IEEE Open Journal of the
Industrial Electronics Society

Received 5 April 2024; accepted 17 May 2024. Date of publication 5 June 2024;
date of current version 21 June 2024. The review of this article was arranged by Associate Editor Luis Gomes.

Digital Object Identifier 10.1109/OJIES.2024.3406059

A Framework for the Generation of Monitor and Plant Model From Event Logs Using Process Mining for Formal Verification of Event-Driven Systems

MIDHUN XAVIER [1] (Graduate Student Member, IEEE), VICTOR DUBININ [2], SANDEEP PATIL [1] (Member, IEEE), AND VALERIY VYATKIN [1,3] (Fellow, IEEE)

[1]Luleå University of Technology, 97187 Luleå, Sweden
[2]Independent researcher, Russian Federation, Penza 440052, Russia
[3]Aalto University, 02150 Helsinki, Finland

CORRESPONDING AUTHOR: VALERIY VYATKIN (e-mail: Valeriy.Vyatkin@aalto.fi)

This work was supported in part by Horizon Europe Project Zero-SWARM funded by European Commission under Grant 101057083.

ABSTRACT This article proposes a method for the automatic generation of a plant model and monitoring using process mining algorithms based on recorded event logs. The behavioral traces of the system are captured by recording event logs during plant operation in either manual control mode or with an automatic controller. Process discovery algorithms are then applied to extract the logic of the process behavior properties from the recorded event logs. The result is represented as a Petri net, which is used to construct the state machine of the plant model and monitor and is in accordance with the IEC 61499 Standard. The monitor is implemented as a function block and can be deployed in real time to trigger an error signal whenever there is a deviation from the actual process scenario. The plant model and controller are connected in a closed loop and are used for the formal verification of the system with the help of the "fb2smv" converter and symbolic model checking tool NuSMV.

INDEX TERMS IEC 61499, formal verification, plant model generation, process mining.

I. INTRODUCTION

The adoption of the IEC 61499 [1] Standard in Industry 4.0 to model distributed systems signifies a reliable and widely accepted approach [2]. In the IEC 61499 Standard, the distributed control system is expressed in terms of a network of function blocks (FBs) executed in an event-driven way, while individual FBs are implemented as state machines [3], [4], [5] called execution control charts (ECCs).

The design of distributed control for systems is composed of several mechatronic components with complex interactions and is susceptible to design faults in the real system. In order to detect potential errors during run-time, monitors [6], [7] can be used as a monitoring technique. The monitor can observe the process sequence and provide an alert or even stop the process if it deviates from the expected process flow. The deployment of distributed control systems without proper verification can cause significant damage to system components. Simulation [8] is another widely used error-capturing method, this method does not guarantee the absence of errors in the system.

Formal verification [9], [10] can be utilized to identify errors in the designed system by using computation tree logic (CTL) and linear temporal logic (LTL) specifications. The development of a formal model for the system requires the creation of models for both the plant and controller. While creating a formal model for the controller is relatively straightforward, modeling the plant can be complicated and it requires manual development. Formal verification [11] offers a more rigorous approach to verifying the correctness of distributed automation systems. Ramdani et al. [12]

introduced reconfigurable CTL, an extension of classical CTL, aimed at streamlining the formal verification process for complex reconfigurable discrete-event systems by reducing the number of properties needing verification through the classification and prioritization of properties based on dominance and equivalence relations. By computationally exploring a broad range of error-causing scenarios within the system, formal verification techniques can provide a more comprehensive analysis of potential errors than simulation-based methods.

The manual implementation of plant models [8], [13] is complex and time-consuming; hence, the automatic generation of plant models [14], [15] helps in many cases. The plant model can be implemented by obtaining system information through the recording of event sequences that occur within the system.

This study focuses on generating a monitor and a plant model by leveraging a process mining methodology on recorded event logs. While the monitor can be represented in different process model paradigms for the purpose of conformance checking, this study focuses on developing a monitor based on the IEC 61499 FB. Subsequently, this monitor is deployed to evaluate the conformance.

The study addresses the challenge of classic process mining, which typically constructs a model of the entire system. In our approach, the focus is on developing a model specifically for the uncontrolled behavior of the plant, with the intention of later integrating it with the control model to create a closed-loop system model. This approach aligns with the requirements of discrete event system verification.

The article is structured as follows: Section II provides an in-depth literature review that compares the proposed approach with existing methods found in prior research. Section III provides an overview of the foundational concepts of the reference model of control/sensor events sequencing and outlines the process of decomposing Petri nets into state machine components (SMCs). The workflow and potential use cases are detailed in Section IV. Section V describes the process of finite state machine (FSM) generation from the event log, including the implementation of both the monitor and the plant model within IEC 61499 FBs. Section VI offers a case study focusing on a pneumatic cylinder, presenting comprehensive results and analysis. Finally, Section IX serves as the conclusion, summarizing the article's findings and delineating future objectives.

II. RELATED WORK

IEC 61499 [1] is an international standard for the design and implementation of industrial automation systems (IASs), particularly in the realm of distributed control systems. Christensen et al. [16] introduced a model-driven approach (MDA) for distributed control systems within the context of IEC 61499 systems, providing a framework for modeling, designing, and deploying distributed control systems using a modular and event-driven approach. IEC 61499 is employed in industrial applications due to its enhanced modularity [17], control logic reusability [18], flexibility in complex processes [19], and improved interoperability among automation components [20].

The paper [21] describes how IEC 61499 standard itself ensures safety, but it is necessary to verify the system before its deployment to guarantee its reliability. Hegny et al. [8] presented IEC 61499 based simulation framework for model-driven production systems development. While simulation is valuable for identifying bugs and assessing system behavior, it alone does not guarantee error-free operation. In [10], the comparison of formal verification approaches for IEC 61499 to enhance the safety assurance is explained. Formal analysis of IEC 61499 applications [11], [22] involves mathematically proving that the control system meets specified safety properties and requirements. By applying formal methods [23], [24], potential hazards and errors can be rigorously identified and corrected.

In [25] and [26], systematic closed-loop modeling in IEC 61499 FBs and its formal verification is presented. Closed-loop formal verification in IEC 61499, where the plant model and controller are connected in a closed-loop, is a better approach [27], [28] as it allows for the verification of system properties under realistic operating conditions, ensuring that the control system behaves correctly and safely in response to dynamic and unpredictable environmental changes [29].

In [13], the plant model is presented not only for simulation but also serves the purpose of formal verification. Malik et al. [30] presented an efficient approach wherein the plant model is constructed within an IEC 61499 FB, simplifying the integration of the controller and plant within a closed-loop system. The introduction of the fb2SMV tool [31] facilitates the conversion of FB code into SMV format, enabling the verification of CTL or LTL specifications through NuSMV model checker [32], thus streamlining the formal verification process. In [33], a plant model in IEC 61499 FB with nondeterministic transitions (NDTs) is introduced for formal verification. While this method provides a more realistic means of implementing the plant model, it does entail manual work that is susceptible to errors and can be time-intensive. In our previous study [14], we proposed a semiautomatic plant model generation in SMV, but it posed challenges in integrating with the SMV model of the controller for closed-loop formal verification.

The research gap centers on the automatic generation of a plant model within IEC 61499 FBs to facilitate formal verification, and this article aims to fill this void. In this study, we employ recorded events and leverage a process mining approach to develop a plant model. While process mining typically yields an overview of the entire system's behavior, this article specifically details the extraction of the plant model using IEC 61499 FBs from this broader system behavior.

Process mining is a data-driven technique used to discover, monitor, and improve real-world processes by analyzing event logs and extracting valuable insights about how these processes are executed [34], [35]. Beyond the realm of business process modeling, such methods are also applied in anomaly detection, cyber-attack detection, and alarm analysis within industrial control systems, utilizing a range of control flow

discovery algorithms [36]. By analyzing event logs from various components and sensors, it can provide a comprehensive view of the system's operational behavior, enabling the identification of bottlenecks, deviations, and inefficiencies. This study applies similar principles to derive the process sequence of a controlled system, utilizing a mining algorithm for process discovery. ProM [37] and DISCO [38] are the widely used process mining tools for the preparation of event logs, process discovery, visualization, and conformance checking. ProM [39] which is used in this study, is an open-source tool, and comprises several process discovery algorithms and plugins for conformance checking and visualization. The **alpha** algorithm [40] is selected as the process discovery algorithm in this study due to its simplicity and transparency. Other process mining algorithms, such as the Heuristics Miner, Inductive Miner, or Genetic Miner are suggested when more complex or detailed insights are required [41], [42]. In recent research, process mining algorithms have shown the capability to handle multiple-concurrency short-loop structures effectively [43], [44]. These algorithms can handle larger and more intricate event logs, making them suitable for in-depth analysis, optimization, and automation of processes. The selection of the algorithm should align with the objectives of the level of complexity in the process of being analyzed [36].

The study provides a comprehensive understanding of the system's behavior through event logs is available. This sets the stage for performing conformance checking based on the model derived from these event logs [45], [46]. This article introduces a method for the automated generation of a monitor in IEC 61499 FBs, facilitating real-time deployment for monitoring purposes. This IEC 61499 monitor FB analyzes all events to detect any deviations from the expected flow. The combination of this monitor in the form of an IEC 61499 FB and the plant model represented similarly in IEC 61499 FBs becomes particularly advantageous when introducing new controllers or migrating to IEC 61499 FBs. The newly introduced controller can seamlessly connect in a closed-loop configuration with the plant model, and the developed IEC 61499 monitor FB allows us to monitor system behavior. Any deviation in events leading to an error state triggers the monitor, ensuring that the newly developed controller performs in accordance with the previous controller. The option to create a formal model of the system and perform formal verification adds an extra layer of safety assurance before deployment into the operational system.

III. BACKGROUND

A. RMCSES CONCEPT

The Reference Model of Control/Sensor Events Sequencing (RMCSES) is a formal model that developed from the process mining [47] of event logs describing the functioning of a closed-loop IAS in an "error-free" mode over an extended period. RMCSES offers a condensed representation of a vast event log, encompassing all possible signals from all components. Conceptually, if one imagines the event log as a collection of sentences or phrases of a formal language, describing the behavior of the IAS [27], then RMCSES functions as a sentence generator for this language. Due to its conciseness, RMCSES can be readily implemented in software or hardware.

The interface of the RMCSES is shown in Fig. 1(d). The RMCSES only takes into account control signals originating from the controller and informative signals originating from sensors as events. These signals represent the interface between the controller and the plant. Internal signals circulating within the control system and the plant are not considered, but it is possible to include control signals from external sources such as an operator.

In an IAS operating in an "error-free" mode, the functioning of each component is characterized by cyclical, meaningful, and locally complete processes. The insignificant operation of components, such as when an ejector piston moves back and forth indefinitely, is not considered. The component is identified to have meaningful behavior when it has a specific goal and actively works toward achieving it. Each component or device follows a specific scenario that begins with its initialization and ends with its termination. A multifunctional device may execute different scenarios depending on the specific task or operation it is performing.

Fig. 1(b) illustrates the cyclical operation of a component (device) and the possible cutoff points of the event log. As the functioning of a component (device) is cyclical, meaningful, and locally complete, the event log represents only a part of the process history of this component (device), leading to incomplete operation traces in the log that will be mapped onto the formal model. Therefore, a question arises as to how these incomplete traces, or "scraps," affect the extraction of an event log. Our experimental results indicate that the addition of such scraps to the event log does not result in significant changes.

Given the convenience of implementing the RMCSES as an IEC 61499 FB, we have opted to employ it as an FSM in our research; we do not preclude the use of alternative models. The RMCSES will constitute a comprehensive representation of the lower level operation of the system; for intricate IAS, this model may become unwieldy.

Petri nets [48] are frequently employed as a formalism for system modeling in the domain of process mining, as they offer greater expressive power in comparison to finite-automata models. The Petri nets are subsequently decomposed into SMCs and then composed to create an FSM through the application of specific refactoring rules. This FSM can be readily mapped onto an ECC diagram of a basic FB, with a near "one-to-one" correspondence. The primary goal of this refactoring procedure is to eliminate constructs such as "Fork into parallel branches" and "Join parallel branches." It should be noted that such transformations may not always be feasible.

B. OVERVIEW OF PETRI NET DECOMPOSITION TECHNIQUES

Petri nets are essential for modeling concurrent systems [49], but can become complex as systems grow in scale. To manage this complexity, various decomposition techniques have

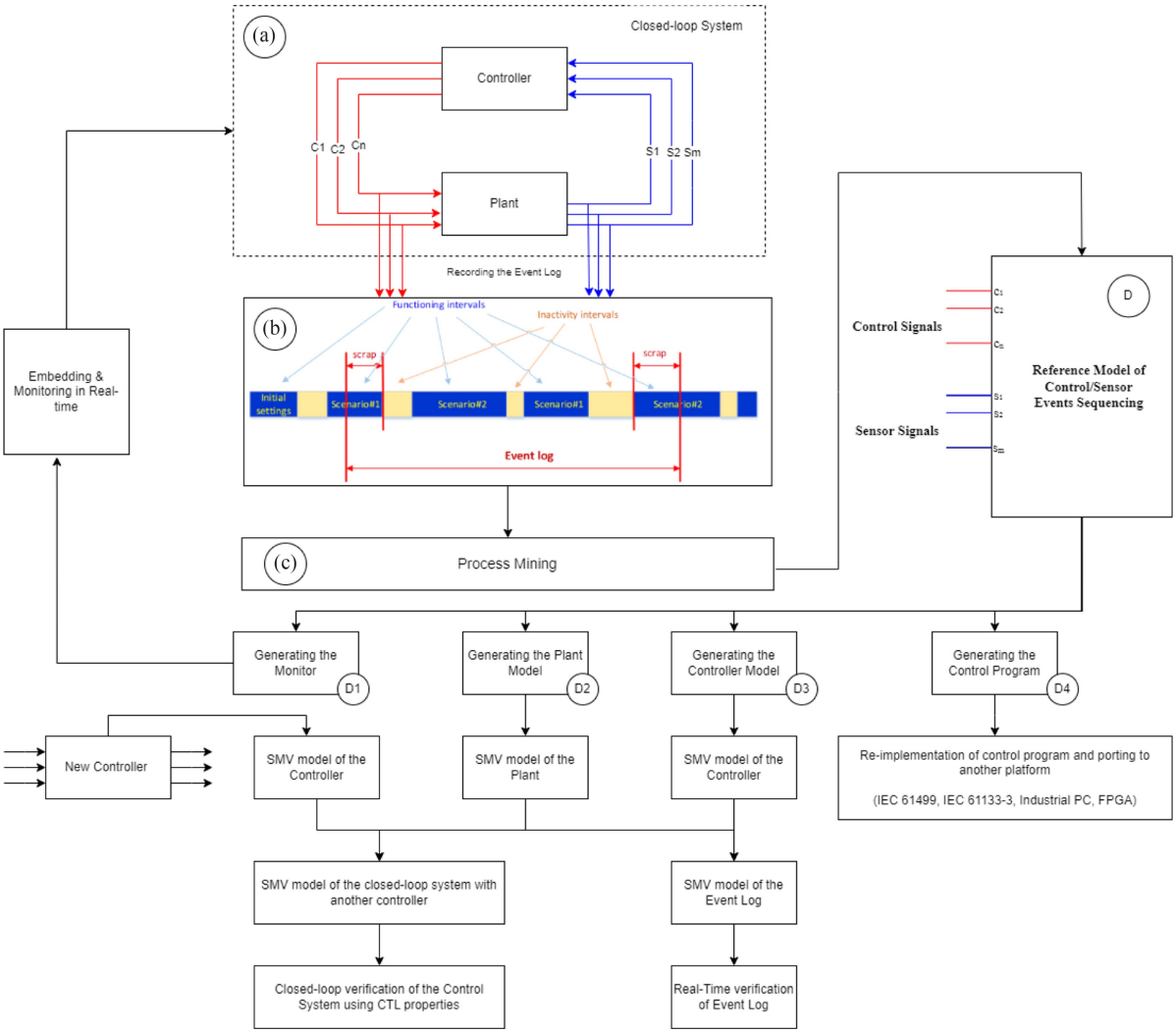

FIGURE 1. **Workflow and use cases.**

been developed. These methods, including the reachability graph method, structural decomposition methods, and linear algebra-based methods, aim to break down Petri nets into more understandable components while preserving their essential behavioral characteristics. The reachability graph method [50] involves analyzing the reachability graph to identify subsets of states and transitions, facilitating the partitioning of the Petri net into manageable modules. Structural decomposition methods [51], [52] leverage properties like perfect graph theory to systematically break down complex nets into simpler subnets. Linear algebra-based methods [53], [54] use matrix manipulation to extract sequential components from place invariants, aiding in the identification of independent modules within the Petri net. These techniques offer distinct advantages in analyzing, verifying, and understanding concurrent systems modeled with Petri nets.

In this article, the focus lies on accurately modeling a plant using IEC 61499 FBs to ensure the comprehensive capture of all its behaviors, vital for the proper functioning of the closed-loop system with a controller. To achieve this, the reachability graph [55], [56] method is employed as a decomposition technique to derive SMCs from Petri nets. This method offers a structured and systematic approach to exploring all reachable states and transitions within the system, guaranteeing completeness in capturing all potential behaviors. By exhaustively mapping out all possible states and transitions, it ensures that no aspects are overlooked during analysis, providing a thorough understanding of the system's behavior dynamics and potential outcomes. It is susceptible to the state explosion problem [50], [57], particularly in larger and more complex Petri nets, leading to computational inefficiency and difficulty in analysis. In contrast, structural decomposition methods leverage the Petri net's structural properties [58] to identify cohesive subsets that can be modeled as individual SMCs, providing flexibility in decomposition criteria. While these methods can offer more manageable representations and may be less prone to state explosion, their effectiveness heavily depends on the chosen

structural properties and decomposition criteria, and may require domain-specific knowledge for optimal results.

This article discusses on the complete automatic generation of a plant model in IEC 61499 FB, identifying structural properties poses a challenge and necessitates domain-specific knowledge. Structural decomposition methods may offer a different approach by breaking down the system into smaller components based on structural properties [59]. This approach may overlook certain behaviors arising from interactions between components, unlike the comprehensive exploration provided by reachability graphs. Reachability graphs provide a fine-grained analysis of the system's behavior, capturing individual states and transitions, which may be lacking in granularity in structural decomposition methods. Linear algebra-based methods [60] provide a systematic approach to extracting sequential components from place invariants, offering insights into the structural properties of the Petri net. They may have limited applicability to Petri nets with complex or nonlinear dynamics and may be computationally intensive for large systems, requiring expertise in linear algebra [53], [61] for implementation and analysis. Overall, while the reachability graph method ensures completeness but suffers from state explosion, structural decomposition methods offer flexibility but depend on chosen criteria, and linear algebra-based methods provide insights but may have limited applicability and computational complexity, highlighting the tradeoffs between completeness, flexibility, and computational efficiency in decomposing Petri nets into SMCs.

Considering the tradeoffs among different decomposition techniques, the need for completeness in capturing all system behaviors to ensure successful formal verification of closed-loop system properties is paramount. The challenge of state space explosion looms large, particularly in the reachability graph method [62], [63]. To address this, several strategies can be implemented [64]. State space reduction techniques involve the adoption of symbolic representations such as binary decision diagrams (BDDs) or decision diagrams to compactly represent sets of states [65], thereby reducing memory requirements and mitigating state space explosion. A recent study [66] presents an approach utilizing reduced ordered binary decision diagrams to address state explosion issues in verifying privacy properties of multiagent systems modeled with knowledge-oriented Petri nets and CTL of knowledge, significantly reducing verification time even for large-scale systems such as the dining cryptographers protocol. Abstraction and Aggregation methods [67] selectively abstract or aggregate unnecessary details in the reachability graph, focusing on essential behavioral properties while discarding less critical information, thus reducing the size of the state space and enhancing efficiency. Partial order reduction techniques [68], [69] aim to diminish redundant interleavings explored during state space traversal, avoiding the exploration of equivalent states and further reducing the size of the state space. clarke1999state, a Hybrid Exploration Strategy is suggested. It involves incremental construction, where the reachability graph is dynamically constructed as needed during analysis, conserving memory and computational resources. On-the-fly exploration is employed to explore states as needed rather than generating all states upfront, efficiently managing state space explosion. Structural decomposition techniques are integrated into the hybrid method, wherein the Petri net is decomposed into smaller subsystems, and reachability analysis is performed on each subsystem independently. This approach reduces the overall size of the state space that needs to be explored and analyzed, further mitigating state space explosion while ensuring comprehensive system behavior coverage.

Combining SMCs into a cohesive FSM necessitates careful integration to ensure synchronization and coordination between them. Several methods facilitate this process: Sequential composition [70], where the output of one SMC becomes the input of another, suitable for systems with well-defined sequential behavior; parallel composition, involving concurrent execution of SMCs and combining their states and transitions to represent parallel behavior [71]; synchronization and handshaking techniques [72] ensuring coordination between SMCs through protocols and clock domain crossing mechanisms; global state encoding [73], where individual SMC states are merged into a unified state space for comprehensive system representation; state merging, identifying and collapsing equivalent states to manage complexity [74]; and interface design and refinement, establishing clear interfaces and refining interactions for modularity and maintainability. These methods offer practitioners a toolkit to effectively integrate SMCs, with choices guided by system characteristics such as concurrency and synchronization needs, enabling the creation of a coherent FSM reflecting the overall system behavior.

For developing an FSM, a combination of parallel composition and synchronization techniques is preferred. Given the distributed nature and modularity requirements of IEC 61499 systems, parallel composition allows individual SMCs to represent concurrent activities effectively, aligning with the system's modular architecture. This approach facilitates easier maintenance, debugging, and resource utilization optimization. Synchronization techniques ensure proper coordination and communication between distributed SMCs, meeting real-time constraints and enhancing system correctness and reliability. By leveraging parallel composition and synchronization, developers can seamlessly integrate SMCs into a coherent FSM, enabling the development of robust and efficient ECC in IEC 61499 FB environments.

IV. WORKFLOW

This section outlines the workflow and potential use cases enabled by RMCSES, as illustrated in Fig. 1. Initially, signals from the distributed control system are recorded to construct RMCSES using process mining techniques. RMCSES then serves as the basis for generating the monitor (D1), plant model (D2), control model (D3), and control program (D4). In our prior research, we presented (D3) an interactive learning approach for deriving controller logic following the IEC

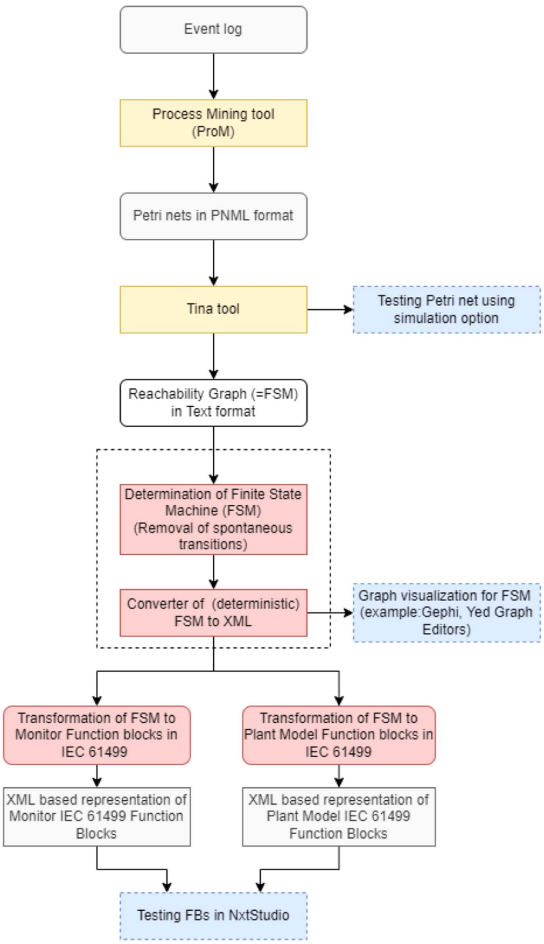

FIGURE 2. Tool chain and data flow for generating FSMs from event logs and their implementation in the form of IEC 61499 FBs.

61499 standard [75]. Our previous study detailed (D2) the methodology and implementation of a basic plant model (a horizontal cylinder) [15]. In this study, we enhance the methodology by incorporating global state information to create a precise plant model, subsequently closing the loop with a controller for formal verification using a toolchain [33]. Developing the monitor in IEC 61499 FBs is instrumental for real-time error identification and prediction within the controller. In this article, we develop such a monitor from event logs for conformance checking. Finally, future work is planned to reimplement control and facilitate platform migration by deriving the control program from RMCSES.

V. FSM FROM EVENT LOGS AND THEIR IMPLEMENTATION IN IEC 61499 FBS

This section describes the steps involved in generating IEC 61499 FBs from the recorded event log.

A. PETRI NET CONSTRUCTION FROM EVENT LOGS

The process of generating FSM from event logs and their implementation in the IEC 61499 FB is shown in Fig. 2. To start the process, the event log is given as input to the process mining tool called ProM [39], which is an open-source tool that offers various features such as representing process logic in different models, event log preprocessing, format conversion, conformance checking, LTL specification testing, and visualizations. The input event log is provided to ProM in CSV format, which then preprocesses the data and converts it into eXtensible Event Stream (XES) format. ProM then applies a process mining algorithm to the event log in XES format to extract processes in the form of Petri nets.

B. FSM GENERATION FROM PETRI NET

1) FROM PETRI NETS TO SMCS: UTILIZING REACHABILITY GRAPHS FOR DECOMPOSITION

To depict a Petri net within the ECC of IEC 61499 FBs, it is crucial to first decompose the Petri net into SMCss. Subsequently, these SMCs can be composed into a FSM. The extracted Petri net can be exported in the Petri net Markup Language (PNML) format using the ProM tool. The PNML-formatted Petri net is then given as input to the TINA tool [76], which constructs a RG and can perform Petri net simulation. The finite RG is decomposed to SMC with spontaneous transitions as follows.

Consider a running example "Repairing telephones" for the implementation of the IEC 61499 FB of the FSM for conformance checking. The description of the "Repairing telephones" event log is given in ProM 6 tutorial, 2010 [77]. The generated Petri net from the event log using an inductive miner is shown in Fig. 3(a). A supplementary transition is added to the Petri net to make the process cyclic. In Fig. 3(a), the Petri net is under stepwise simulation, and the "Repeat" transition is added to make this process cyclic. Graphical representation of the RG is drawn manually using the RG text from TINA is outlined in Fig. 3(b). The SMC is derived from the RG, and the transitions in the SMC are designated with the same names as the corresponding transitions in the Petri net. The "spontaneous" transitions are marked in blue color.

There appears to be a discrepancy regarding the potentially infinite nature of constructing a reachability set for a Petri net, which, in theory, could be infinite. It is essential to clarify that this set should not be infinite, as the event log, by its very nature, is finite. From a finite event log, a Petri net should indeed yield a finite number of reachable markings. The count of these markings should not exceed the number of entries in the event log, serving as an upper bound. This is grounded in the event log's representation as a very large RG with linear topology. Each event (entry) in the log distinctly defines a transition from one state (marking) to another. Modern tools for Petri net analysis can construct high-dimensional RGs, like those achieved through the BDD method. The challenge lies not in constructing the RG but rather in its implementation using IEC 61499 FBs. Nevertheless, even in cases where the RG comprises several thousand states, this challenge could be surmountable with the availability of appropriate CAD tools. It is worth noting that reduction methods for Petri nets

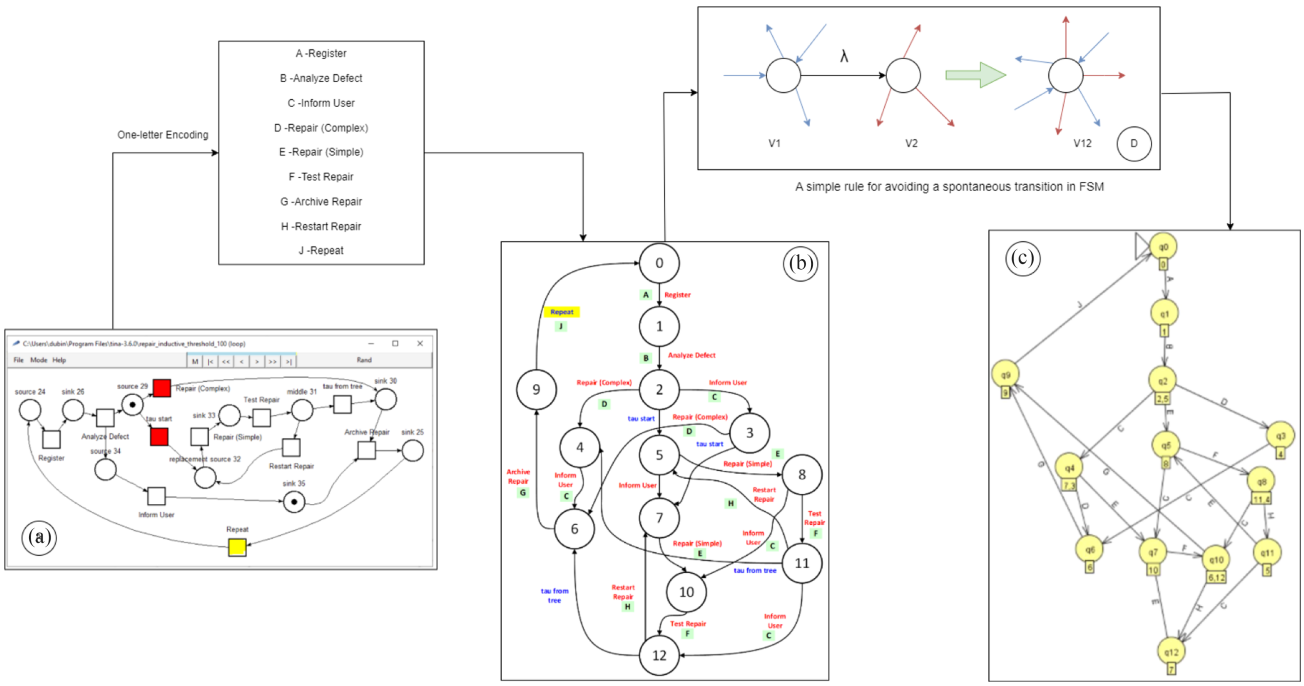

FIGURE 3. Transformation of Petri net to FSM for monitor implementation.

may offer a viable strategy to significantly reduce the size of the RG, further enhancing the manageability of this complex process.

2) TACKLING STATE SPACE EXPLOSION: EFFECTIVE STRATEGIES FOR PETRI NET ANALYSIS

Addressing the state space explosion issue in the reachability graph method for Petri nets involves employing various strategies to manage the exponential growth of states and transitions. Techniques such as abstraction and aggregation identify high-level patterns and structures for simplification, while partial order reduction reduces redundancy by exploring subsets of transitions together. Symmetry reduction exploits symmetrical properties to collapse equivalent states and transitions, effectively pruning redundant branches. On-the-fly exploration dynamically generates states and transitions only when needed, reducing memory consumption. State compression techniques compress state representations for efficient storage, and parallel and distributed exploration leverage parallelism to speed up analysis. By utilizing these methods, practitioners can navigate the complexities of larger Petri nets while efficiently managing computational resources, with the choice of approach guided by specific system characteristics and resource constraints.

3) COMPOSING FSM FROM SMCS

Developing a FSM from SMCs demands meticulous synchronization of decomposed components, a challenging task influenced by the system's clock domains. Within the framework of IEC 61499 FB, creating an ECC requires selecting an appropriate method for merging SMCs into an FSM, tailored to the specific requirements and characteristics of the distributed control system. The combined utilization of parallel composition and synchronization techniques emerges as the favored approach, offering concurrency management, modularity preservation, resource optimization, adherence to real-time constraints, and reinforcement of system correctness and reliability. These methodologies seamlessly align with the distributed and modular architecture of IEC 61499 systems, making them well-suited for consolidating SMCs into a coherent FSM.

To implement the strategy of parallel composition and synchronization techniques, the initial step involves identifying individual SMCs representing distinct functionalities within the distributed control system. Clear interfaces must be defined to delineate inputs, outputs, events, and communication protocols, facilitating seamless coordination among SMCs. Each SMC is then implemented as a separate module, enabling concurrent execution to handle various tasks efficiently. Event-driven communication serves as a pivotal synchronization mechanism, establishing communication channels between SMCs to facilitate information exchange and event triggering. Integration of SMCs into a unified system requires ensuring accurate interaction and synchronization based on defined interfaces and communication channels. Rigorous testing is essential to validate correctness, reliability, and real-time performance, with iterative refinement and optimization of the FSM to meet desired behavior and performance criteria. This approach, aligned with the distributed and modular nature of IEC 61499 systems, fosters the development of robust and efficient distributed control systems.

C. DETERMINIZATION OF FSM WITH SPONTANEOUS TRANSITIONS

Nondeterminism may occur in FSMs when employing "spontaneous" transitions within Petri nets. This means that the transitions of the Petri net are not related to any event. In the ProM system, they are indicated by black rectangles in Petri nets. The rule of mapping the transitions is as follows: A transition in an FSM will be spontaneous if it corresponds to a "spontaneous" transition in the Petri net. In FSM, spontaneous transitions are denoted by the symbol λ ("lambda") or ϵ ("epsilon").

The process of determinization can significantly simplify the implementation of a FSM in software. In this case, this process consists of getting rid of λ-arcs representing spontaneous transitions.

In this study, we use two approaches to the determinization of FSM that contain spontaneous transitions. The essence of the first approach is as follows: two vertices (states) connected by a λ-arc are combined into one vertex. In this case, the incoming and outgoing arcs of both vertices are combined, and the λ-arc is removed [see Fig. 3(d)]. This rule is applied until there are no λ-arcs left in the transformed graph. This method is not universal and it is applicable only in the case of a tree-like topology of λ-arc relationships. In the second (universal) method, two vertices are contracted into one vertex if one vertex is reachable from another vertex through a chain of λ-arcs.

Determinization of an FSM can be omitted when there are no states with two or more spontaneous transitions outgoing from a state of the FSM. If there are no such situations, then a spontaneous transition can be interpreted in the ECC as a transition with an always true condition "1."

D. GENERATION OF DETERMINISTIC FSM

The JFLAP tool [78] is used to implement the determinization of FSM. In this process, only one-letter designations of input symbols (signals) can be used. To encode the long names of input events used in the Petri net for the given example, we use a one-letter encoding (see Fig. 3).

The nondeterministic FSM is shown in Fig. 3(b), and spontaneous transitions are labeled with the symbol λ ("lambda"). The transition labeled as J (i.e., "Repeat") from q9 to q0 will not take part in the determinization process. Instead, it will be replaced with the symbol "1" in the corresponding ECC. After determinization, the deterministic FSM is obtained as shown in Fig. 3(c). GraphML format [79] was used to visualize the reachability graph with the help of visualization tools like Gephi [80], Yed [81], etc.

We employ a process discovery algorithm to generate a Petri net from the event log. Subsequently, we utilize the TINA tool to convert this Petri net into SMCs. These SMCs are further transformed into a deterministic FSM with the assistance of the JFLAP tool. These tools have gained widespread acceptance and are known for their accuracy and reliability in performing these conversions. By following this step-by-step approach and analyzing the model at each stage, we ensure the correctness of the transformation process. Attempting a direct conversion could be error-prone, making these integrated tools the preferred choice to prevent inaccuracies and streamline the workflow.

E. TRANSFORMATION OF FSM TO IEC 61499 FBS FOR MONITOR IMPLEMENTATION

The structure of the application for conformance checking is depicted in Fig. 4. The application is based on a closed-loop system that includes a plant and a controller. The RMCSES obtained from the event log is connected to the closed-loop system for the purpose of conformance checking. The RMCSES is implemented as a FSM that uses input data to drive its logic. The FSM is capable of detecting errors and verifying the correct sequence of steps in the process flow by monitoring events from the sensor and actuator in real time.

Fig. 4 depicts the conversion of the RMCSES into the monitor. The monitor interface FB receives input signals in the form of controller signals $c_1 \dots c_n$ and sensor signals $s_1 \dots s_m$. The ECC is generated through the utilization of the transformation rule depicted in Fig. 4. An algorithm, based on this transformation rule and facilitating the conversion of FSM to ECC with error handling, is presented in Algorithm 1. When the monitor FB detects a valid transition from state q_i to q_j triggered by event X_k, it generates an output indicating a successful transition, denoted as "OK." If an unexpected event occurs, it transitions to the ERROR state, producing an ERROR event and providing details about the event that caused the error (EventID) and the StateID, which identifies the state where the error took place. This transformation rule ensures that the ECC reflects the behavior of the FSM and is equipped to handle errors.

The Monitor FB shown in Fig. 4 is a direct implementation of the FSM, representing the formal reference model. The FSM is mapped to the ECC practically "one-to-one" and the FB interface is formed as follows. Each event is assigned its own event input. The INIT input signal is intended to set the model to the initial state. The signal from the event output "OK" is issued when the received input event is in conformance with the formal reference model. In this case, the number (id) of the ECC state is issued, to which the transition took place (the output variable StateID). When an error is detected, not only the state number (StateID) where the error occurred but also the number of the input event (EventID) that led to the error is issued. Thus, the error is captured. The FB works until the first error appears, after that the FB is blocked, i.e., it becomes not responsive to any inputs. The ECC model is complete in the sense that from each basic ECC state, there are outgoing arcs labeled by each of the input events. Input events that are not specified for a state of the formal reference model transition the ECC to one of the error states. The ECC representation for the running example "Repairing telephones" is shown in Fig. 5(b). For the sake of simplicity transitions to error, states are shown for only one test state q11.

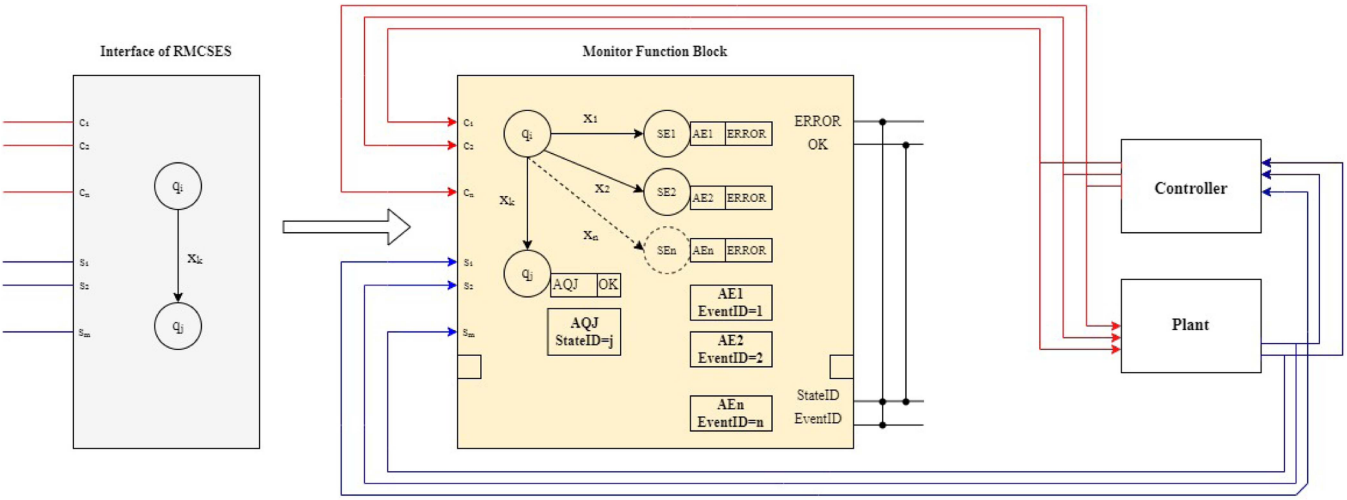

FIGURE 4. Application for conformance checking.

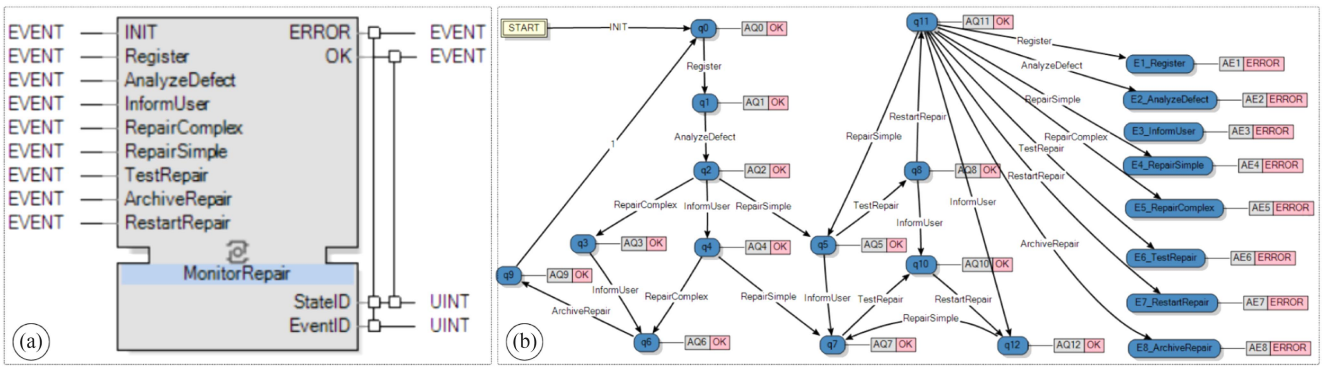

FIGURE 5. FB interface and ECC of telephone repairing monitor.

It should be noted that the implementation of the monitor may differ in different tools. For example, in FBDK [82] it is simpler than in NxtStudio. This is due to the difference in execution models of basic FBs in these tools. While in FBDK the input event is immediately cleared after the activation (firing) of the ECC transition, in NxtStudio it is not cleared during the entire execution time of the FB (run-to-complete). As a result, during the execution of the FB, several ECC transitions marked with the same input event can be triggered.

F. EXTRACTION OF PLANT MODEL FROM THE OVERALL SYSTEM MODEL

The FSM derived from the event log serves as a representation of the overall system behavior. In this study, an innovative approach for extracting the model of the uncontrolled plant is presented. The system is viewed as a closed-loop configuration, encompassing both the plant and the controller. The objective, therefore, centers on obtaining the model of uncontrolled plant behavior, which can subsequently be integrated with the control model, resulting in a closed-loop system model. This approach is particularly tailored to meet the requirements of event-driven system verification. The process of transitioning from the FSM to the plant model in the form of IEC 61499 FBs is elaborated upon below, illustrating our methodology for achieving this integration.

G. TRANSFORMATION OF FSM PLANT MODEL TO IEC 61499 FBS

The presented application for verification follows a structure illustrated in Fig. 6, where a closed-loop system is formed by connecting a plant model obtained from the RMCSES with either a new or an existing controller. An existing controller is utilized to construct the RMCSES, whereas a new controller is connected with RMCSES for the purpose of verification. The verification process involves conformance checking to ensure that the newly developed controller operates in accordance with the previous controller. NuSMV tool can be used for verification through CTL/LTL specifications.

1) TRANSFORMATION RULES FROM FSM TO ECC

The graphical depiction of the transformation of RMCSES to plant model in IEC 61499 FB is illustrated in Fig. 6. The resulting plant model interface comprises control signals $c_1 \ldots c_n$ and NDT as input signals, while sensor signals

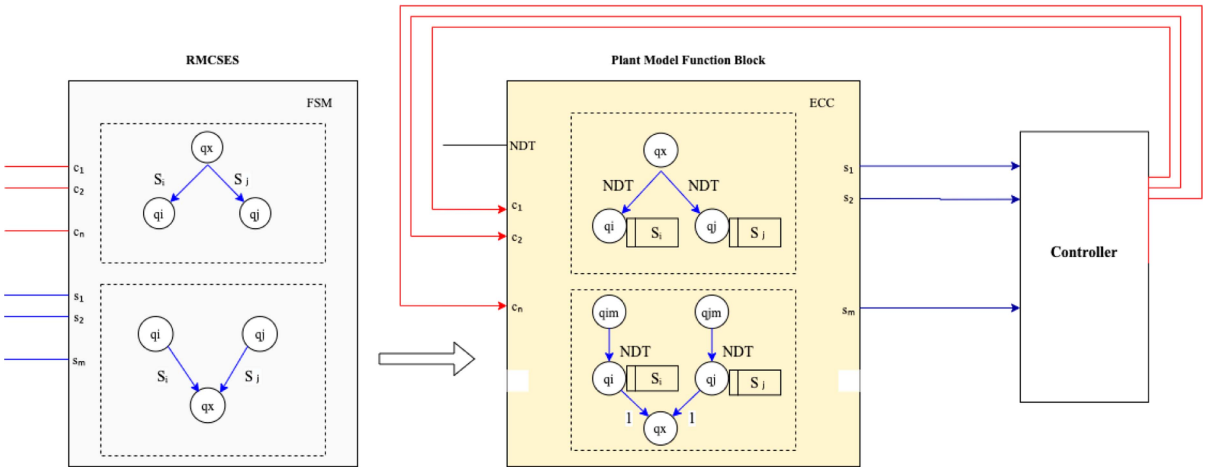

FIGURE 6. Application for verification.

$s_1 \ldots s_m$ are designated as the output signals of the FB. In this transformation, any transition in the FSM triggered by a sensor signal is replaced by an NDT transition, with the output of the next state serving as a sensor event signal. This NDT serves as a mechanism for initiating transitions at arbitrary time intervals. On the other hand, transitions within the FSM that are initiated by control signals remain unaltered when transitioning to the ECC model.

A) CASE 1: DIVERGING SENSOR SIGNALS

The transformation rule when sensor signal arcs diverge from a state is shown in Fig. 6. When signals emanate from one state and branch out to multiple states, the sensor signals are substituted with NDT signals, and each output place generates the corresponding sensor signal as its output.

Given transitions in the FSM:

$$(q_x, s_i) \rightarrow q_i$$
$$(q_x, s_j) \rightarrow q_j.$$

In ECC, these transitions are replaced by NDT transitions with output events s_i and s_j:

$$(q_x, \text{NDT}) \rightarrow (q_i, s_i)$$
$$(q_x, \text{NDT}) \rightarrow (q_j, s_j).$$

B) CASE 2: CONVERGING SENSOR SIGNALS

When signals come together into a single state from various states, they are altered according to the rule depicted in Fig. 6. It is important to note that it is not feasible to bring multiple NDT signals together into the same state and generate two sensor signals as outputs. To address this, we introduced additional intermediary states that produce their respective sensor signals as outputs, and eventually, these states converge into a single state. The transformation rule for this scenario is as follows:

Given transitions in the FSM:

$$(q_i, s_i) \rightarrow q_x$$
$$(q_j, s_j) \rightarrow q_x.$$

In ECC, additional intermediate states q_{im} and q_{jm} are introduced, and NDT transitions are used to connect them to q_i and q_j, respectively:

$$(q_{im}, \text{NDT}) \rightarrow (q_i, s_i)$$
$$(q_{jm}, \text{NDT}) \rightarrow (q_j, s_j).$$

Finally, q_i and q_j converge to q_x:

$$(q_i, 1) \rightarrow q_x$$
$$(q_j, 1) \rightarrow q_x.$$

Algorithm 2 is defined by applying the transformation rules described earlier to facilitate the conversion of RMSCES into a plant model in the IEC 61499 FB.

H. MDA AS A METHODOLOGICAL BASIS FOR THE DEVELOPMENT

In this study, the MDA was used for the development, according to which the design process is represented as a chain of model transformations, starting from the initial model and ending with the target one. In model-driven development the model transformation is "the heart and soul" of the approach [83]. The event log as a source model, and the target models are: 1) the monitor model and 2) the plant model. In accordance with this, two different chains of transformations were used. The plant model is important as an independent result that can be used for various purposes in the design process, including certifying a new controller during transitions between different hardware and software platforms. This challenge of certifying is addressed in this study. The certification process involves two essential activities: one is the creation of a model representing the plant through a series

Algorithm 1: FSM to Monitor ECC.

Input : FSM = { Q: $\{q_0, q_1, \ldots, q_{n-1}\}$ is a set of states, where q_0 is an initial state;
CS: $\{X_1, X_2, \ldots, X_{p+r}\}$ is a set of input symbols, where $X_i \in C \cup S$, where
C= $\{c_1, c_2, ..., c_p\}$ is a set of control signals, and S= $\{s_1, s_2, ..., s_r\}$ is a set of sensor signals;
$\delta \subseteq Q \times CS \times Q$ } is a transition relation;

Output: ECC = { QC:
$\{q_0, q_1, \ldots, q_{n-1}, SE_n, SE_{n+1} \ldots SE_{n+p+r-1}\} \supset Q$ is a set of ECC states;
CS is a set of input events (see above);
Out: {OK, ERROR} is a set of output events;
Alg= $\{a_0, a_1, \ldots, a_{n+p+r-1}\}$ is a set of algorithms;
ϕ : {Q × CS → QC × Alg × Out } is an ECC transition function}

1 StateID=q_0
2 /* Creating the conforming (right) ECC transitions */
3 **foreach** $(q_i, X_k, q_j) \in \delta$ **do**
4 Create ECC Transitions: $(q_i, X_k) \rightarrow (q_j, a_j, \text{OK})$ $a_j.StateID = j$
5 /* Creating the non-conforming (erroneous) ECC transitions */
6 **foreach** $q_w \in Q$ **do**
7 **foreach** $X_e \in CS$ - $T(q_w)$ **do**
8 Create ECC Transitions: $(q_w, X_e) \rightarrow (SE_e, a_e, \text{ERROR})$ $a_e.EventID = e$
9 **return** *ECC*;
10 ***Note* : Used variables and sets**
11 StateID is an output variable of the FB Monitor that stores the identifier of the current ECC state;
12 a_j.StateID is an occurrence of the StateID variable in the a_j algorithm associated with an ECC state;
13 EventID is an output variable of the FB Monitor that stores the identifier of the erroneous (not expected) input event;
14 a_e.EventID is an occurrence of the EventID variable in the a_e algorithm associated with an ECC state;
15 $T(q_w) \subseteq CS$ is a set of signals labelling the transitions outgoing from the ECC state $q_w \in Q$.

Algorithm 2: FSM to Plant Model ECC.

Input : FSM = { Q: $\{q_0, q_1, \ldots, q_{n-1}\}$ is a set of states, where q_0 is an initial state;
CS = $C \cup S$ is a set of input signals, where
C = $\{c_1, c_2, \ldots, c_p\}$ is a set of control signals,
S = $\{s_1, s_2 \ldots, s_r\}$ is a set of sensorsignals;
$\delta \subseteq Q \times CS \times Q$ is a transition relation; }

Output: ECC = { $QC = Q \cup D$ is a set of ECC states, where D is a set of additional ECC states created dynamically;
$Cond = TC \cup \{1\}$ is a set of ECC transition conditions, where $TC = C \cup \{NDT\}$ is a set of input events; S is a set of output events (see above);
$\phi : QC \times Cond \rightarrow QC \cup QC \times S$ is an ECC transition function }

1 /* Processing the FSM transition labelled by control signals */
2 **foreach** $(q_i, \sigma, q_j) \in \delta$ **do**
3 **if** $\sigma \in C$ **then**
4 Create ECC Transition: $(q_i, \sigma) \rightarrow (q_j)$;
5 /* Processing the FSM transition labelled by sensor signals which are not converged */
6 **foreach** $(q_i, \sigma, q_j) \in \delta$ **do**
7 **if** $\sigma \in S$ *AND* $q_i \notin QJ$ **then**
8 Create ECC Transition: $(q_i, NDT) \rightarrow (q_j, \sigma)$;
9 /* Processing the FSM transition labelled by converging sensor signals */
10 **foreach** $(q_w \in QJ)$ **do**
11 **foreach** $(q_i, \sigma, q_w) \in \delta$ **do**
12 **if** $\sigma \in S$ **then**
13 Create ECC State $d \in D$
14 Create ECC Transition: $(q_i, NDT) \rightarrow (d, \sigma)$;
15 $(d, 1) \rightarrow (q_w)$;
16 **return** *ECC*;
17 ***Note* : Used variables and sets**
18 $QJ = \{q_j \in Q, |\{(q_i, s_k, q_j) \in \delta'\}| > 1\}$, where $\delta' = \delta \cap Q \times S \times Q$ is a set of ECC states which have converging sensor signals in the FSM

of transformations from event log to Petri net, RG, FSM of the plant, resulting in an IEC 61499 FB-based plant model. Next is the development of a comprehensive model for the new controller using IEC 61499 FBs. As the new controller is developed in IEC 61499 FBs, it is advisable to maintain consistency by expressing the plant model in the same notation. This design not only ensures uniformity but is also supported by the availability of the fb2smv, a tool that streamlines the conversion of IEC 61499 FBs into SMV code. These conversions, while intricate, are significant for systematic controller verification and migration, ensuring uniformity and reliability in the certification process. While many conversions are required primarily for certification and monitoring system implementation, a more direct approach is available when the sole goal is to verify an existing system based on event log data. In this scenario, the Petri net generated in ProM can be directly analyzed in dedicated platforms like TINA, which

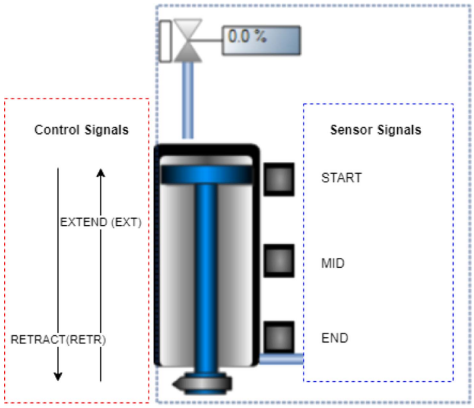

FIGURE 7. Pneumatic cylinder HMI representation NxtStudio.

offers compatible tools for this purpose. There are different corresponding tools available within this system. There exist methods for directly converting Petri nets into SMV code [84]. The resulting SMV code can be employed for analysis using queries rooted in LTL and CTL within systems like NuSMV.

VI. CASE STUDY: A PNEUMATIC CYLINDER

A. GENERAL DESCRIPTION

Visualization of the cylinder's generated by its IEC 61499 simulation model in NxtStudio is shown in Fig. 7. The cylinder has three sensors START, MID, and END indicating the position of the piston. The vertical cylinder's motion is controlled by EXT and RETR actuator signals'. EXT to move downwards and RETR to move upwards. The Actuator and Sensor signals of the pneumatic cylinder are represented in Fig. 7.

B. EVENT LOG DESCRIPTION

The event log of the different processing scenarios of the vertical cylinder is recorded in CSV format, which captures the activities in chronological order. The event log, depicted in Fig. 8(a), comprises of three columns: CaseId, State, and Activity. The CaseId is unique for each processing scenario, while the Activity column represents the events that occurred during the processing of a scenario. Signals (like events) happen instantaneously, and it is a good practice to use the Boolean vector to store them, with "1" indicating it is set and "0" indicating it is reset. The timestamp information is used solely for sorting the activities in the event log.

The event log captures various processing scenarios of the vertical cylinder, including movements from START to END via MID, and returns to the START position. The event log also includes random movements of the cylinder captured by pressing the HMI buttons. These events are recorded using the OPC UA communication protocol.

C. FSM GENERATION FROM EVENT LOG

The ProM process mining tool is utilized for constructing a Petri net from the event log. ProM offers several process discovery algorithms, and the **alpha** algorithm is utilized here for extracting the process from the event log. The event log, in CSV format, is first converted to XES format, and the "Case" column is selected as "CaseId," while the event column is a combination of "Activity" and "State" columns. This XES format is then provided as input to the alpha algorithm. The resulting Petri net is shown in Fig. 8(b).

The TINA tool is used to generate the RG for the Petri net. To achieve cyclic processing behavior, a new transition named "Repeat" is added to the Petri net, connecting the "END" state to the "START" state. Stepwise simulation is performed to test the Petri net, as shown in Fig. 8(b). The RG is then used to generate the FSM in text format. The RG obtained from the Petri net can contain spontaneous transitions, rendering it a nondeterministic FSM. To convert this nondeterministic FSM to a deterministic one, the "Converter of TINA RG to GraphML" software tool [75] is utilized. The resulting deterministic FSM is presented in Fig. 8(c) in GraphML format. The FSM is visualized using the "Yed" GraphML editor [81], which provides a clear process logic behind the system. The state information on the edges of the FSM is removed since it is unnecessary to differentiate control and sensor signals.

D. IEC 61499 REPRESENTATION OF MONITOR

To implement the monitor in the IEC 61499 FBs, the RM-CSES' FSM representation is taken as a starting point. The transformation rules are then applied to this FSM to create the monitor's FB interface. The interface, represented in Fig. 9(a), includes input signals for sensors and control signals. An additional event "R" is included for the "Repeat" event, which is used for the cyclic operation of the system's process.

The monitor FB generates an "OK" event with the corresponding StateID when the system process executes in the correct order. In the case of any deviation from the expected process behavior, an "ERROR" signal is emitted along with the "EventID" indicating the specific event responsible for the deviation. The ECC for the monitor FB is represented partially in Fig. 9 (E2). If any event other than "VCEXT_TRUE" and "VCRETR_FALSE" occurs in "STATE4," an "ERROR" event is triggered, along with the respective EventID. The ECC in Fig. 9 (E2) only shows errors occurring from "STATE 4," but in the actual scenario, there will be multiple events from different states (STATE1, STATE 2, etc.) leading to their corresponding "ERROR" states (VCEXT_TRUE, VCRETR_FALSE, etc.).

E. IEC 61499 REPRESENTATION OF PLANT MODEL

The ECC of the plant model is generated using the deterministic FSM obtained from the Petri net. Transformation rules discussed in Section V-G are applied to the FSM to derive the ECC. The interface of the plant model mirrors the controller depicted in Fig. 9(a). The plant model has an additional signal called NDT, which provides a nondeterministic delay before producing the sensor signals as output. The behavior of the system is represented by the ECC and is embedded inside a FB. The ECC of the plant model is illustrated in Fig. 9 (E1).

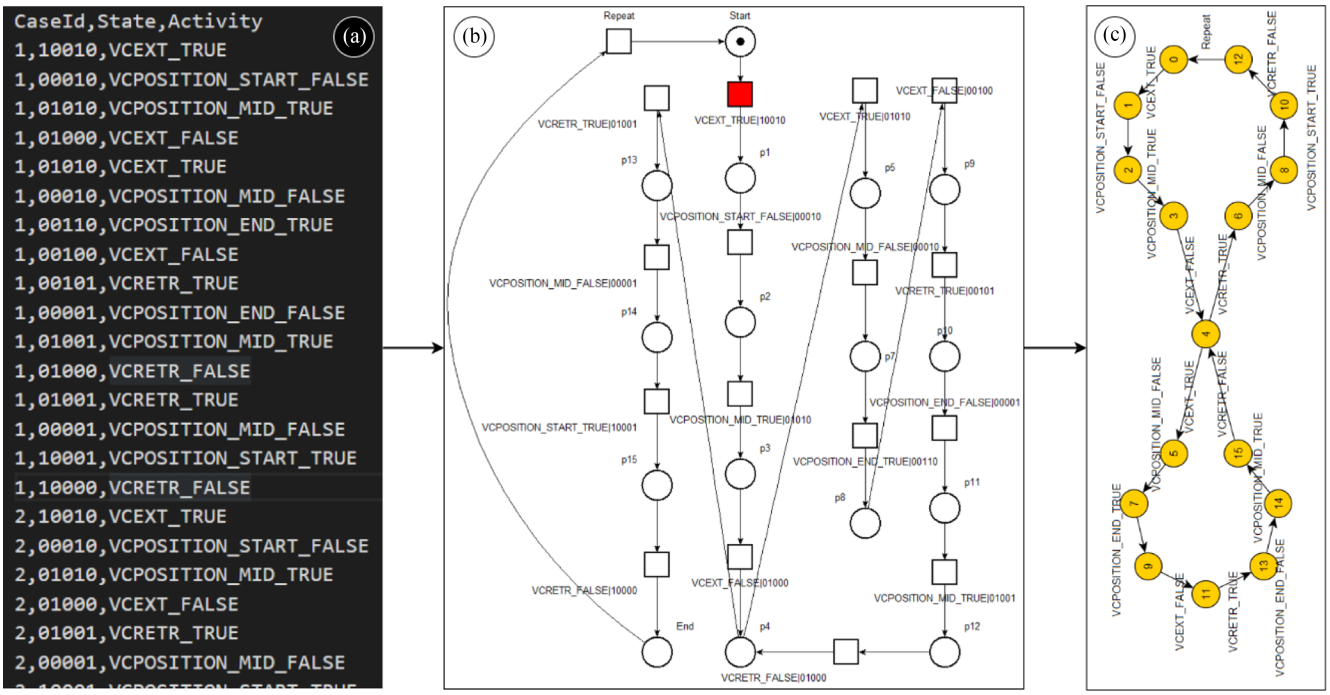

FIGURE 8. (a) Vertical cylinder event log representation in CSV, (b) Stepwise simulation of Petri net in TINA, and (c) FSM representation in Yed editor.

VII. TRACES WITH ONLY EVENTS, NO GLOBAL STATE

A. GLOBAL STATE

The global state of the system is defined as a vector of boolean values that represents the combination of sensor and actuator signals. This vector is used to construct the global state context of the system. The global state context represents a unique frame of reference for the system. To ensure the accuracy of the global state, clock synchronization and time stamping are used across all distributed controllers in the network.

B. IMPORTANCE OF STATE INFORMATION

In order to obtain accurate results from the process mining algorithm, it is essential to include state information in the event log. The activity in the event log should reflect the combination of the event signal value and state information. By doing so, the generated Petri net will consist of two loops, which will showcase the different actions of the pneumatic cylinder. This confirms that the system is capable of supporting multiple processing paths or options.

A) HOW NEW TRANSITION IN PETRI NET IS CREATED WHEN STATE INFORMATION IS CONSIDERED?

A new transition in Petri net is created only when the combination of signal value and state of the system is different.

1) If the state of the system is the same and signals are different, then a new transition occurs.
2) If the signals are the same but occur in different state conditions, then a new transition occurs.

So, the number of events in the event log would not be the same as the number of transitions in the Petri net. The Petri net constructed without the state information for the same experiment is shown in Fig. 10. When comparing the Petri net created using state information Figs. 8(b) and 10, the Petri net without state information, Fig. 10, is inaccurate and shows misleading transitions which never occur in the process scenarios.

VIII. RESULTS AND ANALYSIS

A. FORMAL VERIFICATION OF THE SYSTEM USING GENERATED PLANT MODEL

The IEC 61499 FB plant model can be used in conjunction with the controller for formal verification purposes. The fb2smv tool is used to convert the IEC 61499 FBs into the SMV formal model of the closed-loop system, which is then verified with a symbolic model checker tool called NuSMV. Various CTL or LTL specifications can be used to check the formal model against the desired system behavior. The NuSMV tool allows one to interactively explore the states of the system and observe its behavior. The simulation mode in NuSMV is useful for validating the system works according to the correct process sequence. The closed-loop model of the system, illustrated in Fig. 9(a) but without the monitor FB, was transformed into a formal model and simulated using NuSMV. The simulation confirmed that the system followed the correct path. To detect possible failure situations that could arise in critical scenarios, the system can be verified using CTL specifications. The SMV specification of the system guarantees that the specified property will never occur in the system at any time.

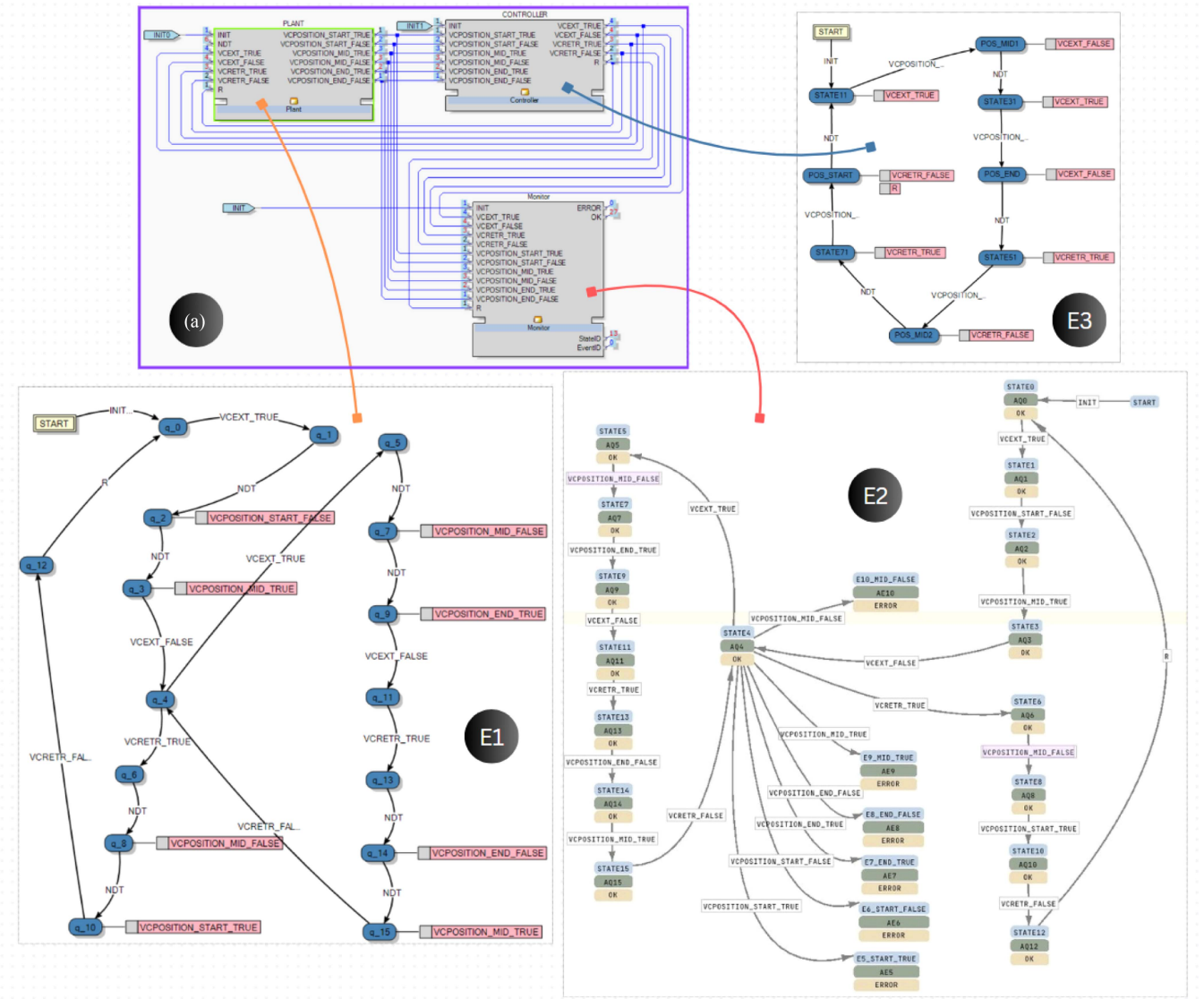

FIGURE 9. Closed-loop system of plant and controller model with the monitor.

```
check_ltlspec -p "G (ClosedLoopModel_inst.
   PLANT_VCPOSITION_START_TRUE -> F (
   ClosedLoopModel_inst.
   CONTROLLER_VCRETR_FALSE ))"
```

The above specification is checked using NuSMV proving that when the vertical cylinder reaches the start position the controller always produces a RETRACT_FALSE event signal in the future. Like this, it is possible to verify the system properties by connecting it with the controller in closed-loop with the help of CTL or LTL specifications.

B. CLOSED-LOOP SYSTEM INTEGRATION WITH MONITOR

The closed-loop system of the plant model and new controller is integrated with a monitor, as depicted in Fig. 9(a). The integration of the closed-loop system is achieved by connecting the output sensor and output actuator signals to the input of the previously generated monitor 9 (A). The monitor checks the conformance of the system by analyzing the signal values. If the signals occur according to the actual process scenario, then it produces an "OK" event with its corresponding "StateID." In case of nonconformance, the monitor produces an "RROR" signal along with the "StateID" and "EventID." The "StateID" specifies the state in which the error occurred, while the "EventID" represents the event that caused the error.

In this experiment, the model produces the right events and it follows the process scenario; therefore, there are no ERROR signals produced after one cycle. Fig. 9(a) shows that it reached "StateID" = 13 and ran one and a half cycles ($R = 1$) is completed without any occurrence of errors ("ERROR" = 0). In order to check whether the monitor captures the error, "VCEXT_TRUE" event was added instead of "VCRETR_TRUE" at "STATE51" of the controller in Fig. 9(E3). Then the monitor produced an "ERROR" signal

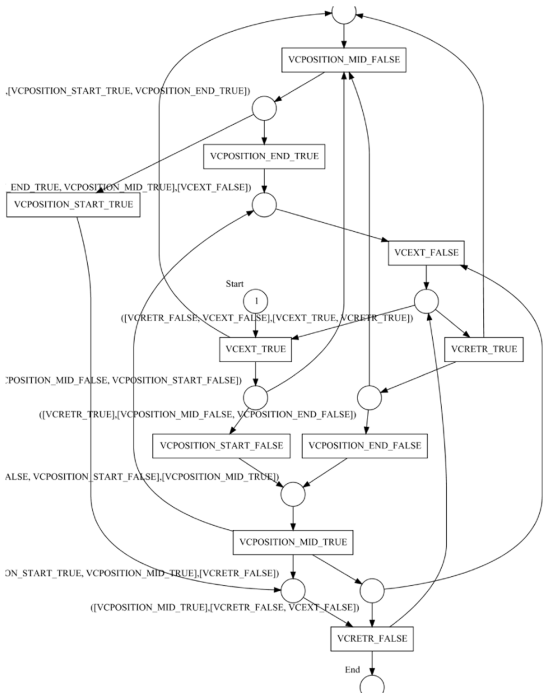

FIGURE 10. Petri net constructed without state information.

with "EventID" = 1 and "StateID" = 11. The monitor at "STATE11" [see Fig. 9 (E2)] checks if any event other than "VCRETR_TRUE" occurs then it directs toward the respective event error state, i.e., in this case, it goes to "E1_VCEXT_TRUE" state and executes "AE1" algorithm and produces ERROR signal as output along with the "EventID" = 1 and "StateID" = 11.

IX. CONCLUSION AND FUTURE WORK

The study proposes a novel method for generating a monitor and plant model from behavioral traces. The monitor is used to detect any deviation in the process sequence, while the automatic generation of a plant model is a challenging task without adequate domain knowledge and system behavior. The study also demonstrates the automatic generation of a plant model solely based on recorded event signals. The resulting plant model is useful for the formal verification of closed-loop systems in compliance with IEC 61499.

Further testing with additional use cases is required to validate the effectiveness of this approach. Future work could involve connecting the monitor to the actual plant in order to detect deviations from the expected process scenario. The global state of the system is the essential information needed for this approach, but recording all sensor and actuator signal values whenever an event occurs makes this process difficult. It is possible to poll all sensors and actuators' signal values with the help of synchronization and timestamp, but for a complex system, there will be a considerable amount of time delays. The retrieval of state information for a complex system needs to be analyzed.

In future research, the significance lies in exploring the performance of the approach, particularly in handling different levels of system complexity. This entails conducting comprehensive analyses of time complexity and engaging in quantitative experiments to gain deeper insights into the effectiveness and scalability of the proposed methodology. Such efforts are essential for establishing a robust understanding of the approach's capabilities across various scenarios and ensuring its applicability in real-world contexts.

APPENDIX A: LOW LEVEL EVENT LOG

This section describes the formal definition of the low-level event log. Fig. 1(a) shows the closed-loop system consisting of the plant and the controller. The set of (abstract) signal lines, L in a closed-loop system is defined as

$$L = S \cup C; S \cap C = \phi$$

where $S = \{s_1, s_2, \ldots, s_m\}$ is a set of lines from Plant's sensor to Controller and $C = \{c_1, c_2, \ldots, c_n\}$ is a set of lines from Controller to Plant's actuator. Fig. 1(a) shows a closed-loop system with indicated signal lines. The set L includes all $(n + m)$ lines:

$$L = \{l_1, l_2, \ldots, l_{n+m}\}.$$

Let there be a set of attributes or parameters that can be associated with signals transmitted over the lines

$$A = \{a_1, a_2, \ldots, a_k\}.$$

Each signal line can have its own set of attributes, defined by a function:

$$p : L \rightarrow 2^A.$$

Thus, signals with the following set of attributes are transmitted over the line $l_i \in L$. This set of attributes is assumed to be ordered. A function is defined to assign attribute values to each of the lines $l_i \in L$

$$z_i : li \rightarrow Dom\left(a_i^1\right) * Dom\left(a_i^2\right) * \cdots * Dom(a_i^t)$$

where $Dom(a_i^j)$ is a domain for the attribute a_i^j. In the case of a large dimension of these domains, the total number of possible signal values can be very large. For convenience, a generalized value function is introduced

$$z = z_1 \cup z_2 \cup \cdots \cup z_{n+m}.$$

A signal on the line $l_i \in L$ is a set of attribute values on this line, i.e., $z(l_i)$. $z(l_i)[t]$ denotes a set of values of attributes of the line $l_i \in L$ at the moment of time $t \in N^+$, where N^+ is the set of positive integer numbers. The use of discrete time does not change the general situation.

An event on the line $l_i \in L$ is the moment when the signal on this line changes. The signal changes if the value of at least one of its attributes changes. The condition that determines the occurrence of an event on the line $l_i \in L$ is the following:

$$\exists t(z(l_i)[t] \neq z(l_i)[t+1]).$$

Theoretically, at the same time, different events can occur on two or more lines. It is possible, but the probability of this is very small. All events in the system can be enumerated

$$E = \{e_1, e_2, \ldots, e_h\}.$$

Each event $e_i \in E$ is defined by the following tuple:

$$e_i = (t_i, ne_i)$$

where t_i is the time at which the event occurred and ne_i is the name of the event. The name of the event $e_i \in E$ arising at the line $l_j \in L$ is defined as follows:

$$ne_i = (l_j, z(l_j)[t_i]). \quad (1)$$

From this formula, one can see that the event name consists of the line ID and the values of all line attributes. We define a function that maps a signal line to a component (device) with which it is connected

$$r : L \to D$$

where $D = \{d_1, d_2, \ldots, d_v\}$ is a set of components (devices) in the system.

An event log entry for event $e_i \in E$ at line $l_j \in L$ is defined by the following tuple:

$$w_i = (CaseID, t_i, ne_i, r(l_j))$$

where CaseID is the constant identifier for each cyclic operation of the processes, t_i (that is, Timestamp) is the time when the event occurred, ne_i (that is, Activity) is the name of the event according to the formula 1 above, $r(l_j)$ is the component with which the corresponding line is associated. It should be noted that the component $r(l_j)$ may not be used. The component t_i may not be used if the event log is chronologically ordered or if timing is not taken into account.

REFERENCES

[1] "International Standard IEC 61499-1: Function blocks, Part 1: Architecture, International Electrotechnical Commission," 2nd ed., 2012.

[2] 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 Trans. Syst., Man, Cybern. Syst.*, vol. 50, no. 3, pp. 851–862, Mar. 2020.

[3] W. Dai and V. Vyatkin, "Redesign distributed PLC control systems using IEC 61499 function blocks," *IEEE Trans. Autom. Sci. Eng.*, vol. 9, no. 2, pp. 390–401, Apr. 2012.

[4] R. Lewis, *Modelling Control Systems Using IEC 61499: Applying Function Blocks to Distributed Systems*. London, U.K.: Institution of Engineering and Technology, 2001.

[5] G. Frey and T. Hussain, "Modeling techniques for distributed control systems based on the IEC 61499 Standard-current approaches and open problems," in *Proc. 8th Int. Workshop Discrete Event Syst.*, 2006, pp. 176–181.

[6] M. Wenger, A. Zoitl, and J. O. Blech, "Behavioral type-based monitoring for IEC 61499," in *Proc. IEEE 20th Conf. Emerg. Technol. Factory Autom.*, 2015, pp. 1–8.

[7] D. Do Tran, J. Walter, K. Grüttner, and F. Oppenheimer, "Towards time-sensitive behavioral contract monitors for IEC 61499 function blocks," in *Proc. IEEE Conf. Ind. Cyberphysical Syst.*, 2020, pp. 27–34.

[8] I. Hegny, M. Wenger, and A. Zoitl, "IEC 61499 based simulation framework for model-driven production systems development," in *Proc. IEEE 15th Conf. Emerg. Technol. Factory Autom.*, 2010, pp. 1–8.

[9] S. Patil, V. Dubinin, and V. Vyatkin, "Formal verification of IEC61499 function blocks with abstract state machines and SMV–modelling," in *Proc. IEEE Trustcom/BigDataSE/ISPA*, 2015, pp. 313–320.

[10] J. O. Blech, P. Lindgren, D. Pereira, V. Vyatkin, and A. Zoitl, "A comparison of formal verification approaches for IEC 61499," in *Proc. IEEE 21st Int. Conf. Emerg. Technol. Factory Autom.*, 2016, pp. 1–4.

[11] G. Cengic and K. Akesson, "On formal analysis of IEC 61499 applications, part A: Modeling," *IEEE Trans. Ind. Inform.*, vol. 6, no. 2, pp. 136–144, May 2010.

[12] M. Ramdani, L. Kahloul, M. Khalgui, Z. Li, and M. Zhou, "RCTL: New temporal logic for improved formal verification of reconfigurable discrete-event systems," *IEEE Trans. Automat. Sci. Eng.*, vol. 18, no. 3, pp. 1392–1405, Jul. 2021.

[13] 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 Proc. Vol.*, vol. 42, no. 4, pp. 211–216, 2009.

[14] M. Xavier, J. Håkansson, S. Patil, and V. Vyatkin, "Plant model generator from digital twin for purpose of formal verification," in *Proc. 26th IEEE Int. Conf. Emerg. Technol. Factory Autom.*, 2021, pp. 1–4.

[15] M. Xavier, V. Dubinin, S. Patil, and V. Vyatkin, "Plant model generation from event log using prom for formal verification of CPS," 2022, *arXiv:2211.03681*.

[16] J. H. Christensen, *Design patterns for systems engineering in IEC 61499, Verteilte Automatisierung - Modelle und Methoden für Entwurf, Verifikation, Engineering und Instrumentierung (VA2000)*. Germany: Otto-von-Guericke-Universität Magdeburg, Mar. 22–23, 2000, pp. 63–71.

[17] K. Thramboulidis et al., "IEC 61499 as an enabler of distributed and intelligent automation: A. state-of-the-art review–a different view," *J. Eng.*, vol. 2013, pp. 1–9, 2013.

[18] I. Hegny, T. Strasser, M. Melik-Merkumians, M. Wenger, and A. Zoitl, "Towards an increased reusability of distributed control applications modeled in IEC 61499," in *Proc. IEEE 17th Int. Conf. Emerg. Technol. Factory Autom.*, 2012, pp. 1–8.

[19] W. Dai, V. Vyatkin, J. H. Christensen, and V. N. Dubinin, "Bridging service-oriented architecture and IEC 61499 for flexibility and interoperability," *IEEE Trans. Ind. Inform.*, vol. 11, no. 3, pp. 771–781, Jun. 2015.

[20] C. Sunder et al., "Usability and interoperability of IEC 61499 based distributed automation systems," in *Proc. 4th IEEE Int. Conf. Ind. Inform.*, 2006, pp. 31–37.

[21] V. Vyatkin, "The IEC 61499 standard and its semantics," *IEEE Ind. Electron. Mag.*, vol. 3, no. 4, pp. 40–48, Dec. 2009.

[22] G. Cengic and K. Akesson, "On formal analysis of IEC 61499 applications, part B: Execution semantics," *IEEE Trans. Ind. Inform.*, vol. 6, no. 2, pp. 145–154, May 2010.

[23] C. Schnakenbourg, J.-M. Faure, and J.-J. Lesage, "Towards IEC 61499 function blocks diagrams verification," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, 2002, pp. 1–6.

[24] L. Yoong, "Modelling and synthesis of safety-critical software with IEC 61499," Ph.D. dissertation, ResearchSpace, Auckland, 2010.

[25] V. Vyatkin, H.-M. Hanisch, C. Pang, and C.-H. Yang, "Closed-loop modeling in future automation system engineering and validation," *IEEE Trans. Syst., Man, Cybern., C*, vol. 39, no. 1, pp. 17–28, Jan. 2009.

[26] C. Pang and V. Vyatkin, "Systematic closed-loop modelling in IEC 61499 function blocks: A case study," *IFAC Proc. Vol.*, vol. 42, no. 4, pp. 199–204, 2009.

[27] R. Sinha, S. Patil, L. Gomes, and V. Vyatkin, "A survey of static formal methods for building dependable industrial automation systems," *IEEE Trans. Ind. Inform.*, vol. 15, no. 7, pp. 3772–3783, Jul. 2019.

[28] M. Xavier, S. Patil, V. Dubinin, and V. Vyatkin, "Formal modelling, analysis, and synthesis of modular industrial systems inspired by net condition/event systems," in *Proc. Int. Conf. Appl. Theory Petri Nets Concurrency*, 2023, pp. 16–33.

[29] P. Ovsiannikova, E. Le Priol, V. Perret, P. Jhunjhunwala, M. Xavier, and V. Vyatkin, "Formal verification of observers supervising a cyber-physical system implemented using IEC 61499," in *Proc. IEEE 32nd Int. Symp. Ind. Electron.*, 2023, pp. 1–6.

[30] A. Malik, P. S. Roop, N. Allen, and T. Steger, "Emulation of cyber-physical systems using IEC-61499," *IEEE Trans. Ind. Informat.*, vol. 14, no. 1, pp. 380–389, Jan. 2018.

[31] "FB2SMV model generator." Accessed: Mar. 5 2024. [Online]. Available: https://github.com/dmitrydrozdov/fb2smv

[32] A. Cimatti et al., "NUSMV 2: An opensource tool for symbolic model checking," in *Proc. Int. Conf. Comput. Aided Verification*, 2002, pp. 359–364.

[33] M. Xavier, S. Patil, and V. Vyatkin, "Cyber-physical automation systems modelling with IEC 61499 for their formal verification," in *Proc. IEEE 19th Int. Conf. Ind. Inform.*, 2021, pp. 1–6.

[34] R. Agrawal, D. Gunopulos, and F. Leymann, "Mining process models from workflow logs," in *Proc. Int. Conf. Extending Database Technol.*, 1998, pp. 467–483.

[35] W. Van Der Aalst, "Process mining: Overview and opportunities," *ACM Trans. Manage. Inf. Syst.*, vol. 3, no. 2, pp. 1–17, 2012.

[36] M. Xavier, V. Dubinin, S. Patil, and V. Vyatkin, "Process mining in industrial control systems," in *Proc. IEEE 20th Int. Conf. Ind. Inform.*, 2022, pp. 1–6.

[37] B. F. Van Dongen, A. K. A. de Medeiros, H. Verbeek, A. Weijters, and W. M. van Der Aalst, "The prom framework: A new era in process mining tool support," in *Proc. Int. Conf. Appl. Theory Petri Nets*, 2005, pp. 444–454.

[38] C. W. Günther and A. Rozinat, "DISCO: Discover your processes," *BPM*, vol. 940, no. 1, pp. 40–44, 2012.

[39] "Prom." Accessed: Jan. 17, 2024. [Online]. Available: https://www.promtools.org/

[40] C. A. Petri, "Kommunikation mit automaten," dissertation, Dept. Comput. Sci., 1962. Accessed: Jun. 9, 2024. [Online]. Available: http://edoc.sub.uni-hamburg.de/informatik/volltexte/2011/160/

[41] S. Dunzer, M. Stierle, M. Matzner, and S. Baier, "Conformance checking: A state-of-the-art literature review," in *Proc. 11th Int. Conf. Subject-Oriented Bus. Process Manage.*, 2019, pp. 1–10.

[42] V. Naderifar, S. Sahran, and Z. Shukur, "A review on conformance checking technique for the evaluation of process mining algorithms," *TEM J.*, vol. 8, no. 4, 2019, Art. no. 1232.

[43] H. Sun, W. Liu, L. Qi, X. Ren, and Y. Du, "An algorithm for mining indirect dependencies from loop-choice-driven loop structure via petri nets," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 52, no. 9, pp. 5411–5423, Sep. 2022.

[44] H. Sun, W. Liu, L. Qi, Y. Du, X. Ren, and X. Liu, "A process mining algorithm to mixed multiple-concurrency short-loop structures," *Inf. Sci.*, vol. 542, 2021, pp. 453–475.

[45] W. Van der Aalst, A. Adriansyah, and B. Van Dongen, "Replaying history on process models for conformance checking and performance analysis," *Wiley Interdiscipl. Rev., Data Mining Knowl. Discov.*, vol. 2, no. 2, pp. 182–192, 2012.

[46] J. Munoz-Gama et al., *Conformance Checking and Diagnosis in Process Mining*. Berlin, Germany: Springer, 2016.

[47] J. E. Cook and A. L. Wolf, "Discovering models of software processes from event-based data," *ACM Trans. Softw. Eng. Methodol.*, vol. 7, no. 3, pp. 215–249, 1998.

[48] W. M. Van Der Aalst and B. F. V. Dongen, "Discovering petri nets from event logs," in *Transactions on Petri Nets and Other Models of Concurrency VII*. Berlin, Germany: Springer, 2013, pp. 372–422.

[49] G. Liu, *Petri Nets: Theoretical Models and Analysis Methods for Concurrent Systems*. Berlin, Germany: Springer Nature, 2022.

[50] P. Buchholz and P. Kemper, "Hierarchical reachability graph generation for petri nets," *Formal Methods System Des.*, vol. 21, pp. 281–315, 2002.

[51] J. Ye, M. Zhou, Z. Li, and A. Al-Ahmari, "Structural decomposition and decentralized control of petri nets," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 48, no. 8, pp. 1360–1369, Aug. 2018.

[52] F.-S. Hsieh, "Robustness analysis of non-ordinary petri nets for flexible assembly/disassembly processes based on structural decomposition," *Int. J. Control*, vol. 84, no. 3, pp. 496–510, 2011.

[53] R. Wiśniewski et al., "Decomposition of distributed edge systems based on the petri nets and linear algebra technique," *J. Syst. Archit.*, vol. 96, pp. 20–31, 2019.

[54] E. Best, R. Devillers, and M. Koutny, *Petri Net Algebra*. Berlin, Germany: Springer Science & Business Media, 2013.

[55] T. Miyamoto and K. Horiguchi, "Modular reachability analysis of petri nets for multiagent systems," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 43, no. 6, pp. 1411–1423, Nov. 2013.

[56] A. Giua and F. DiCesare, "Petri net structural analysis for supervisory control," *IEEE Trans. Robot. Autom.*, vol. 10, no. 2, pp. 185–195, Apr. 1994.

[57] P. Küngas, "Petri net reachability checking is polynomial with optimal abstraction hierarchies," in *Proc. Int. Symp. Abstraction, Reformulation, Approximation*, 2005, pp. 149–164.

[58] R. Wiśniewski, G. Bazydło, L. Gomes, A. Costa, and M. Wojnakowski, "Analysis and design automation of cyber-physical system with hippo and IoPT-tools," in *Proc. 45th Annu. Conf. IEEE Ind. Electron. Soc.*, 2019, pp. 5843–5848.

[59] A. Kiviriga, "Efficient model checking: The power of randomness," Ph.D. dissertation, Aalborg Univ. 2023.

[60] R. Wiśniewski, Ł. Stefanowicz, A. Bukowiec, and J. Lipiński, "Theoretical aspects of Petri nets decomposition based on invariants and hypergraphs," in *Multimedia and Ubiquitous Engineering*. Berlin, Germany: Springer, 2014, pp. 371–376.

[61] Y. Cai, I. Nishii, and T. Sekiguchi, "Modeling by petri net with place invariants for sequential control systems," *Elect. Eng. Jpn.*, vol. 115, no. 5, pp. 100–111, 1995.

[62] E. Zambon and A. Rensink, "Graph subsumption in abstract state space exploration," in *Proc. Conf. Comput. Graph. Interactive Techn. Australas. Southeast Asia*, 2012. [Online]. Available: https://api.semanticscholar.org/CorpusID:2828529

[63] S. Apel, D. Beyer, V. Mordan, V. Mutilin, and A. Stahlbauer, "On-the-fly decomposition of specifications in software model checking," in *Proc. 24th ACM SIGSOFT Int. Symp. Found. Softw. Eng.*, 2016, pp. 349–361.

[64] Z. Xin-feng, W. Jian-dong, L. Bin, Z. Jun-wu, and W. Jun, "Methods to tackle state explosion problem in model checking," in *Proc. 3rd Int. Symp. Intell. Inf. Technol. Appl.*, 2009, pp. 329–331.

[65] A. Shrestha, L. Xing, and Y. Dai, "Decision diagram based methods and complexity analysis for multi-state systems," *IEEE Trans. Rel.*, vol. 59, no. 1, pp. 145–161, Mar. 2010.

[66] L. He, G. Liu, and M. Zhou, "Petri-net-based model checking for privacy-critical multiagent systems," *IEEE Trans. Comput. Social Syst.*, vol. 10, no. 2, pp. 563–576, Apr. 2023.

[67] S. Lüdtke, M. Schröder, F. Krüger, S. Bader, and T. Kirste, "State-space abstractions for probabilistic inference: A systematic review," *J. Artif. Intell. Res.*, vol. 63, pp. 789–848, 2018.

[68] P. Godefroid, *Partial-Order Methods for the Verification of Concurrent Systems: An Approach to the State-Explosion Problem*. Berlin, Germany: Springer, 1996.

[69] E. M. Clarke, O. Grumberg, M. Minea, and D. Peled, "State space reduction using partial order techniques," *Int. J. Softw. Tools Technol. Transfer*, vol. 2, pp. 279–287, 1999.

[70] R. Wiśniewski, A. Karatkevich, M. Adamski, A. Costa, and L. Gomes, "Prototyping of concurrent control systems with application of petri nets and comparability graphs," *IEEE Trans. Control Syst. Technol.*, vol. 26, no. 2, pp. 575–586, Mar. 2018.

[71] R. Czerwinski and D. Kania, *Finite State Machine Logic Synthesis for Complex Programmable Logic Devices*, vol. 231. Berlin, Germany: Springer Science & Business Media, 2013.

[72] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev, *Logic Synthesis for Asynchronous Controllers and Interfaces*, vol. 8. Berlin, Germany: Springer Science & Business Media, 2012.

[73] A. El-Maleh, S. M. Sait, and F. N. Khan, "Finite state machine state assignment for area and power minimization," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2006, pp. 1–4.

[74] R. Czerwinski, D. Kania, R. Czerwinski, and D. Kania, "Synthesis of FSMs," in *Finite State Machine Logic Synthesis for Complex Programmable Logic Devices*. London, U.K.: Springer, 2013, pp. 25–48.

[75] M. Xavier, V. Dubinin, S. Patil, and V. Vyatkin, "An interactive learning approach on digital twin for deriving the controller logic in IEC 61499 standard," in *Proc. 27th Int. Conf. Emerg. Technol. Factory Autom.*, 2022, pp. 1–7.

[76] B. Berthomieu, P.-O. Ribet, and F. Vernadat, "The tool TINA–construction of abstract state spaces for petri nets and time petri nets," *Int. J. Prod. Res.*, vol. 42, no. 14, pp. 2741–2756, 2004.

[77] H. E. Verbeek and R. Jagadeesh Chandra Bose, "ProM 6.0 tutorial," 2010, last accessed: Mar. 7th, 2024. [Online]. Available: http://www.promtools.org/prom6/downloads/prom-6.0-tutorial.pdf

[78] S. Rodger and T. Finley, "JFLAP," 2015, last accessed: Jan. 20, 2024.

[79] "Graphml," last accessed: Jan. 11, 2024. [Online]. Available: http://graphml.graphdrawing.org/specification/dtd.html

[80] M. Bastian, S. Heymann, and M. Jacomy, "GEPHI: An open source software for exploring and manipulating networks," in *Proc. Int. AAAI Conf. Web Social Media*, 2009, pp. 361–362.

[81] "Yworks," last accessed: Jan. 15, 2024. [Online]. Available: https://www.yworks.com/

[82] "Holobloc," last accessed: Mar. 5, 2024. [Online]. Available: https://holobloc.com/
[83] S. Sendall and W. Kozaczynski, "Model transformation: The heart and soul of model-driven software development," *IEEE Softw.*, vol. 20, no. 5, pp. 42–45, Sep./Oct. 2003.
[84] G. Wimmel, "A BDD-based model checker for the PEP tool," Major Individual Project Report, Dept, 1997.

MIDHUN XAVIER (Graduate Student Member, IEEE) received the B.Tech. degree in electronics and communication engineering from MG University, Kottayam, India, in 2014, the master's degree in computer science from Indian Institute of Information Technology, NIT Trichy Campus, India, in 2017. He is currently working toward the Ph.D. degree in dependable communication and computation systems with Luleå University of Technology, Luleå, Sweden, with a major in formal verification and modeling of industrial automation systems using IEC 61499 Standard.

He is also an accomplished Software Engineer with 3 years of experience in data analytics and web3 development. He has worked with several esteemed organizations such as Uvionics Pvt. Ltd., TCS, and RCKR Software Pvt. Ltd. in India as a Software Engineer.

VICTOR DUBININ received the Diploma degree in computer engineering, the Ph.D. degree in computer engineering and the Dr.Sc. degree in computer science from the University of Penza, Penza, Russia, in 1981, 1989, and 2014, respectively.

From 1981 to 1989, he was a Researcher, from 1989 to 1995, he was a Senior Lecturer, and from 1995 to 2015, he was an Associate Professor with the University of Penza. Since 2015, he has been a Professor with the Department of Computer Science, University of Penza. He was a Visiting Researcher position with The University of Auckland, Auckland, New Zealand, and from 2013 to 2019, he was with the Luleå University of Technology, Luleå, Sweden. His research interests include formal methods for specification, verification, synthesis, and implementation of distributed and CPS. He was a recipient of DAAD-grants to work as a Guest Scientist with Martin-Luther-University Halle-Wittenberg, Halle, Germany, in 2003, 2006, and 2010, respectively.

SANDEEP PATIL (Member, IEEE) received the bachelor's degree in computer science engineering from the CMR Institute of Technology, Bangalore, India, in 2005, the master's of computer science in software engineering degree from the Illinois Institute of Technology, Chicago, IL, USA, in 2010, the master's of engineering studies (computer systems) degree from the University of Auckland, Auckland, New Zealand, in 2011, and the Ph.D. degree in formal verification of cyber-physical systems from the Lulea University of Technology, Lulea, Sweden.

His research interests include software engineering principles and methodologies in distributed industrial automation, especially using the IEC 61499 paradigm. He also works with formal verification techniques in the same application field. He is an accomplished software engineering professional with over 16 years of research and development experience in systems and application software, including 4 years as a Senior Software Engineer at Motorola India Pvt. Ltd., India.

VALERIY VYATKIN (Fellow, IEEE) received the Ph.D. degree from Taganrog Radio Engineering Institute, Taganrog, Russia and Nagoya Institute of Technology, Nagoya, Japan, in 1992 and 1999, respectively, and the Habilitation degree from Conferred by Ministry of Science and Education Saxony-Anhalt Country, Germany, in 2002.

He is currently on Joint Appointment as the Chaired Professor with the Luleå University of Technology, Luleå, Sweden, and a Full Professor with Aalto University, Helsinki, Finland. Previously, he was a Visiting Scholar with Cambridge University, Cambridge, U.K., and had permanent academic appointments with New Zealand, Germany, Japan, and Russia. His research interests include dependable distributed automation and industrial informatics, software engineering for industrial automation systems, artificial intelligence, distributed architectures, and multiagent systems applied in various industry sectors, including smart grid, material handling, building management systems, data centers, and reconfigurable manufacturing.

Dr. Vyatkin was a recipient of the Andrew P. Sage Award for the Best IEEE Transactions Paper in 2012. He has been the Chair of the IEEE IES Technical Committee on Industrial Informatics since 2016 and the Vice President of IES for Technical Activities for the term 2022–2025.