Towards Formal Verification of Smart Grid Distributed Intelligence: FREEDM case

Sandeep Patil[1], Gulnara Zhabelova[1], Valeriy Vyatkin[1,2], Bruce McMillin[3]
[1]Department of Computer Science, Electrical and Space Engineering, Luleå University of Technology, Luleå, Sweden.
[2]Department of Electrical Engineering and Automation, Aalto University, Finland.
[3]Department of Computer Science, Missouri University of Science and Technology, Rolla, Missouri
{sandeep.patil, gulnara.zhabelova}@ltu.se, vyatkin@ieee.org, ff@mst.edu

***Abstract*—This paper presents a model-checking framework for the purpose of design and implementation of robust smart grid applications based on distributed intelligence. The paper first introduces distributed grid intelligence approach to smart grid automation and related challenges of their verification. . We then introduce the case study example and how model-checking can be applied to the presented system implemented in IEC 61499 standard.**

In the end we present the initial results of our model-checking application to smart grid applications. The paper will conclude with some issues faced during the research and corrective steps to address these issues in future.

Keywords—model-checking, smart grid, IEC 61499, automation systems

I. INTRODUCTION

Modelling of cyber-physical systems (CPS) [1] with block diagram languages is becoming a mainstream trend. A well-known academic example of such heterogeneous modelling environment is Ptolemy II [2] developed at Berkeley. Industrial automation has been one of the sources for challenges in CPS design and analysis. The executable component architecture of IEC 61499 allows modelling CPS composed of physical processes (a.k.a. plant) combined with control and communication. The model can be used for validation of system-level properties before deployment.

Smart Grid [3] can be viewed as on such CPS. The term smart grid is used to denote novel power distribution infrastructure (grid) enhanced with latest computer and communication technologies that is being widely investigated worldwide as the replacement for current systems. It is intended that it will prevent the cascading failures and blackouts that currently occur, as well as allowing the easy integration of renewable energy sources such as wind, solar, and geothermal generators, together with the widespread use of electric vehicles. Current grid architecture with centralized and hierarchical control is one of the principal barriers to microgeneration (e.g. via solar photovoltaics) at the household level.

The Smart Grid concept has been widely researched and prototyped in the last decade. Investigation towards the sustainable and efficient energy delivery system, has identified two significant paradigm shifts [4]. The future infrastructure should be ready for wide spread of distributed renewable energy resources (RES) and bi-directional energy flow. In order to integrate components of the future grid such as distributed renewable energy resource (DRER), Plugin Electric Vehicle (PEV) and distributed energy storage device (DESD), the grid needs to be smarter. The Smart Grid promises efficient, sustainable and reliable electricity infrastructure, where every member of the grid (consumer, energy retailer, operator and generation.) is an active participant in fast-paced real-time energy market. While the creation of 'distributed nervous system' architecture for the grid is still in the research domain, great effort is being undertaken to bring advanced research results into industrial practice. Our team has recently championed the development of a new SmartGrid multi-agent automation architecture, based on decentralized intelligent decision-making nodes [5, 6] and utilizing several advanced embedded computing and software technologies, such as function blocks[7], software agents [8, 9] and communications [10].

The Future Renewable Electric Energy Delivery and Management (FREEDM) centre is a part of the world's research effort to realise Smart Grid. FREEDM project envisions the backbone of future grid to be a new power distribution infrastructure, allowing for integration of Smart Grid components in a "plug and play" manner and enabling bi-directional flow of electricity. This new energy delivery infrastructure forms the "Energy Internet" [4].

To control such a complex infrastructure, the FREEDM system needs to perform energy and power management, energy dispatch, dynamic reconfiguration, power balancing, maintaining system stability and reliability, fault identification and location (self-healing), and managing business policies while maintaining security of grid participants. Within the FREEDM context, these functionalities constitute Distributed Grid Intelligence (DGI).

FREEDM also emphasizes on the Robustness of the DGI applications. Towards the goal of achieving robustness in DGI applications, in this paper we present a model checking approach to check robustness of two DGI applications implemented using IEC 61499 reference architecture.

Modelling of smart grid systems with block diagram languages is becoming a mainstream trend [11-15]. The function block architecture of the IEC 61499 standard [7] is based on the event-driven block diagrams and is used for modelling and implementation of distributed automation

systems in various branches of industry. The executable component architecture of IEC 61499 allows modelling smart grid systems composed of physical processes (a.k.a. plant) combined with control and communication. The model can be used for validation of system-level properties before deployment. The most common validation method is simulation, but it has well known limitations in the discovery of possible faults. Model-checking [16, 17] presents a complementary method of exhaustive testing, however it is also limited by possible complexity explosion when the original system uses rich data-types and computations. In the past 15 years of research certain progress [18] was achieved in formal modelling of the IEC 61499 architecture, limited by support of very simple data types, non-timed semantics and small scale systems. In the meantime the power of model-checking tools and sophistication of supported model-checking techniques has substantially improved and new opportunities emerge.

With increasing complexity of developing program systems, visual languages has become more and more popular [19]. In applied computer programming, for example, UML graphical notation is in fact a standard for representing program architecture, use cases etc. In industrial automation, the IEC 61131-3 standard defines Function Blocks Diagrams (FBD), Ladder Diagrams (LD), and Sequential Function Charts (SFC) as graphical languages for high-level control software definition. The new standard IEC 61499 defines function block diagram language as a basis one.

The rest of the paper is organized as follows: The section II presents the two case study examples. A brief introduction to model checking is given in sections III, section IV presents some robustness requirements and properties to be verified. Section V presents the initial results. The paper then concludes and presents the future work contribution in section VI.

II. Case Study Examples

Two DGI applications created in the FREEDM project can be exemplified by the following two: Intelligent Fault Management (IFM) [20] and Load Balancing [21].

A. Intelligent Fault Management

IFM device is a new generation fault isolation device with intelligent control.

Fig. 1 shows FREEDM's zone concept protection scheme where the system is divided into zones, using fault isolation devices (FID, new generation circuit breakers). These FID's are places at the borders of each zonal section. In Fig. 1 (a) the protection scheme is divided into three zones and an overall zone (henceforth referred to as zone 0) as a backup protection for overall system. Fig. 1 (b) shows one such zone with energy resources, storage devices and some load. At each zone merging unit is placed at the terminal of distribution line and the feeder of the load to measure current, digitize and transfer the sensed values to IFM controller. Each zone has an IFM, which runs the protection algorithms and incorporates into DGI – distributed grid intelligence. The overall protection scheme consists of primary and secondary protection. Primary protection used is the differential scheme: if the sum of current in a zone equals zero indicates either there is no fault or fault is outside the zone. In case the sum is not zero then the fault is within the zone and IFM makes decisions to trip the FID's at the border of the faulty section (eq.1).

$$\forall i \sum_j \left(in_i^j + out_i^j\right) = 0 \qquad (1)$$

where $i = zone\ and\ j = current\ in\ the\ zone$

The secondary protection (overcurrent protection) of IFM (Zone 0) examines each incoming current sample; then compares against pre-set current value, which can be 3-5 times of the rated current. The current from zone 1, 2, 3 are evaluated at the central IFM, and in case of fault, trip is sent to corresponding FID's (eq.2).

$$\forall i\ current_i^j > \varepsilon \implies fault_i \qquad (2)$$

where $i = zone, j = current\ in\ the\ zone$

$and\ \varepsilon\ is\ the\ thresshold$

IEC 61499 implementation of this control scheme can be found in [20].

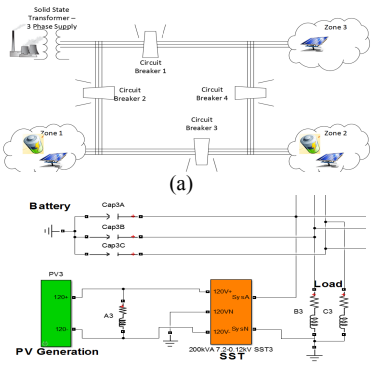

Fig. 1: (a) FREEDM's zone concept protection scheme, (b) a zone with energy storage device, energy resource, SST (DGI with IFM) and some load.

B. Load Balancing

Another DGI application critical for FREEDM operation is Load Balancing. The LB algorithm is used to normalize the overall load within the FREEDM distribution network among the IEM peers. The DGI nodes, running the load balancing algorithm, communicate their load changes with each other in order to transfer the load from a node with excess load to a node that is need of additional load in order to maintain stability of the distribution system. The LB algorithm was developed by the FREEDM centre and presented in [22].

Fig. 2 shows the state flow diagram of the distributed load balancing algorithm as presented in [22].

A simplified version of the algorithm is explained below in three simple steps:

1) Every IEM peer (DGI node) computes the actual

(house) load of the SST (represented by X_{Actual}). Actual load is defined as the difference between total load consumption (represented by X_{Load}) and total load (power) production by the energy resources (DRER) (represented by X_{DRER}), i.e

$$X_{Actual} = X_{Load} - X_{DRER} \quad (3)$$

2) An IEM node is considered to be in a *Low* state (excess load, i.e. *Supply*), if actual load is less than *Threshold*, i.e. $X_{Actual} < Threshold$. It is thought to be in a *High* state (inadequate load, i.e. *Demand*), if actual load exceeds the *Threshold*, i.e. $X_{Actual} > Threshold$. Threshold is defined based on an optimization heuristic. In any other cases, the IEM is in a *Normal* state (has just sufficient load);

3) The IEM node in a *Low* state then communicates with the IEM node in a *High* state for power migration following the state chart as shown in Fig. 2

Complete algorithm details can be found in [22].

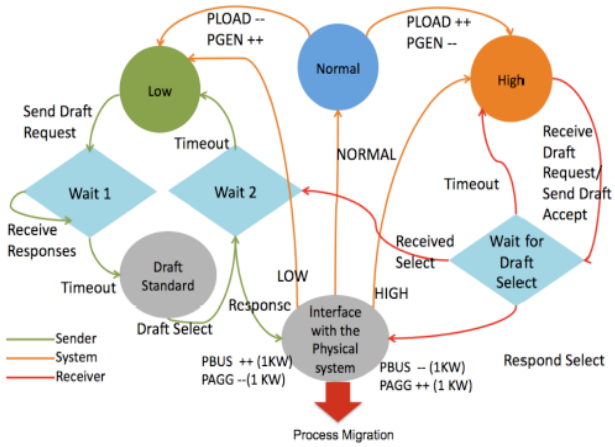

Fig. 2: State diagram of a distributed load balancing scheme [22]

III. Fomral Verification and Model Checking

Formal verification is an act of proving or disproving an algorithm with respect to some specification or property. Model-checking is one such formal verification approach introduced in early 1980s by Clarke and Emerson [16, 17]. A model-checker generates state space of the model that includes all or some states and transitions. Each path in the state space corresponds to one system's run or a single test case. Model-checking has three notable advantages:

1. It enables the unsupervised automatic verification process of a system;
2. It identifies system failure via counter examples(a scenario (state trace) that breaks the safety property of the system); and
3. It makes use of temporal logic for specifications so that model checker can automatically check for different properties (including safety properties).

While model-checking is hungry on computational resources, it has been successfully used in other adjacent areas of computer systems engineering, such as hardware design, thus proving its ability to handle problems of reasonably large complexity[23, 24]. This suggests that it can be also applied in the industrial automation domain, and there have been impressive number of research projects towards this goal such as [18, 25]. Block-diagram system models have been formally modelled by Bae, K. et al.[26], and IEC 61499 has been modelled using various formalisms [27-29]. In works [30-32] modelling and verification of closed-loop automation systems using NCES was described. In work [33] the methods to model Petri net in SMV [34] were proposed.. In [35] a new modelling approach was proposed based on abstract state machines (ASM) that allowed application of the most developed SMV model checker. The latter enables modelling and verification of more complex system specifications with arithmetic computations. In this paper, we use the methodology presented in [35].

IV. Robustness Requirements

A. IFM

The following requirements were derived from the actual robustness requirements that were drafted for the IFM. More of the Robustness requirements can be found in [36]

REQ1. The system shall have '*n*' number of isolated fault tolerant zones.

REQ2. Each IFM controller should support a differential protection scheme (eq.1) of the zone.

REQ2.1. If eq(1) does not hold for 10 consecutive readings, fault is triggered.

REQ2.2. If eq(1) holds, then counter (used in REQ2.1 will be reset)

REQ3. Each IFM controller should support a overcurrent protection scheme (eq.2) of the entire system (all the zones, Zone 0 in our nomenclature).

REQ3.1. If eq(2) does not hold for 20 consecutive readings, fault is triggered.

REQ3.2. If eq(2) holds, then counter (used in REQ3.1will be reset).

B. Load Balancing

REQ1. If there is a zone with excess power and a zone in need of power, the load balance must happen.

REQ2. The actual load in the zone must be less than the threshold for a zone with excess load.

REQ3. The actual load must be greater than the threshold for a zone in need of additional load.

C. CTL Properties for model checking the IFM and Load Balancing

Table 1 below summarizes the CTL properties that are checked for the IFM use case and Table 2 summarizes the CTL properties for Load Balancing use case.

Table 1: CTL properties to be satisfied for the IFM case

Tests	CTL Properties
If sum of currents is non zero (± ε), differential_counter is incremented	$AG(sum_k > \varepsilon \rightarrow EX(diff_{count_k} ++))$
If current is greater than a threshold, overcurrent_counter is incremented	$AG(\bigvee_0^i i_k > \omega \rightarrow EX(overcuurent_{count_k} ++))$
If overcurrent_counter reaches n (n = 20 in our example), trigger overcurrent_fault	$AG(overcuurent_{count_k} = 20 \rightarrow EX(OCFault_k))$
If differential_counter reaches n (n = 10 in our example), trigger differential_fault	$AG(diff_{count_k} = 10 \rightarrow EX(DIFFFault_k))$
If either different_fault or overcurrent_fault trigger zone fault	$AG(DIFFFault_k \vee OCFault_k \rightarrow EX(ZoneFault_k))$
Resetting of the counter	$AG\left(sum_k < \varepsilon \rightarrow EX\left(diff_{count_k} = 0\right)\right)$ $AG(\bigvee_0^i i_k < \omega \rightarrow EX(overcuurent_{count_k} ++))$

Where, k is the zone and i is the current in the zone. sum_k refers to the summation used in eq(1). $diff_{count_k}$ is the counter as referred to in REQ2.1. $overcuurent_{count_k}$ is the counter as referred to in REQ3.1. $DIFFFault_k$ is the differential protection scheme fault and $OCFault_k$ is the overcurrent protection scheme fault.

Table 2: CTL properties to be satisfied for the Load Balancing case

Tests	CTL Properties
Invariant holds[37]	$AG\left(\sum_0^k high_k + \sum_0^k low_k + k\gamma = g\right)$
If a node in low state (and there is at least one node in high state), eventually there will be energy migration	$AG(\bigvee_0^k XActual_k < \omega \; AND \bigvee_0^k XActual_k > \omega \rightarrow EF(PStar_k = PStar_k - \delta)$

Where $XActual_k$ is as referred to in eq(3) and $PStar_k$ is the load adjustment as part of the load balancing agreement[21].

Apart from the above CTL properties, the following liveliness properties are also checked.

1. The protection check module is invoked always.
2. The protection check module is terminated always.
3. Once the protection module is invoked, it must be termiated evetually.

V. THE SMV MODEL OF THE IFM CASE STUDY AND RESULTS

A. The SMV Model.

In this section we briefly present the SMV model (important snippets) of the IFM system (we will not present the model for the load balancing) auto-generated using the Function block to SMV code generator developed by us.

The main module declaration of the overcurrent protection unit is given below.

```
MODULE overcurrentUnit(INIT_smv, REQ, INITO, Operate, Counter, variable_, CompareValue_, CounterVal_, OperateVal_, count_, alpha, beta)
```

The main statement that checks for overcurrent is as below

```
next(QO):= case
        S_smv=s2_smv & Q_smv=S_REQ & NA=1 & NI=1 &
        (Current > comparevalue ): TRUE;
        S_smv=s2_smv & Q_smv=S_REQ & NA=1 & NI=1 &
        (Current <= comparevalue ): FALSE;
        TRUE: QO;
esac;
```

The main module declaration for the differential protection is given below

```
MODULE DifferentialUnit4(INIT0, REQ0, INITO, Operate, current1_, current2_, current3_, current4_, CountVal_, HiSet_, OperateVal_, alpha, beta)
```

The main statement that checks for differential protection is given below in two parts, the first calculates the sum of all the currents and second the comparison (same compare value block used for the previous case).

```
next(sumAll):= case
        S_smv=s2_smv & Q_smv=S_REQ & NA=1 & NI=8 :
        (IntCurrent1+IntCurrent2+IntCurrent3);
        TRUE: sumAll;
esac;


next(QO):= case
        S_smv=s2_smv & Q_smv=S_REQ & NA=1 & NI=1 &
        (Current_Sum > comparevalue ): TRUE;
        S_smv=s2_smv & Q_smv=S_REQ & NA=1 & NI=1 &
        (Current_Sum <= comparevalue ): FALSE;
        TRUE: QO;
esac;
```

B. Prelimnary Reults

The SMV modelling language gives us the option to mention the range of values a variable can take (Fig. 3) and we put this to good use. For example, we know that counter value for differential protection cannot be more than '10' according to requirements.

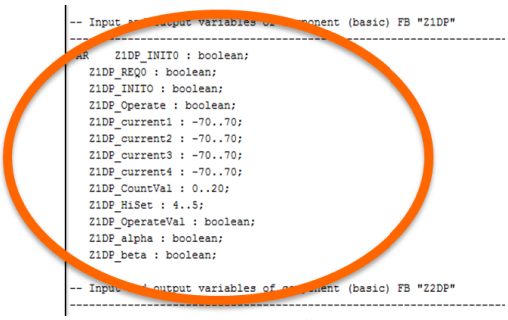

Fig. 3: Ranges in SMV modelling language

Hence, even before the CTL properties can be checked, the NuSMV model checker showed us error saying we were trying to assign value '15' to the count (Fig. 4). It was indeed the case as we forgot to reset the counter value (REQ2.2), once we triggered a fault.

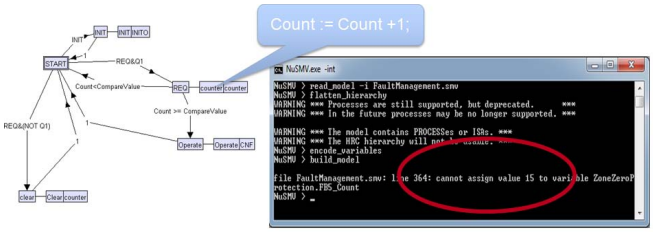

Fig. 4: Invalid counter value

All the liveliness properties were also successfully verified. Some of the other CTL properties were also verified and resulted in no counter examples. However some of the properties could not be verified due to the limitations of the resources (RAM).

VI. Conclusions and Futuire Outlook

In this paper we briefly presented how model-checking can be applied in the context of smart grid automation. We auto generated formal model of two DGI applications in SMV modelling language and model checked with NuSMV [38] model checker. The primary results were great, we were able to liveliness check and also boundary value checks successfully.

The immediate future work is to optimize our SMV model so that we are able to model check all the CTL properties. We have developed a new framework [39] towards optimization using cloud infrastructure and we will be applying it to our current SMV model.

We also did not account for modelling time in the current SMV model and hence we will be incorporating model of time and notion of time in our SMV model. We have already integrated it[40] in our auto generator tool

References

[1] E. A. Lee and S. A. Seshia, *Introduction to Embedded Systems: A Cyber-physical Systems Approach*: Lulu.com, 2011.

[2] C. Ptolemaeus, *System Design, Modeling, and Simulation: Using Ptolemy II*: Ptolemy.org, 2014.

[3] European Commission. (2006). *European Smart Grids Technology Platform—Vision and strategy for Europe's electricity networks of the future. 6.*

[4] 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(1), pp. 133-148, 2011.

[5] N. Higgins, V. Vyatkin, N. K. C. Nair, and K. Schwarz, "Distributed Power System Automation With IEC 61850, IEC 61499, and Intelligent Control," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on,* vol. 41(1), pp. 81-92, 2011.

[6] G. Zhabelova and V. Vyatkin, "Multiagent Smart Grid Automation Architecture Based on IEC 61850/61499 Intelligent Logical Nodes," *Industrial Electronics, IEEE Transactions on,* vol. 59(5), pp. 2351-2362, 2012.

[7] "Function blocks — Part 1: Architecture, IEC Standard 61499-1," Second ed, 2012.

[8] K. Decker, "A Vision for Multi-agent Systems Programming," in *Programming Multi-Agent Systems*. vol. 3067, M. Dastani, J. Dix, and A. El Fallah-Seghrouchni, Eds., ed: Springer Berlin Heidelberg, 2004, pp. 1-17.

[9] G. Black and V. Vyatkin, "Intelligent Component-Based Automation of Baggage Handling Systems With IEC 61499," *IEEE Transactions on Automation Science and Engineering,* vol. 7(2), pp. 337-351, 2010.

[10] *IEC 61850 Communication networks and systems for power utility automation,ed. 2,* , 2009.

[11] G. Zhabelova, C.-W. Yang, S. Patil, C. Pang, J. Yan, A. Shalyto, *et al.*, "Cyber-physical components for heterogeneous modelling, validation and implementation of smart grid intelligence," in *Industrial Informatics (INDIN), 2014 12th IEEE International Conference on*, 2014, pp. 411-417.

[12] V. Vyatkin, "IEC 61499 as Enabler of Distributed and Intelligent Automation: State-of-the-Art Review," *IEEE Transactions on Industrial Informatics,* vol. 7(4), pp. 768-781, 2011.

[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(5), pp. 2351 - 2362 2011.

[14] G. Zhabelova, S. Patil, Y. Chen-Wei, and V. Vyatkin, "Smart Grid applications with IEC 61499 reference architecture," in *Industrial Informatics (INDIN), 2013 11th IEEE International Conference on*, 2013, pp. 458-463.

[15] 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.

[16] E. A. Emerson and E. Clarke, "Characterizing correctness properties of parallel programs using fixpoints," in *Automata, Languages and Programming*. vol. 85, J. de Bakker and J. van Leeuwen, Eds., ed: Springer Berlin Heidelberg, 1980, pp. 169-181.

[17] E. M. Clarke and E. A. Emerson, "Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic," presented at the Logic of Programs, Workshop, 1982.

[18] H.-M. Hanisch, M. Hirsch, D. Missal, S. Preuße, and C. Gerber, "One Decade of IEC 61499 Modeling and Verification-Results and Open Issues," in *13th IFAC Symposium on Information Control Problems in Manufacturing*, V.A. Trapeznikov Institute of Control Sciences, Russia, 2009.

[19] F. Ferri, *Visual languages for interactive computing: definitions and formalizations*: IGI Global, 2008.

[20] G. Zhabelova, V. Vyatkin, and N. C. Nair, "Standard-based engineering and distributed execution framework for intelligent fault management for FREEDM system," in *IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society*, 2011, pp. 2724-2729.

[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] 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.

[23] L. Fix, "Fifteen Years of Formal Property Verification in Intel," in *25 Years of Model Checking*, G. Orna and V. Helmut, Eds., ed: Springer-Verlag, 2008, pp. 139-144.

[24] C. Kern and M. R. Greenstreet, "Formal verification in hardware design: a survey," *ACM Trans. Des. Autom. Electron. Syst.,* vol. 4(2), pp. 123-193, 1999.

[25] V. Vyatkin and H. M. Hanisch, "Formal modeling and verification in the software engineering framework of IEC 61499: a way to self-verifying systems," in *Emerging Technologies and Factory Automation, 2001. Proceedings. 2001 8th IEEE International Conference on*, 2001, pp. 113-118 vol.2.

[26] K. Bae, P. C. Ölveczky, T. H. Feng, E. A. Lee, and S. Tripakis, "Verifying hierarchical Ptolemy II discrete-event models using Real-Time Maude," *Science of Computer Programming,* vol. 77(12), pp. 1235-1271, 10/1/ 2012.

[27] V. Vyatkin and H. M. Hanisch, "A modeling approach for verification of IEC1499 function blocks using net condition/event systems," in *Emerging Technologies and Factory Automation, 1999. Proceedings. ETFA '99. 1999 7th IEEE International Conference on*, 1999, pp. 261-270 vol.1.

[28] V. Dubinin, V. Vyatkin, and H. M. Hanisch, "Modelling and Verification of IEC 61499 Applications using Prolog," in *Emerging Technologies and Factory Automation, 2006. ETFA '06. IEEE Conference on*, 2006, pp. 774-781.

[29] Y. Junbeom, C. Sungdeok, and J. Eunkyung, "A Verification Framework for FBD Based Software in Nuclear Power Plants," in *Software Engineering Conference, 2008. APSEC '08. 15th Asia-Pacific*, 2008, pp. 385-392.

[30] S. Patil, S. Bhadra, and V. Vyatkin, "Closed-loop formal verification framework with non-determinism, configurable by meta-modelling," in *IECON 2011 - 37th Annual Conference on IEEE Industrial Electronics Society*, 2011, pp. 3770-3775.

[31] S. Patil, V. Vyatkin, and M. Sorouri, "Formal verification of Intelligent Mechatronic Systems with decentralized control logic," in *Emerging Technologies & Factory Automation (ETFA), 2012 IEEE 17th Conference on*, 2012, pp. 1-7.

[32] M. Sorouri, S. Patil, and V. Vyatkin, "Distributed control patterns for intelligent mechatronic systems," in *Industrial Informatics (INDIN), 2012 10th IEEE International Conference on*, 2012, pp. 259-264.

[33] G. Wimmel, "A BDD-based Model Checker for the PEP Tool," *Major Individual Project Report, Dept*, 1997.

[34] *Cadence SMV Model Checker*. Available: http://www.kenmcmil.com/smv.html

[35] S. Patil, V. Dubinin, C. Pang, and V. Vyatkin, "Neutralizing Semantic Ambiguities of Function Block Architecture by Modeling with ASM," in *Perspectives of System Informatics*. vol. 8974, A. Voronkov and I. Virbitskaite, Eds., ed: Springer Berlin Heidelberg, 2015, pp. 76-91.

[36] R. Sinha, S. Patil, C. Pang, V. Vyatkin, and B. Dowdeswell, "Requirements Engineering of Industrial Automation Systems - Adapting the CESAR Requirements Meta Model for Safety-Critical Smart Grid Software [Submitted]," in *Industrial Electronics Society, IECON 2015 - 41st Annual Conference of the IEEE*, Yokohama, Japan, 2015.

[37] T. Paul, J. W. Kimball, M. Zawodniok, T. P. Roth, B. McMillin, and S. Chellappan, "Unified Invariants for Cyber-Physical Switched System Stability," *Smart Grid, IEEE Transactions on,* vol. 5(1), pp. 112-120, 2014.

[38] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri*, et al.*, "NuSMV 2: An OpenSource Tool for Symbolic Model Checking," in *Computer Aided Verification*. vol. 2404, E. Brinksma and K. Larsen, Eds., ed: Springer Berlin / Heidelberg, 2002, pp. 241-268.

[39] S. Patil, D. Drozdov, V. Dubinin, and V. Vyatkin, "Cloud-Based Framework for Practical Model-Checking of Industrial Automation Applications," in *Technological Innovation for Cloud-Based Engineering Systems*. vol. 450, L. M. Camarinha-Matos, T. A. Baldissera, G. Di Orio, and F. Marques, Eds., ed: Springer International Publishing, 2015, pp. 73-81.

[40] D. Drozdov, S. Patil, V. Dubinin, and V. Vyatkin, "Model-checking of cyber-physical systems modelled with timed distributed block diagram language [Submitted]," in *Perspectives of System Informatics (PSI'15)*, Kazan, Russia, 2015.