Cyber-Physical Systems with Process-Oriented Paradigm

Subjects: Computer Science, Software Engineering | Computer Science, Hardware & Architecture

Contributor: Vladimir E. Zyubin , Natalia O. Garanina , Igor S. Anureev , Sergey M. Staroletov

Process Oriented Programming (POP) is a programming paradigm based on the concept of "processes" that determine the response (outputs) of a reactive system based on events (states of internal and external variables, timeouts). A process-oriented program is a set of interacting processes that can change their behavior (reaction to events), start and stop other processes, and be executed in parallel. The languages created within the framework of POP (e.g. Reflex, IndustrialC, poST) fit very well for specification various control algorithms, which can consist of hundreds and thousands of processes. In addition to the fact that process-oriented programs structurally and conceptually correspond to the technological description of the plant under control, they also have the following remarkable property. In process-oriented algorithms, the use of data is usually local, limited to one or more processes.

distributed control systems
control software
process-oriented programming
cyber-physical production system

1. Introduction and Motivation

Cyber-physical systems (CPSs) are usually defined as network systems with a computational core that interacts with the physical world through sensors and actuators. Some researchers insist on adding the network architecture to the definition. This remark is confirmed by the widespread use of distributed architectures for the implementation of cyber-physical systems. For example, a modern car control system can include up to a hundred microcontrollers connected to the CAN bus [1]. The distributed architecture of the vehicle control system provides a reduction in the length and weight of connecting wires, ease of maintenance, eliminates the mechanical or hydraulic implementation of various functions, thereby minimizing the weight and size characteristics of the system as a whole, and also makes it possible to implement various control strategies [2][3][4]. Due to these circumstances, software development has a significant part in the costs of creating new and redesigning existing control systems. Usually, software development costs account for more than half of all expenses [5].

Cyber-physical systems (CPSs) are being actively developed as part of the Industry 4.0 concept. According to the concept, which declares the shift from mass production to mass customization, the CPSs must meet requirements such as modularity, adaptability, reusability, and flexibility. The class of cyber-physical systems that integrate physical and virtual components in the automation of production processes is called cyber-physical production systems (CPPSs). IEC 61131-3 languages and a later superstructure over them—IEC 61499 function blocks—are positioned as the main means of specifying the algorithms for the functioning of CPPSs. These approaches are seriously criticized by specialists for their inability to respond to the challenges that arise in the development of industrial automation systems and, in fact, for their inability to solve the problems for which these languages were created: to ensure portability, reconfigurability, interoperability, and the distribution of the developed software [6]. The developers of the IEC 61131-3/IEC 61499 specifications themselves agree with the conclusions about the standard's inconsistency with the declared goals [7].

The problems of existing approaches lead to the fact that, in practice, a module-oriented (device-centric) approach is used [8], which involves the separate programming of each controller of the control system. In particular, this is caused by the current "one function per module" paradigm used in the development of control system components in accordance with the V-cycle technology [9]. The main disadvantages of this approach are the complexity of software maintenance, its readability and modification, flexibility, and the complexity of verification, which is crucial for most cyber-physical systems. Based on the current problems of programming cyber-physical systems, the need for the development of the topologically independent (application-centric) programming of distributed control systems was stated in [9]. The indisputable advantage of this concept is the possibility of a simple and, in the most preferable case, sequential description of the control algorithm, independent from the type of nodes, their number, and the topology of the distributed control system. Topology-free programming could provide a drastic reduction in the cost of developing and maintaining the control software being created. The implementation of the concept involves creating a toolchain for the design and development of the control algorithm: first of all, automatically dividing the algorithm into parts and deploying them on controllers. The concept weeds out the tedious and error-prone manual process of dividing it into parallel (more precisely, independent or loosely connected) parts. Although the quality of automatic parallelization has improved over the past few decades, the fully automatic parallelization of sequential programs by means of compilers remains a significant challenge due to the need for complex program analysis and unknown factors (such as input range) at compilation time [10]. This gives rise to serious problems associated with the automatic generation of executable code. Automatic parallelization by compilers or tools is very difficult due to different types of race conditions. Dependency analysis is not easy for code that uses indirection, pointers, recursion, and indirect function calls. In addition, access to shared or global resources can lead to bottlenecks, resource starvation, or deadlocks [11]. The main direction for increasing the comprehensibility and reducing the complexity of the verification of a system of concurrent processes is to ensure the determinism of its execution [12].

These circumstances are typical for the field of parallel programming of computational algorithms. In terms of control algorithms, the complexity of the problem can be reduced due to the following features:

- First of all, in the field of automation, parallelism is an integral part of the control algorithm. This circumstance is reflected in so-called process-oriented languages; for example, in the recently developed poST language, which is a process-oriented dialect of the IEC 61131-3 Structured Text [13]. In the poST language, a program is built as a set of weakly dependent processes—FSM-like structures. Therefore, the process-oriented specification of the control algorithm contains information about its possible parallelization.

- Secondly, in contrast to parallel programming goals, the need for parallelization is primarily caused not by the desire to reduce the computation time but by the desire to reduce the cost of the system as a whole and the cost of its maintenance by using smaller, less expensive, and more flexible microprocessors, reducing the length of wires, reducing the complexity of wiring, and improving maintainability, providing the opportunity to implement various control strategies [3][14].

- Thirdly, for control problems, an extremely important circumstance is the possibility of formally verifying the algorithms being created. For the process-oriented paradigm, researchers developed the mathematical model of hyperprocesses. Informally, a hyperprocess involves expanding the processes in the system into one process by periodically turning on the logical processes in the program according to the round robin strategy to execute the code in their current states. A number of formal methods for the process-oriented programming languages have been developed [15][16] based on the semantics of a hyperprocess. Correspondingly, when developing a methodology for the topologically independent programming of control systems based on a process-oriented approach, it is desirable to reuse the existing verification methods.

2. Distributed Control Systems for Vehicles

Probably the most well-known area of the application of distributed control today is the automotive domain. Here, the associated data transmission lines are used to connect the main electronic control unit (ECU) of the car to child ECUs and sensors/actuators. Based on this approach, other systems in different domains are being developed for cases where there is a sufficiently large number of interacting devices and it is required to minimize the number of wires. Since the 90s of the last century, the advent of injection technology and the implementation of closed-loop engine control algorithms have led to the need to take into account the data and synchronization of a large number of signals from sensors connected at different places in the system. Thus, the need for distributed control was recognized early enough in the automotive industry and a number of connectivity protocols were developed to enable in-vehicle network communication. Nowadays, these protocols have been merged into a de facto standard for the in-vehicle controller area network (CAN) [17]. A CAN bus has been successfully integrated into the industry for data exchange between controllers in a distributed system with a distance of several meters.

In [18], the authors explored CAN messages from a vehicular peripheral bus of a real car. It was found that the peripheral bus is disconnected from the main bus, where data are transmitted in the exchange of devices for monitoring engine parameters, is less high-speed, and transmits data between the radio, heating installation, display, and control knobs. There are several identifiers and data bytes are transmitted for each identifier with some given periods. From such experiments, the distributed control approach used in industry today can be stated as: (1) controllers transmit periodic messages to each other about the state of key system variables (encoded information received from closely connected sensors and derived information based on the processing of this data); (2) more important information is transmitted at shorter intervals; (3) periods are calculated so that there is not a large number of collisions during simultaneous transmission; (4) the data network is essentially distributed into independent subnets that do not interfere with each other when transmitting heterogeneous information (as in the example, the engine ECU does not need to know about the radio station currently playing). In the existing literature, such a control algorithm is called “time-triggered control” [19].

In [20], Albert et al. compared the applications of the time-triggered and event-triggered approaches in automotive distributed control systems using the CAN bus and its TTCAN variant (time-triggered CAN) [21]. It is noted that systems whose subsystems’ operation is pre-calculated by time windows are predictable; their components can be interchanged from different vendors, with the requirements that the maximum operating times are not violated. However, such systems cannot properly respond to important (system critical state) messages because the latter can only be processed after bus arbitration. On the other hand, fully event-driven systems can process messages with the desired priorities; however, due to non-determinism, when changing system components to others (for example, faster or slower), the control algorithm must be completely revised. It is concluded that it is necessary to use the event-triggered approach only for a few of the most important messages and to include their processing in a special time window before everyone else, while the processing of the normal messages should be performed using time-triggered control. In this case, it is possible to calculate the change in the control law of the object, and it becomes predictable.

Thus, when planning control algorithms, the delays associated with the bus should be calculated and simulated. For instance, the work by Baek et al. [22] considers the theory and practice of creating a controller for an autonomous all-terrain vehicle using the CAN bus. Using a simple fixed-priority bus model [23], the worst-case response time was calculated and the transit times and loss percentages between key system components were measured. Taking into account the obtained information, a simulation was carried out on the controller model based on control theory and it was concluded that the implementation is built correctly, taking into account the specified delays and losses. The same has been carried out in numerous research projects; for example [24][25].

Of course, if the controllers are placed side by side, it makes no sense to use the CAN bus. For example, Shiau et al. explored the possibility of designing a distributed multi-MCU-based flight control system for unmanned aerial vehicles [26]. The system was built on low-cost microcontrollers. For inter-node communication, the UART interface was used. The distributed system is able to calculate complex non-linear Kalman filters and calculate the attitude angle using base elements with a low performance. The system also provides for error handling when receiving sensor data. The quality of the resulting design was verified using MATLAB.

3. Design of Distributed Control Systems Based on IEC 61131-3/61499

Christensen in [27] suggested and adopted the model–view–control (MVC) design pattern to the domain of industrial automation and integrated it with the IEC 61499 standard architecture. According to the adopted pattern, control software is organized from two components connected in a closed loop: a controller, implementing a set of control operations available as published services, and a model, simulating the plant. Vyatkin et al. in [28] extended the approach to include the formal verification of function block systems, appropriate for more rigorous verification by means of model checking.

Thramboulidis in [29] adopted a system-based approach for the development of industrial automation systems (IASs). Based on the proposed methodology, the UML software model of the system was obtained by extracting it from a formal system model and was further developed into implementation code. The approach presented by the author enables the development of control systems from diverse perspectives to meet the demands of CPSs. The author provided evidence of the need for such an approach by demonstrating the simultaneous engineering of various components of the system, including mechanical, electrical, and software parts. This approach offers the advantage of integrating the relationships between the physical and software aspects of components in CPS, thereby improving the quality of IAS development.

In [30], Schwab et al. described the TORERO tool, which achieved a semi-automatic allocation of the IEC 1499 control application to the distributed hardware underneath, and generated communication-related code automatically based on the allocation.

In [31], Dai and Vyatkin proposed three different approaches to redesigning existing PLC programs to the distributed architecture based on the IEC 61499 functional blocks: two object-oriented approaches and a service-oriented one (class-oriented approach), in which the program is structured based on the plant functionality. A study of the effectiveness of these approaches on an airport baggage handling system showed the improved effectiveness of the latter approach, even when compared to the original IEC 61131-3 program.

Ribeiro and Bjorkman in [32] analyzed and identified several fundamental challenges that need to be addressed before one can start to design cyber-physical production systems consistently. Among other things, the authors stated the existence of two different approaches to the decomposition (structuring) of the system: functional or structural (object-oriented) decomposition. In addition, the authors, analyzing the concept of holonic manufacturing systems, noted the lack of dedicated domain-specific language, leading to agent or service-based architectures or a combination of both in articulation with other technologies.

Patil et al. in [33] proposed refined design patterns for cyber-physical system architecture. These patterns were empirically tested in a series of projects and showed a reduction in development complexity [34].

In [35], Cruz Salazar et al. proposed a classification of multi-agent approaches to the design and implementation of cyber-physical production systems, classified them, and showed their satisfactory properties, particularly for the implementation of the field control level. The analytical part of the work provides an excellent overview of multi-agent technologies that are alternative to the conventional one based on the IEC 61131-3/61499 standard.

Zyubin and Rozov in [36] proposed a conceptual approach to the IEC 61499-based specification for control software using the poST language. The approach is based on the idea of a single reduced function block with one event input invoking the algorithm specified in poST. The approach allows for developing distributed event-driven algorithms using only one ST-like language, and thus drastically simplifies program maintenance. The question of ensuring determinism was left out of consideration.

In [37], the researchers concluded that managing scenarios for reconfiguration within the function block can be difficult at this stage as the reconfiguration and control models are merged into a single ECC. This results in overlapped error handling, initialization, and reconfiguration. Additionally, there is no differentiation between the control level and higher levels within the ECC. This leads to a large number of states and transitions and low readability and maintainability. Furthermore, due to the high dependency between the states, it is challenging for the developer to extend or maintain an ECC.

Sinha et al. in [38] proposed a hierarchical and concurrent ECC (HCECC) to model concurrent behaviors within the IEC 61499 paradigm. The HCECC-based function blocks utilize a multilevel hierarchical state machine to integrate concurrent and hierarchical behaviors and reduce system complexity. However, the authors did not tackle the reconfiguration issue, which is critical in distributed control systems where execution semantics are influenced by environmental changes and user requirements.

Marschall et al. in [39] proposed an agent-controlled approach for CPPSs. Open Platform Communications Unified Architecture (OPC UA) was selected as the standard protocol for the data exchange in the multiagent control system. The internal control logics of the system were implemented using the state machine design pattern. When implementing the system, no verification tools were used and the resulting program contained errors during execution.

Despite the fact that the IEC 61499 standard aims to achieve interoperability and portability, the results of the experiments show that the options for exchanging data between different tools within this standard are restricted. As part of their efforts to tackle the challenge of porting IEC 1499 applications, Hopsu et al. in [40] suggested streamlining manual processes for relocating functional blocks in a distributed system by creating a standalone application in order to use the NxtStudio package, which automatically manages communication between different devices.

In paper [41], the authors proposed an IEC-61499-based model for CPSs to distribute the complexity of control software over numerous small devices. The approach enables the creation of a comprehensive structure that can combine the design, simulation, and distributed deployment of automation software. The proposed scheme was validated through a packet-sorting system implemented in the nxtSTUDIO platform. Thus, the article outlines one of the variants of the so-called device-centric approach.

Parant et al. in [42] proposed a methodology for building the knowledge base from the product’s specifications and a formal diagram linkage table (DLT) approach to ensure the domains’ coherence and interdependence. This approach facilitates the identification of a perturbation’s impact on the system and proposes the appropriate response during a reconfiguration of its functionality. In fact, the authors extended the functional approach to designing CPPSs and included in the development process not only the question “what the system should do” but also the question “what the system can do”.

4. Solving Conflicts in Distributed Systems

Well-known approaches that deal with non-deterministic behavior during conflicts over shared resources have been developed for concurrent processes. The development of these methods is traditionally associated with Dijkstra's classic work, e.g., [43]. Reviews of the state of the art in this area can be found in [44]. However, since the concept makes stronger demands and has specifics, particularly the interaction with the environment, these algorithms are only partially applicable to the tasks of creating distributed automation systems.

Eidson et al. introduced a programming model that captures the physical notion of time for the model-based design of distributed real-time embedded systems, called programming temporally integrated distributed embedded systems (PTIDESs). PTIDES structures distributed software as an interconnection of components communicating using timestamped events in order to provide determinism in CPSs [45].

Among the works on modeling cyber-physical systems, the papers of Edward Lee should be especially noted. In his book [46] and also in his article [12], he dwells in detail on aspects of the correct definition of the concept of a cyber-physical system. He notes that this concept was introduced by Helen Gill in 2006 and is just a modern rethinking of cybernetic systems, which, in turn, are systems studied in control theory. However, modern cyber-physical systems are associated with large amounts of information from heterogeneous sensors. Such systems operate in a distributed manner and have an impact on people, and, when designing them, modeling cannot be dispensed with, since their correct functioning can have consequences for the environment and humans. In their works, Lee raises the key problem of modeling a non-deterministic world based on different types of models, both deterministic and non-deterministic. For instance, in [12], he gives a detailed description of PRET and Ptides projects, which aimed to use deterministic models for CPS with faithful physical realizations. The author argues that such an approach is practical due to deterministic models being easier to understand and analyze. The PRET project shows that the timing precision of synchronous digital logic can be practically made available at the software level of abstraction. The Ptides project shows that deterministic models for distributed cyber-physical systems have practical faithful realizations. The time stamp mechanism proposed in these projects allows for detecting synchronism violations due to network delay or the clock synchronization error and processing these violations in accordance with the context; for example, rejecting a database transaction.

In [47], in order to avoid undesirable non-determinism in the system of parallel processes (e.g., data racing), M. Lohstroh et al. suggested time-tagging actions of parallel processes that are sensitive to non-deterministic execution. These tags are used in additional timing constraints that are supposed to ensure that concurrent processes run in a consistent manner that does not allow for undesirable non-determinism. This technique requires a careful study of the sequence of the processes' execution, which is quite labor-intensive.

In paper [48], H. Li et al. studied the possibilities of ensuring the robustness for a distributed consumption–production system with respect to deadlocks. They proposed a strict proxy communication protocol for the system agents (processes) that limits the number and frequency of process interactions. On the one hand, such restrictions strongly lead to the absence of deadlocks in the system, but, on the other hand, they significantly complicate the interaction of parallel processes.

5. Design of Distributed Control Systems Based on Process-Oriented paradigm

5.1. Plain method

The main idea of mapping a process-oriented algorithm to a distributed architecture is shown in **Figure 1**. The mapping preserves the semantics of a hyperprocess [49] since the algorithm is in fact executed sequentially. In accordance with this scheme, the execution of a process-oriented algorithm occurs cyclically while maintaining the "read-calculate-write" cycle. The reading of the input signals is performed in parallel, and then the cycle of the inter-node synchronization of the read values is performed. After the synchronization cycle, the processes on the first computing node are activated. At the end of the calculations, the control (interprocess) signals and the calculated values of the output signals are transmitted to the stakeholder nodes; at the end of the synchronization, the second (in the general case, to the next) node computer executes its part of the control algorithm. After the sequential execution of the algorithm on all computing nodes of the distributed control system, the last microprocessor node initiates a parallel writing of output signals. After this, the next cycle begins.

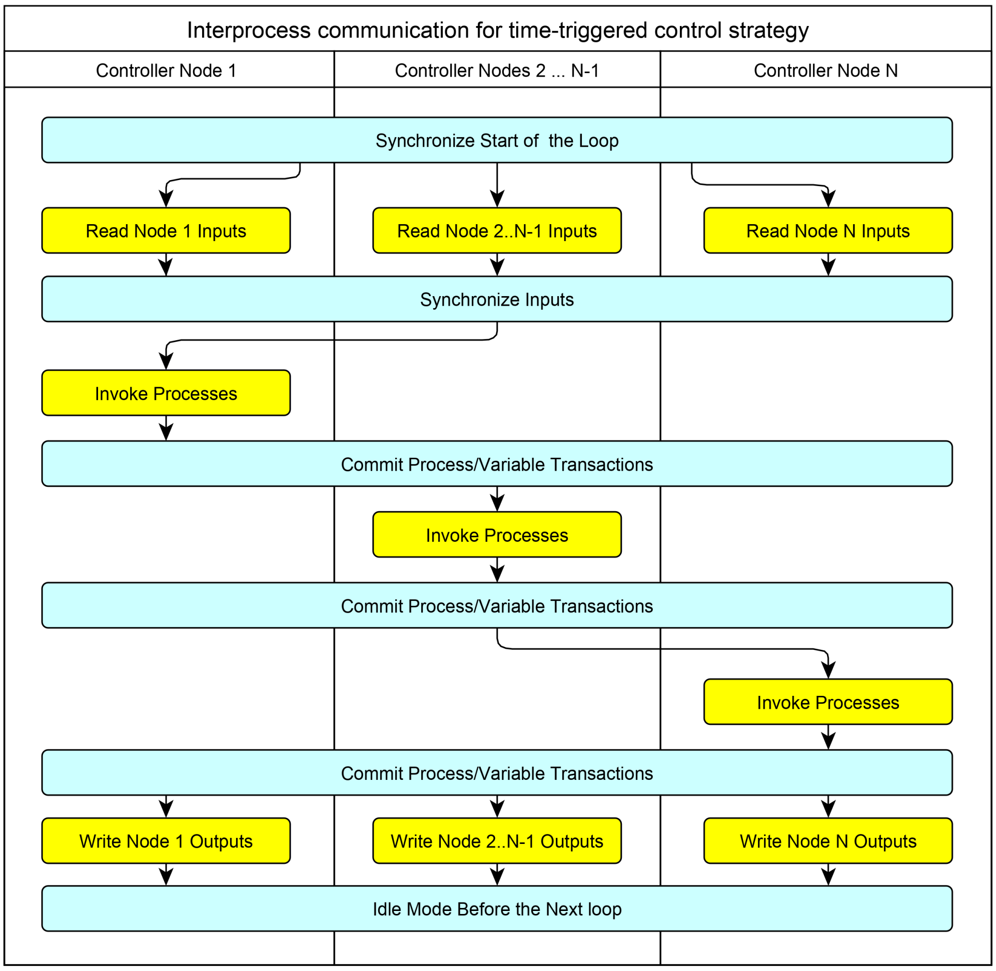
Interprocess communication for time-triggered control strategy
Controller Node 1
Controller Nodes 2 ... N-1
Controller Node N
Synchronize Start of the Loop
Read Node 1 Inputs
Read Node 2..N-1 Inputs
Read Node N Inputs
Synchronize Inputs
Invoke Processes
Commit Process/Variable Transactions
Invoke Processes
Commit Process/Variable Transactions
Invoke Processes
Commit Process/Variable Transactions
Write Node 1 Outputs
Write Node 2..N-1 Outputs
Write Node N Outputs
Idle Mode Before the Next loop

Figure 1. Preserving the hyperprocess semantics while deploying a process-oriented specification on a distributed architecture.

This approach enables researchers to generate a distributed application for any topology, comprising nodes up to the number of processes in the original process-oriented specification. When specifying the control algorithm, researchers do not care about the number of nodes, their type, and the network topology. This makes this approach topology-free programming.

Advantages of the approach:

- deterministic behavior of a distributed system;
- a high degree of granularity up to placing only one process on a node of the distributed system;
- preserving the semantics of the monolithic implementation.

Besides the advantages, the designed model also leads to a more intricate software architecture as only one PLC can have access to a single output module. It also requires additional communication between multiple PLCs to share information such as input/output module data and program data. As a result, there is an increase in both hardware costs and engineering effort required [31].

5.2. *Ad Hoc* method

There are various alternative approaches aimed at minimizing the overhead associated with inter-process data synchronization. To minimize the computational complexity of inter-node synchronization, which should ideally be excluded, the set of program processes can be divided into independent groups (clusters) with the finest possible granularity. The requirements for partitioning a control algorithm according to a process-oriented specification into clusters are as follows:

- processes using the same variable should both be in the same cluster;
- two processes using the same process should both be in the same cluster;
- processes forming a loop relative to the use relation should be in the same cluster.

Figure 2 demonstrates the algorithm flowchart that extracts clasters from a process-oriented specification.

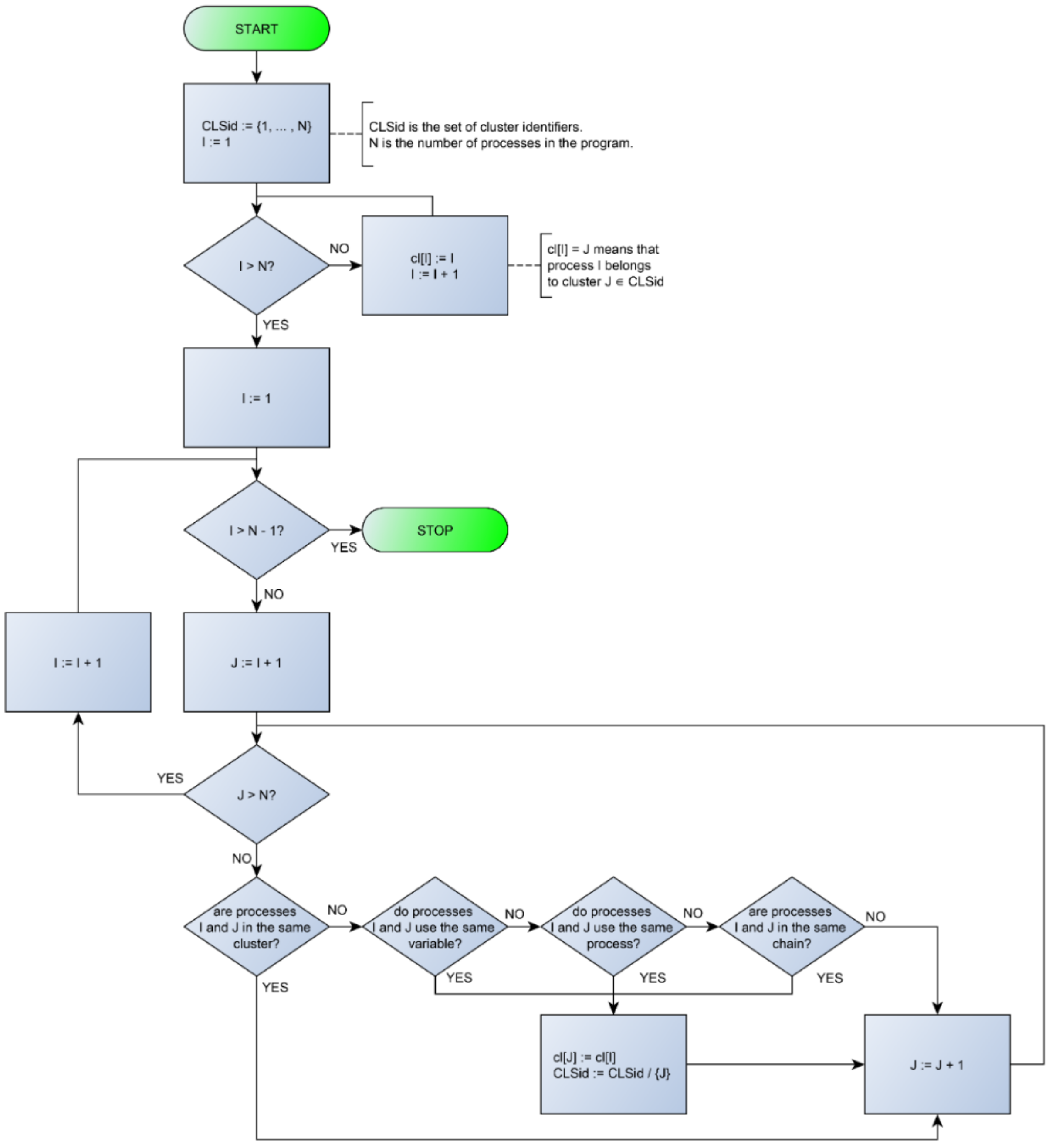
START
CLSid := {1, ... , N}
I := 1
CLSid is the set of cluster identifiers.
N is the number of processes in the program.
I > N?
NO
cl[I] := I
I := I + 1
cl[I] = J means that process I belongs to cluster J ∈ CLSid
YES
I := 1
I > N - 1?
YES
STOP
NO
I := I + 1
J := I + 1
YES
J > N?
NO
are processes I and J in the same cluster?
NO
do processes I and J use the same variable?
NO
do processes I and J use the same process?
NO
are processes I and J in the same chain?
NO
YES
YES
YES
YES
cl[J] := cl[I]
CLSid := CLSid / {J}
J := J + 1

Figure 2. Partitioning algorithm flowchart.

6. Summary

A topologically independent specification of distributed control algorithms with the process-oriented programming paradigm can be deployed with the plain approach based on a network of microcontroller nodes. In this schema, only operations for reading input and writing output signals are physically parallelized, and the algorithm itself is executed sequentially, similarly to a centralized implementation, with synchronization of information messages and the order of execution over the bus. The approach constructively ensures the absence of data races and the preservation of the semantics of the original process-oriented program, which means the applicability of existing methods for verifying centralized process-oriented programs to distributed process-oriented programs. This opens the door to the formal verification of distributed programs using already developed approaches.

In particular, model checking method for process-oriented programs proposed in [50][51] can be applied with minor corrections that concern constructing several independent action lines of parallel processes with regard to given clusters. Also it is possible to use process-oriented deductive verification methods proposed in [15][52] by applying them to each cluster separately with the original program annotations.

In [53] an algorithm for splitting a process-oriented specification into clusters, which eliminates the need for data synchronization, is proposed. The study of the algorithm on case study shows that clusters can be arranged in any arbitrary combination on the computing nodes of a distributed system. In any combination, the physical signals are processed locally and the implementation does not require data synchronization.

References

1. Capehart, B.L.; Capehart, L.C. Web Based Energy Information and Control Systems: Case Studies and Applications; CRC Press: Boca Raton, FL, USA , 2021.
2. Chakraborty, S.; Ramesh, S. Programming and Performance Modelling of Automotive ECU Networks. In Proceedings of the 21st International Conference on VLSI Design (VLSID 2008), Hyderabad, India, 4–8 January 2008 ; pp. 8–9.
3. Dongik Lee; Jeff Allan; Stuart Bennett. Distributed Real-Time Control Systems using CAN; Springer Science and Business Media LLC: Dordrecht, GX, Netherlands, 2003; pp. 357-385.
4. Sakle, S.V.; Kadam, M.R.; Dhande, M.J. Review paper of Vehicle control system using CAN protocol. Int. J. Orange Technol. 2021, 3, 40–45.
5. Drozdov, D.; Atmojo, U.D.; Pang, C.; Patil, S.; Ali, M.I.; Tenhunen, A.; Oksanen, T.; Cheremetiev, K.; Vyatkin, V. Utilizing software design patterns in product-driven manufacturing system. In Proceedings of the International Workshop on Service Orientation in Holonic and Multi-Agent Manufacturing, Paris, France, 1–2 October 2020.
6. Kleanthis Thramboulidis; IEC 61499 vs. 61131: A Comparison Based on Misperceptions. *J. Softw. Eng. Appl.* **2013**, *06*, 405-415.
7. Thomas Strasser; Alois Zoitl; James H. Christensen; Christoph Sünder; Design and Execution Issues in IEC 61499 Distributed Automation and Control Systems. *null* **2010**, *41*, 41-51.
8. Thramboulidis, K. Different perspectives [Face to Face; “IEC 61499 function block model: Facts and fallacies”]. IEEE Ind. Electron. Mag. 2009, 3, 7–26.

9. Kim, J.C.; We, K.S.; Lee, C.G.; Lin, K.J.; Lee, Y.S. HW resource componentizing for smooth migration from single-function ECU to multi-function ECU. In Proceedings of the 27th Annual ACM Symposium on Applied Computing, Trento, Italy, 25–29 March 2012; pp. 1821–1828.
10. Fox, G.C.; Williams, R.D.; Messina, P.C. Parallel Computing Works!; Elsevier: Amsterdam, The Netherlands, 2014.
11. Vajk, T.; Dávid, Z.; Asztalos, M.; Mezei, G.; Levendovszky, T. Runtime model validation with parallel object constraint language. In Proceedings of the 8th International Workshop on Model-Driven Engineering, Verification and Validation, Wellington, New Zealand, 17 October 2011; pp. 1–8.
12. Edward A. Lee; The Past, Present and Future of Cyber-Physical Systems: A Focus on Models. *Sensors* **2015**, *15*, 4837-4869.
13. Vladimir E. Zyubin; Andrei S. Rozov; Igor S. Anureev; Natalia O. Garanina; Valeriy Vyatkin; poST: A Process-Oriented Extension of the IEC 61131-3 Structured Text Language. *IEEE Access* **2022**, *10*, 35238-35250.
14. Shaila P. Kharde Rachana V. Vairal; A Review on Vehicle Control System by Using CAN Protocol. *Int. J. Adv. Res. Electr. Electron. Instrum. Eng.* **2015**, *4*, 8678-8682.
15. Igor Anureev; Natalia Garanina; Tatiana Liakh; Andrei Rozov; Vladimir Zyubin; Sergei Gorlatch. Two-Step Deductive Verification of Control Software Using Reflex; Springer Science and Business Media LLC: Dordrecht, GX, Netherlands, 2019; pp. 50-63.
16. Vladimir Zyubin; Igor Anureev; Natalia Garanina; Sergey Staroletov; Andrei Rozov; Tatiana Liakh. Event-Driven Temporal Logic Pattern for Control Software Requirements Specification; Springer Science and Business Media LLC: Dordrecht, GX, Netherlands, 2021; pp. 92-107.
17. Controller Area Network CAN, an Invehicle Serial Communication Protocol. J1583_199003. Available online: https://www.sae.org/standards/content/j1583_199003/ (accessed on 6 July 2023)
18. Sergey Staroletov. A Software Framework for Jetson Nano to Detect Anomalies in CAN Data; Institute of Electrical and Electronics Engineers (IEEE): Piscataway, NJ, United States, 2023; pp. 490-498.
19. Obermaisser, R. Event-Triggered and Time-Triggered Control Paradigms; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2004; Volume 22.
20. Albert, A. Comparison of event-triggered and time-triggered concepts with regard to distributed control systems. Embed. World 2004, 2004, 235–252.
21. G Leen; D Heffernan; TTCAN: a new time-triggered controller area network. *Microprocess. Microsystems* **2002**, *26*, 77-94.

22. Woonhyuk Baek; Seyong Jang; Hoin Song; Soontae Kim; Bongsob Song; Dongkyoung Chwa; A CAN-based Distributed Control System for Autonomous All-Terrain Vehicle (ATV). *null* **2008**, *41*, 9505-9510.

23. K. Tindell; A. Burns; A.J. Wellings; Calculating controller area network (can) message response times. *Control. Eng. Pr.* **1995**, *3*, 1163-1169.

24. Chen, R.; Liu, B.; Pan, M.; Zhou, H. Design of Distributed Control System for the Pick-up Robot Based on CAN Bus. In Proceedings of the 2019 IEEE International Conference on Mechatronics and Automation (ICMA), Tianjin, China, 4–7 August 2019; pp. 102–107.

25. Liangfei, X.; Jianfeng, H.; Xiangjun, L.; Jianqiu, L.; Minggao, O. Distributed control system based on CAN bus for fuel cell/battery hybrid vehicle. In Proceedings of the 2009 IEEE International Symposium on Industrial Electronics, Seoul, Korea, 5–8 July 2009; pp. 183–188.

26. Shiau, J.K.; Hung, W.S.; Chang, C.M. Development of a Distributed Multi-MCU Based Flight Control System for Unmanned Aerial Vehicle. J. Appl. Sci. Eng. 2015, 18, 251–258.

27. Christensen, J.H. Design patterns for systems engineering with IEC 61499. In Proceedings of the Verteilte Automatisierung-Modelle und Methoden für Entwurf, Verifikation, Engineering und Instrumentierung (VA2000), Magdeburg, Germany, 22–23March 2000; pp. 63–71. Available online: http://www.holobloc.com/papers (accessed on 6 July 2023).

28. Valeriy Vyatkin; Hans-Michael Hanisch; Cheng Pang; Chia-Han Yang; Closed-Loop Modeling in Future Automation System Engineering and Validation. *null* **2008**, *39*, 17-28.

29. Kleanthis Thramboulidis; A cyber–physical system-based approach for industrial automation systems. *Comput. Ind.* **2015**, *72*, 92-102.

30. Schwab, C.; Tangermann, M.; Lueder, A. The modular TORERO IEC 61499 engineering platform-Eclipse in automation. In Proceedings of the 2005 IEEE Conference on Emerging Technologies and Factory Automation, Catania, Italy, 19–22 September 2005; Volume 2.

31. Wenbin Dai; Valeriy Vyatkin; Redesign Distributed PLC Control Systems Using IEC 61499 Function Blocks. *IEEE Trans. Autom. Sci. Eng.* **2012**, *9*, 390-401.

32. Luis Ribeiro; Mats Bjorkman; Transitioning From Standard Automation Solutions to Cyber-Physical Production Systems: An Assessment of Critical Conceptual and Technical Challenges. *IEEE Syst. J.* **2017**, *12*, 3816-3827.

33. Patil, S.; Drozdov, D.; Vyatkin, V. Adapting software design patterns to develop reusable IEC 61499 function block applications. In Proceedings of the 2018 IEEE 16th International Conference on Industrial Informatics (INDIN), Porto, Portugal, 18–20 July 2018; pp. 725–732.

34. Patil, S.; Drozdov, D.; Zhabelova, G.; Vyatkin, V. Refactoring of IEC 61499 function block application—A case study. In Proceedings of the 2018 IEEE Industrial Cyber-Physical Systems

(ICPS), Saint Petersburg, Russia, 15–18 May 2018; pp. 726–733.

35. Luis Alberto Cruz Salazar; Daria Ryashentseva; Arndt Lüder; Birgit Vogel-Heuser; Cyber-physical production systems architecture based on multi-agent's design pattern—comparison of selected approaches mapping four agent patterns. *Int. J. Adv. Manuf. Technol.* **2019**, *105*, 4005-4034.
36. Vladimir Zyubin; Andrei Rozov. Using Process-Oriented Structured Text for IEC 61499 Function Block Specification; Springer Science and Business Media LLC: Dordrecht, GX, Netherlands, 2021; pp. 217-227.
37. Guellouz Ep Addad, S. Towards a New Methodology for Design, Modelling, and Verification of Re configurable Distributed Control Systems Based on a New Extension to the IEC 61499 Standard. Dissertation zur Erlangung des Grades des Doktors der Ingenieurwissenschaften der Naturwisse nschaftlich–Technischen Fakultät der Universität des Saarlandes und Tunisia Polytechnic School, Carthage University, 2021. (accessed on 6 July 2023). . Zertifiziert mit dem DINI-Zertifikat 2019 für Open-Access-Publikationsdienste. Retrieved 2023-7-26
38. Roopak Sinha; Partha S. Roop; Gareth Shaw; Zoran Salcic; Matthew M. Y. Kuo; Hierarchical and Concurrent ECCs for IEC 61499 Function Blocks. *IEEE Trans. Ind. Informatics* **2015**, *12*, 59-68.
39. Benedikt Marschall; Markus Schleicher; Axel Sollich; Thomas Becker; Tobias Voigt; Design and Installation of an Agent-Controlled Cyber-Physical Production System Using the Example of a Beverage Bottling Plant. *IEEE J. Emerg. Sel. Top. Ind. Electron.* **2021**, *3*, 39-47.
40. Hopsu, A.; Atmojo, U.D.; Vyatkin, V. On portability of IEC 61499 compliant structures and systems. In Proceedings of the 2019 IEEE 28th International Symposium on Industrial Electronics (ISIE), Vancouver, BC, Canada, 12–14 June 2019; pp. 1306–1311.
41. Ernesto Monroy Cruz; Luis Rodolfo Garcia Carrillo; Luis Alberto Cruz Salazar; Structuring Cyber-Physical Systems for Distributed Control with IEC 61499 Standard. *null* **2023**, *21*, 251-259.
42. Alexandre Parant; François Gellot; Damien Zander; Véronique Carré-Ménétrier; Alexandre Philippot; Model-based engineering for designing cyber-physical systems from product specifications. *Comput. Ind.* **2023**, *145*, 103808.
43. Edsger W. Dijkstra; The structure of the "THE"-multiprogramming system. *Commun. ACM* **1968**, *11*, 341-346.
44. Dhoked, S.; Golab, W.; Mittal, N. Recoverable Mutual Exclusion; Springer Nature: Berlin/Heidelberg, Germany, 2023.
45. John C. Eidson; Edward A. Lee; Slobodan Matic; Sanjit A. Seshia; Jia Zou; Distributed Real-Time Software for Cyber–Physical Systems. *null* **2011**, *100*, 45-59.
46. Lee, E.A. Plato and the Nerd: The Creative Partnership of Humans and Technology; MIT Press: Cambridge, MA, USA, 2017.

47. Marten Lohstroh; Christian Menard; Soroush Bateni; Edward A. Lee; Toward a Lingua Franca for Deterministic Concurrent Systems. *ACM Trans. Embed. Comput. Syst.* **2021**, *20*, 1-27.
48. Huailin Li; Qinsen Liu; Mengnan Liu; Bangyong Sun; Bin Du; Robust Deadlock Control for Reconfigurable Printing Manufacturing System Based on Process Algebra. *IEEE Access* **2023**, *11*, 42473-42484.
49. Vladimir E. Zyubin. Hyper-automaton: a Model of Control Algorithms; Institute of Electrical and Electronics Engineers (IEEE): Piscataway, NJ, United States, 2007; pp. 51-57.
50. Tatiana V. Liakh; Natalia O. Garanina; Igor S. Anureev; Vladimir E. Zyubin. Verifying Reflex-software with SPIN: Hand Dryer Case Study; Institute of Electrical and Electronics Engineers (IEEE): Piscataway, NJ, United States, 2020; pp. 210-214.
51. Anna A. Ponomarenko; Natalia O. Garanina; Sergey M. Staroletov; Vladimir E. Zyubin. Towards the Translation of Reflex Programs to Promela: Model Checking Wheelchair Lift Software; Institute of Electrical and Electronics Engineers (IEEE): Piscataway, NJ, United States, 2021; pp. 493-498.
52. Ivan Chernenko; Igor Anureev; Natalia Garanina. Proving Reflex Program Verification Conditions in Coq Proof Assistant; Institute of Electrical and Electronics Engineers (IEEE): Piscataway, NJ, United States, 2021; pp. 485-488.
53. Vladimir E. Zyubin; Natalia O. Garanina; Igor S. Anureev; Sergey M. Staroletov; Towards Topology-Free Programming for Cyber-Physical Systems with Process-Oriented Paradigm. *Sensors* **2023**, *23*, 6216.