Cyber-physical automation systems modelling with IEC 61499 for their formal verification

Midhun Xavier *, Sandeep Patil*, Valeriy Vyatkin* †
* Department of Computer Science, Computer and Space Engineering, Lulea Tekniska Universitet, Sweden
†Department of Electrical Engineering and Automation, Aalto University, Espoo, Finland

Email: {midhun.xavier, sandeep.patil}@ltu.se, vyatkin@ieee.org

***Abstract*—This paper introduces a problem-oriented notation within the IEC 61499 syntax to be used for creating formal closed-loop models of cyber-physical automation systems. The proposed notation enables creation of a comprehensive tool-chain that can combine design, simulation, formal verification and distributed deployment of automation software. The proposed notation allows for definition of non-deterministic transitions in ECC of basic function blocks of IEC 61499.**

The tool chain includes an IEC 61499 compliant engineering environment, FB2SMV converter of functions blocks to SMV code, the NuSMV model-checker and utilities for interpreting counterexamples.

***Index Terms*—Formal verification, simulation, IEC 61499, cyber-physical automation systems**

I. Introduction

Distributed industrial automation systems pose a significant challenge for their efficient verification and validation due to their heterogeneous structure, use of wireless communication and decentralised logic. The inherent inter twinning of computational and communication processes with complex physical dynamics has called for the term cyber-physical systems (CPS) [1] to emphasize the challenges and the need for new development approaches.

The IEC 61499 architecture [2] is getting increasingly recognised as a powerful mechanism for engineering such systems. It has been proven also as an efficient way of modeling CPS in automation [3].

The challenge of IEC 61499 verification has been well-recognized from the early stages of the standard's development and evaluation [4], [5]. Closed-loop modelling has been proposed for the most comprehensive verification [6], which implies the need for modelling the plant.

In quite many works, the plant modelling [7] was done in the same formalism, which was used eventually to represent the model for the model-checker. Graphical modelling languages of finite-state machines and Petri nets [8], [9] were used in particular, and the models were prepared using the corresponding graphical editors. However, the IEC 61499 itself provides a graphical engineering interface and supports programming in terms of state machines. Therefore, a problem-oriented notation could be proposed to take advantage of the existing tools and avoid using additional ones in the process of modelling. This paper proposes such an approach by introducing a tool chain.

The paper is structured as follows: Section II discusses the related work and problem statement. Section III and IV illustrates an example and simulation model in detail. Section V describes the discrete-state modelling approach including the implementation of non-deterministic transition in smv. Section VI gives an overview of FB2SMV tool's functionalities and features. Section VII presents the results and analysis of the work. Finally, Section VIII concludes the paper and outlines future goals.

II. Related works and problem statement

Christensen suggested a model-driven development approach [10] for distributed automation systems that is based on the use of the model-view-control object-oriented design pattern [11]. The approach supports several development stages, from simulation in the loop to the deployment. In [6] that approach was extended to include also formal verification into the verification and validation of function block systems.

The suggested framework is heavily based on the closed-loop architecture of the model, where the plant part is explicitly represented in the overall system model. This architecture allows for easy integration with a simulation model for the virtual commissioning purposes, which can be seamlessly converted to the deployment configuration. In addition, the closed-loop configuration could be transformed to a structurally similar formal model [12], appropriate for more exhaustive verification by means of model-checking [13], [14]. The concept of an integrated environment VEDA presented in [15] had already supported closed-loop verification of IEC 61499 function block systems. While the controller parts of the closed-loop models were automatically translated into the corresponding formal model in a Petri-net like language, the model of the plant parts had to be developed manually in the same formalism. This made the process of model creation quite difficult, not allowing for systematic use of the tool by control engineers. The emergence of the automatic model generator FB2SMV [16] that is capable of creating SMV models from IEC 61499 function block systems, promises increased potential of formal verification on account of using the industry grade model-checkers of the NuSMV [17] and NuXMV family. However, the problem of creating the model of plant in SMV remains to be the limiting factor for industrial application of the corresponding verification tool-chain.

Fig. 1. Drilling station system.

In this paper, we attempt to overcome this hurdle by proposing a CPS modelling method that is entirely based on IEC 61499. By means of the same modelling language, we represent both simulation models of the plant, which are equivalent to hybrid automata, and the models for formal verification [18], which are equivalent to discrete-state automata with non-determinism. The latter model can be derived from the former by applying a sequence of transformation steps.

Both kinds of plant models, implemented as IEC 61499 function blocks can be included to the multi-closed-loop model connected to the real controllers. This opens the opportunity of checking the distributed control logic of CPS in simulation and model-checking.

III. Illustrative Example

The modelling method and tool-chain used for CPS verification are illustrated in this paper using a laboratory-scale distributed automation system "Drilling station", described in the next subsection. In this case study, we selected and created a formal model for the same system. Implementation of formal models of real systems as well as verification is done with the help of the tool chain.

The drilling station system in Fig. 1 is composed of several mechatronic components, among which, in this study we selected only the Drill and rotating Table. It is assumed that the mechatronic components are smart, i.e. they are equipped with their own control devices, implementing their basic operations, which are as follows.

The Drill moves in upward or downward direction. Whenever a workpiece is detected by the sensor under the drill, it moves downward and starts drilling. Once it completes drilling, it moves upwards and rests at the home position.

The Table rotates from one fixed position to another. The cycle is completed when it rotates six times. When a workpiece is placed in the loading positions, the table rotates to bring it under the drill.

The control logic of each mechatronic component is implemented as a function block which follows the IEC 61499 standard. The function block diagram shown in Figure 2 consists of the two function blocks orchestrated to work together by means of event and data connections between

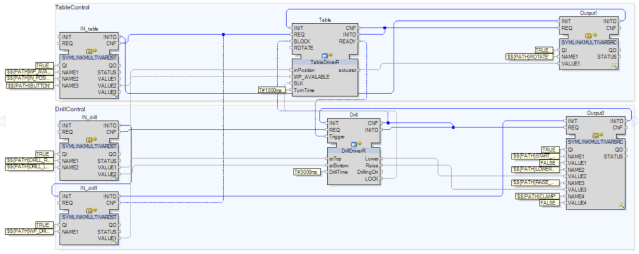

Fig. 2. Function block representation of the distributed automation of drilling station.

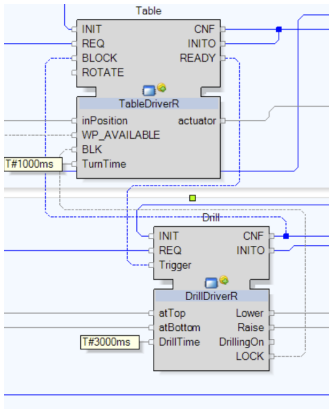

Fig. 3. Close-up on the decentralised controllers' interaction.

each other and sources and sinks of sensor inputs and actuator outputs (function blocks on the left hand side and right hand side respectively). A close-up on the interacting controllers is presented in Figure 3.

It is assumed that the smart mechatronic components are delivered by their vendors together with the software components for implementing their control logic. They are integrated to the drilling station in a way, assuming that the control of internal operations in each mechatronic component is implemented by its predefined function block, and the integrator tries to minimise its software development effort by reusing the software components received from the vendors. This approach requires exhaustive testing of the orchestrated system on compliance with functional and non-functional requirements.

Hence, we will use both simulation in the loop with the plant model and formal verification in closed loop for more exhaustive exploration of the state space.

IV. Simulation Model

The simulation-in-the-loop environment is shown in Fig. 4. The original function blocks containing the autonomous control logic of drill and table are connected with the function blocks implementing simulation models of the drill and table respectively. The controllers are also connected with each other exactly same way as in the real configuration in Fig. 3. For example, the function block TableMod11 of type TableModTop represents a model of the table, that is driven by one control signal `fwd`. When this signal receives the value

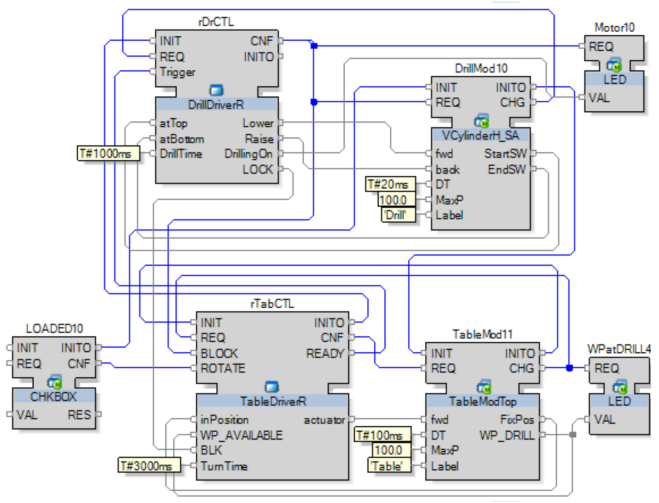

Fig. 4. The Function Block representation of the simulation-in-the-loop configuration.

TRUE, the simulated table starts continuous rotation clockwise. The rotation is stopped if fwd resets to FALSE.

The model outputs the Boolean value FixPos which becomes TRUE when the table comes to one of the six fixed positions. To keep table in the fixed position, the controller has to stop motion by resetting the control signal fwd to FALSE. Besides, the model produces the WP_DRILL signal, which is the reading from sensor indicating the presence of workpiece under the drill.

The simulation environment reproduces the working behavior of the real plant which helps to visually identify the behavior of the system before deploying the distributed control to the real hardware. Besides, the errors identified during the formal verification can be represented in simulation.

The simulation models, used in this configuration, have continuous dynamics, which is the time-domain implementation of hybrid state machine as discussed in [6]. The core part of the plant simulation function block is based on the hybrid automaton model of the process, as illustrated in Fig. 5. The state machine has three states corresponding to the static position of the moving object, such as vertical axis of the drill, or the rotating table. These are stHOME, stEND and stSTOP. There are also two dynamic states, when the coordinate of the object is changing: dMOVETO and dRETURN.

When the state machine is in the one of the dynamic states (say, dMOVETO), it emits the START output event which invokes the external E_CYCLE FB, which starts emitting periodic events, activating the function block with the state machine. The state machine remains in the dMOVETO state until the position reaches the the end position, i.e. Pos=DIST. Until then the loopback transition condition is true, so the state-machine remains in the dMOVETO state. Every time the loopback transition is executed, the event CHG is emitted which also invokes the external "Integrator" FB, which recalculates the new value of the process variable based on the current value, duration of the time interval between recalculations is determined by the DT parameter of the E_CYCLE and speed of the motion.

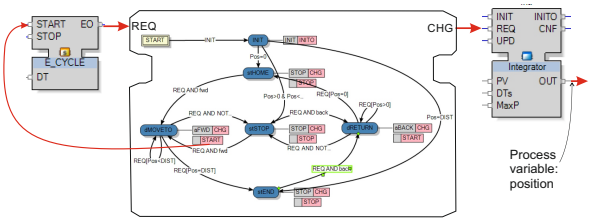

Fig. 5. Computational implementation of a hybrid automaton in function block.

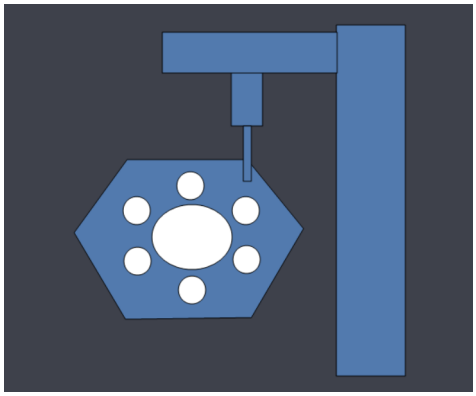

Fig. 6. The visual representation of the simulation process.

Given the ever changing process values, the evolution of the model can be visually displayed using internal or external means. The development, reported in this paper, was done using the NxtStudio of NxtControl, which offers a proprietary visualisation technology called CAT. The plant model blocks were implemented as the CATs, therefore the model behaviour was implemented internally, within the same development environment. The interactive system visualisation by means of CATs is shown in Fig. 6.

V. Discrete-State Modelling Approach

The discrete-state model of the system is created in IEC 61499 based on the simulation model described in the previous section.

The discrete-state equivalent of the simulation configuration is shown in Fig. 7. Here the function blocks simulating the drill and table are substituted by their analogs operating in the discrete state domain, instead of modelling the continuous process parameters, such as the drill's and table's numeric position. The model can be then simulated in the IEC 61499 IDE with values of function block inputs and outputs displayed and modified interactively. It can be translated to the SMV model using FB2SMV tool and exposed to the formal verification by model-checking.

A. Notation for Plant Modelling

In the current example, the drill model is represented by an instance of a basic discrete motion model LinearDA. Its state-machine implementation is shown in Fig. 8. The execution

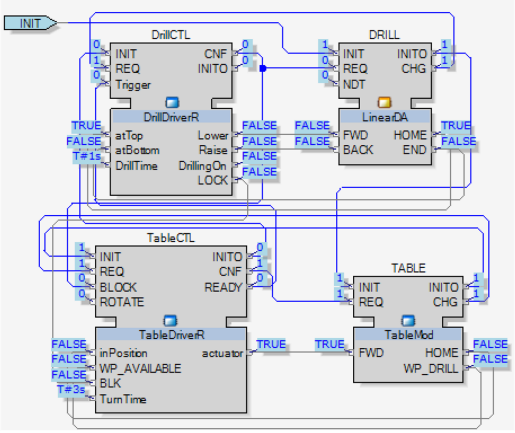

Fig. 7. Discrete state function block model of the Drilling station.

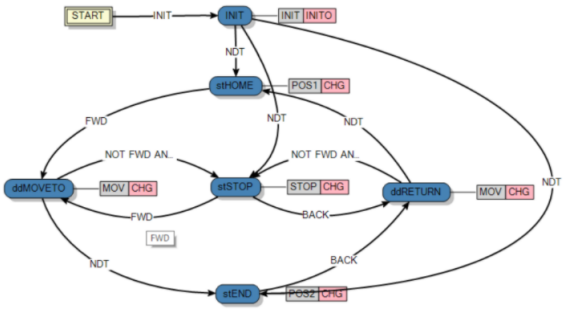

Fig. 8. Discrete state linear motion process model with NDT.

semantics of the state-machine follows the rules of IEC 61499, i.e. the function block is activated by an input event, and the state machine evolution is following the rules for execution control chart (ECC) of basic function blocks.

Similarly to the hybrid state-machine in Fig. 5, the discrete state model of the drill specifies three static and two dynamic states, but it does not model the position as a numeric value. The drill moves from stHOME to stEND state via a motion state called ddMOVETO. The state transition occurs from stHOME to ddMOVETO whenever FWD signal is TRUE. It is remarkable to note the NDT event input of the LinearDA function block, which remained unassigned in the application in Fig.7. The NDT is reserved in the proposed modelling notation for Non-Deterministic Transition. Whenever the formal model generator will encounter NDT in the state machine, it will interpret it accordingly. For example, the SMV modelling of NDT will be described in section V-C

In terms of our model, the use of NDT in the transition from the motion state ddMOVETO to the static state stEND models the unknown duration of the motion from one state to another.

B. Non Deterministic Transitions in controllers

Non-deterministic transition can be also helpful for simplification of controller models containing timers. For example, in our case study, the controllers were developed as state

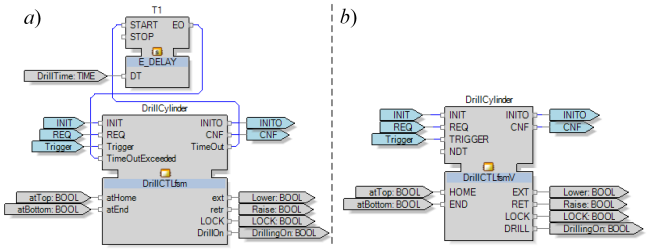

Fig. 9. a) The real drill controller with external timeout. b) The interface of modified drill Controller with non- deterministic transition input.

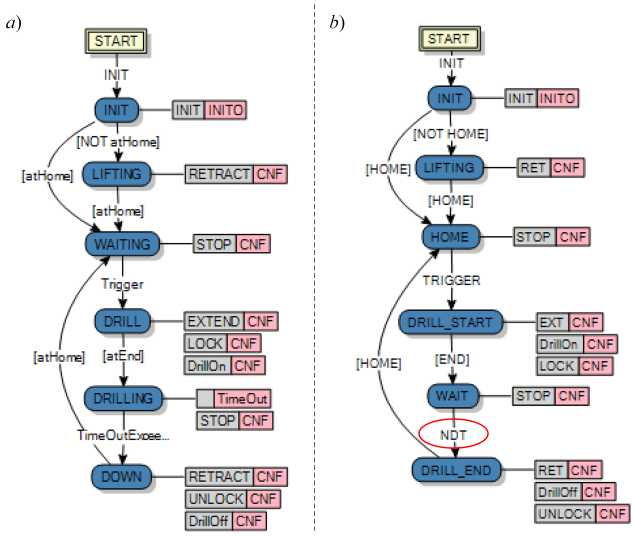

Fig. 10. a) The ECC of the real drill controller; b) The ECC of the modified drill controller with a non-deterministic transition modelling the time delay.

machines with timeouts, therefore they are implemented in composite function blocks. In the drill, drilling process needs to be done for several durations, which is achieved with the help of The E_DELAY function block. The composite function block consists of a real controller and E_DELAY function block as shown in figure 9,a.

However, formal modelling of the timers in SMV is computationally hard. It can be avoided if the concrete delay duration was substituted by non-deterministic transitions with the help of NDT signal. Therefore, the controllers can be modified this way in order to be reduce complexity of model-checking. Therefore, we removed the timeout E_DELAY substituted the corresponding input of the function block and added NDT input. The execution control chart of the real drill controller and the modified drill controller are shown in figure 10.

C. Modelling of non-determinism in SMV

SMV provides a way to accomplish non-deterministic choice by providing a set of values to the signal. The first statement is used to declare the variable NDT as a Boolean type and the second statement is used to initialize the NDT variable to either TRUE or FALSE value.

```
1 | VAR NDT:= boolean;
2 | init(NDT):= { TRUE, FALSE };
```

In every transition we are giving a provision to choose either TRUE or FALSE. This makes the NDT variable unpredictable in each transition.

```
3 | next(NDT):= case
4 |       TRUE : { TRUE, FALSE };
5 |     Esac;
```

Implementing non-determinism in every transition can be limited by introducing conditions in the next statement. If it is not required for the NDT variable to choose values in every transition then we can design like below:

```
6 | next(NDT):= case
7 |       Condition: { TRUE, FALSE };
8 |       TRUE : NDT;
9 | Esac;
```

VI. FB2SMV TOOL

The **fb2smv** tool [16] is a model generator for generating SMV models of function block systems in IEC 61499. It is a part of formal verification tool-chain, that includes the model checker NuSMV and the tool for counterexample analysis in terms of the original FB system.

The tool implements the formal model of IEC 61499 as per the modelling method present in [19]. In order to construct SMV code, the **fb2smv** tool uses Abstract State Machine (ASM) [20] as an intermediate model. The tool takes IEC 61499 function blocks expressed in XML format as input and generates a formal model with the help of ASM semantics. According to [19], the structure of SMV code consists of the declaration part and the rules part, which are the ASM rules.

The tool converts basic and composite function blocks and also includes more additional features like, limiting the boundaries of variables to reduce the state space, changing execution order of FBs, deciding the input event priority by changing its order etc. The proposed non-deterministic transitions notation has been added to the tool as a result of this work.

VII. RESULTS AND ANALYSIS

In this paper, we demonstrated on a case study example, the use of IEC 61499 language for design and formal verification of cyber-physical automation systems. First, we designed the control logic of each mechatronic component in a drilling station and then implemented it as a function block in IEC 61499 standard. The simulation environment of the drilling station is developed with the help of the same controllers which are used in the real configuration. We reproduced the same plant behavior in the simulation model using the function blocks. In order to create a formal model of the system, we transformed existing function blocks of the plant models and controllers by adding non- deterministic transitions, using the proposed NDT notation. Using FB2SMV we converted the functional blocks to SMV code, which was verified using NuSMV on a machine with Intel(R) core(TM) i7-10510U CPU @1.80GHz 2.30GHz with 32GB RAM. The simulate feature of NuSMV was used to check the formal model's working behavior. Simulating the SMV code in interactive mode gives us the provision to go through each state by giving appropriate sensor inputs. We can try all possible input combinations to verify whether the system behaves as we expected. We can also randomly simulate the traces or we can manually go through each state by selecting different input combinations.

The main drawback of simulation is that the number of possible behaviors can be too large or even infinite. Simulation can show the presence of bugs, not their absence.

More comprehensive verification can also be done by model-checking various properties, i.e. checking whether a requirement is true or not in all possible execution traces of control logic. This will allow us to test critical scenarios where there can be failures in some combination of inputs. The formal model of the system was verified with help of CTL [21] specifications. While testing the CTL specifications in NuSMV, we found the following statement is false so it is possible that the table can rotate while the drilling process is going on.

```
-- specification  G !(DRILL_TABLE_CFB3_inst.
   DrillCTL_RET = TRUE &
   DRILL_TABLE_CFB3_inst.ActuatorGen_EO =
   TRUE)
```

While executing the above specification, the NuSmv gave a counterexample that contradicts this statement. The counter-example generation for the above specification took 26000 seconds to complete. The counterexample helps to identify the error in the controller's design. We tested the same logic in the simulation model, the table rotated from its current position to another while drilling the workpiece. The real system also exhibits the same issue.

To fix this issue we need to analyze the counterexample provided by the NuSMV, but identifying the variables changed in each iteration is difficult. The Nutrac tool provides a better way to understand each state. The Nutrac tool converts the counter example to a CSV file. This CSV file's columns represent states and rows represent input/output events or data input/output variables. We analyzed the CSV file and identified the issue. The issue was present in the execution control chart of the table's controller. It is required to add the BLOCK signal and it should be checked before moving from DRILLED state to REMOVED state. We verified the CTL specification again and this time NuSMV gave the TRUE result. The modified controller is tested in the real object, as well as in a simulation system and it was behaving as we expected i.e table was not able to rotate while drilling is going on.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we proposed a tool chain which helps to verify and analyze the function blocks implemented in IEC 61499 standard. This tool chain can be used for continuous development and evaluation of distributed control systems. With the help of this tool chain, it is possible to test the system quickly and efficiently. The accurate implementation formal model is necessary to identify all possible flaws in the system. The existing functionalities of fb2smv along with

the non-deterministic transitions in function blocks help to provide a similar formal model of the system. Previously, the counter-example analysis was complicated but now the Nutrac tool solves this issue by giving a better representation of counter-example in CSV format. The developers working on complex system design can use this tool chain for continuous development and testing.

The non-deterministic transitions in function blocks can be extended by introducing NDT as a variable to the FBs. This NDT variable can be any of any IEC 61499 data type but the NDT variable should be able to choose one value from set values. For example, if we introduce NDT as a variable of integer data type and its values limited 0 to 5 then it should be able to randomly select one value from 0 to 5. In order to introduce more randomness to our formal model, it's better to implement NDT as a variable instead of using NDT as an event signal. An interesting field to explore is if we can generate the specification as well as the plant model automatically then existing manual interventions can be avoided. The tool chain which identifies all possible errors and fixes them automatically could be the next step in the future.

IX. Acknowledgements

This work was sponsored, in part, by the H2020 project 1-SWARM co-funded by the European Commission (grant agreement: 871743).

References

[1] E. A. Lee and S. A. Seshia, *Introduction to embedded systems: A cyber-physical systems approach*. MIT Press, 2017.

[2] "IEC 61499-1: Function Blocks Part 1: Architecture," 2012.

[3] W. Dai, C. Pang, V. Vyatkin, J. H. Christensen, and X. Guan, "Discrete-event-based deterministic execution semantics with timestamps for industrial cyber-physical systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 3, pp. 851–862, 2017.

[4] V. Vyatkin and H.-M. Hanisch, "A modeling approach for verification of IEC 1499 function blocks using net condition/event systems," in *1999 7th IEEE International Conference on Emerging Technologies and Factory Automation. Proceedings ETFA'99 (Cat. No. 99TH8467)*, vol. 1. IEEE, 1999, pp. 261–270.

[5] H.-M. Hanisch, M. Hirsch, D. Missal, S. Preuße, and C. Gerber, "One decade of IEC 61499 modeling and verification-results and open issues," *IFAC Proceedings Volumes*, vol. 42, no. 4, pp. 211–216, 2009.

[6] V. Vyatkin, H.-M. Hanisch, C. Pang, and C.-H. Yang, "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, 2008.

[7] I. Buzhinsky and V. Vyatkin, "Plant model inference for closed-loop verification of control systems: Initial explorations," in *2016 IEEE 14th International Conference on Industrial Informatics (INDIN)*. IEEE, 2016, pp. 736–739.

[8] B. Berthomieu and M. Diaz, "Modeling and verification of time dependent systems using time petri nets," *IEEE transactions on software engineering*, vol. 17, no. 3, p. 259, 1991.

[9] M. Zhou and K. Venkatesh, *Modeling, simulation, and control of flexible manufacturing systems: a Petri net approach*. World Scientific, 1999.

[10] J. H. Christensen, "Design patterns for systems engineering with IEC 61499," *Verteilte Automatisierung-Modelle und Methoden für Entwurf, Verifikation, Engineering und Instrumentierung*, p. 63–71, 2000.

[11] (2008) Model-view-controller design pattern. [Online]. Available: https://folk.universitetetioslo.no/trygver/themes/mvc/mvc-index.html

[12] R. Sinha, S. Patil, L. Gomes, and V. Vyatkin, "A survey of static formal methods for building dependable industrial automation systems," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 7, pp. 3772–3783, 2019.

[13] E. M. Clarke, E. A. Emerson, and A. P. Sistla, "Automatic verification of finite-state concurrent systems using temporal logic specifications," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 8, no. 2, pp. 244–263, 1986.

[14] G. Frey and L. Litz, "Formal methods in plc programming," in *Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics.'cybernetics evolving to systems, humans, organizations, and their complex interactions'(cat. no. 0*, vol. 4. IEEE, 2000, pp. 2431–2436.

[15] V. Vyatkin and H.-M. Hanisch, "Verification of distributed control systems in intelligent manufacturing," *Journal of Intelligent Manufacturing*, vol. 14, no. 1, pp. 123–136, 2003.

[16] fb2smv model generator. [Online]. Available: https://github.com/dmitrydrozdov/fb2smv

[17] A. Cimatti, E. Clarke, F. Giunchiglia, and M. Roveri, "Nusmv: a new symbolic model checker," *International Journal on Software Tools for Technology Transfer*, vol. 2, no. 4, pp. 410–425, 2000.

[18] F. Wang, "Formal verification of timed systems: A survey and perspective," *Proceedings of the IEEE*, vol. 92, no. 8, pp. 1283–1305, 2004.

[19] D. Drozdov, V. Dubinin, S. Patil, and V. Vyatkin, "A formal model of IEC 61499-based industrial automation architecture supporting time-aware computations," *IEEE Open Journal of the Industrial Electronics Society*, vol. 2, pp. 169–183, 2021.

[20] Y. Gurevich and E. Börger, "Evolving algebras 1993: Lipari guide," *Evolving Algebras*, vol. 40, 1995.

[21] E. A. Emerson and J. Y. Halpern, "Decision procedures and expressiveness in the temporal logic of branching time," *Journal of computer and system sciences*, vol. 30, no. 1, pp. 1–24, 1985.