclusions and Future Work

This paper presents a novel "in the loop" co-simulation approach for distributed automation. Several simulation architectures and use-case scenarios have been presented, including software in the loop and hardware in the loop. Main contributions of the paper are as follows:

- It describes the concept of a co-simulation framework for distributed control environment of Smart Grid where the controls are implemented as event-driven communicating components. The event-driven nature of plant-controller interaction eliminates the free-wheeling issue of cyclic scan based platforms.

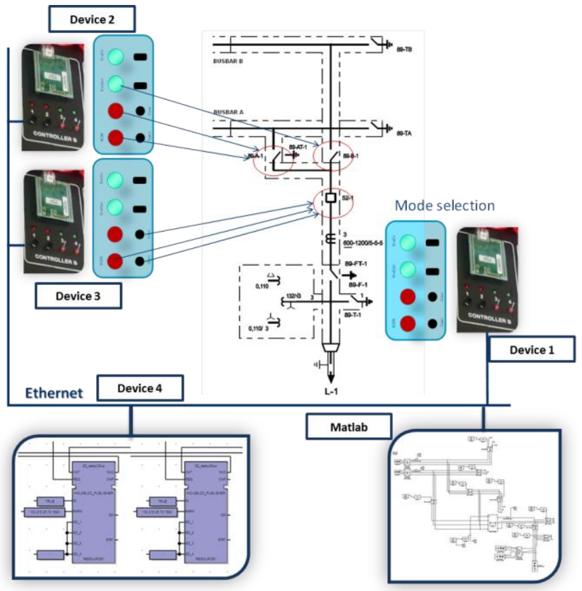


Fig. 14. Distributed setup implementing "52 Blocking" scenario.

- The paper proposed a technique that compensates for computation and communication delays both between plant and controller and controller to controller, and for the time discretization interval (time step) used in the simulation environment;
- The co-simulation design pattern does not require neither deep modification of the simulation model, nor of the control code. The control and simulation environments communicate via a standard communication channel (using UDP). The time compensation technique does not require neither deep modification of the simulation model nor of the control code.
- The criterion of co-simulation correctness has been formulated and implemented in the co-simulation framework.

The results have been demonstrated with three experimental Smart Grid examples reflecting upon different co-simulation architectures, for three different IEC 61499 implementation environments: FBDK, NxtControl and ISaGRAF. Implementation for one more environment Forte has been also developed, but not demonstrated in the paper.

The developed co-simulation approach has proven to be extremely useful in the context of Smart Grid research projects as the means of validation for distributed grid intelligence. The ability to perform system level simulation in the loop provides a convincing argument in favour of distributed control in Smart Grid.

Future work will include extensions of the developed framework for RS-CAD [48] or PS-CAD [49] environments instead of MATLAB, deeper integration of communication networks' properties and integration with the formal verification frameworks, such as the one described in [50].

## ACKNOWLEDGMENTS

This work was supported in part by the research grants of the University of Auckland: FRDF 3622763, PRESS account of the University of Auckland, and NSF FREEDM project ("Executable System Level Smart Micro Grid Model").

## REFERENCES

[1] X. Yu, C. Cecati, T. Dillon, and M. G. Simões, "The New Frontier of Smart Grids," *Industrial Electronics Magazine,* vol. 5, pp. 49-63, 2011.

[2] V. C. Gungor, D. Sahin, T. Kocak, S. Ergut, C. Buccella, C. Cecati, and G. P. Hancke, "Smart Grid Technologies: Communication Technologies and Standards," *Industrial Informatics, IEEE Transactions on,* vol. 7, pp. 529-539, 2011.

[3] M. Liserre, T. Sauter, and J. Y. Hung, "Future Energy Systems: Integrating Renewable Energy Sources into the Smart Power Grid Through Industrial Electronics," *Industrial Electronics Magazine,* vol. 4, pp. 18-37, 2010.

[4] "Smart Grid for Distribution Systems: The Benefits and Challenges of Distribution Automation (DA)(Draft Version 2) White Paper for NIST," ed: IEEE Working Group on Distribution Automation, 2009.

[5] P. Palensky and D. Dietrich, "Demand Side Management: Demand Response, Intelligent Energy Systems, and Smart Loads," *IEEE Transactions on Industrial Informatics,* vol. 7, pp. 381-388, 2011.

[6] C. Cecati, C. Citro, A. Piccolo, and P. Siano, "Smart Operation of Wind Turbines and Diesel Generators According to Economic Criteria," *Industrial Electronics, IEEE Transactions on,* vol. 58, pp. 4514-4525, 2011.

[7] P. Siano, C. Cecati, H. Yu, and J. Kolbusz, "Real Time Operation of Smart Grids via FCN Networks and Optimal Power Flow," *Industrial Informatics, IEEE Transactions on,* vol. 8, pp. 944-952, 2012.

[8] V. Calderaro, C. N. Hadjicostis, A. Piccolo, and P. Siano, "Failure Identification in Smart Grids Based on Petri Net Modeling," *Industrial Electronics, IEEE Transactions on,* vol. 58, pp. 4613-4623, 2011.

[9] T. Godfrey, M. Sara, R. C. Dugan, C. Rodine, D. W. Griffith, and N. T. Golmie, "Modelling Smart Grid Applications with Co-Simulation," in *The 1st IEEE International Conference on Smart Grid Communications (SmartGridComm 2010)*, 2010.

[10] V. Liberatore and A. Al-Hammouri, "Smart grid communication and co-simulation," in *Energytech, 2011 IEEE*, 2011, pp. 1-5.

[11] L. Hua, S. Sambamoorthy, S. Shukla, J. Thorp, and L. Mili, "Power system and communication network co-simulation for smart grid applications," in *Innovative Smart Grid Technologies (ISGT), 2011 IEEE PES*, 2011, pp. 1-6.

[12] T. Strasser, M. Stifter, F. Andren, D. Burnier de Castro, and W. Hribernik, "Applying open standards and open source software for smart grid applications: Simulation of distributed intelligent control of power systems," in *Power and Energy Society General Meeting, 2011 IEEE*, 2011, pp. 1-8.

[13] G. Zhabelova and V. Vyatkin, "Multiagent Smart Grid Automation Architecture Based on IEC 61850/61499 Intelligent Logical Nodes," *IEEE Transactions on Industrial Electronics,* vol. 59, pp. 2351 - 2362 2011.

[14] "Function blocks: International Standard IEC 61499," ed: International Electrotechnical Commission, 2005.

[15] V. Vyatkin, "IEC 61499 as Enabler of Distributed and Intelligent Automation: State of the Art Review," *IEEE Transactions on Industrial Informatics,* vol. 7, pp. 768-781, 2011.

[16] V. Vyatkin, G. Zhabelova, N. Higgins, M. Ulieru, K. Schwarz, and N. K. C. Nair, "Standards-enabled Smart Grid for the future Energy Web," in *Innovative Smart Grid Technologies (ISGT),* Gaithersburg, MD, 2010, pp. 1-9.

[17] T. Strasser, F. Andrén, and M. Stifter, "A test and validation approach for the standard-based implementation of intelligent electronic devices in smart grids," in *Holonic and Multi-Agent Systems for Manufacturing* vol. 6867 V. Marík, P. Vrba, and P. Leitao, Eds., ed Berlin / Heidelberg: Springer 2011, pp. 50–61.

[18] N. Higgins, V. Vyatkin, N. K. C. Nair, and K. Schwarz, "Distributed Power System Automation With IEC 61850, IEC 61499, and Intelligent Control," *IEEE Transactions on Systems Man and Cybernetics Part C-Applications and Reviews,* vol. 41, pp. 81-92, Jan 2011.

[19] R. Bernhardt, G. Schreck, and C. Willnow, "Realistic robot simulation," *Computer & Control Engineering Journal,* vol. 6, pp. 174-176, 1995.

[20] P. Klingstam and P. Gullander, "Overview of simulation tools for computer-aided production engineering," *Computers in Industry,* vol. 38, pp. 173-186, 1999.

[21] H. Carlsson, B. Svensson, F. Danielsson, and B. Lennartson, "Methods for reliable simulation based PLC code verification," *IEEE Transactions on Industrial Informatics,* vol. 8, pp. 267-278, 2011.

[22] V. Vyatkin, H.-M. Hanisch, C. Pang, and J. Yang, "Application of Closed-Loop Modelling in Integrated Component Design and Validation of Manufacturing Automation," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews,* vol. 39, pp. 17-27, 2009.

[23] V. Vyatkin, "Intelligent mechatronic components: control system engineering using an open distributed architecture," presented at the Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA '03. IEEE Conference, 2003.

[24] P. Cheng and V. Vyatkin, "IEC 61499 function block implementation of Intelligent Mechatronic Component," presented at the 8th IEEE International Conference on Industrial Informatics, 2010.

[25] nxtControl.com. (2010). Available: http://www.nxtcontrol.com/

[26] J. Yan and V. V. Vyatkin, "Cyber Physical Approach for Baggage Handling Systems Automation enabled by Multi-Agent Control and IEC 61499 " in *IEEE 9th International Conference on Industrial Informatics (INDIN 2011),* Lisbon, Portugal, 2011.

[27] F. Maturana, R. Ambre, R. Staron, D. Carnahan, and K. Loparo, "Simulation-based environment for modeling distributed agents for smart grid energy management," presented at the Emerging Technologies & Factory Automation (ETFA), 2011 IEEE 16th Conference on, 2011.

[28] C. Chinchul and L. Wootaik, "Analysis and Compensation of Time Delay Effects in Hardware-in-the-Loop Simulation for Automotive PMSM Drive System," *Industrial Electronics, IEEE Transactions on,* vol. 59, pp. 3403-3410, 2012.

[29] C. Dufour and J. Belanger, "Discrete time compensation of switching events for accurate real-time simulation of power systems," in *Industrial Electronics Society, 2001. IECON '01. The 27th Annual Conference of the IEEE*, 2001, pp. 1533-1538 vol.2.

[30] F. Kuhl, R. Weatherly, and J. Dahmann, *Creating computer simulation systems: an introduction to the high level architecture*. Prentice Hall PTR Upper Saddle River, NJ, USA, 1999.

[31] ONERA The French Aerospace Lab, "CERTI - Open Source HLA RTI.," 3.4.0 ed, 2002, pp. HLA 1.3 specification, IEEE 1516-2000.

[32] P. Chombart, "Multidisciplinary modelling and simulation speeds development od automoive systems and software.," ITEA 2, Modelisar, Dassault Systemes2012.

[33] R. Hofer and M. L. Loper, "Dis today [Distributed Interactive Simulation]," *Proceedigs of the IEEE,* vol. 83, 1995.

[34] C. J. Bankier, "GridIQ - A Test Bed for Smart Grid Agents," Bachelor of Engineering (Honours), Department of Information

Technology and Electrical engineering, The University of Queensland, 2010.

[35] MathWorks. (2010). *The MathWorks - MATLAB and Simulink for Technical Computing*. Available: http://www.mathworks.com

[36] B. Vandiver, "Testing of UCA based microprocessor based protective relays," in *Power Engineering Society Summer Meeting, 2002 IEEE*, 2002, pp. 294-296 vol.1.

[37] N.-D. Nhat, K. Gwan-Su, and L. Hong-Hee, "A study on GOOSE communication based on IEC 61850 using MMS ease lite," in *Control, Automation and Systems, 2007. ICCAS '07. International Conference on*, 2007, pp. 1873-1877.

[38] A. P. Apostolov, "Distributed protection, control and recording in IEC 61850 based substation automation systems," in *Developments in Power System Protection, 2004. Eighth IEE International Conference on*, 2004, pp. 647-651 Vol.2.

[39] A. Q. Huang, M. L. Crow, G. T. Heydt, J. P. Zheng, and S. J. Dale, "The Future Renewable Electric Energy Delivery and Management (FREEDM) System: The Energy Internet," *Proceedings of the IEEE*, vol. 99, pp. 133-148, 2011.

[40] A. Thirumalai, X. Liu, and G. Karady, "Novel Digital Protection System for FREEDM Loop," in *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, 2010, pp. 22-26.

[41] G. Zhabelova, V. Vyatkin, and N. Nair, "Standards- based Intelligent Fault Management System for FREEDM Green Hub Model," in *IEEE International Conference on Industrial Electronics ( IECON'11)*, Melbourne, Australia, 2011.

[42] International Electrotechnical Commission, " Basic information and communication structure," in *Communication Networks and Systems in Substations*, ed. Switzerland: International Electrotechnical Commission,, 2010.

[43] G. Zhabelova and V. Vyatkin, "Intelligent Logical Nodes of IEC 61850 and IEC61499 for Multi-agent Smart Grid Automation," *IEEE Transactions on Industrial Electronics,* vol. in print, 2011.

[44] N. Higgins, V. Vyatkin, N. Nair, and K. Schwarz, "Distributed Power System Automation with IEC 61850, IEC 61499 and Holonic Control," in *Proc. IEEE Conference on Systems, Machine and Cybernetics 2008*, Singapore, 2008.

[45] Holobloc Inc. (2008). *Function Block Development Kit (FBDK).* Available: http://www.holobloc.org/

[46] J. G. Moreno, "MODELING OF LOGICS APLICATION USE CASES - FUNCTION: 52 BLOCKING (FOR OPENING AND CLOSING)," 26th September 2010.

[47] C.-W. Yang, V. Vyatkin, and N. Nair, "Implementation of Editable Logic in IEC 61850 Logical Nodes by means of IEC 61499," in *IEEE International Conference on Industrial Electronics (IECON'11)*, Melbourne, Australia, 2011.

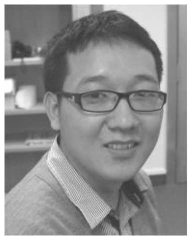
[48] (2010). *RTDS Technologies: The World Standard for Real Time Power System Simulation*. Available: http://www.rtds.com/index/index.html

[49] HVDC. (2010). *PSCAD*. Available: https://pscad.com/products/pscad/

[50] S. Patil, S. Bhadra, and V. Vyatkin, "Configurable non-determinism in a closed-loop modelling and verification framework for embedded control systems," in *IEEE International Conference on Industrial Electronics (IECON'11)*, Melbourne, Australia, 2011.

**Chia-han Yang** received his B.Eng degree (1st class hons.) and PhD in the Department of Electrical and Electronics Engineering from The University of Auckland, Auckland, New Zealand in 2006 and 2011 respectively. His PhD research is investigating methodologies of improving distributed control system design based on IEC61499 standard. His research interests are in the area of distributed control and automation, control system modelling/simulation and robotics.

**Gulnara Zhabelova** is a PhD student at the University of Auckland, New Zealand. Gulnara comleted BE in Mechatronics and Robotics (Hons), ME in Automation and Control (Hons) and ME in Compter Systems (Hons) in 2006, 2008, 2009 respectively. Former two degrees are with Karaganda State Technical University, Kazakhstan, and later one is with The University of Auckland, New Zealand. She is working on application of multi-agent technologies to distributed automation, control and protection of power grid at any level: from transmission to residential consumption. Her interest is in application of information and communication technologies in current power grid and future Smart(er) Grid. Also her interest covers AMI and demand side management within energy infrastructure involving distributed generation and PEV.



**Chen-Wei Yang** received his B.Eng degree (Hons.) and ME degree (Hons) in the Department of Electrical and Computer Systems Engineering from The University of Auckland, Auckland, New Zealand in 2011 and 2012 respectively. Currently, he is pursuing his Ph.D degree at the Department of Electrical and Computer Systems Engineering at the University of Auckland, Auckland, New Zealand. His research interests are in the area of distributed automation and industrial informatics in the field of Smart Grid systems for IEC61850 based systems.

**Valeriy Vyatkin** is Chaired Professor of Dependable Computation and Communication Systems at Luleå University of Technology, Sweden, and visiting scholar at Cambridge University, U.K., on leave from The University of Auckland, New Zealand, where he has been Associate Professor and Director of the InfoMechatronics and Industrial Automation lab (MITRA) at the Department of Electrical and Computer Engineering. He graduated with the Engineer degree in applied mathematics in 1988 from Taganrog State University of Radio Engineering (TSURE), Taganrog, Russia. Later he received the Ph.D. (1992) and Dr. Sci. degree (1998) from the same university, and the Dr. Eng. Degree from Nagoya Institute of Technology, Nagoya, Japan, in 1999. His previous faculty positions were with Martin Luther University of Halle-Wittenberg in Germany (Senior researcher and lecturer, 1999–2004), and with TSURE (Associate Professor, Professor, 1991–2002).

Research interests of professor Vyatkin are in the area of dependable distributed automation and industrial informatics, including software engineering for industrial automation systems, distributed architectures and multi-agent systems applied in various industry sectors: Smart Grid, material handling, building management systems, reconfigurable manufacturing, etc. Dr Vyatkin is also active in research on dependability provisions for industrial automation systems, such as methods of formal verification and validation, and theoretical algorithms for improving their performance. In 2012, Prof Vyatkin has been awarded Andrew P. Sage Award for best IEEE Transactions paper.