Distribution of Control Tasks to Smart Devices in Industrial Control Systems: a Case Study

Zivana Jakovljevic, *Member, IEEE* and Dusan Nedeljkovic

***Abstract*—Cyber Physical Systems (CPS) and Internet of Things (IoT) open the way for new generation of Industrial Control Systems (ICS) characterized by high flexibility, modularity and reconfigurability necessary within Industry 4.0. Inevitable shift from centralized to distributed control systems is underway, but the changes are not as rapid as expected. One of the limiting factors is the lack of engineering techniques for distributed control systems design, simulation and verification. In this paper we analyze recently proposed techniques for distributed control systems development using an example of a simple transport system consisting of two CPS – smart conveyor belt and smart cylinder. In particular we consider the methods based on Control Interpreted Petri Nets (CIPN), Supervisory Control Theory (SCT) and IEC 61499 standard.**

***Index Terms*—Distributed Control; Industrial Control Systems; Cyber Physical Systems; Industry 4.0.**

I. Introduction

Industry 4.0 and introduction of Cyber Physical Systems (CPS) at manufacturing shop floor lead to significant changes in Industrial Control Systems (ICS) [1]. The demand for highly flexible automation and reconfigurable manufacturing induced by fluctuating market needs and high product variety on one [2], and the development of CPS based systems as the leading enabling technology on the other hand [1] represent the main drivers of these changes. CPS based smart devices with integrated computational and communication capabilities open up new possibilities in terms of ICS modularity, flexibility and reconfigurability. It is expected that with the full extent implementation of CPS at manufacturing shop floor the traditional automation hierarchy standardized through IEC 62264 will be replaced with truly distributed control systems where the control tasks will be carried out through interoperability of networked smart devices – Fig. 1. It is expected that all elements of automation hierarchy will remain, but in terms of functional hierarchy distributed over network without the pyramidal structure of the corresponding devices [3].

CPS are already readily employed at manufacturing shop floors in different automation tasks primary as smart sensors and actuators, and strict automation hierarchy is already broken down as there exist communication of different devices intra and over non-adjacent levels of automation pyramid. Nevertheless, as a rule, smart sensor and actuators are integrated in ICS in traditional manner – they are connected to the central controller (e.g., Programmable Logic Controller - PLC) that carries out the control task. In this way, computational capabilities of CPS, their modularity and ability to make manufacturing systems adaptable to new products are not fully exploited.

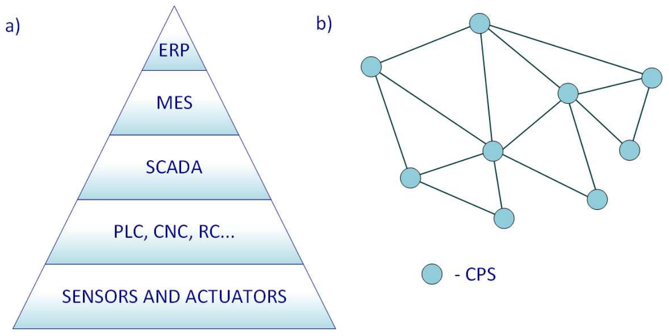


Fig. 1. Change of control paradigm in Industry 4.0: a) IEC 62264 automation hierarchy; b) Distributed control

There are several reasons for this. One is the inertia of control engineers community to implement new trends and their rationalle to keep the existing techniques for ICS design, that were practicaly tested and proven in a myraid of real-world examples. Yet, due to inability of traditional ICS to deal with high product variaty, there is a trend to perform a number of processes in manufacturing manually which represents a step backwards with respect to process sophistication; in this context, modularity of control system and distribution of control tasks are paramount for automation in Industry 4.0. The other reason is the lack of well-proven formaly-consequent engineering techniques for the desing of distributed control systems, i.e., for the distribution of control taks to smart devices [4]. Finally, when CPS are employed at manufacturing shop floor (regardless if centrally or distributed) cybersecurity related issues on communication links emerge.

Recently, a number of techniques for distribution of control tasks to smart devices and for dealing with cybersecurity within ICS have been proposed. Within this paper we will illustrate some of them using an example of simple system for parts transport that is presented in Section II. In particular, we will consider the method for control system distribution that is based on Control Interpreted Petri Nets (CIPN) that we proposed in [5], as well as the method based on Supervisory Control Theory [6] – Section III. Furthermore, in Section IV

Zivana Jakovljevic is with the University of Belgrade – Faculty of Mechanical Engineering, 16 Kraljice Marije, 11000 Belgrade, Serbia (email: zjakovljevic@mas.bg.ac.rs).

Dusan Nedeljkovic is with the University of Belgrade – Faculty of Mechanical Engineering, 16 Kraljice Marije, 11000 Belgrade, Serbia (email: dnedeljkovic@mas.bg.ac.rs).

we will show how the distributed control system can be simulated using IEC 61499 standard [7], and in Section V we will consider the possible effects of cyber-attacks on discrete event system using the methodology that we presented in [8]. Finally, Section VI gives some concluding remarks.

II. Description of the System and Control Task

The system that we consider in this paper is used for parts transport and consists of: 1) smart conveyer belt and 2) smart cylinder (Fig. 2). Conveyor belt transfers parts to position II and it is actuated by step motor M that is started/stopped using signal *m*. In addition, the belt contains sensor *s* that detects the presence of parts in removing position II, as well as start switch *st* that is used for the start of the system operation. Both sensors and actuator are connected to the belt's local controller denoted as LC_1. Smart cylinder, on the other hand, besides the actuator, contains a monostable 5/2 dual control valve controlled by signal *ap* (*ap* = 1 for cylinder advancing and *ap* = 0 for cylinder retracting), as well as limit switches $a1/a0$ for detecting final advanced/retracted position. Pneumatic and sensing devices are augmented with local controller LC_2 that controls the smart cylinder. The allocation of sensing and actuation signals to smart devices' local controllers is presented in Table I.

TABLE I
Sensors and Actuators Signals Mapping to Local Controllers

Device	Loc. Cont.	Signal	Description
Smart conveyer	LC_1	*m*	Motor actuation
		s	Sensor for part detection
		st	Start switch
Smart cylinder	LC_2	*ap*	Cylinder actuation
		$a1$	Advanced position sensor
		$a0$	Retracted position sensor

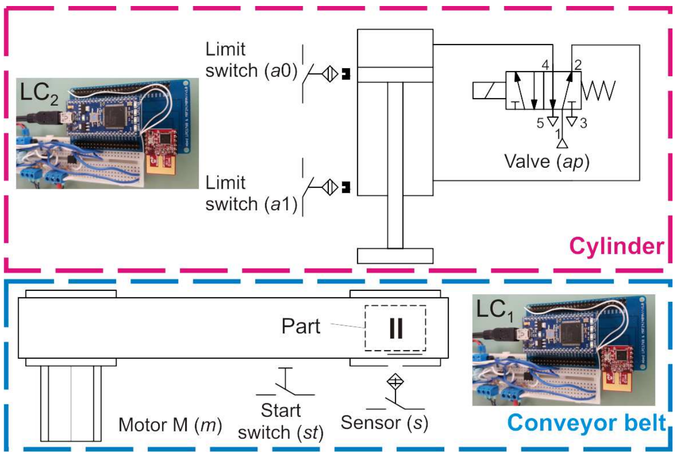


Fig. 2. Graphical representation of the system used in case study

The system should function as follows. After pressing the start switch, conveyor belt starts motion; when the part comes to the conveyor belt, it is transferred to position II where the sensor *s* is activated. After activation of signal *s*, conveyor belt stops and the cylinder removes the part from the belt (it advances and immediately retracts to home position). Once cylinder reaches the retracted position, the conveyor starts moving again to transfer new part to position II and the cycle repeats.

There are a number of different methods for formal description of the controller that would ensure the described system behavior when it is controlled using one (centralized) controller, e.g., PLC. Most frequently employed technique that represents *de facto* standard in ICS design are CIPN and the derived formalisms of GRAFCET (Graphe Fonctionnel de Commande des Étapes et Transitions - Functional Graph of Control by Steps and Transitions) and IEC 61131-3 Sequential Function Chart (SFC). CIPN represent bipartite graphs containing transitions represented by bars and places represented by circles [9]. Within CIPN each transition has associated condition as a Boolean function of sensory signals (although actuator signals can be used as well), whereas the actions which change the values of control system outputs are allocated to the CIPN places. The state of CIPN is represented by token(s) assigned to places, and during system evolution the token passes from previous to the next place when the transition between them fires, i.e., when the associated condition is true. Using CIPN formalism the desired performance of the system from Fig. 2 can be described using CIPN form Fig. 3; this CIPN can be easily transferred to SFC or other embedded devices programming languages and implemented in centralized PLC or some other device (e.g., microcontroller).

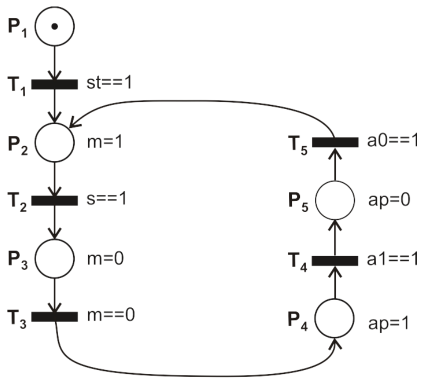


Fig. 3. CIPN describing the desired behavior of the system from Fig. 2.

III. Distribution of Control Tasks to Smart Devices

Opposite to centralized control systems design, generally, there is a lack of methods for the design of distributed controllers [10]. Existing approaches can be classified as *bottom-up* and *top-down*. Using *bottom-up* methods the behavior of each device within the control network is described using Petri nets [11], Supervisory Control Theory (SCT) [12], IEC 61499 [7] or similar formalisms, and the behavior of the system as a whole is obtained through their composition. These methods are characterized by low backwards compatibility since their implementation necessitates completely new mindset in system designers. Furthermore, they are usually based on trial and error and as such are time consuming and require significant verification [4]. *Top-down* approaches, on the other hand, describe the system as if centrally controlled and, using predefined methodology, distribute control tasks to smart devices. These methods are characterized by high backwards compatibility and can be easily embraced by engineers in everyday practice.

A *top-down* approach from [5] is based on CIPN. Global CIPN describing the behavior of the system as a whole and mapping of sensors and actuators to local controllers with direct access to these devices are at the input of this method. Following the set of rules that consider the allocation of sensing and actuation signals to transitions and places within global CIPN on one hand, and physical mapping of sensors and actuators to local controllers LC_i on the other, this method generates a separate $CIPN_i$ to be employed at each LC_i. Basically, this approach extracts from global CIPN into $CIPN_i$ the places that contain actuating signals allocated to LC_i along with preceding transitions, and introduces places with sending commands to compensate missing links. The details regarding the method can be found in [5].

Following this approach, for the CIPN from Fig. 3 and allocation of sensing and actuation signals from Table I, **CIPN$_1$** and **CIPN$_2$** presented in Fig. 4 are obtained. Within these CIPNs, the transitions and places that were extracted from global CIPN (Fig. 3) are denoted in parentheses. The actions associated to places within **CIPN$_1$** and **CIPN$_2$** contain only actuating signals mapped to corresponding devices (conveyor belt and cylinder, respectively). On the other hand the transitions can contain the signals allocated to the other device, as denoted red in T_4^1, T_{init}^2 and T_4^2. These signals are transferred between local controllers using selected communication protocol, and protocol agnostic sending commands are denoted green in corresponding places in Fig. 4. These commands are the result of the application of the procedure from [5].

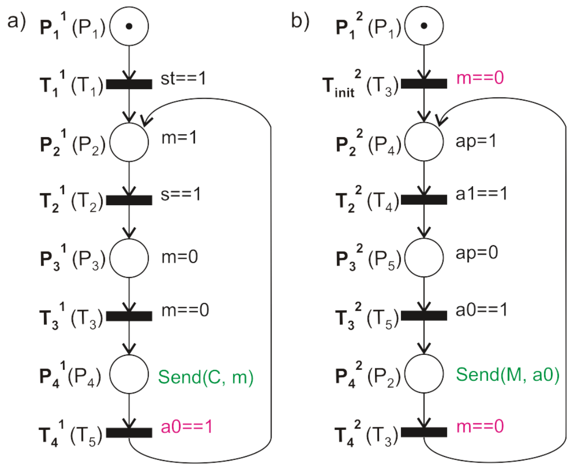


Fig. 4. CIPN representation of the system control distributed to smart devices: a) **CIPN$_1$** representing control tasks distributed to smart conveyer belt; b) **CIPN$_2$** representing control tasks distributed to smart cylinder;

For example, while at P_4^1 smart conveyor belt sends information about motor stopping to smart cylinder which receives it in transition T_{init}^2 or T_4^2 depending on the current state; in this way T_3 from Fig. 3 is performed. Similarly, while at P_4^2 smart cylinder sends information about getting to retracted position to smart conveyer belt, which the latter receives at transition T_4^1, delivering the T_5 from Fig. 3. Once CIPNs for local controllers are generated, their implementation is straightforward as in the case of centralized controllers.

An example of *bottom-up* approach for distribution of control tasks will be illustrated using SCT formalism [6]. The models of the local controllers – supervisors for smart devices can be presented by Finite State Machines (FSM) in which the transition between states occurs on the events that represent the change of certain (actuation or sensory) signal. FSMs representing the desired behavior of LC_1 and LC_2 are given in Fig. 5. In particular, S_B represents smart conveyer belt (Fig. 5a) and S_C (Fig. 5b) smart cylinder cyber part. Note that in smart conveyer belt FSM model motor activation signal (m = 1) is denoted by *mp*, whereas deactivation signal (m = 0) is denoted *mm*. Similarly, for smart cylinder ap = 1 is denoted by *ap* and ap = 0 by *am*. The signals that are transferred from a local controller are marked green, and the signals that are received are marked red.

As can be observed from the Fig. 5, the design of local controllers using this formalism is not straightforward and requires to simultaneously take care about the behavior of both devices and communication of signals between them which can be extremely tedious and error prone when larger number of local controllers is used. The equivalence of S_B and S_C to **CIPN$_1$** and **CIPN$_2$** from Fig. 4 can be observed.

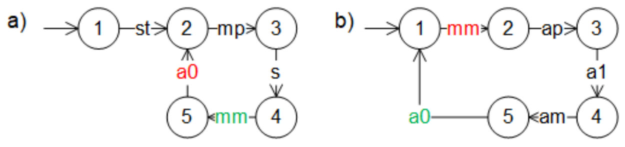


Fig. 5. Case study from Fig. 2 – FSMs representing: a) Conveyer belt local controller – S_B; b) Cylinder local controller – S_C;.

To verify that the system as a whole will have the desired behavior after implementation of the developed controllers, using SCT formalism, the physical part of the system should be modeled as well. FSMs from Fig. 6 - G_B and G_C represent physical parts of smart conveyer belt and smart cylinder, respectively.

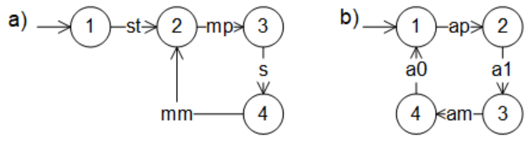


Fig. 6. Case study from Fig. 2 – FSMs representing: a) Conveyer belt physical device - G_B; b) Cylinder physical device - G_C.

Conjoint operation of all four FSMs from Figures 5 and 6 is presented by FSM G from Fig. 7, where x, y, z and v in state notation (x, y, z, v) represent the states of S_C, S_B, G_C and G_B, respectively. FSM G, that represents the behavior of the CPS from Fig. 2, is obtained using the following FSM operation:

$$G = (S_B \,||\, S_C) \times (G_B \,||\, G_C) \qquad (1)$$

where || denotes FSM parallel composition, and × product operator [6][1]. Comparing CIPN from Fig. 3 which represents the desired behavior of the system and FSM from Fig. 6 that represents the conjoint behavior of smart devices with distributed control tasks, the equivalence can be observed. Nevertheless, it should be noted that the implementation of SCT formalism assumes that CIPNs are not used during

[1] All automata operations are carried out using DESUMA software [13]

system representation, and in this paper we use it for the comparison only.

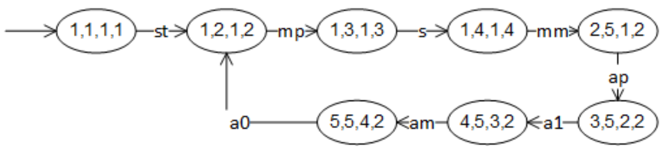


Fig. 7. Case study from Fig. 2 – behavior of the system represented using single FSM – *G*

IV. Modeling and Simulation of Distributed Control Systems using IEC 61499

Once the control task is distributed to smart devices' local controllers, it is beneficial to further verify it, preferably using simulation. IEC 61499: Function blocks [7] represents an international standard intended for distributed control system modeling and simulation. Using this standard the functionality of a system's hardware or software component is encapsulated using function blocks (FBs) that introduce object oriented paradigm into industrial control systems programming [14, 15]. FBs represent classes whose instances can be used for task execution in concrete applications. Using this formalism, CPS are modeled and simulated through interaction between their physical and cyber parts, each represented by separate FB. The blocks are integrated through real-time interaction that is modeled using events and data flows. The functionality of an FB is event driven and it is defined by its Execution Control Chart (ECC) that specifies the behavior of the modeled component when certain events in the system occur. They are in the form similar to CIPN.

IEC 61499 formalism models the behavior of the system through a network of FBs called application. Within application each smart device is introduced using corresponding FBs and multiple devices are interconnected using certain events and data. Fig. 8 represents IEC 61499 application for the system from Fig. 2[2]. Within this application function blocks ConveyCyber and ConveyPhys model cyber and physical parts of the smart conveyer, whereas CylCyber and CylPhys model corresponding parts of smart cylinder. Each FB contains head (upper part) that contains input (left side) and output (right side) events and body that contains input (left side) and output (right side) data. FB refreshes input/output data on the occurrence of the corresponding event, and their interconnections are modeled with connectors - red for event and blue for data flow.

Fig. 9 presents the ECCs for FBs modeling the behavior of the smart conveyor. At the system start, ConveyPhys transfers to S1, activates start switch (action *Start*) and issues **CNF** event to invoke the change of corresponding ConveyPhys output and ConveyCyber input; after that, it waits for **REQ** event and receipt of signal for motor activation (*mot* = *true*) from ConveyCyber when it passes to S2. Within S2 it activates sensor (action *Sensor_act*), issues **CNF** event to change corresponding input in ConveyCyber and waits for **REQ** events with the desired input data from ConveyCyber to return to S1.

[2] For IEC 61499 modeling and simulation 4diac software [16] is used.

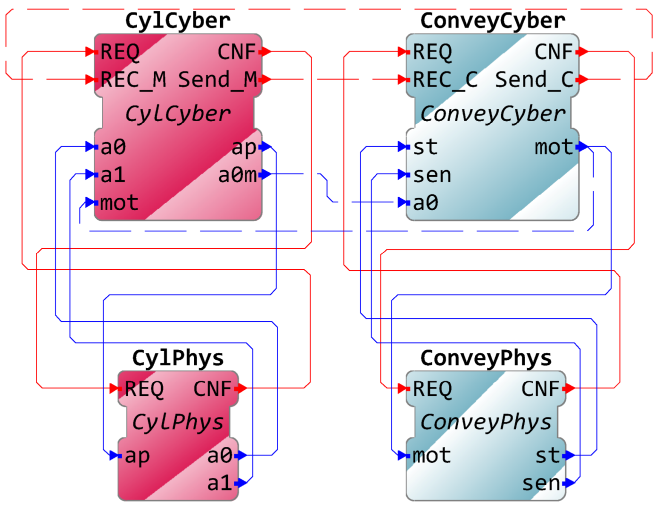
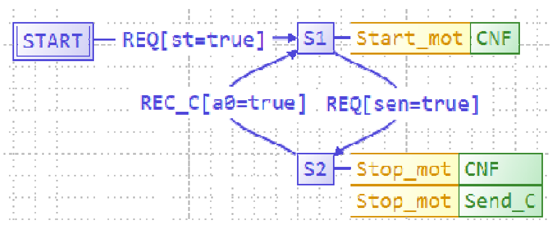
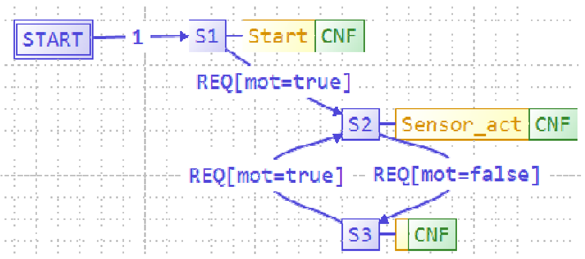


Fig. 8. IEC 61499 application that models the behavior of the system from Fig. 2.

a)



b)



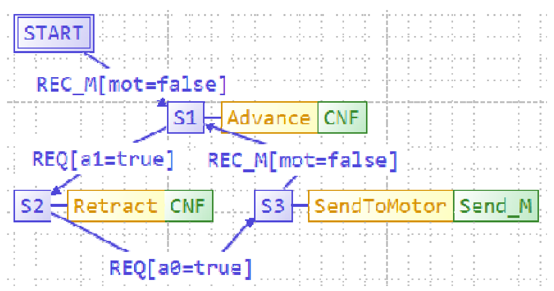
c)

ConveyCyber actions	
Start_mot	mot:=**true**;
Stop_mot	mot:=**false**;
ConveyPhys actions	
Start	st:=**true**;
Sensor_act	sen:=**true**;

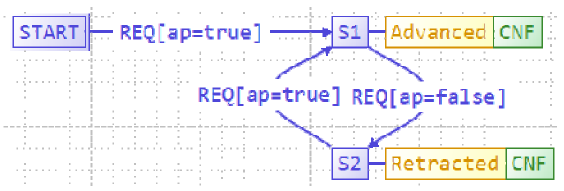
Fig. 9. ECCs for conveyer belt: a) cyber part of the conveyer belt - ConveyCyber ; b) physical part of the conveyer belt ConveyPhys; c) actions definition.

Simultaneously, at the beginning, ConveyCyber waits for **REQ** event (**CNF** event from ConveyPhys – Fig. 8) and the value *st* = *true* from ConveyPhys; on the receipt of this data it transfers to S1 where it carries out *Start_mot* action and invokes **CNF** event to inform the ConveyPhys (which is at that moment in S1 or S3) about signal change. Afterwards it waits in S1 information from ConveyPhys that sensor is activated to transfer to S2. In S2 it invokes the action for motor stopping and informs ConveyPhys and CylCyber about this change using events **CNF** and **Send_C**. Finally, after receipt of information that cylinder removed the part from the belt from CylCyber through event **REC_C** and data $a0 = true$, it returns to S1.

a)



b)



c)

CylCyber actions	
Advance	ap:=**true**;
SendToMotor	a0m:=**true**;
Retract	ap:=**false**;
CylPhys actions	
Advanced	a0:=**false**; a1:=**true**;
Retracted	a0:=**true**; a1:=**false**;

Fig. 10. ECCs for cylinder: a) cyber part of the cylinder - CylCyber ; b) physical part of the cylinder CylPhys; c) actions definition.

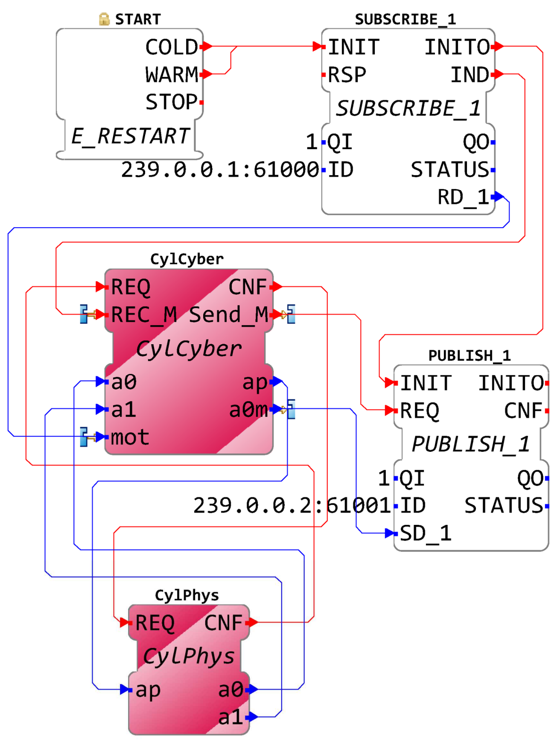


Fig. 11. Cylinder resource with introduced Publish and Subscribe function blocks for simulation of the system performance.

Similarly (Fig. 10), at the begining CylCyber waits for the information from ConveyCyber that motor stopped (**REC_M** event that is connected to **Send_C** from ConveyCyber – Fig. 8), moves to S1, invokes cylinder advancing (action *Advance*), and upon receipt of the event and data referring to the completion of CylPhys advancing, issues commands for its retraction. When it gets the information that CylPhys retracted, CylCyber issues **Send_M** event to inform CylCyber that motor can start motion, and waits for the new signal from ConveyCyber that motor stopped (**REC_M** event connected to **Send_C**) to return to S1 and prepare for the new cycle. Finally, CylPhys moves between states corresponding to advanced and retracted positions in accordance with the events and data received from CylCyber.

Comparing these ECCs with CIPNs and FSMs from Figures 4, 5 and 6, the equivalence can be observed.

To simulate the behavior of the control system defined by application (Fig. 8) function blocks are allocated to different resources and the communication is introduced through publish and subscribe FBs as presented in Fig. 11 using an example of smart cylinder.

V. Security Related Issues

As can be observed from previous sections, the performance of distributed control system is communication intensive, thus openings up the space for malicious cyber-attacks by various adversaries and raising cyber security related issues. A number of different attack scenarios can be identified and in case of discrete event systems such as one at hand two kinds of attacks can be singled out: 1) event removal and 2) event insertion. The adversaries can issue these attacks in various combinations always having the common goal to remain stealthy and to have negative effect on the system performance. Considering the possible consequences of cyber-attacks on ICS that can be not only of economic nature but also safety related (catastrophic damages of the equipment or even threats to human health and lives), it is crucial to analyze possible attack points, scenarios and consequences at early phases of the system design. Within this paper we will briefly illustrate an approach for modeling attacks scenarios using SCT based methodology that we proposed in [8]. We will consider the examples of removal and insertion attacks on $a0$ signal transmitted from smart cylinder to conveyor belt.

Following $a0$ removal attack denoted by $a0r$, smart conveyor belt cyber part will remain in the state at which it waits for $a0$ to progress with functioning, whereas smart cylinder cyber and physical parts will continue functioning as if attack did not occur. Described evolution of the system can be modeled using FSMs under attack S_B^r, S_C^r and G_C^r presented in Fig. 12a-c. In these automata the evolution of the system elements under attack is denoted in red lines. The details of the model generation can be found in [8], and the performance of the system as a whole is represented by automaton:

$$G^r = \left(S_B^r \,||\, S_C^r\right) \times \left(G_B \,||\, G_C^r\right) \qquad (2)$$

which is presented in Fig. 12d. From this figure it can be observed that on the *a*0 removal attack the system stops (it enters the state marked red) and no damage is expected.

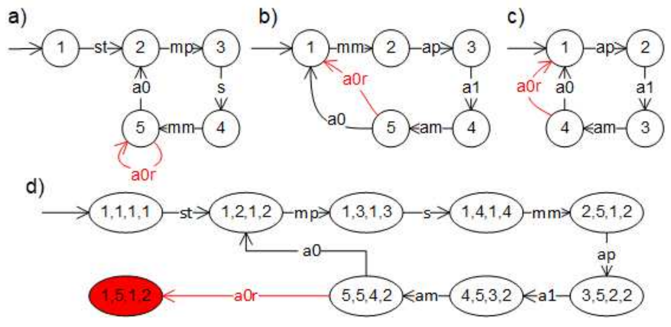


Fig. 12. Model of the system under *a*0 removal attack: a) S_B^r – model of conveyor cyber part; b) S_C^r – model of cylinder cyber part; c) G_C^r – model of cylinder physical part; d) model of the whole system behavior under attack.

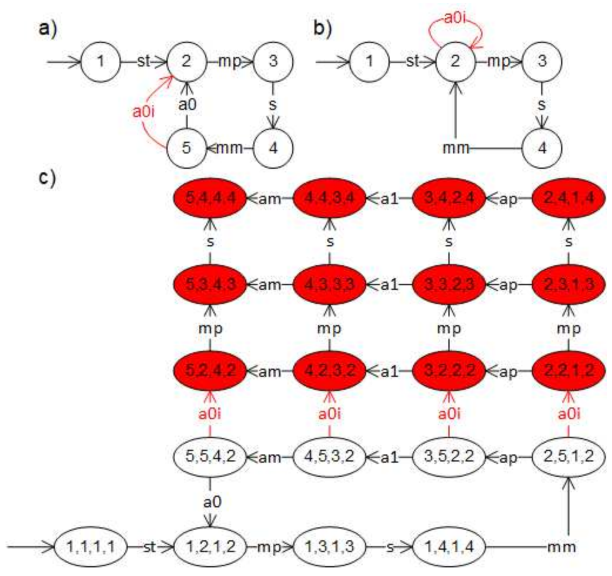


Fig. 13. Model of the system under *a*0 insertion attack: a) S_B^i – model of conveyor cyber part; b) G_B^i – model of conveyor physical part; c) model of the whole system behavior under attack.

When insertion attack is considered, it should be noted that to remain stealthy, the adversary can issue the *a*0 insertion attack only while S_B is in state 5 [8]. Following insertion attack *a*0*i*, the conveyer belt supervisor will transfer from state 5 to state 2 as if real *a*0 was received as presented in red line in FSM S_B^i in Fig. 13a. Simultaneously, conveyor physical part will remain in the state 2 as given in automaton G_B^i from Fig. 13b (the details regarding the formalism for generation of these automata can be found in [8]). The behavior of the whole system under *a*0 insertion attack is modeled through:

$$G^i = \left(S_B^i \,||\, S_C\right) \times \left(G_B^i \,||\, G_C\right) \tag{3}$$

presented in Fig. 13c where the possible evolutions of the system after attack are denoted using red states. These scenarios include starting the motor before the part is removed from the conveyer belt and can lead to the falling of the part from the transporter and its damage.

VI. Conclusion

Within this paper we have summarized and illustrated using a case study our recent research results in the area of CPS based distributed control systems design, verification and cyber protection. In particular, we have presented the application of the *top-down* approach for distributed control system design that is based on CIPN and that is characterized by excellent backwards compatibility. The convenience of this technique is supported through illustrating the application of a *bottom-up* technique based on SCT that requires completely new approach to ICS modeling. Furthermore, we have shown how the developed distributed control system can be simulated using standard IEC 61499. Finally, since the deployment of CPS and IoT at manufacturing shop floor leads to significant cyber security issues, we have shown how the SCT based framework can be applied for the analysis of cyber threats in our case study.

Acknowledgment

This research was supported by the Science Fund of the Republic of Serbia, grant No. 6523109, AI-MISSION 4.0

References

[1] H. Kagermann, W. Wahlster, and J. Helbig, "Recommendations for Implementing Strategic Initiative INDUSTRIE 4.0," Acatech, Germany, 2013 [Online]. Available: http://www.acatech.de

[2] H. ElMaraghy, "Smart changeable manufacturing systems." *Proc Manuf*, vol. 28, pp. 3-9, 2019.

[3] Reference Architecture Model Industrie 4.0 (RAMI4.0), 2015. [Online]. Available: fhttp://www.zvei.org

[4] J. Otto, B. Vogel-Heuser, and O. Niggemann, "Automatic parameter estimation for reusable software components of modular and reconfigurable cyber-physical production systems in the domain of discrete manufacturing," IEEE *Trans. Ind. Informat.*, vol. 14, no. 1, pp. 275–282, Jan. 2018.

[5] Z. Jakovljevic, V. Lesi, S. Mitrovic, and M. Pajic, "Distributing sequential control for manufacturing automation systems," *IEEE Trans. Contr Syst Tech.*, vol. 28, no. 4, pp. 1586-1594, July 2020.

[6] P. Ramadge and W. Murray Wonham, "The control of discrete event systems," *Proc. IEEE*, vol. 77, no. 1, pp. 81–98, 1989.

[7] *Function blocks – Part 1: Architecture*, IEC 61499-1:2012.

[8] Z. Jakovljevic, V. Lesi, and M. Pajic, "Attacks on Distributed Sequential Control in Manufacturing Automation," *IEEE Trans. Ind. Informat.*, vol. 17, no. 2, pp. 775-786, Feb. 2021.

[9] R. David and H. Alla, *Discrete, Continuous, and Hybrid Petri Nets*, 2nd ed. Berlin, Germany: Springer, 2010.

[10] R. Vrabič, D. Kozjek, A. Malus, V. Zaletelj, and P. Butala, "Distributed control with rationally bounded agents in cyber-physical production systems," *CIRP Ann.*, vol. 67, no. 1, pp. 507–510, 2018.

[11] L. Wang, C. Mahulea, and M. Silva, "Distributed model predictive control of timed continuous Petri nets," in Proc. IEEE Conf. Decis. Control, pp. 6317–6322, Dec. 2013.

[12] M. Wakaiki, P. Tabuada, and J. P. Hespanha, "Supervisory control of discrete-event systems under attacks," Dynamic Games and Appl., vol. 9, pp 965-983, Dec. 2019.

[13] L. Ricker, S. Lafortune, and S. Genc, "Desuma: A tool integrating giddes and umdes," in Proceedings - Eighth International Workshop on Discrete Event Systems, WODES 2006, pp. 392–393, 2006.

[14] J. H. Christensen, T. Strasser, A. Valentini, V. Vyatkin, and A. Zoitl, "The IEC 61499 Function Block Standard: Software Tools and Runtime Platforms," in ISA Automation Week, 2012.

[15] V. Vyatkin, *IEC 61499 Function Blocks for Embedded and Distributed Control Systems Design*, ISA, 2012.

[16] https://www.eclipse.org/4diac/ Accessed: 2020-04-24