IEEE Open Journal of the
Industrial Electronics Society

Received 18 August 2023; accepted 25 September 2023. Date of publication 2 October 2023;
date of current version 16 October 2023. The review of this paper was arranged by Associate Editor Paulo Leito.

Digital Object Identifier 10.1109/OJIES.2023.3321084

Formal Verification of the Control Software of a Radioactive Material Remote Handling System, Based on IEC 61499

GIORDANO LILLI [1,2], MIDHUN XAVIER [3] (Student Member, IEEE), ETIENNE LE PRIOL [4], VINCENT PERRET [4], TATIANA LIAKH [3] (Member, IEEE), ROBERTO OBOE [1] (Fellow, IEEE), AND VALERIY VYATKIN [3,5] (Fellow, IEEE)

[1]Department of Management and Engineering, University of Padova, 36100 Vicenza, Italy
[2]Laboratori Nazionali di Legnaro, Istituto Nazionale di Fisica Nucleare, 35020 Legnaro, Italy
[3]Department of Computer Science, Electrical and Space Engineering, Luleå University of Technology, 971 87 Luleå, Sweden
[4]Department of Mechanical Engineering, École Normale Supérieure Paris-Saclay, 91190 Gif-sur-Yvette, France
[5]Department of Electrical Engineering and Automation, Aalto University, 02150 Espoo, Finland

CORRESPONDING AUTHOR: GIORDANO LILLI (e-mail: giordano.lilli@phd.unipd.it)

This work was supported in part by European Commission through the HORIZON 2020 project 1-SWARM under Grant 871743 and in part by the Horizon Europe project Zero-SWARM under Grant 101057083.

ABSTRACT Automation systems within nuclear laboratories are intended to work under harsh operating conditions. Selective Production of Exotic Species (SPES) is a nuclear research facility currently under construction by the Istituto Nazionale di Fisica Nucleare, dedicated to the production and study of radioactive ion beams. Isotopes are produced within the target ion source unit, a vacuum vessel that must be replaced on a regular basis. The highly radioactive environment necessitates the deployment of a set of automated systems dedicated to the unit's remote management. To meet high-level security standards, the design of such instrumentation and control systems must include extensive verification. Based on specific safety requirements, model checking can be used to assess the systems' correctness. This article describes how to employ an integrated toolchain to design, simulate, formally verify, and deploy the control software for the Horizontal Handling Machine, a safety-critical remote handling system in operation at SPES. The IEC 61499 standard's adoption led to a redesign of the control logic. Following a preliminary online simulation, the closed-loop system has been formally verified using the NuSMV symbolic model checker, with the help of the FB2SMV converter. In addition, the Function Blocks Modeling Environment tool was used for automating verification and analyzing counterexamples.

INDEX TERMS Formal verification, IEC 61499, isotope separation online (ISOL), model checking, NuSMV, radioactive ion beams (RIBs), remote handling, Selective Production of Exotic Species (SPES), simulation.

I. INTRODUCTION

Automation systems in nuclear laboratories must comply with strict safety requirements to avoid any potential risk to personnel or equipment. The critical operating environment generally discourages innovation in the design of control software, leading to an old-fashioned approach still today in the Industry 4.0 era. However, the introduction of distributed control systems based on modern standards would be advantageous for operational and safety challenges. This paradigm shift will lead to the development of smarter systems based on flexible and reconfigurable automation architectures. In this context, the evolution from applications based on IEC 61131-3 [1] toward IEC 61499 [2], [3] solutions would provide key tools to face the design and verification challenges typical of complex distributed control systems. The main advantages of this migration include:

1) flexible, reconfigurable, and scalable architecture;
2) modular design and standardized function blocks (FBs);

3) simulations and offline and online verification;
4) formal model checking techniques.

The Selective Production of Exotic Species (SPES) facility [4] can be considered an attractive use case to demonstrate the advantages of implementing safety-critical control systems based on IEC 61499. The Istituto Nazionale di Fisica Nucleare (INFN) is currently developing an experimental plant at Legnaro National Laboratories for multidisciplinary research on radioactive ion beams (RIBs) produced through the isotope separation online technique [5]. Isotopes are generated, as fission reaction products, from the collision of a high-energy proton beam (40 MeV, 200 μA) with a multifoil uranium carbide target [6] consisting of seven UCx disks [7]. The target ion source (TIS) unit [8] is identified as the core of the process. Here, the isotopes are produced and extracted for further studies in the field of nuclear physics and for medical applications. Unfortunately, aging of target and source materials requires regular replacement of the TIS unit to maintain high efficiency. This is difficult in view of the specific operational conditions. Indeed, the highly radioactive environment precludes any human operation. For this reason, the management of the TIS unit is entrusted to a remote handling framework [9], conceived to fulfill its specific life cycle. The safe operation and reliability of mobile robots operating in complex scientific facilities are essential features that need to be assessed [10]. In the SPES case, the automated systems involved in the replacement procedure face an intense radiation field generated by various contributions, such as the TIS unit [8], the residual front-end activation [11], and the isotopes deposition along the RIB line [12]. Operational safety is the outcome of an integrated strategy that combines the formal verification of control software with the deployment of inherently safe design principles to the hardware. In our work, we focused on the most critical remote handling task: the automated removal of a radioactive TIS unit from the SPES Front-End. The main objectives of the study are to demonstrate the benefits of the migration of IEC-61131-based software to an IEC 61499 architecture and to implement offline and online software verification techniques. This contribution describes the development of flexible and reconfigurable control software based on IEC 61499, along with its formal verification through an integrated toolchain, for a safety-critical remote handling system: the Horizontal Handling Machine (HHM) depicted in Fig. 1. The provided implementation demonstrates how to incorporate modular nondeterministic transitions (NDTs) in formal verification to improve the model's realism while limiting complexity.

The rest of this article is organized as follows. Section II discusses the related works and the problem statement. Section III explains the design and formal verification approach adopted for a radioactive material remote handling system. Sections IV and V describe the approach used to perform online simulations and formal verification, respectively. Section VI presents the main results of the work. Finally, Section VII concludes this article.

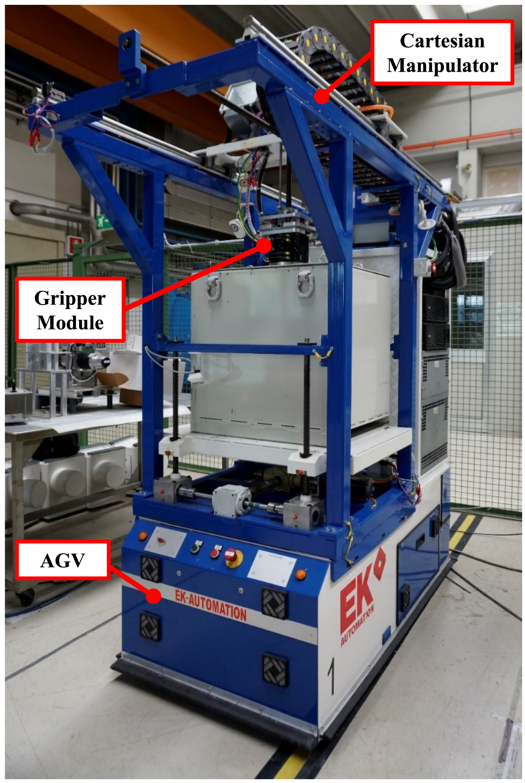

FIGURE 1. HHM is the primary remote handling vehicle in operation at the SPES facility.

II. RELATED WORKS AND PROBLEM STATEMENT

Designing control systems for industrial applications, particularly those involving nuclear-based materials, is of utmost importance. It is essential to adopt an approach that offers a flexible, portable, reconfigurable, and scalable architecture. In this context, the use of design patterns within the field of industrial cyber-physical systems (iCPS) employing the IEC 61499 standard emerges as a favorable strategy for designing such systems [13], [14]. These design patterns, originating from experienced system designers, hold significant value in the field of software engineering [15].

A. IEC 61499: A MODEL-DRIVEN APPROACH TO BUILD COMPLEX CONTROL SYSTEMS

IEC 61499 [2] is a standardized framework employed in the field of industrial automation for the modeling of distributed control systems [16]. The language's flexibility greatly contributed to its popularity for this type of projects. The use of advanced design patterns is crucial during the development of applications, which are defined as platform-independent models composed of modular components, in order to properly exploit this potential, achieving a high level of reusability, and ensuring greater reconfigurability of the underlying systems [17]. Christensen et al. [18] introduced a model-driven approach for distributed control systems, employing the model-view-control design pattern [19] within the context of IEC 61499 systems. The existing body of literature has

extensively investigated a variety of model-driven software design and engineering approaches that can be employed for the development of control systems, with a particular emphasis on their application to iCPS [14]. Bonfé et al. [20] examine the packaging industry practices and present design patterns specifically tailored for model-driven software design and implementation. Moreover, similar design patterns are proposed in [21], [22], and [23].

Safety of control software is an additional factor that becomes crucial in specific applications, such as laboratories dealing with nuclear-based materials. In these contexts, comprehensive testing is imperative to mitigate the risk of potentially catastrophic issues arising from even minor errors. The verification challenge of IEC 61499 has been acknowledged since the early stages of the standard's development and evaluation [24], [25]. To achieve the most thorough verification, closed-loop modeling has been suggested, necessitating the inclusion of plant modeling [26]. The implementation of a model-driven approach facilitates simulation in the loop [27], [28], enabling thorough testing and identification of system errors.

During the design process, simulation plays a crucial role in assessing the overall behavior of the control system, ensuring its compliance with expected outcomes and assisting in the process of virtual commissioning [29]. However, despite being advantageous in identifying flaws, this tool does not provide a guarantee of system reliability. Formal verification techniques have, thus, been proposed, as a promising approach to automatically verify the correctness and safety of automation systems. Specifically, the model checking integration in closed-loop verification [30] assists in the identification of design weaknesses in the model through the use of counterexamples. The closed-loop architecture, encompassing both the controller and the plant, depicts the overall behavior of the system. This design approach works successfully for both simulation and formal modeling applications [31].

B. FORMAL VERIFICATION TECHNIQUES FOR MODULAR INDUSTRIAL CONTROL SYSTEMS: CHALLENGES AND STRATEGIES

Model checking includes formal verification methods that are used as trustworthy tools to grant the correctness of instrumentation and control systems [32], [33], [34]. These techniques have been employed in a wide range of disciplines, including avionics [35], [36], automotive [37], [38], [39], and nuclear power plants [40], [41], [42], [43]. Despite their remarkable benefits for the validation of complex automation control logics, the computing requirements of model checking techniques may frequently represent a bottleneck in their use [44]. Several approaches have, thus, been proposed to reduce their computational complexity [45], [46], [47]. Furthermore, special care should be taken to guarantee that the system model's architecture reflects the behaviors of actual systems [48]. The entire verification procedure is completed in three phases. Creating a formal version of the actual system is the first step. At the second stage, the model and temporal logic specifications are fed into a verification tool. NuSMV [49] is a symbolic model checker based on binary decision diagrams (BBDs), whereas SPIN [50] is an explicit state software verification suite. In the third step, the tool reports whether or not the specification was met. A sequence of the model's states where the specification does not hold will also be provided, if possible, as a counterexample [51]. Unfortunately, despite the values of the model variables being included in each element of the sequence, counterexamples are unable to reveal their inherent dependencies or internal structure [52]. In an effort to make model checking more user-friendly, a number of visualization tools [53] have been created in recent years to assist users in understanding system behavior during specification violations and to identify the source of the problem [54], [55], [56]. This method's ultimate goal is to identify design flaws in the controller. In [57], a novel framework is presented for the design and validation of industrial automation systems using formal methods, specifically leveraging the IEC 61499 architecture and the automation object concept. The proposed approach enables the comprehensive process of designing, simulating, formally verifying, and deploying pick-and-place systems. Several technologies are available for formally modeling and validating industrial control systems [58]. The VEDA tool, for instance, is designed to support the formal verification of IEC 61499 systems within a closed-loop context. Although VEDA supports the modeling of the controller using a Petri net representation, representing the plant component requires manual effort. Recent research demonstrates the automatic generation of plant models from event logs [59], [60], simplifying the arduous task of constructing formal models for control engineers. Furthermore, interactive learning techniques have played a crucial role in enabling the automatic generation of controller FBs [61], thereby greatly facilitating the process of formal verification. The introduction of FB2SMV [62] enabled the creation of SMV models as formal representation derived from the IEC 61499 FBs within the system. This process allows us to take advantage of the great potential provided by NuSMV industry-grade model checker [49]. Given the introduced benefits, a significant effort has been made into including formal verification tools within the design process. The development of a seamless process that links engineering with verification was outlined in [63]. The toolchain includes an IEC-61499-compliant engineering environment, a converter for translating FBs into SMV code, the NuSMV model checker, and utilities for interpreting counterexamples. The presented approach aims to facilitate the design, simulation, formal verification, and distributed deployment of automation software for a cyber-physical system (CPS). To achieve this, a problem-oriented notation within the IEC 61499 syntax is suggested, enabling the creation of comprehensive closed-loop models. The proposed methodology addresses the challenge of verifying and analyzing FBs implemented in the IEC 61499 standard by providing a toolchain that supports continuous development and testing of distributed control

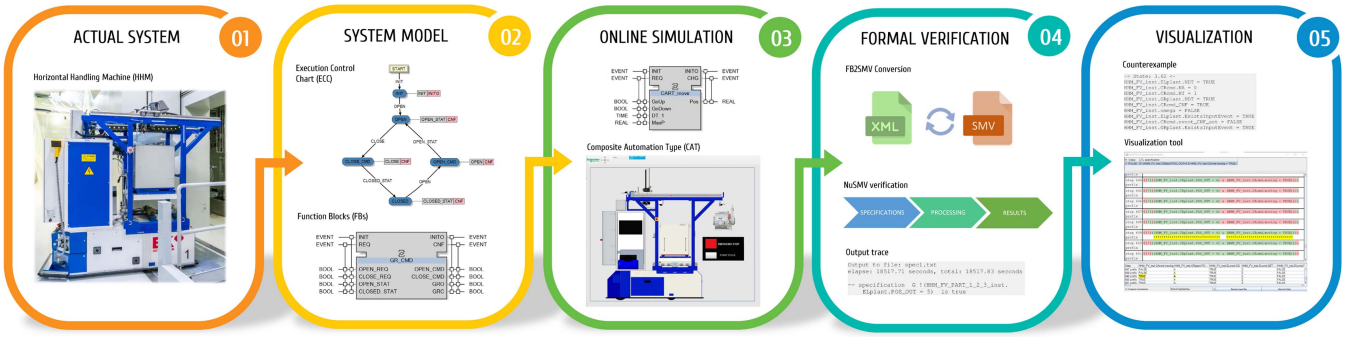

FIGURE 2. **Proposed workflow for the validation of safety-critical automation systems.**

systems. Liakh et al. [64] detail the creation of a model checking plug-in designed for IEC 61499 systems within the Function Blocks Modeling Environment (FBME) [65] graphical development environment. This plug-in automates various stages of the process, including the conversion of the system into a formal model, model checking, and the provision of visual explanations for counterexamples.

Despite the undeniable benefits introduced by the IEC-61499 standard, which provides a reference architecture and models for distributed control system development, the lack of established methodologies and theoretical foundation for iCPS poses a challenge for the development of new CPS applications. In particular, the integration of the aforementioned technologies into real industrial applications might be challenging due to the actual complexity of the solution and the time required for both the CPS implementation and verification. Indeed, while some examples for basic systems are provided in [63], it is still not clear whether the described techniques can be applied to complex CPS. The goal of this work thus, is to provide real-world strategies for developing modular applications, implementing automatic verification procedures, and reducing system complexity. In this article, we demonstrate how the described techniques can be applied in the refactoring and verification of a safety-critical control system. The following sections give a detailed explanation of each component of the implemented workflow, which is illustrated in Fig. 2. The development of a modular and portable system model, the reduction of verification complexity through the partial incorporation of NDTs, and the implementation of an automatic verification procedure are the fundamental novelties of the proposed solution.

III. CASE STUDY

An illustrative example is used to describe the entire formal verification process of a CPS, which includes the IEC 61499 software remodeling, the partial introduction of NDTs to conduct symbolic model checking under realistic conditions, and the visualization of counterexamples. The case study covered in this article is a safety-critical remote handling system used to transport and store radioactive material within a nuclear research facility. In this work, we propose the refactoring of an IEC 61131 control software with a new flexible and reconfigurable architecture based on the IEC 61499 standard. In addition, formal modeling and verification tools have been implemented to validate the effectiveness of the designed solution.

A. HORIZONTAL HANDLING MACHINE

The primary remote handling vehicle used to manipulate and transfer the TIS unit within the SPES target area is known as the HHM. This system, depicted in Fig. 1, enables the safe removal of an irradiated TIS unit from the SPES Front-End and transport to the temporary storage system, an automated storage rack designed to house up to 54 TIS units for long-term radioactive decay. Following the TIS unit removal, the HHM is employed to install a new TIS unit on the SPES Front-End in preparation for a new irradiation cycle. The machine consists of a Cartesian manipulator located on the top of an automated guided vehicle. While the vehicle allows movement between different areas, the manipulator is used for the collection of the TIS unit, its storage, and the upcoming installation. Here, brushless motors are used for the precise positioning of three linear axes. Two of them (*trolley* and *crane*) allow the TIS units to move along the longitudinal and vertical directions, respectively. The third (*elevator*) allows for the vertical movement of a shielded box, which is used to secure the TIS unit during transport. The manipulator's end-effector consists of a redundant pneumatic gripper able to engage both the TIS unit and the shielding box lid. The machine control software is based on IEC 61131 and runs on an onboard programmable logic controller (PLC) (Schneider Electric M340). Since its conception as a safety-critical automation system, the design of the HHM has incorporated inherently safe principles. In addition, the system has been assessed using specific probabilistic risk assessment techniques to evaluate the most severe failure scenarios and validate the implemented independent protection layers. In this context, software formal verification acts as a fundamental protection layer that can reduce the risk of system failure, potentially leading to unintended maintenance interventions in areas with a significant environmental dose rate. The HHM software logic supports multiple operating modes and motion sequences based on the type of remote handling task. Among the existing operational procedures, we focused on the most critical task: the removal of an

irradiated TIS unit and subsequent storage inside the shielding box during transport. During this procedure, the HHM is facing the SPES Front-End, and all actions are carried out by the Cartesian manipulator. The onboard PLC controls the sequence management, which includes the axes movements, the pneumatic gripper, and the reading of the various hard-wired signals from the limit switches demanded to detect the proper positioning of the radioactive TIS unit. This scenario has been considered as critical since a potential fault during the execution would necessitate a maintenance intervention under severe radiological conditions, leading to a significant personnel exposure.

The operation consists of the following steps.

1) The *trolley* initially moves ahead to pick up the TIS unit.
2) The *crane* descends, engages the TIS unit, and rises to the top positions.
3) The *trolley* moves to the middle position on top of the open shield box while holding the TIS unit.
4) The *crane* lowers the TIS unit, while the *elevator* raises the box. Once in position, the *gripper* releases the payload.
5) The manipulator finally closes the box with the lid.

The finite-state machine (FSM) is depicted in Fig. 3.

B. IEC 61499 IMPLEMENTATION

The HHM control software was initially designed in accordance with the IEC 61131-3 standard; an overview of the software section implementing the main state machine as a case structure is reported in Fig. 4. With the advancement of technology, the IEC 61499 introduction suggested a complete code refactoring in the direction of a more modern, modular, and flexible architecture, where it would be possible to change the behavior of the system by acting on a single FB. The remodeled application of the HHM control logic was developed using the EcoStruxure Automation Expert tool. The software architecture is built on FBs linked to Moore-type FSMs known as execution control charts (ECCs) [66]. An overview of the global composite FB model is available in Fig. 5. One of the many benefits provided by the IEC 61499 refactoring, aside from supporting formal verification, is the introduction of a modular, standardized, and reusable architecture for the development of FBs. This strategy results in improved code organization and the potential to "certify" the behavior of the FBs, thus reducing the verification complexity in subsequent applications. In addition, the existing IEC 61131 design, which is based on structured text (ST), incorporates global variables within the program to track the program execution. Since the software's behavior is not always evident, this poses a serious concern. In contrast, IEC 61499 provides for the explicit specification of the dependencies and interactions between different FBs. The *elevator*, *trolley*, *crane*, and *gripper* are the key actuation groups employed in this application. Each of these mechatronic systems, which work together to securely encase the TIS unit in the shielding box, is supervised by a dedicated controller. The following sections provide a detailed description of the main FBs.

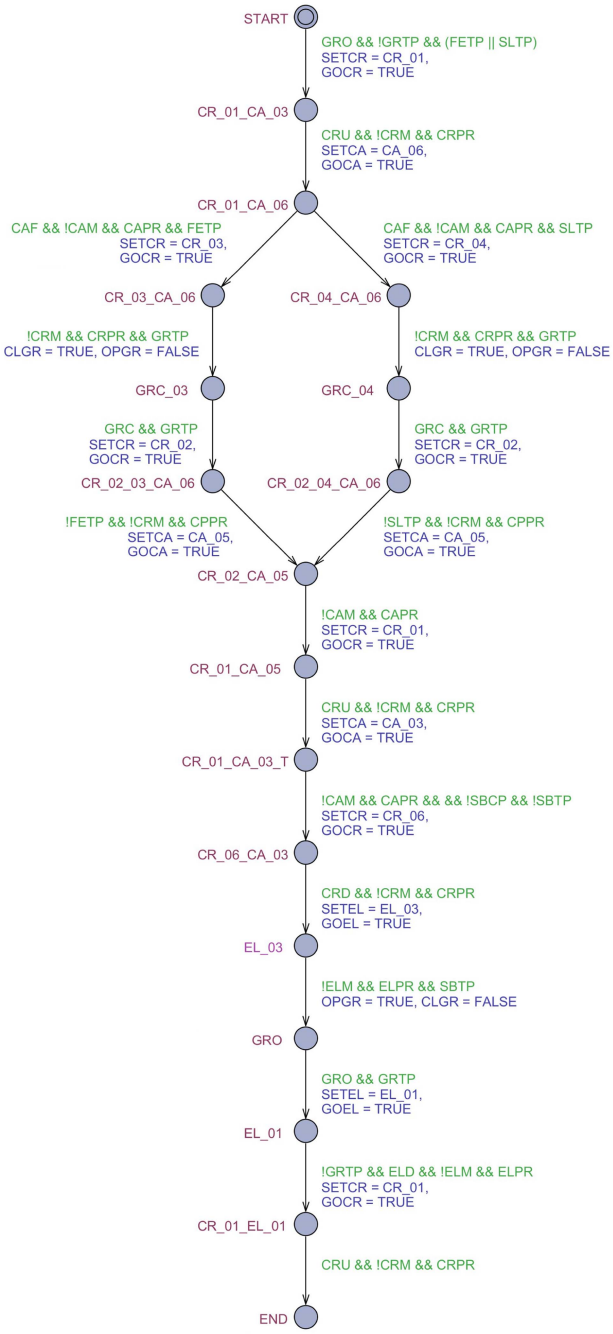

FIGURE 3. TIS unit pick-up sequence's finite state machine.

1) LINEAR MOTION AXES

A standardized pair of controller and plant FBs can be used to conceptually model the three linear axes. Using a modular and reusable strategy, the development work can be significantly decreased. In addition, it makes it possible for the system to be easily reconfigured in order to achieve alternative capabilities in the future. The core FB `AXE_CMD`, which implements an absolute positioning control system, is shared by the three linear axes. The FB and the correspondent ECC are displayed in Fig. 6. The plant FB precisely sets the axis according to the

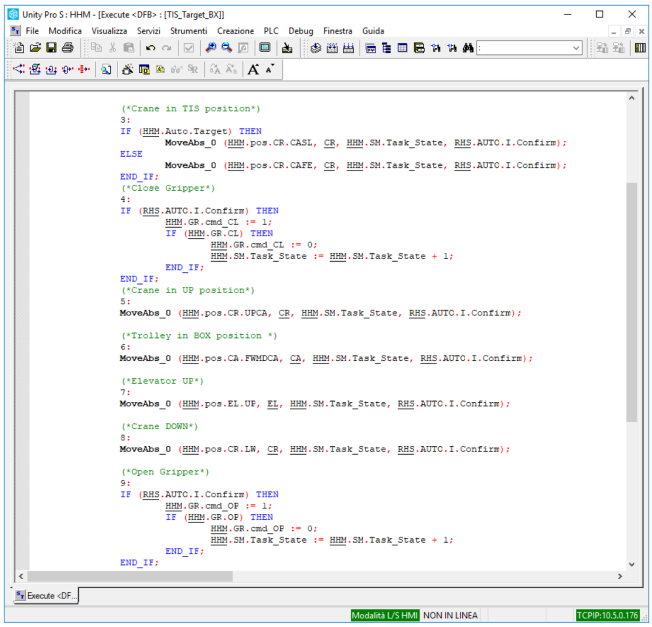

FIGURE 4. Original HHM control program, based on IEC 61131-3 structured text.

destination coordinates and provides the POS_REACHED signal to the controller once the motion is completed. The given target position directs the AXE_CMD to the preset coordinates. The FB acknowledges its arrival and stops it once it reaches the designated spot. A visual representation of the *elevator* linear motion axis is reported as an example in Fig. 7.

2) GRIPPER

The operating mode of the HHM pneumatic *gripper* differs from the abovementioned systems due to its inherent discrete logic. *Gripper* CLOSE or OPEN commands are processed when the REQ event is triggered. The FB provides two output signals to indicate when the relevant "closed" or "open" state has been reached.

3) SEQUENCE CONTROLLER

The SEQUENCE FB manages the integration of the various subsystems and the overall HHM behavior throughout the execution of the remote handling sequence. The precise list of tasks is defined within the correspondent ECC. Each FSM state is associated with a set of actions carried out by a specific algorithm. Motion actions are started by setting the desired position for a specific axis and sending the GO command to the appropriate controller. The reception of the POS_REACHED command from the plant FB causes the transition to the next state. In our case study, the sequence controller FB implements a state machine that refers to a single HHM task: the TIS unit pickup sequence. This sequence has been examined as a representative example. The system's adaptable architecture will make it possible to incorporate more motion sequences in the future by updating a single FB.

4) SUPPORT FUNCTION BLOCKS

The INIT FB initializes the system and prepares it to perform the desired procedure at the start of software execution. The user can then choose between manual and automatic HHM operating modes by using the TRIGGER and MODE_SELECTION FBs. While the first allows the user to direct the HHM behavior, the automatic mode forces the system to stick to the Sequence controller's state machine logic. The ESTOP FB, as the last support FB, offers the ability to stop the execution at any time. This feature protects the system from internal or external failure caused by unfavorable conditions.

IV. SIMULATION MODEL

IEC 61499 applications can typically be tested using dynamic (online) or static (offline) techniques. In order to ensure safety in a system that has already been deployed and is in use, the first group of techniques seeks to monitor it in its operating state. Conversely, offline safety measures are meant to reduce fault risk at the design stage and test the system before use [67]. In our work, we focused on offline verification methods aimed at fault removal. This process can be accomplished at the designed stage using formal verification tools or online testing techniques. Software simulation involves feeding the program with input sequences that replicate the behavior of the actual system and determining whether or not the program's outputs comply with specific requirements. The adopted development suite includes a native human–machine interface (HMI), which may be used as a command center and to simulate system execution. Composite automation types (CATs) were used to model a range of mechatronic components for the simulated plant. This feature facilitates testing of the system's simulation behavior in a common environment because CATs can be directly linked to both HMI objects and FBs. Inputs were used to link the controller FBs to the relevant CAT blocks, replicating the real-world behavior of the mechatronic components in the system. The HHM representation implemented in the HMI is shown in Fig. 8, where the three linear motion axes are linked to distinct CAT blocks. Each axis plant FB is connected to a dedicated AXE_CMD controller, which selects the desired position set point from a predefined pool of coordinates and triggers the motion request. In response to the controller's inputs, the plant block validates the coordinates, performs the movement, and acknowledges its arrival at the predetermined location. The axis motion may be stopped at any time by activating a STOP input event. The GR_CMD FB opens or closes the clamp based on the input signal from the controller. In order to interlock the option of releasing the payload only in particular positions, the GR_CMD is additionally provided with the axes' actual positions.

We should emphasize that the HMI CATs provide a more accurate representation of the system behavior when compared to the axis plant FBs discussed in Section III. While in the basic implementation, the plant FB will only trigger the POS_REACHED signal after an arbitrary time, here, an integrator simulates the linear axis movement and sends the

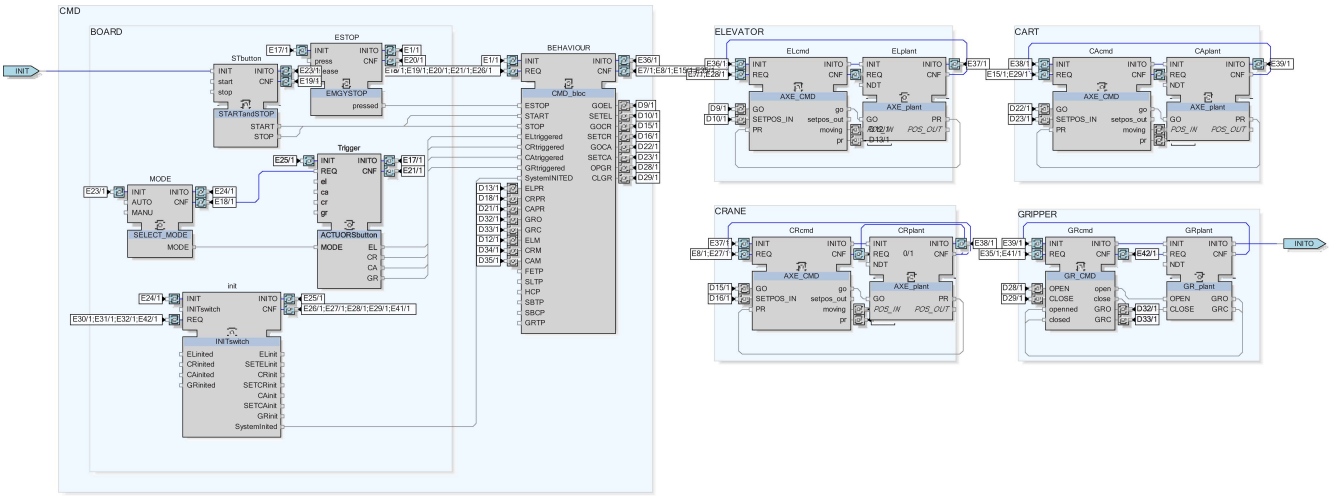

FIGURE 5. IEC 61499 global composite FB of the HHM model.

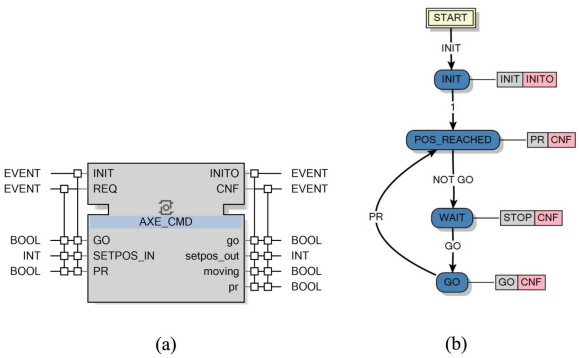

FIGURE 6. Overview of the controllers dedicated to the HHM linear motion axes and gripper. (a) AXE_CMD FB. (b) AXE_CMD ECC.

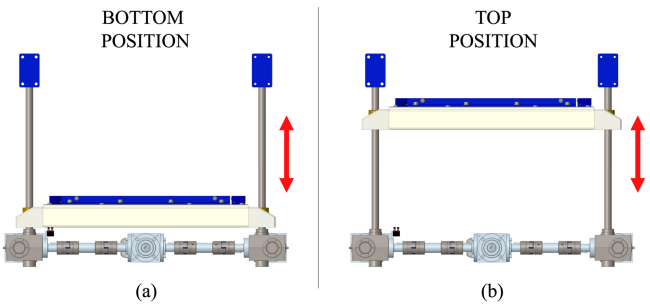

FIGURE 7. Visual representation of the elevator linear motion axis. (a) Bottom position. (b) Top position.

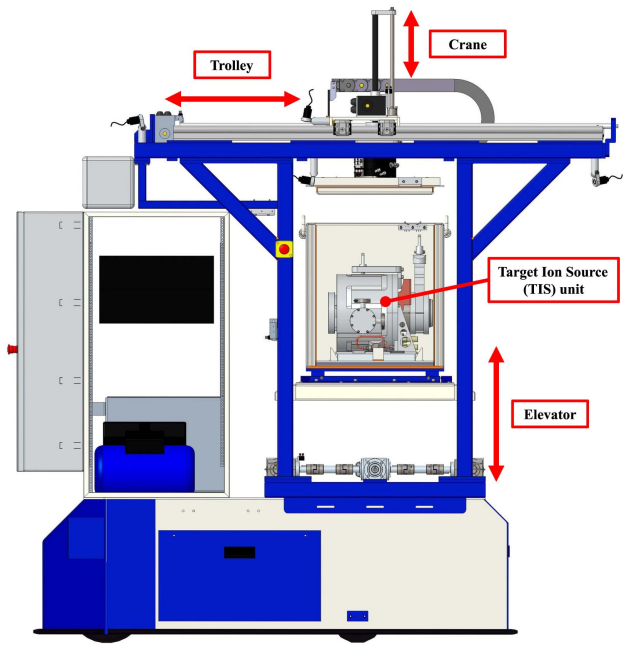

FIGURE 8. Graphical representation of the HHM CAT used in the HMI for online monitoring and simulations.

actual position coordinates to the correspondent object in the HMI allowing the user to follow the motion while it is being executed. Further debugging tools, such as runtime monitoring blocks, can also be employed to detect specific critical conditions. The software's modularity allows for the independent and concurrent development of the controller and plant FBs. Each FB will be initially tested and debugged with the aid of custom mock-up blocks. As they reach maturity, they can then be interconnected to run the simulation. After the verification, the final stage will be to replace the simulation's plant FB with the actual system.

Unfortunately, simulations often cannot explore all possible paths due to the huge size of state automata representing industrial control software. This bottleneck makes them insufficient as an exhaustive verification method since it prevents conclusive verification of program behavior in a reasonable amount of time. Furthermore, the quality of the output is also influenced by the automation engineer's knowledge and experience in selecting pertinent testing sequences that may correspond to typical dangerous circumstances of the controlled process [68]. To address these problems, formal

TABLE 1. Description of the LTL Specifications Verified With NuSMV in the HHM Model

No	Property	Comment
1	`G !(ELplant.POS_OUT = 5)`	The *elevator* plant FB must never reach the *error* state in any of the sequence elements.
2	`G !(CAplant.POS_OUT = 5)`	The *trolley* plant FB must never reach the *error* state in any of the sequence elements.
3	`G !(CRplant.POS_OUT = 5)`	The *crane* plant FB must never reach the *error* state in any of the sequence elements.
4	`G !(CRplant.POS_OUT in (2..4) & CAcmd.moving = TRUE)`	The *crane* must always be in the top position while the *trolley* is moving to prevent mechanical collisions.
5	`G !(ELplant.POS_OUT = 2 & CRplant.POS_OUT = 4 & CAcmd.moving = TRUE)`	To avoid mechanical collisions, the *trolley* must not move while the HHM is lowering the TIS unit inside the HHM shielding box.
6	`G !(ELplant.POS_OUT = 1 & CRplant.POS_OUT = 4 & GRplant.GRO = TRUE)`	The pneumatic *gripper* shouldn't open until the *elevator* is not in the top position, even if the *crane* is in the lower position.

verification techniques have been established, which provide methods for closed-loop (plant and controller) model checking able to analyze a program in its entirety.

V. FORMAL VERIFICATION

The formal verification of finite-state systems, such as closed-loop control algorithms, has been effectively accomplished in the last ten years thanks to symbolic model checking based on BDDs. These tools have been developed in the past to overcome the state explosion problem in finite automata [47]. Model checking is the process of exploring the reachable states of a model, which is described as an FSM, in order to validate temporal logic specifications. When a property is violated, the tool provides a counterexample in the form of a sequence of states [46]. As previously mentioned, the most well-known open-source model detection tools among the available solutions are NuSMV and SPIN. In particular, because of its extensive core capabilities and good scalability, NuSMV is frequently used for reliability and security verification of industrial designs [69]. This tool supports the representation of synchronous and asynchronous finite-state systems, and it allows for the verification of both linear temporal logic (LTL) and computation tree logic (CTL) specifications using implicit methods. In more detail, it compares a model against a property using a symbolic representation of the specification [70].

Accurate modeling of the real system is essential in order to validate the intended behavior of the device and detect potentially undesirable states. This enables simulation and verification of the apparatus prior to its actual operation. Since the model is an abstraction, it may not include all relevant characteristics of the real-world system or the context in which it is embedded. Hence, a condensed version of the plant FB can be used to create a reduced formal model, which can then be verified utilizing symbolic model checking techniques thanks to the NuSMV tool. As an illustration, in the presented use case, the `AXE_CMD` FB only takes into account the beginning, intermediate, and final states rather than the motion dynamic considered in the real system. This approximation is still acceptable because the goal at this stage is to assess the possible blockage within two locations instead of the specific stop positioning.

Table 1 describes a collection of LTL expressions that have been developed to identify potential critical problems. Specifications 1–3 are meant to ensure that none of the three linear motion axis plants enters the error state during system execution. On the other hand, requirements 4 and 5 deal with potential collision detection. More in detail, the first verifies that *trolley* movements are inhibited when the *crane* is not fully raised in the top position, and the second focuses on the system configuration occurring while positioning the TIS unit within the shielding box. Similarly to the last scenario, the *trolley* must not move, while the *elevator* is raised and the *crane* is lowered. Specification 6 aims to confirm that the *gripper* only opens in a specific location: when the TIS unit is lowered within the box (*elevator* up and *crane* down). A batch script, detailed in Listing 1 in the Appendix, has been developed to examine all the aforementioned requirements with NuSMV and log data.

The LTL specifications were evaluated in two different scenarios to test the reliability of the formal verification. As described in Section III, the `SEQUENCE` FB implements an FSM where the HHM axis movements are executed sequentially to prevent any potential collision. If we specifically consider state `GRC_03_GRC_04` in Fig. 9, which corresponds to the TIS unit picked up by the HHM Cartesian manipulator, the subsequent path toward the shielding box shall be carried out in three distinct steps: 1) backward movement of the *trolley* axis; 2) lowering of the *crane* axis; and 3) rising of the *elevator*. The described motion sequence and the correspondent states are visible in Fig. 9(b). In our research, we deliberately induced a design flaw in the control software to determine if NuSMV was able to identify it. Specifically, we updated the main FSM to launch the previously mentioned actions in a parallel execution, with the three motion axes

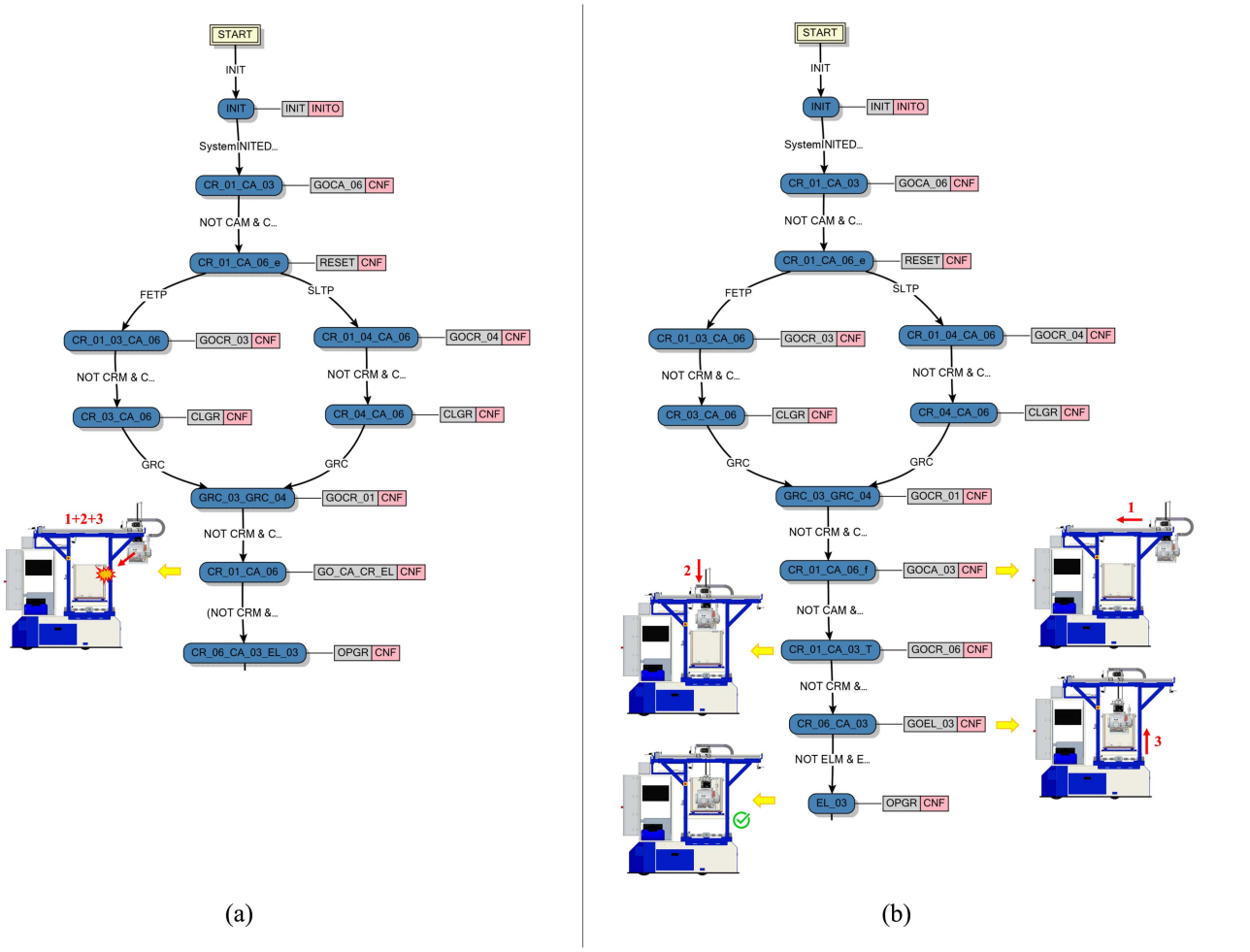

FIGURE 9. Comparison of the two investigated control sequences. (a) ECC of the controller FB implementing the parallel movement of the three linear axes. (b) ECC in which the three movements are executed sequentially.

moving simultaneously, as shown in Fig. 9(a). Since it does not always result in a fault condition, this type of design error is particularly difficult to identify through conventional simulations. The relative motion axes speeds do, in fact, affect the likelihood of a collision. This implies that we may be able to perform multiple simulations without observing any failure event. The following section discusses how adding nondeterminism to the model can make it more realistic by taking into account the impact of nonidealities found in the real world and allowing for the early identification of potential system defects. With regard to the test under discussion, NDTs within axes plant FBs seek to change the amount of time required to get the `POS_REACHED` signal, directly impacting the relative speed between concurrent axis movements. The violation of LTL specification 5 in Table 1 allows for the detection of the collision occurrence.

A. DISCRETE-STATE PLANT MODELING IN FBS WITH NDTS

The original system, created to exploit the visualization and online verification capabilities provided by EcoStruxure Automation Expert, needs to be reduced and adapted in order to apply formal verification methods. The *elevator*, *trolley*, and *crane* components were modeled in this study by a simplified FB that embodies the intended behavior of the actual system while omitting the features used for visualization. As opposed to the simulation scenario where each component was modeled using a single FB, this global plant model has been implemented to capture the behavior of the three linear motion axes collectively. Thus, by discretizing the plant model's FB while maintaining its functional capabilities, the original complex model can be reduced to a simpler representation. The `AXE_PLANT` component features two data inputs (`GO` and `POS_IN`) and two data outputs (`POS_REACHED` and `POS_OUT`). The system may simulate real-world behavior using the NDT event's random signal emission, which enables the discovery of previously undetected faults using CTL or LTL specifications. As an example, Fig. 10 depicts a potential scenario in which an NDT has been introduced in the ECC associated with *elevator* plant FB. In this case, the plant enters the `GO` state upon receiving the controller's `GO` signal, and following the NDT event, it reaches the `END` state. The physical meaning of this NDT is that the transition

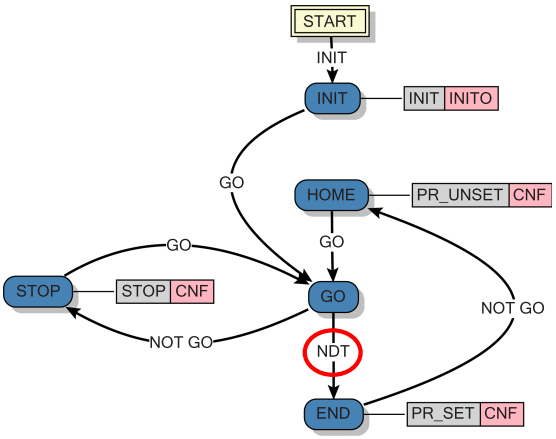

FIGURE 10. Example of injection of an NDT within the elevator plant model.

between the GO and END states, i.e., the axis motion towards a given position, might take an unspecified amount of time. If a NOT_GO signal is generated while the plant is in the GO state, it enters the STOP state and remains there until another GO signal is activated. In the END state, the plant notifies the controller that the task has been completed by setting the value of POS_REACHED signal to TRUE. Following the deactivation of the GO signal, the plant returns to the HOME state.

The *gripper* plant model features two data inputs, OPEN and CLOSE, and two data outputs, GRO and GRC. The model initially enters the OPENING state when the controller sets OPEN to TRUE. Second, it switches to the OPEN state in response to an NDT signal. Similarly, when the controller sets CLOSE to TRUE, the plant reaches the CLOSING state and, following a random time delay caused by the NDT, enters the CLOSED state. If the CLOSE command is activated during the OPENING state, the model transitions to the CLOSING state. If the OPEN command is activated during the CLOSING state, the plant returns to the OPENING state and awaits for the emission of the NDT signal. The discrete-state model of the HHM was converted into an SMV model using the FB2SMV tool. Subsequently, the verification has been carried out by NuSMV, using an Intel core i7-10510U CPU@1.80 GHz 2.30 GHz with 32-Gb RAM. In an effort to mitigate the state-space explosion problem, NDTs have gradually been introduced into different sections of the model according to the scenarios in Table 2. The progressive integration of NDTs might be viewed as a feature of the proposed toolchain. While it is true that critical faults might occur as a result of multiple nondeterministic conditions acting simultaneously, in a first verification stage, distinct blocks can be assessed independently while maintaining the execution time within reasonable limits.

B. TRACE ANALYSIS USING FBME

In our work, we used FBME [65], enhanced with trace visualization and in-depth analysis capabilities, to examine

TABLE 2. NDT Scenarios Analyzed in the Study: NDTs are Progressively Included in the Model

No.	Scenario
1	NDT in *elevator* plant
2	NDT in *trolley* plant
3	NDT in *crane* plant
4	NDT in *gripper* plant
5	NDT in *elevator* and *trolley* plants
6	NDT in *elevator*, *trolley* and *crane* plants
7	NDT in *elevator*, *trolley*, *crane* and *gripper* plants

counterexamples and find the causes of violations of requirements.

One of the nontrivial tasks when using formal verification is to analyze the resulting counterexample. Verifiers frequently offer an output trail in a text format that is inconvenient and confusing to the user. As a result, the user has to make additional efforts to analyze the counterexample. The IEC 61499 code presents additional challenges due to the automatic generation of the model for the verifier, which leads the counterexample to utilize the notations of the input model. Given an output trace, the following natural step is to determine the location of the error and investigate the underlying causes. The nonimperative nature of IEC 61499 further complicates this process. Moreover, current common IEC 61499 software development tools do not provide in-depth trace analysis capabilities.

FBME is an integrated development environment (IDE) for IEC 61499 applications, currently under active development at the Luleå University of Technology (LTU). FBME is a cross-platform, open-source, modular IDE that is based on IntelliJ IDEA and the meta programming system (MPS) [71]. The MPS provides powerful tools for developing custom domain-specific languages and also provides a platform for creating custom IDEs. Modularity and extensibility are key features of FBME, so the LTU has also extended FBME's functionality by adding enhanced capabilities to the visualization and automatic analysis of IEC 61499 program execution traces and counterexamples. The functionality to automatically call NuSMV and generate a model for verification was also seamlessly integrated into FBME.

The trace analysis in FBME is shown in Fig. 11. Trace is stored in unified format [64] and can be obtained from different sources: either from a simulation of the verifier model, as a counterexample, or from the actual execution of the IEC 61499 program. The entire trace history is displayed on panel ①. The user selects the trace step of interest and can examine the state of the system. The values of all variables, message counters and other data are also displayed on the diagram itself (panel ②), where changes that have occurred in the current step are highlighted. In addition to clear visualization of the trace, FBME uses powerful techniques for

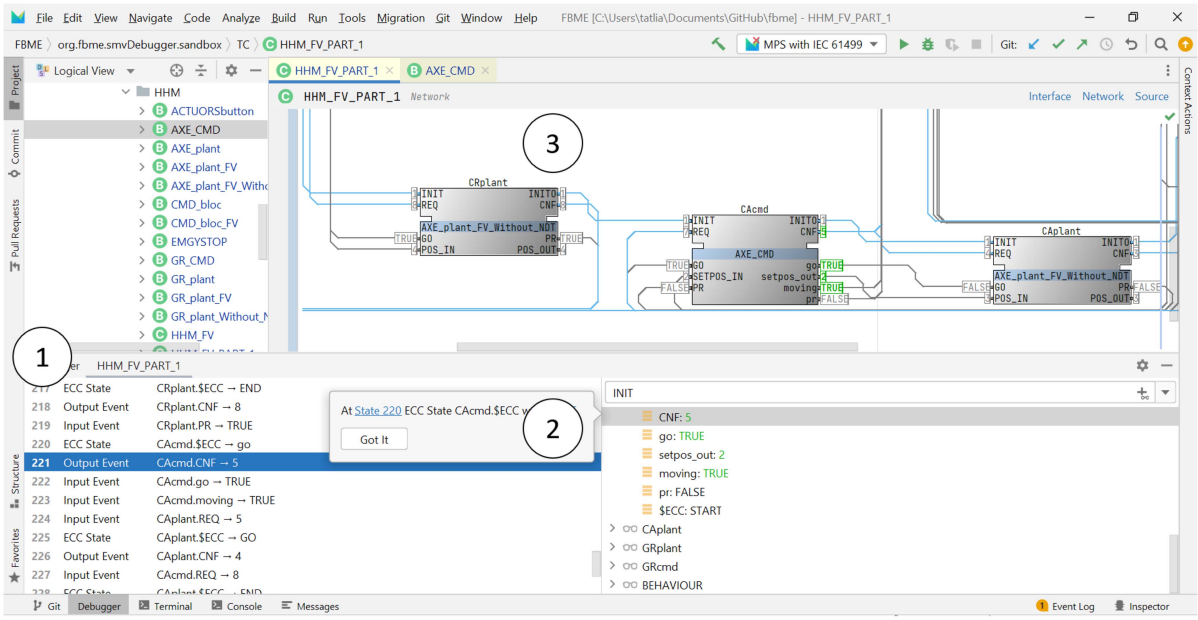

FIGURE 11. **FBME trace analysis.**

its in-depth analysis. By examining the preceding code, this method enables the user to visually locate the section of the code that resulted in the problem. Visual explanation allows us to establish causal relationships between different trace events (note that the term "event" here is used in a broader sense, as a change in system state, rather than an event in IEC 61499 terms). For instance, it might be possible to determine whether a variable modification, an IEC 61499 event emission, or a change in an ECC state produced a specific event. In this way, the cause of the violation of a requirement can be identified. An example of the result of visual explanation technique is available in window ③.

VI. RESULTS AND DISCUSSION

The initial phase of the project, which followed the software remodeling based on IEC 61499, was devoted to validating the model by launching various simulations directly within the EcoStruxure Automation Expert suite. This was achieved through the use of CATs, which allow FBs to be directly linked to HMI objects. The application model and the HMI have been developed independently. Once sufficiently stable, the HMI plant FB was connected to the controller FBs, replacing the existing simplified version of plant FBs. Launching the online simulation, the user can monitor the sequence execution. The software will begin in the initial state, progress through specific checkpoints, and eventually reach the final state. Unfortunately, even if the simulation does not report any errors, this merely indicates that there exists a path where it crosses all the checkpoints. Hence, using symbolic model checking tools will provide a more thorough level of investigation. Prior to the verification procedure, it is crucial to verify the accuracy of the formal model. This can be accomplished by simulating the model in NuSMV, where various paths and random states are explored. The simulation assists in demonstrating that the model properly covers all the ECC states of the behavioral FB by tracing the path of ECC states. It also helps to confirm that the generated formal model behaves in accordance with the discrete-state model by providing information about the values of all the variables in each state. The NuSMV simulation technique can detect changes in the ECC and their impact on system behavior. Initially, using this method, it will be possible to confirm that all paths leading from the beginning to the end will pass through the crucial checkpoint. As the second step, this assertion needs to be proven even in the presence of nondeterminism. Indeed, the introduction of NDTs may have resulted in the inclusion of certain additional pathways in the application, and this is reflected in a larger state space with multiple routes. In contrast to simulation, where we can test only one scenario, NDTs allow us to evaluate several possibilities. The evidence that the given specifications are validated in all of these paths will, thus, extend the results of the online simulation. The six formulated properties have been checked using a batch script that reads the supplied SMV model and performs the verification, logging both the execution time and result for each specification. The quantity of memory needed to store and manipulate BDDs is the primary limitation of model checking methods. In light of this, the proposed implementation allows for the gradual integration of NDTs into the model. This stepwise approach provides better control over the model and allows for faster specification analysis. The time required for NuSMV to execute the formal verification of all the described LTL specifications while altering the number of NDTs is depicted in Fig. 12. It is evident that the gradual inclusion of NDTs resulted in a global increase in execution time. Because of the ample state space, it is feasible that with a larger number of

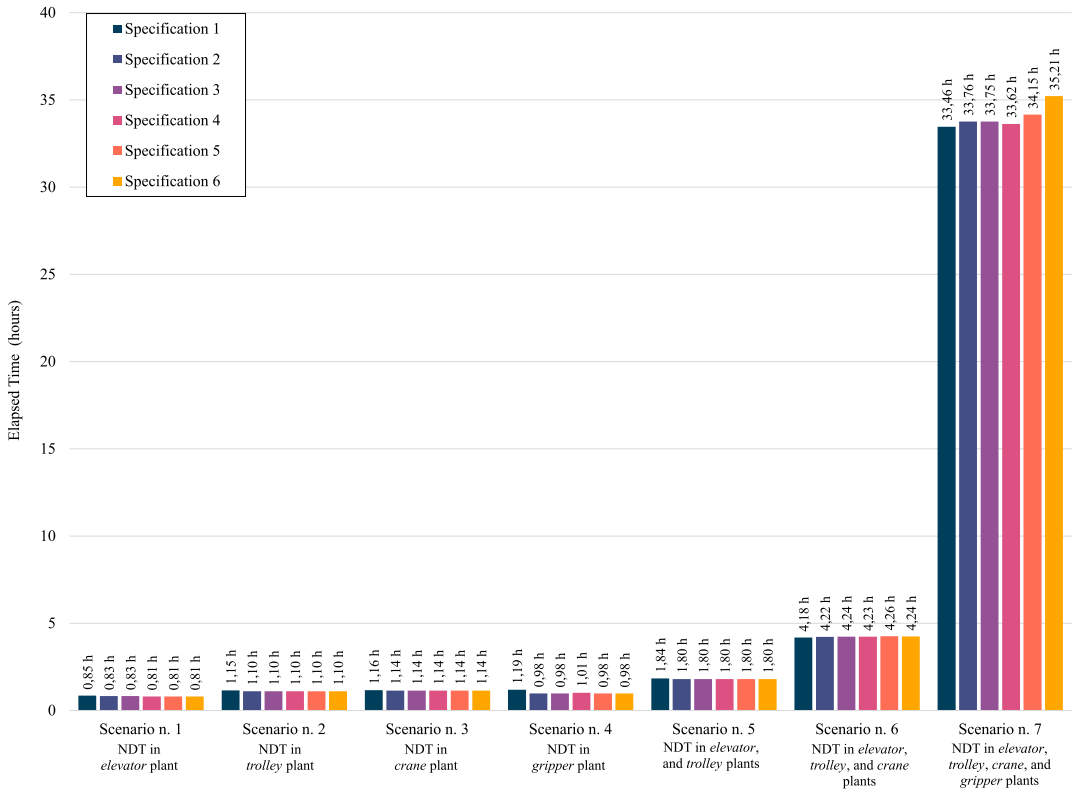

FIGURE 12. **Execution time required by NuSMV in different NDTs' configuration.**

NDTs, global verification of all pathways will fail. Reducing the number of NDT points in this situation may be a viable option for squeezing the state space to a tolerable size and then gradually increasing it. Bounded model checking is an alternate strategy that searches for a counterexample in executions whose length is constrained by some number k. If no bug is discovered, k is increased until either a bug is discovered, the problem becomes unmanageable, or some predetermined upper bound is reached [46]. A key feature of the described engineering framework is the ability to govern nondeterminism. NDTs can be injected into specific locations to perform formal verification in a particular configuration. This method allows us to validate the automation system under particular stress conditions. As discussed in Section V, the IEC 61499 application was formally verified following the purposeful introduction of a design fault that might potentially lead to a collision occurrence. Despite the difficulties in identifying this failure condition using conventional simulations, NuSMV was able to successfully accomplish this task, thus providing a counterexample that demonstrates the violation of LTL property 5 in Table 1. In the case under study, the amount of time needed for the formal verification was comparable with what was required for the same LTL expression in Scenario 7 (see Fig. 12). However, it is difficult to formulate a generic statement because the duration depends on the particular paths that lead to the failure conditions. The evidence of the violation is provided by NuSMV in the form of a failure trace, which

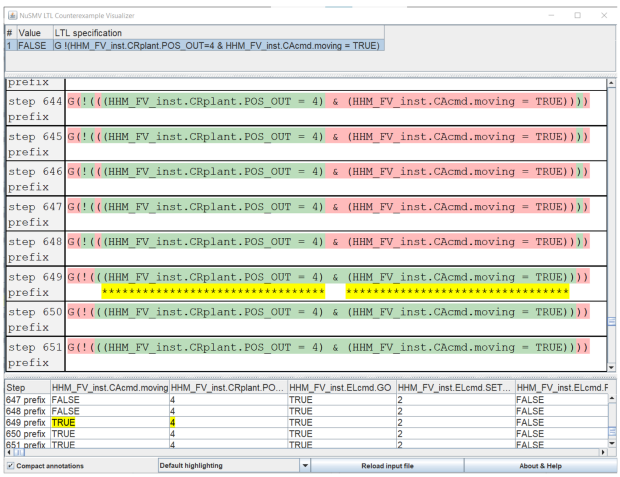

FIGURE 13. **Graphical visualization of the counterexample trace produced by NuSMV when an LTL specification is violated.**

depicts a state sequence of system model transitions where the specification is not met. Fig. 13 shows how, through the use of specific visualization tools [53], it would be possible to decode the output trace and examine the path that led to the violation. This result is of great significance as it showcases how the presented set of tools can be employed in the verification of complex safety-critical control systems, enabling the early detection of potential failure conditions that would

be extremely difficult to spot through traditional simulation and testing techniques.

VII. CONCLUSION AND FUTURE WORK

In this article, we showed how to use an integrated toolchain for the analysis and verification of the control software for a real safety-critical automated system employed in the transport and storage of radioactive material in a nuclear research facility. The provided use case was intended to demonstrate the actual feasibility of integrating the phases of modeling, simulation, verification, and analysis in a complex system using an automatic procedure. The study benefited from the software redesign based on the IEC 61499 standard for several kinds of reasons. First, it enabled the optimization of code structure by defining standardized, modular, and reusable FBs based on specific ECCs. Second, it allowed for the explicit specification of the relationships and dependencies between FBs while eliminating the incorporation of global variables. Third, it supported the translation of the code into an SMV model, thereby enabling formal verification of LTL safety specifications. Finally, the incorporation of NDTs within different FBs facilitated the simulation of sequence execution under realistic conditions. The developed IEC 61499 solution's portability promotes the system to be integrated into various toolchains. In the proposed example, we investigated this feature by combining it with FB2SMV and FBME for the verification of a set of LTL safety specifications. While the first tool is used to extract the software formal model, model verification is subsequently carried out using NuSMV. FBME, on the other hand, is a comprehensive tool, capable of automating the entire verification process by incorporating automatic model generation, NuSMV verification, visualization, and analysis of counterexample trace. The suggested toolchain can be instrumental in the early identification of design flaws that could result in potential mechanical collisions. The presented results emphasize the validity of the toolchain by demonstrating the benefits of formal system verification in detecting nontrivial design errors that may result in a failure event under specific circumstances. A key feature of the proposed solution, in addition to modularity and portability, is the deep control over localized NDT introduction. This capability can be effective in reducing the process complexity, permitting independent testing of specific FBs, and keeping the time required by model checking within reasonable limits. One limitation of the presented methodology resides in the accuracy with which the IEC 61499 model represents the actual system. Indeed, the necessity for mitigating the state explosion problem ultimately led to the adoption of a simplified design, especially with regard to plant FBs. Ensuring a high level of accuracy between the model and its real-world equivalent is crucial during this phase. Furthermore, in the provided use case, we investigated a single, albeit critically important, remote handling procedure. Further developments will allow the software model to be expanded to include more system motion sequences and plant details, thus finalizing the development of a digital twin of the primary SPES remote handling system.

APPENDIX NUSMV SCRIPT

```
time
read_model -i HHM_FV_PART_1.smv
flatten_hierarchy
encode_variables
build_model
time
check_ltlspec -p "G !(HHM_FV_PART_1_inst.ELplant.
    POS_OUT = 5)" -o spec1.txt
time
check_ltlspec -p "G !(HHM_FV_PART_1_inst.CAplant.
    POS_OUT = 5)" -o spec2.txt
time
check_ltlspec -p "G !(HHM_FV_PART_1_inst.CRplant.
    POS_OUT = 5)" -o spec3.txt
time
check_ltlspec -p "G  !(HHM_FV_PART_1_inst.CRplant.
    POS_OUT in (2..4) & HHM_FV_PART_1_inst.CAcmd.
    moving = TRUE)" -o spec4.txt
time
check_ltlspec -p "G  !(HHM_FV_PART_1_inst.ELplant.
    POS_OUT = 2 & HHM_FV_PART_1 _inst.CRplant.
    POS_OUT = 4 & HHM_FV_PART_1_inst.CAcmd.moving
    = TRUE)" -o spec5.txt
time
check_ltlspec -p "G  !(HHM_FV_PART_1_inst.ELplant.
    POS_OUT = 1 & HHM_FV_PART_1_inst.CRplant.
    POS_OUT = 4 & HHM_FV_PART_1_inst.GRplant.GRO =
     TRUE)" -o spec6.txt
time
```

LISTING 1. Batch script used to check the LTL specifications with NuSMV.

REFERENCES

[1] *Programmable Controllers. Part 3: Programming Languages*, Standard IEC 61131-3:2013, Int. Electrotech. Commission, Geneva, Switzerland, 2013.

[2] *Function Blocks—Part 1: Architecture, Second Edition*, Standard IEC 61499, Int. Electrotech. Commission, Geneva, Switzerland, 2012.

[3] A. Zoitl and R. Lewis, *Modelling Control Systems Using IEC 61499*, vol. 95, 2nd ed. London, U.K.: IET, 2014.

[4] T. Marchi et al., "The SPES facility at Legnaro National Laboratories," *J. Phys.: Conf. Ser.*, vol. 1643, no. 1, 2020, Art. no. 12036.

[5] A. Andrighetto et al., "SPES: An intense source of neutron-rich radioactive beams at Legnaro," *J. Phys.: Conf. Ser.*, vol. 966, no. 1, 2018, Art. no. 012028.

[6] A. Andrighetto et al., "The ISOLPHARM project: ISOL-based production of radionuclides for medical applications," *J. Radioanalytical Nucl. Chem.*, vol. 322, no. 1, pp. 73–77, Oct. 2019.

[7] S. Corradetti et al., "The SPES target production and characterization," *Nucl. Instrum. Methods Phys. Res., B: Beam Interact. Mater. At.*, vol. 488, pp. 12–22, Feb. 2021.

[8] A. Monetti et al., "The RIB production target for the SPES project," *Eur. Phys. J. A*, vol. 51, no. 10, pp. 1–11, Oct. 2015.

[9] G. Lilli, L. Centofante, M. Manzolaro, A. Monetti, R. Oboe, and A. Andrighetto, "Remote handling systems for the selective production of exotic species (SPES) facility," *Nucl. Eng. Technol.*, vol. 55, pp. 378–390, Jan. 2023.

[10] D. I. Khan, S. Virtanen, P. Bonnal, and A. K. Verma, "Functional failure modes cause-consequence logic suited for mobile robots used at scientific facilities," *Rel. Eng. Syst. Saf.*, vol. 129, pp. 10–18, Sep. 2014.

[11] A. Donzella et al., "Residual activation of the SPES front-end system: A comparative study between the MCNPX and FLUKA codes," *Eur. Phys. J. A*, vol. 56, no. 2, pp. 1–13, Feb. 2020.

[12] L. Centofante et al., "Study of the radioactive contamination of the ion source complex in the selective production of exotic species (SPES) facility," *Rev. Sci. Instrum.*, vol. 92, no. 5, May 2021, Art. no. 53304.

[13] 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.

[14] S. Patil, D. Drozdov, and V. Vyatkin, "Adapting software design patterns to develop reusable IEC 61499 function block applications," in *Proc. IEEE 16th Int. Conf. Ind. Inform.*, 2018, pp. 725–732.

[15] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. München, Germany: Pearson Deutschland GmbH, 1995.

[16] 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 J. Ind. Electron. Soc.*, vol. 2, pp. 169–183, 2021.

[17] L. Sonnleithner, B. Wiesmayr, V. Ashiwal, and A. Zoitl, "IEC 61499 distributed design patterns," in *Proc. IEEE Int. Conf. Emerg. Technol. Factory Autom.*, 2021, pp. 1–8.

[18] J. H. Christensen, *Design Patterns for Systems Engineering With IEC 61499*. Magdeburg, Germany: Otto-Von-Guericke-Universitaet, 2000.

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

[20] M. Bonfé, C. Fantuzzi, and C. Secchi, "Design patterns for model-based automation software design and implementation," *Control Eng. Pract.*, vol. 21, no. 11, pp. 1608–1619, Nov. 2013.

[21] G. Čengić, O. Ljungkrantz, and K. Åkesson, "A framework for component based distributed control software development using IEC 61499," in *Proc. IEEE Int. Conf. Emerg. Technol. Factory Autom.*, 2006, pp. 782–789.

[22] V. Vyatkin, S. Karras, and T. Pfeiffer, "Architecture for automation system development based on IEC61499 standard," in *Proc. IEEE 3rd Int. Conf. Ind. Inform.*, 2005, pp. 13–18.

[23] R. Hametner, A. Zoitl, and M. Semo, "Automation component architecture for the efficient development of industrial automation systems," in *Proc. IEEE Int. Conf. Autom. Sci. Eng.*, 2010, pp. 156–161.

[24] V. Vyatkin and H. M. Hanisch, "A modeling approach for verification of IEC1499 function blocks using net condition/event systems," in *Proc. IEEE Symp. Emerg. Technol. Factory Autom.*, 1999, pp. 261–270.

[25] 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, Jan. 2009.

[26] 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, Appl. Rev.*, vol. 39, no. 1, pp. 17–28, Jan. 2009.

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

[28] C. H. Yang and V. Vyatkin, "Transformation of simulink models to IEC 61499 function blocks for verification of distributed control systems," *Control Eng. Pract.*, vol. 20, no. 12, pp. 1259–1269, Dec. 2012.

[29] N. Galkin, M. Ruchkin, V. Vyatkin, C. W. Yang, and V. Dubinin, "Automatic generation of data centre digital twins for virtual commissioning of their automation systems," *IEEE Access*, vol. 11, pp. 4633–4644, 2023.

[30] 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.

[31] 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.

[32] E. M. Clarke, O. Grumberg, D. Kroening, D. Peled, and H. Veith, *Model Checking*. Cambridge, MA, USA:MIT Press, 1999.

[33] K. Schneider, *Verification of Reactive Systems, Formal Methods and Algorithms*. Berlin, Germany: Springer, 2004.

[34] C. Baier and J.-P. Katoen, *Principles of Model Checking*. Cambridge, MA, USA: MIT Press, 2008.

[35] G. E. Gelman, K. M. Feigh, and J. Rushby, "Example of a complementary use of model checking and agent-based simulation," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2013, pp. 900–905.

[36] H. Wang, D. Zhong, and T. Zhao, "Avionics system failure analysis and verification based on model checking," *Eng. Failure Anal.*, vol. 105, pp. 373–385, Nov. 2019.

[37] V. Todorov, F. Boulanger, and S. Taha, "Formal verification of automotive embedded software," in *Proc. 6th Int. FME Workshop Formal Methods Softw. Eng.*, 2018, pp. 84–87.

[38] J. H. Kim, K. G. Larsen, B. Nielsen, M. Mikučionis, and P. Olsen, "Formal analysis and testing of real-time automotive systems using UPPAAL tools," in *Proc. 20th Int. Workshop Formal Methods Ind. Crit. Syst.*, 2015, pp. 47–61.

[39] P. Filipovikj, N. Mahmud, R. Marinescu, C. Seceleanu, O. Ljungkrantz, and H. Lönn, "Simulink to UPPAAL statistical model checker: Analyzing automotive industrial systems," in *Int. Symp. Formal Methods*, 2016, pp. 748–756.

[40] A. Pakonen, I. Buzhinsky, and K. Björkman, "Model checking reveals design issues leading to spurious actuation of nuclear instrumentation and control systems," *Rel. Eng. Syst. Saf.*, vol. 205, Jan. 2021, Art. no. 107237.

[41] E. Jee et al., "FBDverifier: Interactive and visual analysis of counterexample in formal verification of function block diagram," *J. Res. Pract. Inf. Technol.*, vol. 42, no. 3, pp. 171–188, 2010.

[42] E. Németh and T. Bartha, "Formal verification of safety functions by reinterpretation of functional block based specifications," in *Proc. Int. Workshop Formal Methods Ind. Crit. Syst.*, 2009, pp. 199–214.

[43] B. F. Adiego et al., "Applying model checking to industrial-sized PLC programs," *IEEE Trans. Ind. Informat.*, vol. 11, no. 6, pp. 1400–1410, Dec. 2015.

[44] I. Buzhinsky and A. Pakonen, "Symmetry breaking in model checking of fault-tolerant nuclear instrumentation and control systems," *IEEE Access*, vol. 8, pp. 197684–197694, 2020.

[45] A. Cimatti and A. Griggio, "Software model checking via IC3," in *Proc. 24th Int. Conf. Comput. Aided Verification*, 2012, pp. 277–293.

[46] A. Biere, A. Cimatti, E. M. Clarke, O. Strichman, and Y. Zhu, "Bounded model checking," *Adv. Comput.*, vol. 58, 2003, Art. no. 117.

[47] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang, "Symbolic model checking: 1020 states and beyond," *Inf. Comput.*, vol. 98, no. 2, pp. 142–170, Jun. 1992.

[48] L. C. Cordeiro, E. B. de L. Filho, and I. V. Bessa, "Survey on automated symbolic verification and its application for synthesising cyber-physical systems," *IET Cyber-Phys. Syst.: Theory Appl.*, vol. 5, no. 1, pp. 1–24, Mar. 2020.

[49] A. Cimatti et al., "NuSMV 2: An opensource tool for symbolic model checking," in *Proc. 14th Int. Conf. Comput. Aided Verification*, 2002, pp. 359–364.

[50] G. J. Holzmann, "The model checker SPIN," *IEEE Trans. Softw. Eng.*, vol. 23, no. 5, pp. 279–295, Mar. 1997.

[51] I. Beer, S. Ben-David, H. Chockler, A. Orni, and R. Trefler, "Explaining counterexamples using causality," *Formal Methods Syst. Des.*, vol. 40, no. 1, pp. 20–40, Feb. 2012.

[52] P. Ovsiannikova, I. Buzhinsky, A. Pakonen, and V. Vyatkin, "Oeritte: User-friendly counterexample explanation for model checking," *IEEE Access*, vol. 9, pp. 61383–61397, 2021.

[53] A. Pakonen, I. Buzhinsky, and V. Vyatkin, "Counterexample visualization and explanation for function block diagrams," in *Proc. IEEE 16th Int. Conf. Ind. Informat.*, 2018, pp. 747–753.

[54] K. Loer and M. D. Harrison, "An integrated framework for the analysis of dependable interactive systems (IFADIS): Its tool support and evaluation," *Autom. Softw. Eng.*, vol. 13, no. 4, pp. 469–496, Oct. 2006.

[55] T. Bochot, P. Virelizier, H. Waeselynck, and V. Wiels, "Paths to property violation: A structural approach for analyzing counterexamples," in *Proc. IEEE Int. Symp. High Assurance Syst. Eng.*, 2010, pp. 74–83.

[56] S. Patil, V. Vyatkin, and C. Pang, "Counterexample-guided simulation framework for formal verification of flexible automation systems," in *Proc. IEEE Int. Conf. Ind. Inform.*, 2015, pp. 1192–1197.

[57] V. Vyatkin, "IEC 61499 as enabler of distributed and intelligent automation: State-of-the-art review," *IEEE Trans. Ind. Inform.*, vol. 7, no. 4, pp. 768–781, Nov. 2011.

[58] V. Vyatkin and H. M. Hanisch, "Formal modeling and verification in the software engineering framework of IEC61499: A way to self-verifying systems," in *Proc. IEEE Int. Conf. Emerg. Technol. Factory Autom.*, 2001, pp. 113–118.

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

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

[61] 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. IEEE Int. Conf. Emerg. Technol. Factory Autom.*, 2022, pp. 1–7.

[62] D. Drozdov, "FB2SMV: IEC 61499 function blocks XML code to SMV converter," 2014. [Online]. Available: https://github.com/dmitrydrozdov/fb2smv

[63] M. Xavier, S. Patil, and V. Vyatkin, "Cyber-physical automation systems modelling with IEC 61499 for their formal verification," in *Proc. IEEE Int. Conf. Ind. Informat.*, 2021, pp. 1–6.
[64] T. Liakh, R. Sorokin, D. Akifev, S. Patil, and V. Vyatkin, "Formal model of IEC 61499 execution trace in FBME IDE," in *Proc. IEEE Int. Conf. Ind. Informat.*, 2022, pp. 588–593.
[65] "Function blocks modelling environment (FBME)," 2020. [Online]. Available: https://github.com/JetBrains/fbme
[66] E. A. Lee and S. A. Seshia, *Introduction to Embedded Systems—A Cyber-Physical Systems Approach*. Cambridge, MA, USA: MIT Press, 2017.
[67] P. Ovsiannikova and V. Vyatkin, "Towards user-friendly model checking of IEC 61499 systems with counterexample explanation," in *Proc. IEEE 26th Int. Conf. Emerg. Technol. Factory Autom.*, 2021, pp. 01–04.
[68] 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. 210–215.
[69] Z. Xu, D. Zhong, W. Li, H. Huang, and Y. Sun, "Formal verification of dynamic hybrid systems: A NuSMV-based model checking approach," in *Proc. ITM Web Conf.*, vol. 17, 2018, Art. no. 03026.
[70] M. Frappier, B. Fraikin, R. Chossart, R. Chane-Yack-Fa, and M. Ouenzar, "Comparison of model checking tools for information systems," in *Proc. 12th Int. Conf. Formal Eng. Methods Formal Methods Softw. Eng.*, 2010, pp. 581–596.
[71] "JetBrains MPS–Meta programming system," 2020. [Online]. Available: https://www.jetbrains.com/mps

GIORDANO LILLI was born in 1990. He received the B.Sc. and M.Sc. degrees in mechatronic engineering in 2012 and 2015, respectively, from the University of Padova, Vicenza, Italy, where he is currently working toward the Ph.D. degree in mechatronics and product innovation engineering.

In 2016, he joined the three-year Fellowship Program with the European Organization for Nuclear Research (CERN), Geneva, Switzerland, as an Automation Control Engineer and worked on the commissioning of safety-critical robots at the ISOLDE and MEDICIS facilities. Since 2019, he has also been responsible for the development of the remote handling systems for the Selective Production of Exotic Species Facility, Legnaro National Laboratories, Istituto Nazionale di Fisica Nucleare, Legnaro, Italy.

MIDHUN XAVIER (Student Member, IEEE) received the B.Tech. degree in electronics and communication engineering from the SCMS School of Engineering and Technology, Kochi, India, in 2014, and the master's degree in computer science from the Indian Institute of Information Technology, Trichy, India, in 2017. He is currently working toward the Ph.D. degree with the 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 three years of experience in data analytics and web application 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.

ETIENNE LE PRIOL was born in Paris, France, in 2000. He received the University Diploma of Technology in mechanical engineering from the University Institute of Technology of Cachan, in 2020.

In 2022, he decided to pursue manufacturing studies involving automation lectures with École Normale Supérieure Paris-Saclay, Gif-sur-Yvette, France. In the summer of the same year, he began automation research in IEC 61499 with Aalto University, Espoo, Finland. In 2023 he became a Mechanical Engineering professor after passing the National Competitive examination "Agrégation".

VINCENT PERRET was born in Dijon, France, in 1999. He is working toward the master's degree in pedagogy applied to higher education with the Department of Mechanical Engineering, École Normale Supérieure Paris-Saclay, Gif-sur-Yvette, France.

In 2021, he studied mechanical engineering and automation with École Normale Supérieure Paris-Saclay. In 2022, he joined Aalto University, Espoo, Finland, as an intern to study the IEC61499 standard and made research about formal verification and monitors.

TATIANA LIAKH (Member, IEEE) received the master's degree in automation of physical and technical research from the Department of Physics, Novosibirsk State University, Novosibirsk, Russia, in 2013, and the Ph.D. degree in engineering (mathematical modeling, numerical methods, and program complexes) from the Institute of Automation and Electrometry of the Siberian Branch of the Russian Academy of Sciences (IA&E SB RAS), in 2021.

Her thesis title was "Dynamic verification of process-oriented control programs for cyber-physical systems." Since 2021, she has been a Postdoctoral Researcher with the Luleå University of Technology, Luleå, Sweden. Her research interests include cyber-physical systems, control software, domain-specific language tools, verification, robotics, industrial automation, and IEC 61499.

ROBERTO OBOE (Fellow, IEEE) was born in Lonigo, Italy, in 1963. He received the Laurea degree (*cum laude*) in electrical engineering and the Ph.D. degree in industrial electronics and informatics from the University of Padova, Padova, Italy, in 1988 and 1992, respectively.

He is currently an Associate Professor of Automatic Control with the Department of Management and Engineering, University of Padova, Vicenza, Italy. His research interests include the fields of motion control, applied digital control, telerobotics, haptic devices, rehabilitation robots, and applications of microelectromechanical systems to motion control.

Dr. Oboe is a Co-Editor-in-Chief for IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS and IEEE OPEN JOURNAL OF THE INDUSTRIAL ELECTRONICS SOCIETY.

VALERIY VYATKIN (Fellow, IEEE) received the Ph.D. degree in applied computer science from Taganrog State University of Radio Engineering, Taganrog, Russia, in 1992, the Dr. Eng. degree in electrical engineering from the Nagoya Institute of Technology, Nagoya, Japan, in 1999, and the Habilitation degree from the Ministry of Science and Technology of Sachsen-Anhalt, Magdeburg, Germany, in 2002.

He is currently the Chaired Professor with the Luleå University of Technology, Luleå, Sweden, and a Full Professor with Aalto University, Espoo, Finland. Previously, he was a Visiting Scholar with the University of Cambridge, Cambridge, U.K., and had permanent academic appointments in 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 Technical Committee on Industrial Informatics of the IEEE Industrial Electronics Society (IES) since 2016 and the Vice-President of the IEEE IES for technical activities for the term 2022–2023.