A modeling approach for Cyber-Physical Systems based on collaborative processes

Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 52-13 (2019) 2764–2769

A modeling approach for Cyber-Physical Systems based on collaborative processes

Giorgio Bruno

DAUIN, Politecnico di Torino, Torino, Italy
(e-mail: giorgio.bruno@ polito.it)

Abstract: Based on the Industry 4.0 initiative, this paper presents a modeling approach for Cyber-Physical Systems (CPS) that is based on collaborative processes. The approach is called MA4.0 (Modeling Approach 4.0) and is made up of four kinds of models: the system model, the collaboration models, the component models, and the process models. The system model shows the types of the components of the system and their relationships. The components may be devices and software systems. The relationships denote collaboraboration models that consist of message-driven interactions and control blocks. Component models show the structure of components: software systems are made up of internal processes while devices also include operating units. Processes are event-driven and leverage the data flow instead of the control flow. The approach is illustrated with the example of a manufacturing cell.

Keywords: Cyber-Physical Systems, Industry 4.0, manufacturing cell, process models, collaboration models.

1. INTRODUCTION

Industry 4.0 is a disruptive German initiative whose aim is to make the industrial world leap forward by taking advantage of information and communication technologies (Hermann et al., 2016). Such technologies include Internet of Things (IoT) and Cyber-Physical Systems (CPS).

IoT is a class of systems in which standard protocols are used to connect not only software systems but also devices of various kinds, i.e., the "things" (Atzori et al., 2010). Devices connect the physical world to the digital one because they are able to send information, respond to queries and execute commands.

IoT has spread to the industrial sector (Xu et al., 2014). In this context, the devices are equipped with internal processes that allow them to collaborate with other devices or software applications, to record data of interest and to perform functional diagnostics. Devices with such features are often referred to as smart objects (Kortuem et al., 2010).

IoT systems are part of CPS because CPS are meant to integrate "computation and physical processes" (Lee, 2008).

In their evolution, CPS have become software intensive and it is expected that they will behave in an increasingly intelligent way thanks to computational algorithms and machine learning capabilities (Müller, 2017).

An example of CPS that will be illustrated in this paper is a manufacturing cell made up of a number of machine tools, a warehouse, an agv (automated guided vehicle) that moves parts from the warehouse to the machine tools and vice versa, and the control system, which executes production orders coming from the plant management system. The cell is an IoT system where the machine tools, the warehouse and the agv are high-level devices (or smart objects). At the same time, it is a CPS because all the components include internal processes that interact with each other to make the cell an intelligent manufacturing system according to the recommendations for implementing Industry 4.0 (Kagermann et al., 2013).

Among the various lines of research spurred by Industry 4.0, IoT and CPS, the one this paper belongs to concerns the conceptual modeling of CPS.

Conceptual modeling means representing a system in an abstract way with the aim of highlighting the aspects that are deemed most important. Those aspects are the architecture of the system, the interactions between the components and the internal processes of the components. The models are abstract in that the implementation details are ignored.

This paper proposes an approach, called MA4.0 (Modeling Approach 4.0) whose major contributions are as follows. The structure of the system is given by a model that shows the types of the components and their relationships. A relationship connects two different types of components and represents their collaboration, which is described by means of a collaboration model. A collaboration model defines the sequence of messages exchanged between two parties and includes the descriptions of the payloads of the messages. A component may be involved in a number of collaborations that are handled by its internal processes. The internal processes are collaboration-aware in that their logic has to comply with the sequences of messages defined in the collaborations the component takes part in.

Processes are mainly event-driven where the events correspond to the input messages. For this reason, their structure leverages the data flow instead of the control flow.

10.1016/j.ifacol.2019.11.626

The data flow establishes the data path in the process and then it consists of the input and output data of the tasks handled by the process. It may happen that the same data enables two or more mutually exclusive tasks: the choice of the task to perform may be automatic or human. In the second case, it is up to a person playing the role associated with the choice to select the task to perform. The notation of the processes draws on high-level Petri nets and leverages the ELBA language (Bruno, 2018); human choices are modeled with a specific construct.
This paper is organized as follows. Section 2 concerns the related work. Section 3 gives a summary of MA4.0, and section 4 illustrates the requirements and the models related to the above-mentioned example of a manufacturing cell.

2. RELATED WORK

As the modeling language of CPS processes, most of the papers have chosen BPMN (Business Process Model and Notation) which is the de facto standard for business process modeling, and have added some extensions to handle the physical devices, such as sensors and actuators. A solution consists in introducing specific tasks to get inputs from sensors and send commands to actuators (Meyer et al., 2013).
Another solution is to associate resources (i.e., IoT devices) to BPMN tasks. In Martins and Domingos (2017), a BPMN model is translated into a programming language for IoT devices.
Bocciarelli et al. (2017) notice that devices must be able to interact with each other, and therefore they introduce the possibility of describing such interactions as process annotations.
Another way of interfacing processes and devices takes advantage of BPMN events. A simple solution is to map events produced by devices to BPMN events (Mandal et al., 2017). A more comprehensive approach leverages the context to decide how to respond to events received from IoT devices. The context links the events to the devices and the devices to the entities of the system; moreover, it defines the relationships between the entities. Song et al. (2018) draws on an ontology to model the context and represent the processes with CAPN (Context-Adaptive Petri nets) (Serral et al., 2015), which allows an ontology to be associated with Colored Petri Nets (Jensen et al., 2007).
BPMN provides pools to show two or more processes and their interactions in the same model. This feature can be exploited to represent CPS digital and physical processes (Graja et al., 2016).
The architectures of CPS and IoT systems have been addressed with reference models, such as IoT-ARM (IoT - Architecture Reference Model) (Bauer et al., 2013) and RAMI 4.0 (Reference Architectural Model Industrie 4.0) (Hankel, and Rexroth, 2015).
A reference model is an abstract framework that encompasses a basic set of unifying concepts. IoT-ARM provides three views: the Functional view addresses the functional building blocks, the Deployment and Operation view is about the behavior of the functional components, and the Information view addresses the information flow through the system.
RAMI 4.0 offers a three-dimensional model: axis 1 shows a hierarchical perspective in which factories are located; axis 2 is about the life cycle of products from design to maintenance; axis 3 addresses the architectural layers that include the Business layer, the Functional layer, and the Information one. Pisching et al. (2018) shows how to define the architecture of a production system that is able to match equipment to parts according to the requirements of part types. RAMI 4.0 does not address the modeling of the processes of CPS; therefore, research has committed to fill the gap with various proposals compliant with the reference architecture. BPMN is used by Suri et al. (2017), Petri nets by Pisching et al. (2018), and S-BPM (Subject-oriented Business Process Management) by Friedl (2018).
Other techniques that are used to model the architecture of CPS are C&C (Component and Connector) models and SysML (Systems Modeling Language). An example of the first technique is given by Kusmenko et al. (2017), while Huang et al. (2018) show an example of the second one.

3. INTRODUCTION TO MA4.0

This section illustrates the MA4.0 modeling approach to CPS.
The basic assumptions are as follows. CPS consist of components that can be devices or software systems. Components interact through messages that are sent and received by their internal processes. The sequence of messages exchanged between two components is defined by a collaboration model, which includes the descriptions of the payloads of messages.
The structure of MA4.0 models is made up of the system model, the collaboration models, the component models, and the process models. The next section gives examples of the models.

3.1 The system model

The system model presents the types of the components and their relationships. Relationships are binary, i.e., they link two types of components, and denote collaboration models. The system model is based on the class diagram provided by UML (Unified Modeling Language) with the difference that the labels of relationships are names of collaboration models.
Component types can be given attributes, such as a unique identifier or a location (for devices). Based on the system model, the configuration of a specific system can be generated by defining the number of components, their attributes and their associations.

3.2 Collaboration models

Collaboration models draw on UML interaction models. They consist of interactions and control blocks. An interaction represents a message and has a name and an optional payload that is defined in the annotations of the collaboration model.
There are three kinds of control blocks: iterative blocks, optional ones and blocks including a number of alternative

sub-blocks. The kinds of the blocks are indicated by the keywords loop, opt, alt, respectively.
The relationships in the system model are oriented: the source element is the component that starts the collaboration, i.e., the one that sends the first message.

3.3 Component models

The model of a device presents the internal processes, the operating units, and their relationships (which refer to internal collaboration models). The processes include the control process (which governs the functionality of the device) and the secondary processes, which deal, for example, with the configuration and initialization of the device. The operating units (sensors, actuators or specific equipment) perform various operations on the underlying physical objects.
The processes handle external collaborations and internal ones. The external collaborations are those defined in the system model, while the internal ones establish the interactions between the internal processes and those between the internal processes and the operating units.
The model of a software component does not include operating units.
An information model can be associated with a component model so as to show the types of the entities (with their attributes and relationships) handled by the internal processes.

3.4 Process models

In MA4.0, process models draw on an extension to Petri nets that associates entities with tokens. Entities refer to system components, messages or internal data of processes.
Entities are contained in places and places belong to three categories: normal places, input places and output ones.
Normal places contain entities that refer to system components, such as a machine or an AGV, or to internal data. Their labels consist of the name of the type of the entities that can be included, followed by an optional state name (appearing withing parentheses). The states represent the stages of the entity life cycles. More information on entity life cycles can be found in the paper by Bruno (2018).
Input and output places can be collectively referred to as collaboration places: they represent the messages that a process receives and sends according to the collaborations it is involved in.
The label of a collaboration place consists of the message name preceded by the collaboration name. If all the collaboration places of the same collaboration can be arranged in a swim lane, then the collaboration name is written in the header of the swim lane and it is omitted from the labels of the places.
The symbol of a normal place is a circle. The symbol of an input (output) place is a circle with a black (white) inner circle. Examples of process models are given in Fig. 4 and Fig. 5.
If a normal place contains initial tokens, its symbol is a circle with a thick edge. Initial tokens are defined by means of initialization rules.
Transitions in a process model represent tasks and their symbol is a rectangle that includes the task name. Human tasks are accompanied by the name of the role of the persons entitled to perform them; the role name is show near the task symbol.
The constraints and the results of tasks are expressed in a declarative way based on pre-conditions and post-conditions. Due to lack of space, they are not shown in the examples presented in the next subsections.
The choice of using extended Petri nets instead of BPMN is motivated by the following reasons. In BPMN, the dataflow is separated from the control flow, while, in MA4.0 process models, the dataflow includes the control flow in that the execution of tasks depends on the presence of suitable entities in their input places. The dataflow consists of the input and output places of tasks.
MA4.0 process models provide some features, such as matching tasks and human choices, which are not easy to handle with BPMN. A matching task has two or more input places and is able to select entities from those places on the basis of a pre-condition. An example is given in Fig. 5.
A human choice is needed when the same input entity can be handled with a number of mutually exclusive tasks and the choice of the suitable task is made by a person who is entitled to do so, on the basis of the role required by the choice. A new construct has been introduced to represent such choices. It consists of a block that encloses the tasks and has a label containing the role of the performer and the name of the choice. An example is shown in Fig. 5. Choices can be more complex as explained in Bruno (2015).

4. CASE STUDY AND MA4.0 MODELS

This section illustrates MA4.0 models with help of a case study concerning a manufacturing cell that is part of a plant. The cell consists of four machine tools, a warehouse, an agv, and is managed by a software control system (CCS, Cell Control System). The CCS interacts with the cell components and with two external software systems, namely the plant control system (PCS, Plant Control system) and the maintenance unit (MU, Maintenance Unit). In addition, the agv interacts with the machines and the warehouse.
It is assumed that the system functionalities are described with two classes of requirements, the general requirements and the specific ones. The system model and the collaboration models are obtained from the former and the models of the components from the latter.

4.1 General requirements, system model and collaboration models

This subsection focuses on the interactions between the components of the system. In particular, it deals with production orders, machining operations, missions of the agv, and maintenance requests. The requirements are as follows.
The CCS receives production orders with PO (Production Order) messages coming from the plant control system (PCS). Each order contains the order code and the type and number of parts to be produced. The CCS informs the PCS of the completion of an order with the POcompleted message, which carries the order code in its payload.

Each part requires a single machining operation. Machine tools work one part at a time. When they are ready to receive a new part, they send the LRequest (LoadingRequest) message to the CCS, and when they are done, they send the URequest (UnloadingRequest) message. For a loading (unloading) request, the CCS assigns a loading (unloading) mission to the agv with message LMission (UMission). The payloads of these messages contain the code of the part type.
The agv performs two types of missions: the loading mission involves the transport of a raw part from the warehouse to a machine tool and the unloading mission involves the transport of a machined part from a machine tool to the warehouse. When a mission has been completed, the agv informs the CCS with message MissionDone.
The agv informs the machine tool (or the warehouse) when the part has been loaded or unloaded with messages PartLoaded and PartUnloaded respectively. The messages contain the code of the part type in their payloads. When the CCS launches a loading mission, it also sends message RawPartNeeded to the warehouse. This message contains the code of the part type in its payload so the agv will get the raw part of the required type.
Machine tools can send maintenance requests to the CCS: they do so by means of message MRequest (Maintenance Request) that may be issued after an unloading request (URequest). The CCS replies in two ways. If it accepts the request, it informs the machine tool with message MRaccepted, and sends message MReq to the MU (Maintenance Unit) with the code of the machine tool in its payload. At the end of the maintenance, the MU replies with message MReqServed, which includes the code of the machine tool in its payload, and the CCS informs the machine tool with message MRserved. Then, the machine tool can resume its work cycle. If the CCS does not accept a maintenance request, it postpones it. In that case, it replies to the machine tool with message MRdelayed, which specifies the amount of delay in its payload. Then, the machine tool resumes its work cycle and issues a new maintenance request at the end of the delay.
The system model of the cell is shown in Fig.1: it includes the types of the components and their relationships. In particular, the type of machine tools is M, and that of the warehouse is W. The attributes of the cell components are omitted for simplicity. The labels of relationships are names of collaboration models.

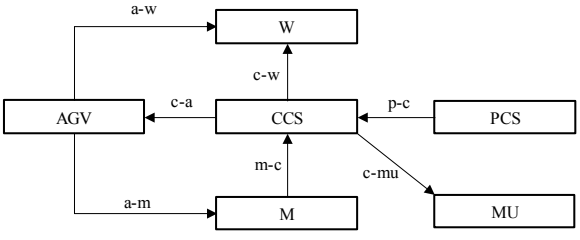

Fig. 1. The system model of the cell

The collaboration between the machine tools and the CCS (m-c), and the one between the agv and the machine tools (a-m) are defined in the models shown in Fig.2. The collaboration models draw on UML interaction diagrams. They consist of interactions and control blocks. An interaction has a name and may include a payload. Payloads are defined in the annotations of the collaboration model. There are two examples of payload in Fig.2: the period of the delay in message MRdelayed and the part type code (ptId) in message PartLoaded.
The interactions in the above-mentioned collaborations are iterative and then they are framed in a loop.
As to the collaboration model m-c, a machine tool continuously performs three work sequences. The main one consists of a loading request and an unloading one. Two variants depend on whether the CCS accepts the request or postpones it.

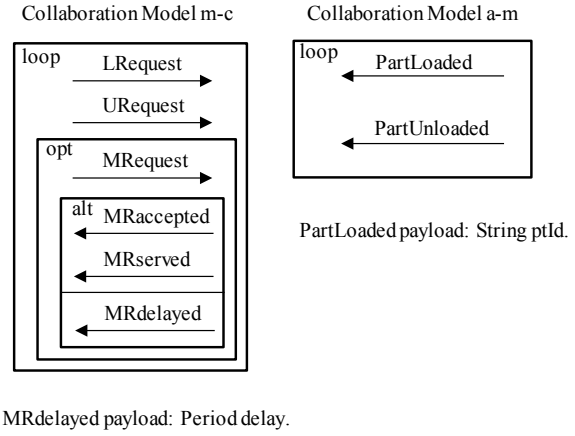

Fig. 2. The models of collaborations m-c and a-m

4.2 Requirements and model of the machine tool

A machine tool is a device including an operating unit that is capable of machining parts of different types by choosing the part program corresponding to the type of the part to be machined. The model of the machine tool is shown in Fig.3: it includes the operating unit (PMT, Physical Machine Tool), the control process and the relationship (c-p) between the two components.

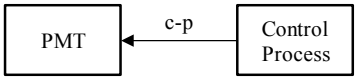

Fig. 3. The components of a machine tool

The work cycle of the control process is as follows. The process starts a new work sequence by sending a loading request (LRequest) to the CCS and then waits for the arrival of the raw part brought by the agv; the presence of the part is notified with message PartLoaded, which carries the part type code. The process sends the WorkPart command (with the part type code) to the operating unit, which chooses the part program corresponding to the part type and performs the machining. At the end, the process receives the WorkDone message and sends the unloading request (URequest) to the CCS. Once the unloading is completed (this is notified with message PartUnloaded), the process makes an event-driven

choice: if there is no MReq (maintenance request) message coming from the operating unit, it starts a new work sequence, otherwise it sends a maintenance request (MRequest) to the CCS. In the second case, the process makes another event-driven choice: if it receives a delay request (MRdelayed), it informs the operating unit with the MReqDelayed message and starts a new work sequence. If, instead, it receives an acceptance message (MRaccepted), it waits until it gets message MRserved and then starts a new work sequence.
The process model is shown in Fig.4.

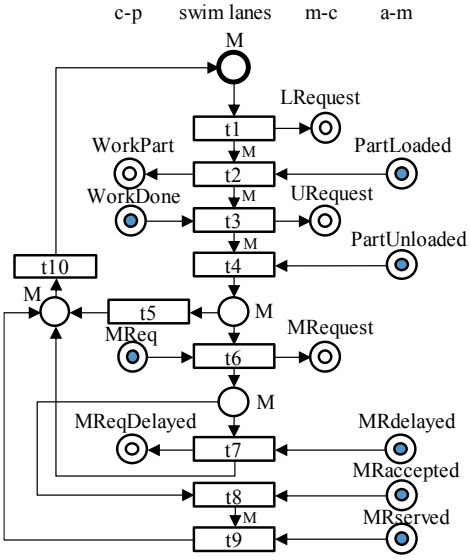

Fig. 4. The process model of the machine tool

The collaboration places appear in swim lanes: m-c, a-m, and c-p contain those related to the collaborations with the CCS, the AGV, and the operating unit, respectively.
A simplification of the notation is shown in the process model and affects tasks in series. If two tasks are in series (Murata, 1989), the place between them can be absorbed in the link, as it takes place with tasks t1 and t2; the label of the link is the label of the place absorbed.
In Fig.4, there is only one initial place. The initialization rule, which is not shown for the lack of space, makes this place receive a token referring to the machine tool acted on by the control process.

4.3 Requirements and model of the CCS

The requirements of the CCS are part of the general requirements section so this subsection comments on the CCS control process model shown in Fig. 5.
The process model is divided into two views. The first view concerns the assignment of missions to the agv while the second one handles the maintenance requests.
There are only two normal places and their type is AGV: they represent the states of the agv handled by the process. The initial state is “idle”. Owing to an initialization rule, the corresponding place will receive a token referring to the AGV that is part of the case study.

Tasks assignLM and assignUM assign a loading mission and an unloading one, respectively; both require that the agv is idle.

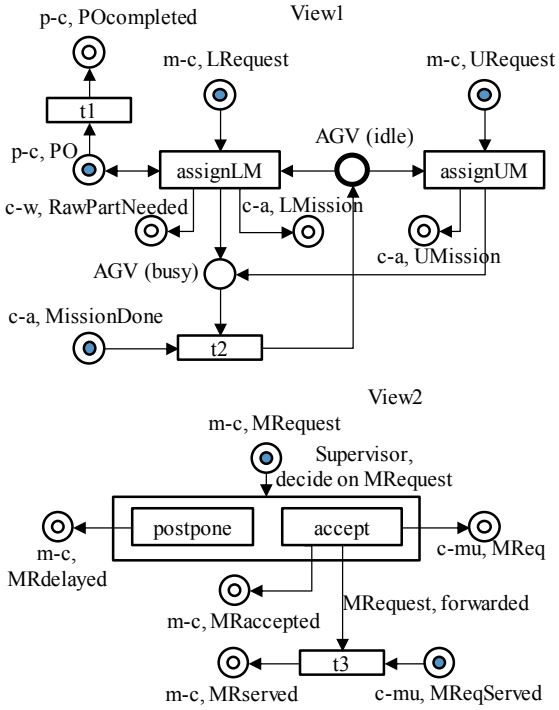

Fig. 5. The process model of the CCS

Task assignLM is a matching task in that it must match a pending order to a requesting machine tool: the match succeeds if the machine tool is able to work the type of parts indicated by the order. The matching rule is written in the pre-condition of the task (not shown for the lack of space). Task AssignLM, besides changing the stage of the agv, is in charge of sending message LMission to the agv and message RawPartNeeded to the warehouse; moreover, it decreases the number of remaining parts of the order selected. The double arrow of the link between task AssignLM and place PO means that the order undergoes a change when the task is performed but it remains in the same place. On the contrary, when the number of remaining parts becomes equal to zero, task t1 removes the order from place PO and sends message POcompleted to the PCS.
View 2 shows how the process handles decisions on maintenance requests. An incoming request may be accepted or postponed and the choice is a human one in that it must be made by a person who is entitled to do so, a supervisor in this case. The human choice is represented by a block that encloses tasks accept and postpone and has a label containing the role of the performer and the name of the choice (decide on MRequest). In case of acceptance, the process sends message MRaccepted to the machine tool and message MReq to the MU, and, when it receives message MReqServed from the MU, it informs the machine tool with message MRserved. In case of postponement, the process sends message MRdelayed to the machine tool.

5. CONCLUSIONS

The modeling approach for Cyber-Physical Systems presented in this paper emphasizes collaborative processes and leverages extended Petri nets to define process models. Further work will be devoted to the development of a simulation environment that can help validate the models and enrich the notation.

REFERENCES

Atzori, L., Iera, A., and Morabito, G. (2010). The Internet of things: a survey. *Computer Networks*, vol. 54, pp. 2787–2805.

Bauer, M., et al. (2013). IoT reference architecture. In Bassi A. et al. (eds.), *Enabling things to talk*, pp. 163–211. Springer, Cham.

Bocciarelli, P., D'Ambrogio, A., Giglio, A., and Paglia, E. (2017). *A BPMN extension for modeling cyber-physical-production-systems in the context of Industry 4.0*. 14th IEEE International Conference on Networking, Sensing and Control, pp. 599–604.

Bruno, G. (2015). Data flow and human tasks in business process models. *Procedia Computer Science*, vol. 64, pp. 379–386.

Bruno, G. (2018). Business process modeling based on entity life cycles. *Procedia Computer Science*, vol. 138, pp. 462–469.

Friedl, A. (2018) *Meeting Industrie 4.0 challenges with S-BPM*. 10th ACM International Conference on Subject-Oriented Business Process Management.

Graja, I., Kallel, S., Guermouche, N., and Kacem, A.H. (2016). BPMN4CPS: *A BPMN extension for modeling cyber-physical systems*. 25th IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, pp. 152–157.

Hankel, M., and Rexroth, B. (2015). *The reference architectural model industrie 4.0 (RAMI 4.0)* [online]. Available at: https://www.zvei.org/en/press-media/publications/the-reference-architectural-model-industrie-40-rami-40/ (Accessed: 13 March 2019).

Hermann, M., Pentek, T., and Otto, B. (2016). *Design principles for Industrie 4.0 scenarios*. 49th IEEE Hawaii International Conference on System Sciences, pp. 3928–3937.

Huang, P., Jiang, K., Guan, C., and Du, D. (2018). *Towards modeling cyber-physical systems with SysML/MARTE/ pCCSL*. 42nd IEEE Computer Software and Applications Conference, pp. 264–269.

Jensen K., Kristensen L.M., and Wells, L. (2007). Coloured petri nets and cpn tools for modelling and validation of concurrent systems. *International Journal on Software Tools for Technology Transfer*, vol. 9, pp. 213–254.

Kagermann, H., Wahlster, W., Helbig, J. (2013). *Recommendations for implementing the strategic initiative INDUSTRIE 4.0* [online]. Available at: https://www.acatech.de/Publikation/recommendations-for-implementing-the-strategic-initiative-industrie-4-0-final-report-of-the-industrie-4-0-working-group/ (Accessed: 13 March 2019).

Kortuem, G., Kawsar, F., Sundramoorthy, V., and Fitton, D. (2010). Smart objects as building blocks for the Internet of things. *IEEE Internet Computing*, vol. 14, pp. 44–51.

Kusmenko E., Roth A., Rumpe B., and von Wenckstern M. (2017). Modeling architectures of cyber-physical systems. In: Anjorin A., Espinoza H. (eds) *Modelling Foundations and Applications*. Lecture Notes in Computer Science, vol. 10376, pp. 34–50. Springer, Cham.

Lee, E.A. (2008). *Cyber physical systems: design challenges*. 11th IEEE Symposium on Object Oriented Real-Time Distributed Computing, pp. 363–369.

Mandal S., Hewelt M., and Weske, M. (2017). A framework for integrating real-world events and business processes in an IoT environment. In: Panetto H. et al. (eds) *On the Move to Meaningful Internet Systems*. Lecture Notes in Computer Science, vol. 10573, pp. 194–212. Springer, Berlin, Heidelberg.

Martins F., and Domingos D. (2017). Modelling IoT behaviour within BPMN business processes. *Procedia Computer Science*, vol. 121, pp. 1014–1022.

Meyer S., Ruppen A., and Magerkurth, C. (2013). Internet of things-aware process modeling: integrating IoT devices as business process resources. In: Salinesi C., Norrie M.C., Pastor Ó. (eds) *Advanced Information Systems Engineering*. Lecture Notes in Computer Science, vol. 7908, pp. 84–98. Springer, Berlin, Heidelberg.

Müller, H.A. (2017). The rise of intelligent cyber-physical systems. *Computer*, vol. 50, pp. 7 – 9.

Murata T. (1989). Petri nets: properties, analysis and applications. *Proceedings of the IEEE*, vol. 77, pp. 541–580.

Pisching, M.A., Pessoa, M.A.O., Junqueira, F., dos Santos Filho, D.J., and Miyagi, P.E. (2018). An architecture based on RAMI 4.0 to discover equipment to process operations required by products. *Computers & Industrial Engineering*, vol. 125, pp. 574–591.

Serral, E., De Smedt, J., Snoeck. M., and Vanthienen, J. (2015). Context-adaptive Petri nets: supporting adaptation for the execution context. *Expert Systems with Applications*, vol. 42, pp. 9307–9317.

Song, R., Wang, Y., Cui, W., Vanthienen, J., and Huang, L. (2018). *Towards improving context interpretation in the IoT paradigm: a solution to integrate context information in process models*. 2nd ACM International Conference on Management Engineering, Software Engineering and Service Sciences, pp. 223–228.

Suri, K., Cadavid, J., Alferez, M., Dhouib S., and Tucci-Piergiovanni, S. (2017). *Modeling business motivation and underlying processes for RAMI 4.0-aligned cyber-physical production systems*. 22nd IEEE International Conference on Emerging Technologies and Factory Automation, pp. 1–6.

Xu, L.D., He, W., and Li, S. (2014). Internet of things in industries: A Survey. *IEEE Transactions on Industrial Informatics*, vol. 10, pp. 2233–2243.