Cyber-Physical Components for Heterogeneous Modelling, Validation and Implementation of Smart Grid Intelligence

Gulnara Zhabelova[1], Chen-Wei Yang[1], Sandeep Patil[1], Cheng Pang[1], Jeffrey Yan[2], Anatoly Shalyto[4], Valeriy Vyatkin[1,3]
[1]Department of Computer Science, Computer and Space Engineering, Lulea Tekniska Universitet, Sweden
{Gulnara.Zhabelova, Chen-Wei.Yang, Sandeep Patil, Cheng.Pang}@ltu.se, vyatkin@ieee.org
[2]Dept. of Electrical and Computer Engineering, University of Auckland, New Zealand, jyan101@auckland.ac.nz
[3]Department of Electrical Engineering and Automation, Aalto University, Finland
[4] ITMO University, St. Petersburg, Russia

Abstract— **This paper presents a practical framework to bring the cyber-physical block diagram models, such as Ptolemy, to the practice of industrial automation. Cyber-Physical Component (CPC) architecture is suggested. CPC aims at the improvement of design, verification and validation practices in automation of Smart Grid. IEC 61499 standard is used as a basis for this architecture. This architecture addresses several design software and system engineering challenges: right equilibrium between abstract representation and "executability" and round-trip engineering. An CPC exhibit such properties as portability, interoperability and configurability thanks to the reliance on open standards. The use of time stamp based execution paradigm adds determinism and predictability at the run-time.**

Keywords—distributed grid intelligence, heterogeneous systems, simulation in the loop, component-based engineering, energy management

I. INTRODUCTION

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]. The result will be more efficient power systems capable of better managing the growing power consumption, providing fault resilience and seamlessly integrating Distributed Renewable Energy Resources (DRER) e.g. wind and solar.

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 [2, 3].

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. 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.

Similar challenges in the design and validation process are faced by applications such as control systems, manufacturing, robotics, transportation etc. [4, 5]. A common umbrella for mentioned domains and Smart Grid is cyber-physical systems (CPS). A CPS is an integration of computation (or control) with physical processes [5]. The computing seats within every physical component [4]. It is not enough to understand physical and computational components individually, it is important to understand their interaction [5].

The design of CPS imposes challenges related to time- and event-driven control, variable time delays, failures, reconfiguration and distributed decision support system [4]. Therefore, design of such systems requires understanding joint dynamics of computers, software, networks and physical processes [5]. One of the biggest challenges is validation and certification of complex designs and systems [4]. CPS design approach requires new models, tools and methods, that will incorporate verification and validation of control and physical components at the control design stage [4].

This paper proposes an approach to systems engineering for Smart Grid applications based on the concept of Cyber-Physical Components (CPC). The paper presents cyber-physical design framework with the co-simulation capabilities, that guarantees correct handling of process dynamics in presence of control delays. Co-simulation enables validation of the CPC alone and a system as a whole at the design stages.

Proposed CPC architecture (CPCA) enable provisions for capturing control, simulation model, and visualisation related to a single component of the physical system. Thus, an CPC acquires a complete description of a single element of the system. Proposed CPC is built on IEC 61499, that encourages a component based design, and communication standard for power system automation IEC 61850.

The IEC 61499 standard [6] 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 [7]. 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 [8]. The IEC 61850 standard suggests the concept of Logical Nodes (LN) for object-oriented design of functionalities in Intelligent Electronic Devices (IED). It was proposed in [9] 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.

The structure of the paper is as follows. Section II presents an outline of the proposed CPC architecture. Section III discusses a challenge of combining the CPC architecture with external plant simulation tools into a (multi) closed-loop simulation environment. The challenge is related to the correctness of time synchronisation in the different parts of the environment that impacts on the accuracy and correctness of modelling. A solution is proposed that is essentially based on time stamping of events and correction of the timestamps when events propagate between the parts of the environment. Section IV presents implementation if mentioned time synchronisation mechanism. Section V illustrates the application of this concept on a Smart Grid related use-case. Conclusion summarizes performed work and outlines future goals.

II. Cyber-Physical Components Architecture

As it can be concluded from the previous discussion, there are attempts to address design complexity of automation systems both in terms of model-driven software design, and in terms of their model-driven verification and validation. In this section, a synergy of these activities is presented resulting in the concept of CPC architecture. An CPC is composed according to the Model-View-Control (MVC) pattern described in the following subsection.

The CPC concept and its implementation with Function Blocks can be seen as an attempt to encapsulate in a reusable component the control, simulation, and other functional elements related to a structurally distinguishable part of the system. By doing so, the approach addresses engineering challenges of distributed heterogeneous cyber-physical systems design, following the path of the Ptolemy II environment [10] that addresses the operational semantics of cyber-physical systems using the block diagram notation of their representation.

A. Cyber-physical components architecture

The CPC architecture (a particular case of which, called Intelligent Mechatronic Component (IMC), was proposed in [11]) is an attempt to address both design and validation challenges of distributed systems. The idea of IMC was to allow the developer thinking in terms of machines or their autonomous parts thereof by increasing the level of abstraction.

The CPC architecture is based on the MVC design pattern [12] adapted by Christensen in [13] to the domain of industrial automation and integrated with the IEC 61499 standard architecture. The building blocks of the architecture are CPCs integrating controllers and simulation models. The plant is composed of a set of mechatronic components, denoted by M. Each component $\gamma \epsilon M$ has corresponding software sub-components $S(\gamma) = \{ctl, sim, hmi, view\}$ following the taxonomy of the CPC architecture [14]:

1. $\gamma(ctl)$ is the controller of the component. The vendor of the component can provide at least a controller implementing certain basic operations or services, invoked by a higher level controller, or, in some cases, a more sophisticated software agent capable of self-organization with other such components.
2. $\gamma(sim)$ is a simulation model of the component intended to be used in closed-loop simulation with the control. Paper [11] proposes a framework for a systematic approach for the composition of simulation and control in a distributed system of intelligent mechatronic objects. One should note that this element can be just a placeholder for the simulation model, with actual functionality implemented in an "external simulation environment", as illustrated in Figure 1, where each component γ_i is represented by the corresponding simulation component $e(\gamma_i)$.
3. $\gamma(hmi)$ implements the human-machine-interface (HMI) that provides interfacing to the component and is analogous to the HMI of Supervisory Control and Data Acquisition (SCADA) systems. The HMI is connected in closed-loop with $\gamma(ctl)$ and facilitates user interaction with the control application.
4. $\gamma(view)$ is a visualization of the dynamics of the simulation model. Since the simulation model is executing in closed-loop with the controller, a live view of the component dynamics can be monitored at the runtime.

This methodology will mainly focus on the physical relationships between components. Topology of the plant configuration can be expressed in the form of an attributed directed graph. Representing this information as a graph opens up the possibility of using a variety of graph transformation techniques [15, 16] to assist in automatic software generation.

B. Composition of components

Specific inter-component and intra-component communication is defined by the tuple $C = \langle R, Q\rangle$, where $R \subseteq S(\gamma) \times S(\gamma)$ specifies the intra-component communication between software sub-components $S(\gamma)$ within a single component γ.

Figure 1 shows an example of two components in the downstream relation, including their internal architecture. For a single component, the set of communication defining relations is expressed as follows:

1. $R(\gamma(sim), \gamma(ctl))$ and $R(\gamma(ctl), \gamma(sim))$: Bidirectional communication between the simulation and control.
2. $R(\gamma(sim), \gamma(view))$: Unidirectional communication between simulation and view representing state data from the simulation passed to the view for rendering.
3. $R(\gamma(ctl), \gamma(hmi))$ and $R(\gamma(hmi), \gamma(ctl))$: Bidirectional communication between control and HMI representing control panel interfacing with the controller application.

Communication between two components γ_i and γ_{i+1} is shown in Figure 1

An arc connects these two components and is assigned with the *downstream* relation attribute.

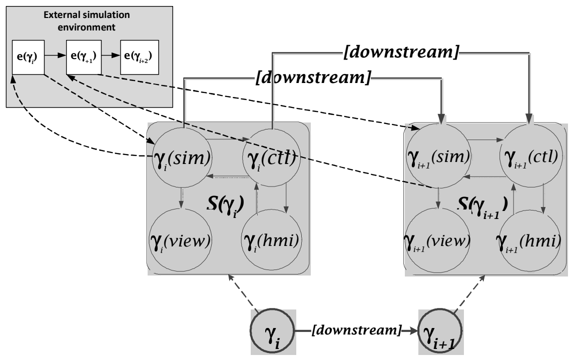

Figure 1 - Communication between two components with a downstream topological relationship [17].

As a result of applying this pattern, the model of the system's behaviour can be designed with nearly complete re-use of the component models.

III. Event-driven Closed-Loop Simulation Environment

Once a control-system model is composed from CPCs its functionality must be verified. The "*sim*" parts of the CPCs provide a ready to use simulation model of the system that looks like a multi- closed-loop (CL) simulation environment. It should be noted, though, that the functionality of the "sim" models does not need to be implemented in the same language as the control part (i.e. IEC 61499). Instead, advantage can be taken of the available and well-developed simulation environments such as MATLAB, OpalRT, PowerWorldSimulator and Eurostag, connected to the "sim" sub-components via a communication channel (as shown in Figure 1).

Model evaluation in some of the mentioned simulation environments is based on time steps that 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 is capable to process. 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.

In case when CPC are implemented using IEC 61499 function blocks, 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 language. IEC 61499 follows an event-driven control paradigm therefore the classic control loop setup in Figure 2 (a) needs to be modified as follows. As an example of controller algorithm, suppose the controller is reading a plant parameter $s(t)$ trying to keep it in certain boundaries by changing a control variable $a(t)$. Whenever $s(t)$ crosses a threshold boundary (event e_1 in Figure 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 its output o and notifies the plant by a message (received at e_4), when the value of $a(t)$ changes to the updated controller output o.

As plant and controller are concurrently active, the plant state variables change values following the plant's dynamics in the interval $[e_1, e_4]$ while the controller takes time to change the value of o.

In a CL-simulation environment composed of concurrently running event-driven controller and model of the plant the difference in behaviour is as presented in Figure 4Figure 3. The plant model operates in discretized time intervals (steps S_1, S_2, …, S_i), of a fixed (Δt) 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.

As the simulation is performed using fixed or variable step durations, the controller communication feedback will be read at the beginning of such a 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.

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 making the reaction time of the controller longer than it really is. This may allow for error to accumulate and result in a greater overshoot.

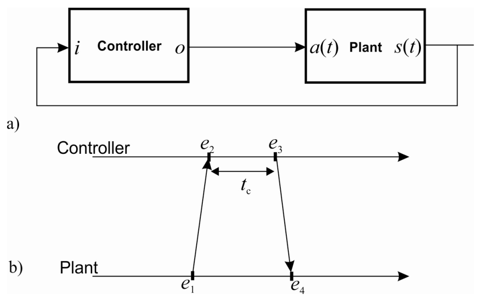

Figure 2. a) Closed-loop plant-controller system; b) Event-driven communication between plant and controller.

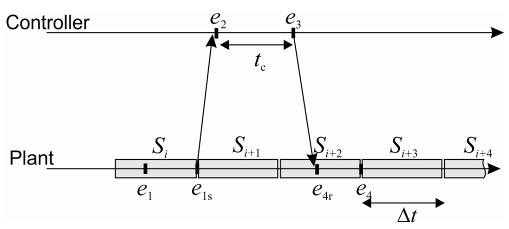

Figure 3. Plant-Controller interaction in CL-simulation.

If the computation time is shorter than the simulation step size the controller command will take effect only in the next simulation step. In both cases the overshoot of the control parameter $s(t)$ can be significant. This can make the simulation results much different from the real life ones which diminishes 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 Δt 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 Figure 4.

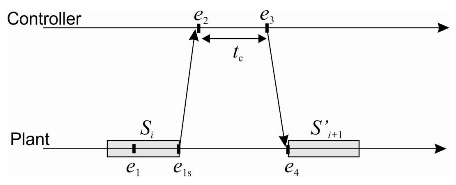

Figure 4. Adjusted CL-simulation environment.

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.

This setup would be free of the problem discussed above, but would have another problem: the controller’s 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 s(t) based on this adjusted time. Implementation of this adjustment will be considered in Section IV.

IV. Implementation of Closed-Loop Simulation Framework

Based on the observations presented in Section III, the CL-simulation environment can be created in a general form presented in Figure 5 (a). The “Controller“ and “Plant model” blocks represent the controller execution environment in function blocks and model of the plant respectively.

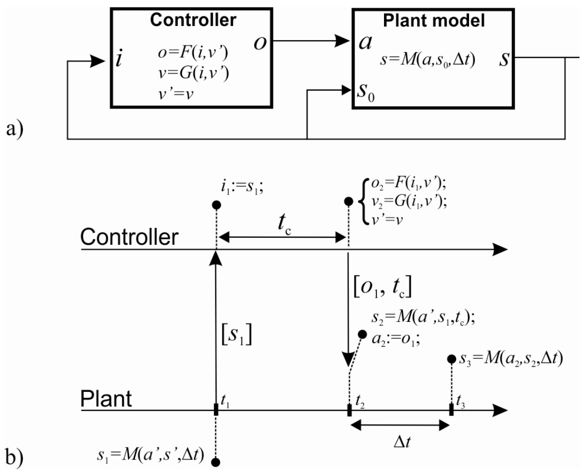

Figure 5. Closed-loop simulation framework for CPC.

It is assumed that the controller has a reactive discrete state logic that determines outputs as a function F of the current inputs i and internal state v. The internal state also changes as a function G of inputs and internal state in the previous controller invocation (v’).

Model of the plant calculates the vector of system variables s as a function of control signals a and of initial values of system variables, denoted by s_0, and duration of the modelling step Δt. The exchange of values between the controller and plant model is implemented by message passing, that is implemented in IEC 61499 framework by event-passing mechanism with associated data.

The mechanism is illustrated in Figure 5 (b) and is commented as follows:

1. Precondition: Before the time moment t_1, the value of model’s variables has been evaluated to s_1.
2. At the moment t_1 this value is sent to the controller in message [s1].
3. Controller, that has vector of internal variables v' from the previous invocation, assigns s_1 to its inputs i and starts execution in order to evaluate the outputs o_2 and internal variables v_2. Duration of the controller execution is t_c.
4. The value of outputs and the duration are sent to the Model in a message $[o_1, t_c]$. Measuring of the t_c could have been done also on the Model’s side, however, in the case of “software in the loop” configuration, actual execution time may be different from that on actual hardware platform. The software in the loop environment can however provide an accurate estimation of the real t_c knowing the profile of the target controller.
5. Upon receiving this message, the model of the plant recalculates its state s_2 to compensate for the inactivity during the controller’s calculation.
6. Then, the model of the plant is invoked one more time in order to calculate system’s state for the regular step Δt.

The proposed CL-simulation method is implemented 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. There is a proxy function block that cooperates with service interface FB sending and receiving data from the Matlab side as shown in Figure 6. 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).

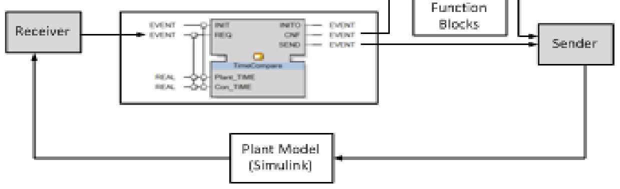

Figure 6. Implementation of the proposed CL-simulation framework.

One can possibly note the conceptual similarity of this approach with time-stamping of events used in the pTides environment [18] for achieving functional invariance of distributed system model in different heterogeneous environments.

V. Distributed Grid Intelligence Application

This section will briefly illustrate an implementation of a "Load Balancing" application developed using the proposed CPC architecture and tested using the described simulation method. The load balancing algorithm is used to normalize the overall load within FREEDM distribution network among the Intelligent Energy Management (IEM) nodes i.e. peers. FREEDM system [19] is a prototype of Smart Grid of a reasonable complexity, with modular architecture. Devices such as (DRER), Distributed energy storage device (DESD), Smart Meters and loads are grouped and managed by Energy Hub based on Solid State Transformer (SST) with embedded intelligence. CPC concept is demonstrated on SST. The SST intelligence is an IEM node. IEM node is essentially an agent, participating in distributed algorithms collaborating with other IEM nodes (agents). IEM nodes comprise an DGI, i.e. multi-agent system.

The DGI nodes run the load balancing algorithm, communicating their energy balance information with each other in order to transfer the load to the nodes with excess generation. The load balancing algorithm is presented in [20] and its implementation with CPC architecture and IEC 61499 function blocks technology was reported in [21]. This design enables modularity of the code, easy re-configurability of the developed system and results in executable system specification. The developed modular code is directly deployable to the embedded hardware platform of the energy hub that is an ARM board.

The developed FB network of Load Balancing application of three IEM nodes is directly executable. Since SST controller is realized on the ARM board, the FB network of Load Balancing application was deployed to three ARM boards. The results of testing via co-simulation are described in the following section.

A. Results and Discussion

Brief literature search shows, that current research is more focused on design and modelling of CPS, rather the component i.e. CPC [22-25]. The common architecture emploed by the authors includes: physical process, computational process and feedback or closed loop control. Not all approaches mention communication network as part of the architecture. The physical part of the CPS models the entire plant, while the computational part models the entire control for the system. The physical part is modelled in power system simlation tools. The control system might take form of distributed control or multi-agent control, and executed on a PC or PC like platfroms.

In this work the CPC is proposed, which are the components used to build the CPS. The CPC architecture includes: physical process or device, controller, HMI and view. These are clearly separated, enabling modular strucutre of teh CPC itself. Each of teh parts of the CPC can be changed without affecting the rest of the CPC. The communication and synchronisation mechanism between the physical and computational parts are formalised as described in previous sections.

Similar approach is described in [26]. Simko in [26], the CPS component is described as a set of computational and physical variables and physical and computational ports and power port dependent on modeling environment and physical domain. Ports defines the interface of the component. The component variables and ports are changed over the time, charachterizing the behaviour of the component. Thus, the component is defined as tuple cosisting of set of variables, a set of ports, and a set of behaviours. The CPS the is the composition of the defined components. Then the definition and semantics of the connections defined in terms of power physical, and computational connections. In this approach, phycial and computational parts of the CPS component are decomposed into the variables and ports, and defined flat in sets.

Another similar approach is described by Macana *et. al.* [25], where to each physical component in the system (micro-grid) corresponds a controller (computational process). These componensts are the distributed generation, loads and the electrical network. A several micro-grids designed in such way, compose the Smart Grid as CPS. However, the paper does not detail the approach and the CPS architecture.

Figure 7 presents the developed CPC for SST. The physical component (*sim*) is SST. SST is modelled in Matlab. Using UDP sockets, the SST communicates its parameters such as P_GATEWAY, P_LOAD, P_SOC and P_PV to the controller, that corresponds to the power supplied by the main grid, load of each IEM node, battery state of cahrge and PV power respectively. Controller (*ctrl*) is modeled in IEC 61499 as a composite FB "*LBNode*". Controller communicates commands such as P* (operating set point) to the SST using UDP sockets. SST and *LBNode* are in the closed loop (feedback loop).. The communication is synchronised as described above. Additionally, the communication network can be modeled and impact of delays, network overload and failures can be studied. Figure 7 shows the *view* part of the CPC. It is connected to the physical component via UDP and displays real-time information of SST parameters as graphs. In the case of SST CPC, the *hmi* component is not present, as teh user is not given control over the SST. SST is cotroller by the *LBNode* according to the load balancing algrotihm.

The system consists of three SST and therefore three CPCs. The designed CPC is instantiated for each SST. The resultant system is shown on Figure 8. Each SST in physical world (Matlab model) has a corresponding part in controller *LBNode*. The node to node communciation is carried out via UDP blocks. The *hmi* and *view* parts of each CPC form HMI of the distribution system, similar to SCADA HMI.

CPC architecture is modular. This allows for system to be tested at the design stage. The unit testing can be perfomed for each CPC. Then the CPS integration testing can be performed. In both cases Matlab, representing physical world, and nxtStudio, IED for controller parts, are running in parallel. Issues of synchronising two environemts are addressed with the synchronisation mechanism.

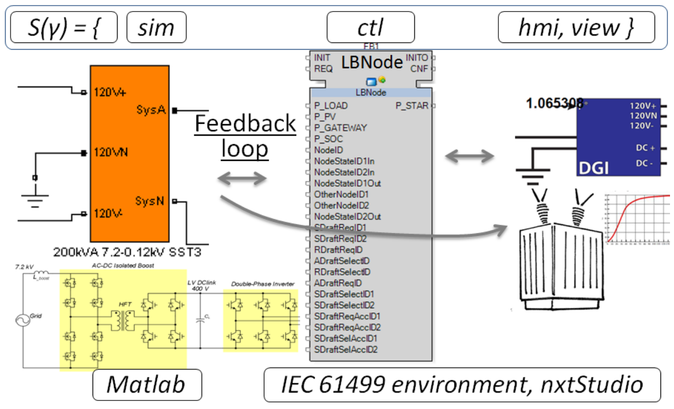

Figure 7. CPC for and SST.

The CPS in load balancing application was tested with *LBNodes* are executed on ARMboard and FREEDM distribution network is modelled in Matlab.

Conclusion

This paper presents a practical implementation of the cyber-physical component architecture using the IEC 61499 standard connected with external simulation environments, such as Matlab. The proposed time correction technique ensures correct closed-loop simulation results independently of actual speeds at which IEC 61499 execution environment and simulation environment runs. An application of this architecture for development and validation of a distributed grid intelligence application has been demonstrated.

This approach potentially paves the way to the use of multi-agent control technology in power system domain. The use of IEC 61499 as underlying technology enables direct execution of such applications on industrial-grade hardware platforms. Practical result was achieved with Load Balancing application which was deployed to ARM board (SST controller) and co-simulated with Matlab FREEDM system model.

Acknowledgment

This work has been supported, in part, by FREEDM NSF Centre grant Y6.F7 and by Luleå Tekniska Universitet.

References

[1] X. Yu, C. Cecati, T. Dillon, and M. G. Simões, "The New Frontier of Smart Grids," *Industrial Electronics Magazine,* vol. 5, no. 3, pp. 49-63, 2011.

[2] "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.

[3] P. Palensky and D. Dietrich, "Demand Side Management: Demand Response, Intelligent Energy Systems, and Smart Loads," *IEEE Transactions on Industrial Informatics,* vol. 7, no. 3, pp. 381-388, 2011.

[4] R. Baheti and H. Gill, "Cyber-physical Systems," in *The Impact of Control Technology*, T. Samad and A. Annaswamy, Eds., ed: IEEE Control Systems Society, 2011.

[5] E. A. Lee and S. A. Seshia. (2011). *Introduction to Embedded Systems - A Cyber-Physical Systems Approach*. Available: http://LeeSeshia.org

[6] V. Vyatkin, "IEC 61499 as Enabler of Distributed and Intelligent Automation: State of the Art Review," *IEEE Transactions on Industrial Informatics,* vol. 7, no. 4, pp. 768-781, 2011.

[7] 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.

[8] 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, no. 1, pp. 81-92, Jan 2011.

[9] G. Zhabelova and V. Vyatkin, "Multiagent Smart Grid Automation Architecture Based on IEC 61850/61499 Intelligent Logical Nodes,"

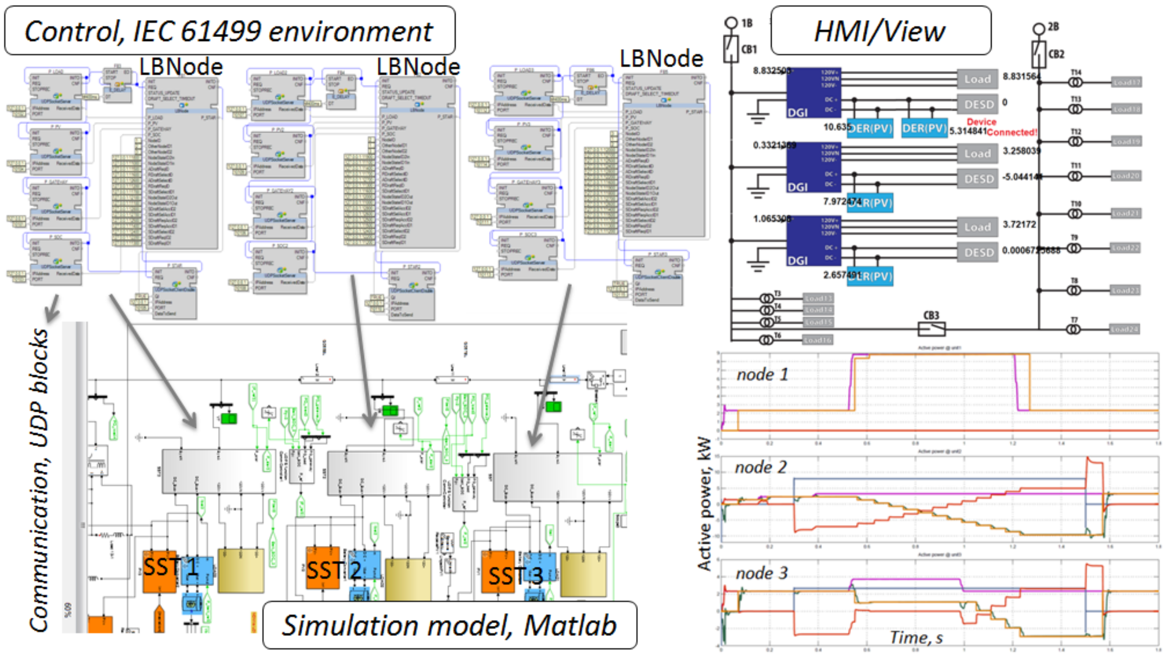

Figure 8. Load Balancing CPS consisting of three CPC components. Each SST in Matlab model has corresponding *LBNode* in the controller, and corresponding representation in HMI/View. UDP FBs are used to interface the controller to the model.

IEEE Transactions on Industrial Electronics, vol. 59, no. 5, pp. 2351 - 2362 2011.

[10] E. A. Lee and H. Zheng, "Operational semantics of hybrid systems," in *Hybrid Systems: Computation and Control*, ed: Springer, 2005, pp. 25-53.

[11] V. Vyatkin, H. M. Hanisch, P. Cheng, and Y. Chia-Han, "Closed-Loop Modeling in Future Automation System Engineering and Validation," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews,* vol. 39, no. 1, pp. 17-28, 2009.

[12] (1979). *Model-View-Controller design pattern*. Available: http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html

[13] J. H. Christensen, "Design patterns for systems engineering with IEC 61499," presented at the Verteilte Automatisierung - Modelle und Methoden für Entwurf, Verifikation, Engineering und Instrumentierung, Magdeburg, Germany, 2000.

[14] V. Vyatkin, "Intelligent mechatronic components: control system engineering using an open distributed architecture," in *Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA '03. IEEE Conference*, 2003, pp. 277-284 vol.2.

[15] U. Ryssel, J. Ploennigs, and K. Kabitzsch, "Generative function block design and composition," in *Factory Communication Systems, 2006 IEEE International Workshop on*, 2006, p. 253.

[16] C. Huijs, "A graph rewriting approach for transformational design of digital systems," in *EUROMICRO 96. 'Beyond 2000: Hardware and Software Design Strategies'., Proceedings of the 22nd EUROMICRO Conference*, 1996, pp. 177-184.

[17] J. Yan and V. Vyatkin, "Distributed Software Architecture Enabling Peer-to-Peer Communicating Controllers," *Industrial Informatics, IEEE Transactions on,* vol. 9, no. 4, pp. 2200-2209, 2013.

[18] J. Zou, S. Matic, E. A. Lee, T. H. Feng, and P. Derler, "Execution strategies for ptides, a programming model for distributed embedded systems," in *Real-Time and Embedded Technology and Applications Symposium, 2009. RTAS 2009. 15th IEEE*, 2009, pp. 77-86.

[19] 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, no. 1, pp. 133-148, 2011.

[20] R. Akella, M. Fanjun, D. Ditch, B. McMillin, and M. Crow, "Distributed Power Balancing for the FREEDM System," in *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*, 2010, pp. 7-12.

[21] S. Patil, V. Vyatkin, and B. McMillin, "Implementation of FREEDM Smart Grid distributed load balancing using IEC 61499 function blocks," in *Industrial Electronics Society, IECON 2013 - 39th Annual Conference of the IEEE*, 2013, pp. 8154-8159.

[22] T. Sanislav and M. Liviu, "Cyber-Physical Systems - Concept, Challenges and Research Areas," *Control Engineering and Applied Informatics, CEAI,* vol. 14, no. 2, pp. 28-33, 2012.

[23] A. Choudhari, H. Ramaprasad, T. Paul, J. W. Kimball, M. Zawodniok, B. McMillin, and S. Chellappan, "Stability of a Cyber-physical Smart Grid System Using Cooperating Invariants," in *Computer Software and Applications Conference (COMPSAC), 2013 IEEE 37th Annual*, 2013, pp. 760-769.

[24] M. J. Stanovich, I. Leonard, S. K. Srivastava, M. Steurer, T. P. Roth, S. Jackson, and B. M. McMillin, "Development of a smart-grid cyber-physical systems testbed," in *Innovative Smart Grid Technologies (ISGT), 2013 IEEE PES*, 2013, pp. 1-6.

[25] C. A. Macana, N. Quijano, and E. Mojica-Nava, "A survey on Cyber Physical Energy Systems and their applications on smart grids," in *Innovative Smart Grid Technologies (ISGT Latin America), 2011 IEEE PES Conference on*, 2011, pp. 1-7.

[26] G. Simko, T. Levendovszky, M. Maroti, and J. Sztipanovits, "Towards a Theory for Cyber-Physical Systems Modeling," in *Proceedings of the 3rd Workshop on Design, Modeling and Evaluation of Cyber Physical Systems (CyPhy)*, Vancouver, Canada, 2013.