Jhunjhunwala, Pranay; Zoitl, Alois; Atmojo, Udayanto Dwi; Vyatkin, Valeriy

Monitoring design pattern for distributed automation systems in IEC 61499 and its formal modelling

Published in:
2022 IEEE 31st International Symposium on Industrial Electronics, ISIE 2022

DOI:
10.1109/ISIE51582.2022.9831470

Published: 01/01/2022

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Please cite the original version:
Jhunjhunwala, P., Zoitl, A., Atmojo, U. D., & Vyatkin, V. (2022). Monitoring design pattern for distributed automation systems in IEC 61499 and its formal modelling. In *2022 IEEE 31st International Symposium on Industrial Electronics, ISIE 2022* (pp. 220-225). (Proceedings of the IEEE International Symposium on Industrial Electronics; Vol. 2022-June). IEEE. https://doi.org/10.1109/ISIE51582.2022.9831470

Monitoring design pattern for distributed automation systems in IEC 61499 and its formal modelling

Pranay Jhunjhunwala*, Alois Zoitl‡, Udayanto Dwi Atmojo*, Valeriy Vyatkin*†
*Department of Electrical Engineering and Automation, Aalto University, Espoo, Finland
†Department of Computer Science, Computer and Space Engineering, Lulea Tekniska Universitet, Sweden
‡ Johannes Kepler University Linz, Linz, Austria

Email: {pranay.jhunjhunwala, alois.zoitl@jku.at, udayanto.atmojo@aalto.fi, vyatkin@ieee.org

***Abstract*—This paper addresses the challenge of achieving reliable and predictable operation of flexible and modular production systems with distributed control and potentially wireless communication. Such systems are envisaged as common in the future Industry 4.0 production facilities.**

A software design pattern is proposed to implement online monitoring of requirements. The IEC 61499 architecture is selected as the implementation platform and its benefits are essentially used by utilisation of the adapter interface mechanism. The paper also outlines a pathway to designing and verifying the monitors based on formal methods.

***Index Terms*—Monitoring, Agent-based approach, Production planning, Service computing**

I. Introduction

The distributed automation systems architecture promises many benefits for factory operations, for example flexibility and reconfigurability. By that distributed architectures eventually contribute significantly to the shift from mass production to mass customisation. It is a natural solution to automate modular production systems implementing plug and produce concept of flexibility as highlighted by [1].

The multi-agent software architecture fits perfectly to the requirements of such systems. It has been investigated in the automation system context for the last three decades. Its main attractive features include the ability to achieve self-adaptation of the participating agents to the ever-changing requirements to the production.

However, along with the benefits, there are some challenges associated with this architecture related to the performance, determinism, and correctness assurance. The execution performance and determinism can be improved by implementing agents in well-established automation frameworks, for example, using the [2] standard.

The execution correctness assurance has roots in proving properties related to the global state of distributed systems. This is a fundamental problem in the theory of distributed systems. The global state properties can be observed by a distributed software application, composed of several communicating software components, called observers, which can be implemented in a centralised or decentralised manner.

In this paper, we are investigating ways to incorporate distributed observers to the distributed automation applications, implemented in IEC 61499 and implementing the multi-agent architecture.

The rest of the paper is structured as follows: in Section II we highlight the literature and past works, followed by the control architecture in Section III. In Section IV we discuss the proposed monitor architecture and its implementation, followed by the formal modelling of the monitor in Section V, and conclusions in Section VI.

II. Literature Review

Multi-agent approach to programming of intelligent automation systems is outlined by [3]. With IEC 61499 it was demonstrated by [4] in the mechatronics context and by [5] in the smart energy area (a comprehensive survey is in [6]).

The fundamental problem of checking properties pertaining to the global state in a distributed system has been addressed in many works. The seminal work of [7] deserves a special credit. In order to address this problem especially during the development time of multi agent systems [8] developed the so called agent sniffer tool. This tool captures all messages exchanged by the agents and allows the user to manually analyse it. This has been further expanded in [9] by an automatics analysis of the captured messages and deriving system properties from them (e.g. derive the system structure). However both works are focusing on the engineering setup phase of systems and not on the normal execution phase.

The growing industrial interest to the IEC 61499 architecture motivates the research on efficient software engineering as applied to this particular technology. [10] highlights classifications of design patterns used in IEC 61499. One-line engineering proposed by [11] is an efficient way to organize the IEC 61499 function block (FB) application, avoiding the messy event and data connections between FBs by using adapter interface connections. Its usage was demonstrated by [12] for intelligent mechatronics and by [13] for process automation systems.

The monitor design pattern has been recently introduced in the IEC 61499 context by [14]. This work raises the question of how the monitors shall be developed based on the given requirements and how their correctness could be verified. Besides, application of the monitor design pattern in distributed systems, which IEC 61499 is destined for, raises

the problem of decentralised implementation of the monitors. [15] introduced the observer-based verification of IEC 61499 applications. The concept implies appending the so called observer FBs to a FB application, which encapsulate checking of the requirements of interest during the run-time.

The ongoing transition to the IoT-inspired distributed automation architectures requires to implement monitors in a decentralised way. This motivates the need for developing the distributed monitoring design pattern, presented in this paper. The shortly mentioned related works create the base for that. The distributed monitoring also needs to be verified, and the idea of its formal verification is investigated in this paper. This novel development aims at applications in strongly modular manufacturing systems with plug and produce capabilities and multi-agent application logic.

III. Agent Based Architecture for control applications

In this work, an agent based control architecture enhanced with the one-line engineering concept has been developed and proposed. The control application is based on the service oriented architecture (SOA).

A. The EnAs Demonstrator

The EnAS Demonstrator[1] is a laboratory scale production island, is used as the test bed in this work. The EnAS demonstrator is composed of six mechanical conveyors and two pneumatic production stations consisting of jacks and sledges.

Shown in top-right of Fig. 1, is the top view diagram of the EnAS demonstrator, in which all the six conveyors are connected in a cyclic-chain. Each conveyor is driven by an electric motor and has a light sensor to detect the presence of a work-piece at a certain position on the production line. In the control application, all the components have been considered as autonomous collaborative agents, responsible for their designated stations. Once agents receive commands from upper layers of the control application, they act as autonomously, collaborating with their peers rather than with the higher-level layers. Agents use information from neighbouring agents to ensure smooth and successful operation.

In Fig. 1, the composition of the control application, along with it's functional decomposition based on the SOA, is showcased. The hardware components of the demonstrator are controlled in the bottom layer, the so-called 'Execution Services Layer'. In the 'Planning Services Layer', the FB's plan the required delivery or placement operations, to meet the requirements generated by the 'Production Services Layer'.

Alongside the various layers of the architecture, the connections between the agents have been highlighted with the help of the 2-D diagram. Each agent has it's individual control FB, which is connected via adapter links to the previous and the following agent, just like the connections of the physical conveyors in EnAS.

[1]https://www.energieautark.com/

B. Agent Operation

Agents operate by receiving control commands either from the production and planning services layers of the control application or agents connected upstream. Commands from the production and planning services layers are sent to the first agent. The agent processes the received command, and forwards the processed information to the agents downstream. In the 2-D diagram in Fig. 1, we see Conveyor Agent 1 receives the command from the Production recipe agent, processes it and passes it to Conveyor Agent 2 downstream.

The similar schematic is followed in the control application. There the 'Conveyor1' FB receives its command from the higher-level layers and is connected to the 'Conveyor2' FB using the adapter link. The similar connection has been observed in all other agents, replicating the cyclic connection pattern of the demonstrator.

Control commands received are processed by the respective agent/conveyor FB. If the command corresponds to the agent, it stores its command and forwards the remaining sequence to the agents downstream. Fig. 2, showcases operation and the chain of message passing between agents.

Each conveyor agent can operate in 3 different modes as explained below:

- Mode C: In mode C, the respective conveyor start running irrespective of the position of the work piece. The conveyor stops running when the work piece crosses the sensor of the next conveyor.
- Mode P: In mode P, the respective agent starts the conveyor when the work piece crosses the sensor of the previous conveyor and stops the conveyor when the work piece crosses the sensor of the next conveyor. 'P' is used to signify pass, meaning the work piece passes the conveyor without stopping.
- Mode D: This is the delivery mode, in which when the work piece comes in front of the sensor of the respective agent, the conveyor is stopped. In the 'D' mode, the agent is designed to perform an additional task of confirming the work piece delivery to the 'Planning Services' and 'Production Services' layers. A confirmation is needed, so as to proceed to the next step of production.

As an example, based on the command used to demonstrate the message passing in Fig. 2, the work piece would start from conveyor 4, pass through conveyors 5-6-1, and stop in front of the conveyor 2 sensor.

The agents act autonomously, but each agent has knowledge about certain critical information from adjacent agents required for the smooth operation. As shown in Fig. 1, each agent is connected to a sensor to detect the position of the work piece. Whenever there is a change in the sensor reading, the agent not only performs the desired operation but at the same time communicates the sensors readings to the adjacent agents. For example, agent 1 shown in Fig. 1, has the following information:

- Sensor for it's responsible conveyor (Yellow)
- Sensor information reported by Agent 2 (Purple)

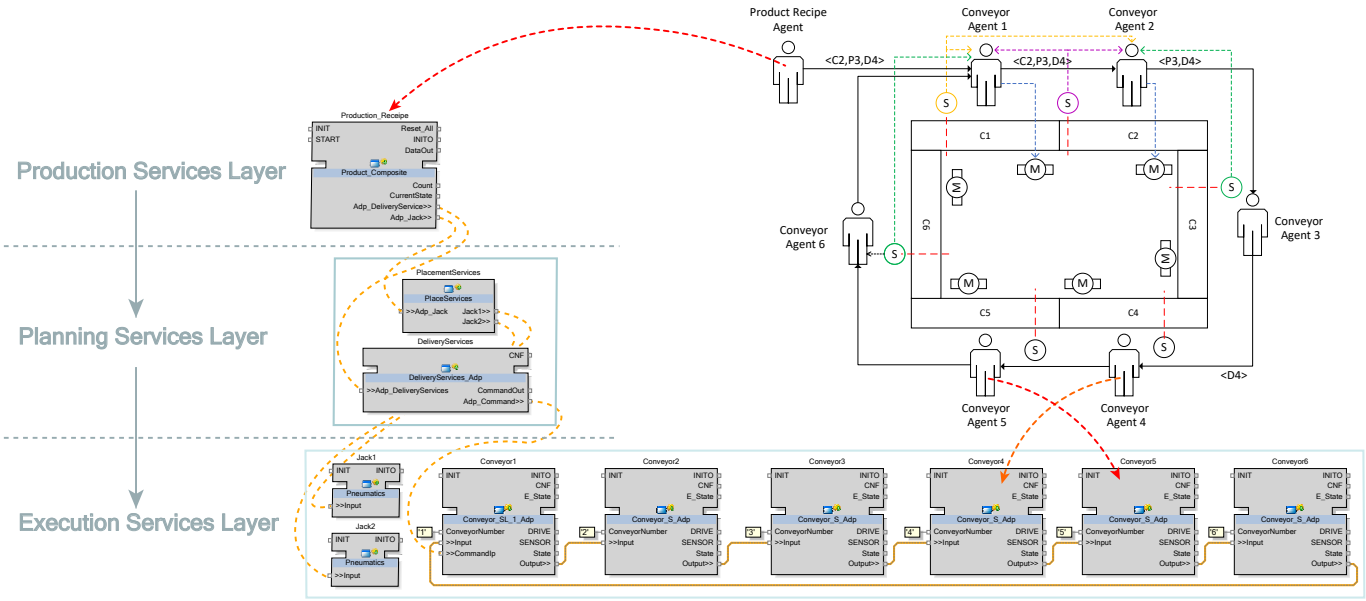

Fig. 1. Implementation of the agent-based control logic of EnAS using SOA and one-line engineering pattern of IEC 61499.

- Sensor information reported by Agent 6 (Green)

This method of having the agents as individual entities which operate on their own reduces the need of communication within the various layers of the control application.

IV. Monitor

The previous section explained the application of the agent-based architecture for design and operation of decentralised control of industrial processes. A drawback of the proposed architecture would be a possible inconsistency of observed input data in case of a failure or malfunction of any one of the service providing agents. This may lead to a wrong decision taken by the agents.

As stated by [14], the idea behind monitoring is to observe a particular consistency condition and raise a violation event or automatically take over a situation violating the consistency conditions.

The goal of the designed monitor would be to observe the operation of each of the control agent present in the control application and raise a violation event or notification to the higher-level operators in case of agent failure or operational faults. Shown in Fig. 3 is the general idea of the proposed mechanism in which each agent communicates with the monitor. The monitor observes the data sent by each agent and checks for the consistency conditions but does not modify any information or control the operation of the respective control agents.

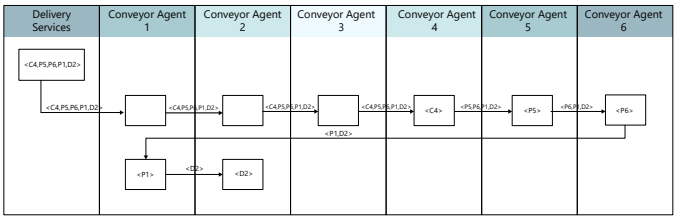

Fig. 2. Message passing between agents.

A. Ideal Monitoring Scenario

Fig. 4, showcases the ideal operation of the proposed monitor. The monitor and the first agent in the queue receive the command generated by the production services. When the monitor receives the command, it is processed within the monitor and the required state of each agent is identified. As soon as the received command is processed the monitor initiates a delay period[2]. The monitoring is carried out once the delay period elapses.

As soon as each agent receives its command, it reports the same to the monitor. As shown in Fig. 4, all three conveyor agents reported their respective states to the monitor. Upon the delay period elapsing, the monitor enters the 'Monitoring'

[2]In our case study: 500ms is the delay period used to give agents some time to receive and process the commands.

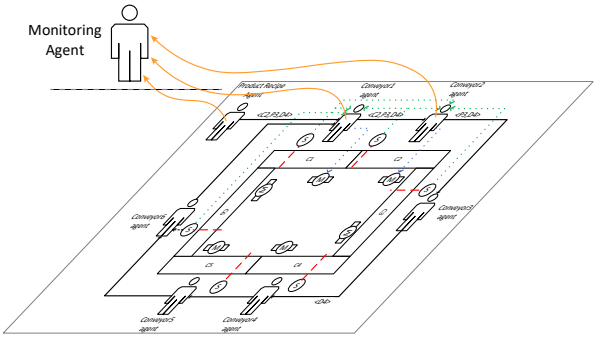

Fig. 3. General Monitor Design Pattern for Agent Based Architecture.

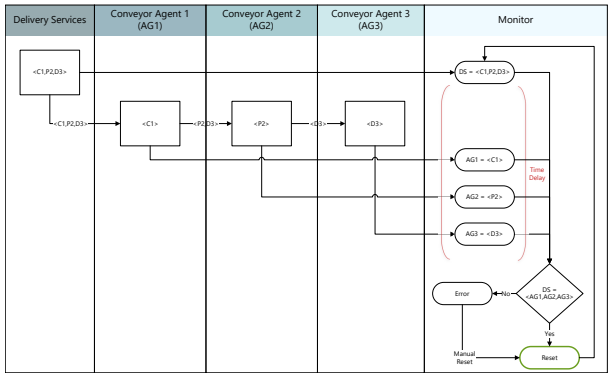

Fig. 4. Monitor : Regular Operation.

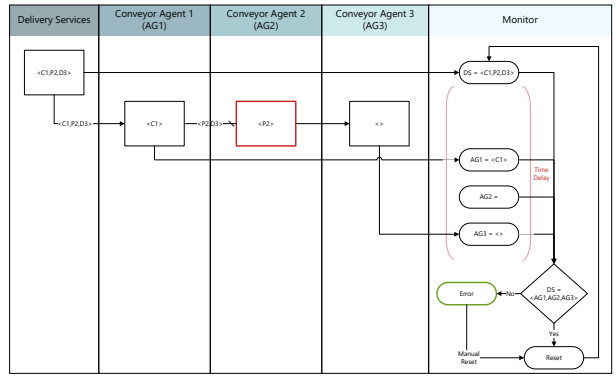

Fig. 5. Monitor : Agent failure condition.

state in which it compares the required state of each agent to the actual state of each agent. In an ideal system operation as highlighted in Fig. 4, the required and attained for each agent match, thereby the monitor automatically resets and waits for the next round of monitoring.

B. Error Monitoring Scenario

To demonstrate a failure case detected by the monitor, we assume that conveyor agent 2 has failed meaning that the respective device could be spoilt, or the device could have lost network connectivity and is unable to communicate with other agents connected upstream or downstream.

As shown in Fig. 5, the command generated by the production services is processed by 'Conveyor Agent 1' and sent downstream to 'Conveyor Agent 2', which due to failure does not receive the command. As stated as drawback above, in this case the chain of message passing is broken, thereby making Agent 3 and all others downstream useless even though they are functionally capable.

Due to the device malfunction, Agent 2 did not respond with a state and Agent 3 not receiving an input command, reported a blank state to the monitor. As highlighted in Fig. 5, upon entering the monitoring state, the desired states for each agent weren't achieved, following which the monitor enters the Error state and raises a violation event. The monitor stays in the error state until attended by the operator and manually reset.

C. Implementation & Working

Shown in Fig. 6 is the developed FB for the monitor. The FB receives as input, the desired state or the command sent to the agents along with the operational state of each agent as and when reported. Inputs "C1_Status to C6_Status" of the FB are individually connected to each conveyor agent and receive the status of the agent. The services of the monitor can be utilized by enabling or activating it using the Enable input of the FB. The input can be triggered using an HMI or force triggering by the operator.

Fig. 7 shows the state machine(SM) for the monitor and displays the respective algorithms in each state. Upon enabling the monitor, the monitor waits in the 'WAIT' state and enters the 'CheckIpCommand' state upon receiving the desired command from the production services. In the state, the monitor analyses the command send to the agents, and stores the desired state of each agent in their respective internal variables.

The monitor enters the 'Monitoring' state only upon completion of the delay period, until which it stays in the 'CheckIpCommand' state, but it can receive the status reports by the agents at their respective inputs. For each round of monitoring, the status reports from the agents will only be considered if sent within the delay period.

In the monitoring state, the algorithm first checks if all the agents have attained the required states, in which it sets the Boolean internal variable 'AllOkay' as TRUE, due to which the SM exits the monitoring state and reset's itself for the next round of monitoring by entering the reset state.

If all agents do not attain the required state, internal variable 'Error' is flagged TRUE, following which in the monitoring state, the algorithm individually checks for the agents which have not attained the desired stated and creates a record for the operator to check. Once all agents are individually checked, the monitor enters the 'Error' state in which the monitor raises the violation condition and sets the flag ErrorV as TRUE and waits for a manual reset from the operator.

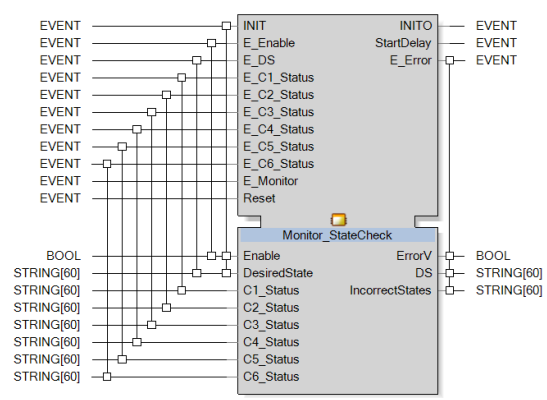

Fig. 6. Function Block interface of the Monitor.

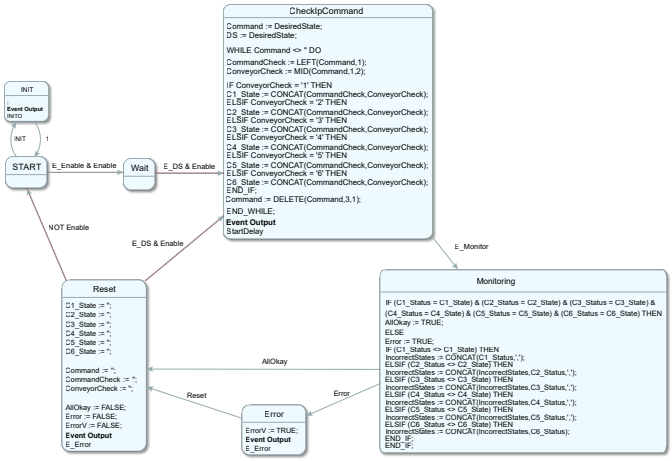

Fig. 7. Monitor State Machine.

D. Scaling the Monitor

The proposed monitor is developed, keeping in mind, that it could be executed on the same or different device, thanks to the deployment flexibility of IEC 61499. The options include running on a controller separate from the main system controllers or edge devices operating locally or even deployment on the cloud. The deployment of the monitor on a standalone device would ensure the monitor does not interfere with the process in consideration or would not bottleneck the operation of the distributed system. On the other hand, it may bring communication overheads.

The monitor pattern aims at verifying the operation of distributed agents during runtime. The pattern showcased in the paper can be used for monitoring a number of distributed agents, by simply extending the needed inputs for the monitor and modifying the algorithm based on the operation of the agents. In this case the agents reported their respective values, which were monitored. The correctness of the monitor can be verified and assured using the means of formal modelling and verification which has been further described in section V.

V. Formal model of monitor

Properties of the monitor can be studied using formal modelling, for example using the Petri Net dialect called Timed Net Condition-Event Systems (TNCES) [16]. The model in Fig. 8 is TNCES, modelling a simplified system of two conveyors' control agents receiving the motion request from the Delivery Services. The timing characteristics are such that the message delay to Conveyor 1 is 3 time units and to the Conveyor 2 is 2 time units, which is reflected in the time constants, assigned to the arcs (p_7, t_3) and (p_9, t_5) respectively. These timings should be derived from the real properties of the system under study. One should note that the transitions t_3 and t_5, modelling the receipt of the message, in the conveyor models are spontaneous, modelling the possibility of message to be also lost, thus not received by the corresponding recipient. The Monitor, in this particular case, detects the situation that both both conveyors have not received the message after 5 time units. In this case the token will be placed to the place p_4 and the Monitor will no longer be operational.

The model, described above, can be investigated using reachability analysis. The software tool [17] can build the reachability space of the model, as illustrated in Fig. 9. The graph does not have deadlocks, proving liveness of the system and does contain states where token is in place p_4, which proves that there are states in which the Monitor would detect the failure.

One can have a closer look at the developments in the model by extracting and visualising a particular trajectory in the reachability graph, as shown in Fig. 10. From here one can see that this particular trace illustrates the scenario, in which Conveyor 1 has not received the message twice in the row which was followed by the Monitor issuing the failure warning event.

VI. Conclusion

In this paper, the monitor design pattern for IEC 61499 has been extended to the case of distributed implementation. The challenges, related to the distributed implementation of the monitoring logic have been tackled by the use of formal modelling and verification as a prerequisite to the monitor coding. The future work may apply methods of the formal monitor synthesis and investigation of the method's limitation in terms of the timing requirements.

VII. Acknowledgement

We dedicate this work to the memory of our dear colleague Prof. Jan Olaf Blech, with whom we had many fruitful discussions on the topic and who especially insisted on the formal modelling of monitors with Petri nets.

References

[1] U. D. Atmojo, J. O. Blech, and V. Vyatkin, "A plug and produce-inspired approach in distributed control architecture: A flexible assembly line and product centric control example," in *2020 IEEE International Conference on Industrial Technology (ICIT)*, pp. 271–277, 2020.

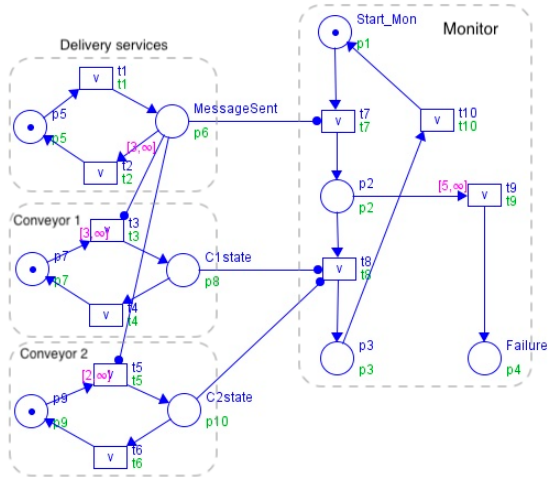

Fig. 8. Condition-Event system model of the monitor applied to a system of two conveyors.

[2] IEC 61499, *Function blocks–Part 1: architecture, Second edition*. 2012.
[3] P. Leitao, S. Karnouskos, L. Ribeiro, J. Lee, T. Strasser, and A. W. Colombo, "Smart agents in industrial cyber–physical systems," *Proceedings of the IEEE*, vol. 104, no. 5, pp. 1086–1101, 2016.
[4] M. Sorouri, S. Patil, Z. Salcic, and V. Vyatkin, "Software composition and distributed operation scheduling in modular automated machines," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 4, pp. 865–878, 2015.
[5] G. Zhabelova, V. Vyatkin, and V. N. Dubinin, "Toward industrially usable agent technology for smart grid automation," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 4, pp. 2629–2641, 2014.
[6] P. Vrba, V. Mařík, P. Siano, P. Leitão, G. Zhabelova, V. Vyatkin, and T. Strasser, "A review of agent and service-oriented concepts applied to intelligent energy systems," *IEEE transactions on industrial informatics*, vol. 10, no. 3, pp. 1890–1903, 2014.
[7] F. Mattern, "Virtual time and global states of distributed systems," *Parallel and Distributed Algorithms*, pp. 215–226, 1989.
[8] P. Vrba, P. Tichý, V. Mařík, K. H. Hall, R. J. Staron, F. P. Maturana, and P. Kadera, "Rockwell automation's holonic and multiagent control systems compendium," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 41, no. 1, pp. 14–30, 2011.
[9] P. Vrba, P. Kadera, M. Myslík, and M. Klíma, "Jboss esb sniffer," in *2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE)*, pp. 1724–1729, 2014.
[10] S. Patil, D. Drozdov, and V. Vyatkin, "Adapting software design patterns to develop reusable IEC 61499 function block applications," in *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, pp. 725–732, IEEE, 2018.
[11] G. Kolleger and A. Kopitar, *Flexible and Reusable Industrial Control Application*, pp. 243–273. CRC Press, 2017.
[12] W. W. Dai, J. H. Christensen, V. Vyatkin, and V. Dubinin, "Function block implementation of service oriented architecture: Case study," in *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*, pp. 112–117, IEEE, 2014.
[13] P. Jhunjhunwala, U. D. Atmojo, and V. Vyatkin, "Towards implementation of interoperable smart sensor services in IEC 61499 for process automation," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, pp. 1409–1412, IEEE, 2020.
[14] P. Jhunjhunwala, J. O. Blech, A. Zoitl, U. D. Atmojo, and V. Vyatkin, "A design pattern for monitoring adapter connections in IEC 61499," in *2021 22nd IEEE International Conference on Industrial Technology (ICIT)*, vol. 1, pp. 967–972, 2021.
[15] Z. E. Bhatti, R. Sinha, and P. S. Roop, "Observer based verification of IEC 61499 function blocks," in *2011 9th IEEE International Conference on Industrial Informatics*, pp. 609–614, IEEE, 2011.
[16] H.-M. Hanisch, J. Thieme, A. Luder, and O. Wienhold, "Modeling of plc behavior by means of timed net condition/event systems," in *1997 IEEE 6th International Conference on Emerging Technologies and Factory Automation Proceedings, EFTA'97*, pp. 391–396, IEEE, 1997.
[17] ViVe, "Visual Verifier." http://vyatkin.org/tools/modelchekers.html, 2011. Accessed: 2021-09-21.

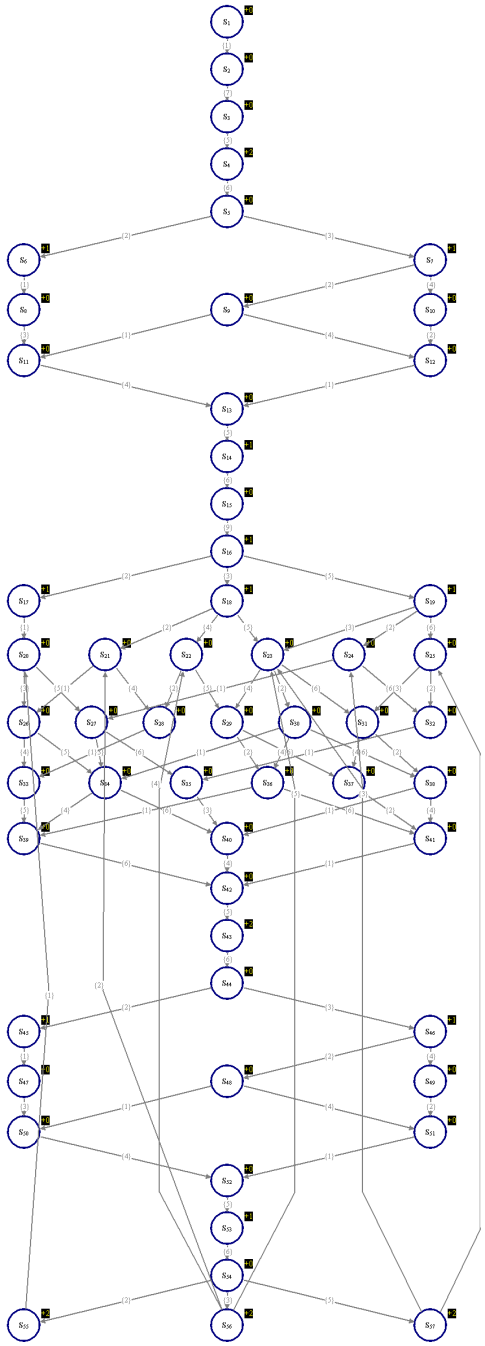

Fig. 9. Reachability graph of the model.

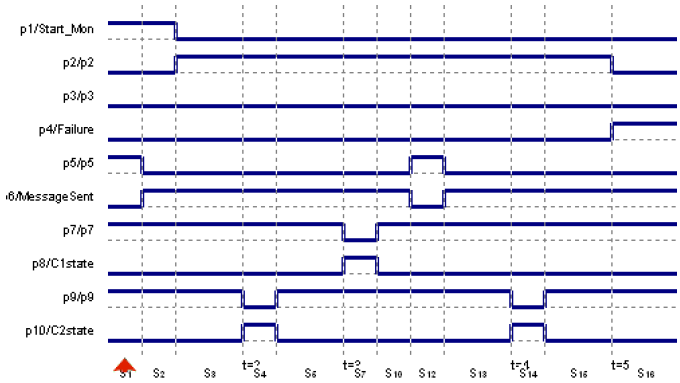

Fig. 10. Timing diagram, illustrating the failure detection scenario.