Shop Floor Control: A Physical Agents Approach for PLC-Controlled Systems

Javier de las Morenas, Andrés García-Higuera, and Pablo García-Ansola

***Abstract*—Multiagent technology raised great expectations from the outset, and these expectations are still high today. A technology that allows the creation of systems that are autonomous, reactive, proactive, and have entities with social skills is necessarily very appealing. Many initiatives have emerged to develop the application of this technology in different areas, including manufacturing. Multiagent-based solutions typically use this technology to create distributed systems with decentralized decision making as a way to tackle complex systems or problems by dividing the complexity among agents. Unfortunately, most works do not go further than the definition of agents or architectures, which prevents industry from adopting this technology. This paper presents an approach to implementing the shop floor control of an automated distribution center following a multiagent approach. This implementation divides the system into agents in charge of the management of the system while highlighting the communication channels with shop floor control equipment.**

***Index Terms*—Multiagent systems (MAS), petri nets (PNs), programmable logic controller (PLC), shop floor control.**

Nomenclature

List of Acronyms

BDI	Beliefs–desires–intentions.
EPCIS	Electronic product code information services.
HMS	Holonic manufacturing systems.
IS	Information Services.
MAS	Multiagent systems.
MDP	Markov decision process
MHS	Material handling systems.
PLC	Programmable logic controller.
PN	Petri net.
PRS	Procedural reasoning system.
RFID	Radio-frequency IDentification.

Manuscript received October 31, 2015; revised August 3, 2016, January 6, 2017, February 20, 2017, and April 24, 2017; accepted June 9, 2017. Date of publication June 28, 2017; date of current version October 3, 2017. Paper no. TII-15-1612. *(Corresponding author: Javier de las Morenas.)*

J. de las Morenas and P. García-Ansola are with the AutoLog Group, School of Mining and Industrial Engineering, University of Castilla-La Mancha, Almadén 13400, Spain (e-mail: javier.delasmorenas@uclm.es; pablo.garcia@uclm.es).

A. García-Higuera is with the AutoLog Group, School of Industrial Engineering, University of Castilla-La Mancha, Ciudad Real 13071, Spain (e-mail: andres.garcia@uclm.es).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2017.2720696

List of Symbols

R_p	MDP. Set of the reader points of a product p.
B_p	MDP. Set of the business steps of a product p.
S_p	MDP. Set of possible states of a product p.
S	MDP. Best transition.
$Q^{\pi}(s,a)$	MDP. Reward value function.
$V^{\pi}(s)$	MDP. Maximum reward function.
P	PN. Set of places.
T	PN. Set of transitions.
F	PN. Set of arcs.

I. Introduction

In a market demanding customized products within a short delivery time, manufacturers are being forced to mass produce customized products, but without ignoring changes in the demand. Under these conditions, companies have to improve their capacity for flexibility and agility in response to consumers' needs, while maintaining the production and quality level so as not to lose competitiveness. The problem of resource allocation becomes a key issue, but that allocation has to be continuously revised to introduce the required modifications as new orders arrive. Negotiation-based solutions provide the required agility to deal with that dynamic approach to resource allocation.

The measurement of the requirements placed on companies have evolved from traditional performance indicators, described in terms of optimal statics, to new indicators that quantify reactivity, adaptability, agility, and robustness [1]. At the same time, material handling systems (MHS) are of high value in industry, as these processes represent 15% of the total production cost [2]. Material handling includes movement, storage, protection, and control of goods during production, distribution, selling, and consumption. An automated distribution center is composed of several MHS, such as conveyors and elevators.

Traditionally, control systems were developed under centralized architectures optimized to state conditions. However, centralized control is currently inefficient as the complexity of design increases as a system enlarges. Moreover, these solutions are not capable of addressing the occurrence of unexpected disturbances because of the rigidity of their structures. Unforeseen events, such as last minute orders, delays in raw materials, breakdowns, or other incidents produce distortions in the initial plan and cause delays and nonoperative situations.

One way to address this problem is by distributing decision making across different entities, creating a noncentralized control system (distributed) that allows the division of the entire system into small and simple subsystems. A promising approach is to consider the elements of the system as being like a community of autonomous, smart, and reusable distributed units that operate as a collaborative set of entities [3]. These encouraging ideas have led to the rise of some initiatives for the development of distributed control systems for their use in industry, and have highlighted multiagent systems (MAS) and holonic manufacturing systems (HMS) [4]; with HMS being a specific implementation of MAS.

On these negotiation-based systems, it is crucial to have information of the environment in order to reach the correct decision making. Techniques to identify and track products by radio-frequency tags (RFID) [5] seem to be a good solution to integrate highly distributed systems.

In this paper, agents, holons, and RFID are put in common in order to define and implement a decentralized shop floor control over industrial controllers in a distribution center.

Section II presents the state of the art of this work, and it is focused on the use of agents in industry. In Section III, a proposal to combine the MAS with RFID is presented, introducing a two-layer MAS called MAS-DUO, and physical beliefs–desires–intentions (BDI) agents based on Markov decision process (MDP) reasoning. Section IV presents the interaction between MAS and PLCs, and the decision making at the shop floor level modeled by PNs. Then, in Section V, the proposal is implemented using a test bench that represents the installations of a distribution center controlled by programmable logic controllers (PLCs). And finally, Section VI presents the outcomes, conclusions, and future work.

II. Literature Review

Agent architectures are characterized by being autonomous, reactive, and proactive, and as having social capacities [6] that allow their application in highly stochastic or dynamic environments.

In the last few years, the technology of agents proceeding from computational science has been used first in manufacturing planning, and then in the control of the execution of the plan, to help deal with the increase in complexity. This approach was pioneered by the architecture proposed by Van Brussel [7] known as PROSA (product, resource, order, staff architecture). It consists of identifying the products, resources, and orders as "holons" that negotiate among themselves to reach a successful solution. ADACOR (ADAptive holonic COntrol aRchitecture) [8] defines an architecture very similar to PROSA, but includes an extra holon, the coordinator holon. This is because the operation of the system can be diverted from the desired one as the decision making is distributed among several entities, and the coordinator holon avoids this situation by aligning the global performance. The evolution of ADACOR includes self-organization mechanisms to cope with perturbations [9]. García Ansola [10] in MAS-DUO proposed an architecture where agents are separated into information system agents and physical agents. This approach reflects the reality of companies, where management and shop floor control are separated. This division happens at the beginning of the company, where the management system or enterprise resource planning (ERP) is contracted to one enterprise and the commissioning of the plant to another. There is no commercial interest in full integration, thus making the startup of the plant harder; take, for example, the industrial manufacturing branch of a company such as Siemens, which provides PLC equipment, industrial communications, and manufacturing execution systems dedicated to shop floor control, and in contrast there are ERP providers as SAP.

From the methodologies point of view, designing agent-based control systems [11] is a starting point. This methodology is based on three steps: the analysis of control decisions, the identification of agents, and the selection of interaction protocols. The use of the JADE framework [12], based on JAVA, is widely used [13], [14]. da Silva *et al.* [13] proposed a control architecture for reconfigurable manufacturing systems taking ADACOR as reference, and modeling the system in two phases: conceptual and dynamic. In the GRACE European Project, [14], four agents similar to the ones used in ADACOR but having in mind a product-driven production approach involving intelligent products are proposed. This work also presents an application in a factory plant producing laundry washing machines. An alternative to the use of JADE is the Erlang programming language, which appears to be especially suitable for programming concurrent, scalable, and distributed systems using a holonic approach [15]. A comprehensive comparative survey of the software engineering methods that have been used in this area can be found in [16].

Some test benches have been developed to test the agent technology or RFID technology. Soylemezoglu *et al.* [17] implemented electronic product code information services (EPCIS) [18] by combining RFID with web repositories to share product information with everyone in the supply chain. Zacharewicz *et al.* [19] focused on the use of RFID in a refrigerated environment, and in this way checked how RFID works in hostile environments. Zawodniok *et al.* [20] or Vrba and Mařík [21] presented experimental platforms where they tested multiagent systems in a flexible manufacturing cell at Cambridge University and in a simulated environment, respectively. Vrba *et al.* [22] presented the application of agents in a manufacturing cell at the ACIN Laboratories of the Technical University of Vienna. An approach combining both technologies can be found in Vrba *et al.* [23] in a simulation software using the MAS framework.

The agents' technology is planned to work on personal computers. However, this presents a problem for the application of agents in industrial environments, specifically for shop floor control. Currently, industrial controllers consist of PLCs or robots that do not allow agent applications to be run, or at most allow some reactive agents with low capacities. The evolution of PLCs seems to be close to the field-programmable gate array [24], increasing their reactivity.

The first approach to solve this problem is by means of holons. Holons are defined as autonomous and cooperative elements that represent the components that form a manufacturing system with decentralized control. Holons can be organized in hierarchical or heterarchical structures. Each holon mainly contains a hardware part that is able to physically influence the real world (e.g., tool, device, and machine) and a control plant responsible for the control of the hardware part and the communication with the rest of the holons in a holonic community [25].

Examples of the implementation of these holons–also known as "physical agents" to highlight their capacity to influence in their environment–can be found in several works in the literature [22], [26], [27]. All these works use FIPA [28] for the control part and the IEC-61499 standard for the physical part. Yu *et al.* [29] proposed the use of function blocks to define a reference model for agent-based systems using services.

Using physical agents or holons (a concept used indistinctly in literature), the control is decentralized but is still far from the industrial controllers. This very promising solution is not robust and reactive enough to be applied in real time at the shop floor control, as moving the decision making away from the controller does not guarantee the correct real-time answer for the system. Moreover, the complexity of the systems makes it difficult for real-time applications to arise [30]. On the one hand, it is clear that the own controller has to carry out the decision making to achieve the robustness and reactivity necessary for the system. On the other hand, agents mainly respond to a high level of abstraction, which is useful for providing autonomy and intelligence, but it fails to deal with real-time constraints. For that reason, this paper presents a way to include the advantages that provide the use of distributed systems, as agents, but keeping the real-time response of the control system using the two independent platforms of agents described in the following section.

III. Proposal

Based on previous works [10], [31], a horizontal-layered MAS acting as a new product scheduling system, so as to better reflect the physical implementation of control systems, seems adequate with the current requirements instead of a unique agent platform; however, a full design and validation process needs to be addressed. Fig. 1, based on the manufacturing control [32], details the layering of the product scheduling system. This layering proposes a division of the agent space into two platforms to make possible a full implementation of these MAS in current manufacturing/logistics scenarios.

This paper introduces an adaptation of the holonic concept into two separate levels, which are developed by using agents. The research community addresses holons as logical units with hardware integration, but real implementation problems arise. In this paper, a BDI physical reasoning is presented [33] to provide an algorithm to divide holons into different logical units. The first platform is an agent structure related to the physical elements of the plant and directly connected to the physical world, which is called "Physical Agent Platform." The second platform is an interface with the upper Information Services (IS) of the company, which is called "IS Agent Platform." This idea comes from the holonic concept of communication in manufacturing control but using a service-oriented division based on protocols and decision-making techniques, instead of a unique holonic structure incorporating these two expert systems.

This division puts special emphasis on applying strategy policies (IS Agent Platform) while attending to the information

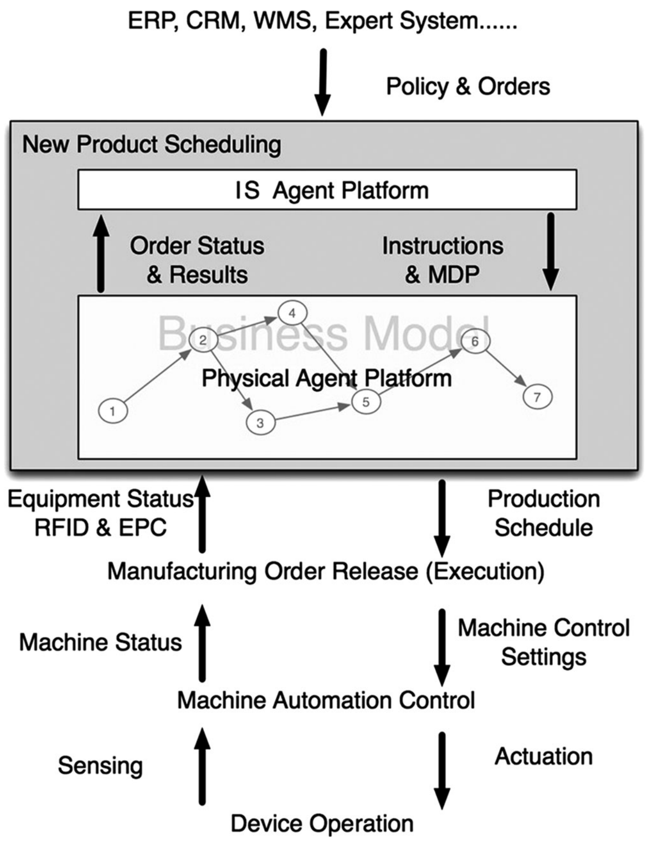


Fig. 1. MAS-DUO proposal: The new product scheduling.

being feed from reality (Physical Agent Platform), which includes constant deviations, disturbances, and restrictions from the initial schedule. It means that an independent low-level real-time control is connected to the hardware, while a deliberative intelligent system provides high-level decision making. However, both layers are not integrated or coupled in a unique holon.

A. Belief Updating Process

Physical BDI agents can directly benefit from visibility frameworks through a software interface, which in the case at hand is the standard EPCIS used to implement common RFID standards. The EPCIS specification helps the definition of this dynamic interface by using the existing services, which automatically publish the real-time information coming from the plant and its circumstances, reporting treated information and abstracting upper information technology (IT) levels. From the BDI point of view, a well-formed reasoning requires a constant information feedback from the environment to constantly feed their beliefs. MAS-DUO [31] proposes the EPC subscriptions to connect the physical products/resources with the internal reasoning of the affected agents; the beliefs of these agents are updated through the standard EPCIS XML specification. The junction of BDI and EPC in the physical BDI agents constitutes a procedural reasoning system (PRS) with an updated RFID event database. It is a framework for constructing real-time product-driven reasoning, which can perform decision making in dynamic shop floor environments. The states of the product are defined and detected by the "What?" (product identification) "Where?" (read point) "When?" (timestamp), and "Why?" (business step) EPC information.

A set of the reader points of a product p, R_p is defined as

$$R_p = (r_1, r_2, \ldots r_n)\ ,\ R_p \in \mathcal{M}_{1xn}. \tag{1}$$

A set of the business steps of a product p, B_p is defined as

$$B_p = (b_1, b_2, \ldots b_m)\ ,\ B_p \in \mathcal{M}_{1xm}. \tag{2}$$

A set of the possible states of a product p, S_p is defined in (3), being each of the states s_{ij} calculated using a simple inference of R_p and B_p.

$$S_p = \begin{bmatrix} s_{00} & \cdots & s_{0n} \\ \vdots & \ddots & \vdots \\ s_{m0} & \cdots & s_{mn} \end{bmatrix}\ ,\ s_{ij} = r_i\ x b_j,\ S_p \in \mathcal{M}_{mxn}. \tag{3}$$

As the number of business steps and read points increases, so does the precision during the decision making, because of the corresponding increase in the number of states. The definition of states is flexible, so that new read points and business steps can be added during operations.

Once the product states are identified along the product schedule, it is possible to define the transactions between the current state (s_{ij}) and the desired state (s'_{ij}). The full transitions do not have to be direct and take place in only one step; there can be intermediate states, which have to be identified during the reasoning process in a dynamic programing process. This is the definition of task, which means the sequential definition of states until the goal state (s'); in which every instruction corresponds with a unique state transition. In the problem at hand, there are available technologies to solve state transition problems, as is the case of MDP techniques, but any such solution needs to be integrated into the BDI reasoning process. The objective of these techniques is to obtain the best transitions or tasks (S) from the product state s_{ij} to the product state s'_{ij} depending on configurable rewards as

$$S\ = \left(s_{ij}, \ldots, s'_{ij}\right). \tag{4}$$

The state transition problems require calculating a reward in every transition in order to schedule the best plan. Therefore, the generation of new beliefs, such as costs, times, importance of clients, or energies consumed that can be valuable, are useful in the search of the best choice of schedule. These new beliefs in the reward reasoning define the decision policy in the product scheduling; the selection depends on a reward calculated by MDP.

B. BDI Reasoning Process and MDP

The BDI reasoning process is divided into three stages: the option generation, the generation of focused beliefs and the selection of the instructions states. MDP is the decision technique chosen for the development of the stated stages.

MDP is a theoretical framework that outcomes a sequential decision processes in which states and transitions are quantified in calculated rewards. Therefore, the accurate identification of the current state of the product is essential to generate accurate rewards during the transitions to the next states. The physical BDI agent calculates the reward during the transition between two Markov states.

MDP defines a reward function using decision parameters submitted by the information services (IS) agent platform and the continuous rewards generated by the beliefs of the physical BDI agents that represent the real situation of the shop floor in a valuable state transition by using rewards matrices. By using EPCIS, any event of the tracked resource in the shop floor is submitted to the involved agent by the subscription service. Once an event has occurred or a new task is required, the captured "What? Where? When? Why?" information allow detecting the state of the product.

Once the value of the reward has been obtained for a specific task–by using the reward matrices and the state transitions-the physical BDI agents, which are in a constant product-driven negotiation, take the outcomes using the MDP. The scheduling of tasks is based in a negotiation where products and resources compete to obtain the best MDP reward during operations. This reward function reflects a strategic decision making in every single operational decision taking into account physical constraints. The value function $V^\pi : S \rightarrow \Re$ specifies the value of the policy Π in the state S. This policy will define the best instruction in every state of the system, even when disturbances occur in order to achieve the environmental alignment. Then, the Q function represents the reward value of the chosen options in the space of possible actions and in all the previously selected steps

$$Q^\pi\ (s, a) =\ R\,(s) + \gamma \sum_{s \in S} P\,(s'|s, a)\, V^\pi\ (s')\,. \tag{5}$$

The V function defines the selection of the maximum reward in the space of possible options obtained by the Q function

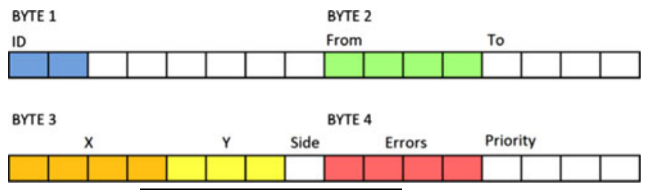
$$V^\pi\ (s) = \max\nolimits_{a \in Ap(s)}\ [Q^\pi\ (s, a)]\,. \tag{6}$$

Combining BDI and MDP techniques, the internal beliefs situate the agent at one of the defined states in the MDP. The intentions of the agent define the actions that will define the state transitions; they allow the generation of a plan based on the Markov parameters depending on the weights used for allocating rewards. The desires are the goal state or final state on the MDP that the agents try to achieve. Therefore, there is an internal MDP in each physical BDI agent, which chooses the actions with the maximum reward trying to reach the goal state.

Readers interested on further details of the BDI reasoning process based on MDP defined can refer to [31].

IV. Connecting MAS With PLC Network Devices

In the most common cases of industrial deployment, it is usually necessary to be able to send and receive instructions from the physical BDI agent to its implementation module residing at a PLC and vice versa. The physical control maintains the operation and the independence of the agent; it is only a way to share variables, not direct instructions. Due to limitations in most PLC programming languages, the PLC may operate by using its function blocks as selected by the module of the agent that implements the negotiation process; else the program could lead to a loss of robustness, and to out of control situations.



Zones Level 0	
	0000
Production	0001
Rejection	0010
Reception	0011
Picking 1	0100
Picking	0101
Picking Rejection	0110
Expedition	0111
Warehouse	1xxx

Fig. 2. PLC message and position codes.

A dispatcher program in the PLC is essential to translate the instructions coming from the rest of the agent implementation.

This way of working needs to program the PLC by using divided function blocks while checking these variables as they are being submitted by the MAS-DUO implementation of the IS. The PLC has a series of instructions implemented and divided into specific tasks as function blocks. These function blocks implement the different operations that have to be carried out such as: moving products along conveyor belts until reaching a determined part of the warehouse, the control of the pallets when reaching meeting points on the conveyor belt or the movement of the pallets until a junction.

Therefore, to preserve the independence of the PLC during operations, the only instruction that the agent can order to the control platform is the moving of pallets from one point in the plant to another. It is then the PLC that implements lower level operations, which facilitates the integration by making it more similar to that being used by previous control schemes. The use of shared information about the localization of the products at different control levels makes the integration possible. As this even allows the higher level IS to submit orders such as changes of destination to a pallet that is already moving along the plant. This scheme has in common with EPCIS the possibility of sharing data with several business models by a common access model. It is not a pure agentification process where every functional resource has an agent avatar connected to every single PLC, but is enough to detect the product states with the information coming from the PLCs. There is a PLC module of the agent that sends and receives the information packets about the new instructions to the PLCs. Therefore, a previous common definition of data is required; these data have parameters such as the identification, the origin, and the destination of the pallet. These instructions have a determined structure in which the necessary data appears coded in an efficient way (see Fig. 2).

In the case at hand, a 4-B binary code is used, which incorporates the identification of the pallet within the control system, the origin and destination of the pallet, a series of flags equipped for possible errors, which could take place during the fulfillment of the order and a reserved space to define the priority of each order. Specifically, the package contains the next fields.

1) ID. The EPC of the product is mapped in the ID field. The relation between the ID and the EPC is stored in the physical BDI agent. It would be very complex to the PLC network to handle the whole EPC.
2) From. The initial localization defines the current position of the pallet. The codifications of the localizations are detailed in the table of Fig. 2.
3) To. The target localization of the instruction corresponds with the goal position of the pallet defined by the physical BDI agent.
4) *X*, *Y*, and side. Details about the localization are the position in the warehouse or the specific picking zone.
5) Priority. The priority assigned to that instruction using the decision parameter by the global policy.

The format of the instructions presented in Fig. 2 can be better understood with the example of a pallet going, for instance, from the input point in the docks "Reception" to a specific location within the "Warehouse"-let us say: location horizontal 5 and vertical 2 on the left in corridor 3. In this case, Byte 1 would contain the ID of the pallet as read at the input identification point (typically by an RFID reader at that location or a bar code reader in previous versions). Byte 2 would contain a 0011 meaning "Reception" and a 1011 meaning "Warehouse-corridor3." Byte 3 would contain the location within that corridor: 0101 ($x = 5$), 010 ($y = 2$), and a 1 for the left-hand side of the corridor. Byte 4 would be typically set to cero unless the crane finds out for instance that the location is already occupied (because of an error in the database used to allocate locations) in which case the corresponding error code would be added here in the four most significant bits (the four bits on the left of the byte) following a specific code including all the different errors that can happen for the different instruction combinations. Byte 4 can also include in its four least significant bits (the four on the right) a priority value that can be modified on-line to make the pallet move faster at some points (e.g., intersections). Evidently, the management system will be constantly checking that Byte 4 to launch the required procedures in case of error and to take priorities into account when the pallet reaches specific decision points.

The system works with a series of these instructions managed by a dispatcher program in the PLC master but previously generated by the agent-based system. The dispatcher program is in charge of the communication between the physical BDI agents and the PLC network. The PLC module of the physical BDI agent is in charge of translating the intentions as instructions and sends them to the dispatcher program of the PLC, and vice versa. The PLC module of the physical BDI agent prepares a package as the represented in Fig. 2 by using the state transitions calculated during the reasoning process.

A. Interface Between PLC

In order to allow physical BDI agents to lead the shop floor control of a distribution center carried out by PLCs following

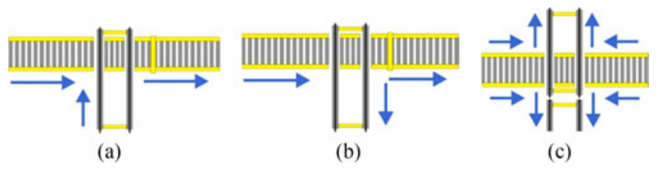


Fig. 3. Decision making at shop floor level. (a) Intersection. (b) Diverter. (c) Crossing.

the instructions presented, it is necessary to model the decision making at shop floor level. The decision making at shop floor level are related with the material handling of products/pallets on conveyor belts and processes provided by machines. In the case of diverters, products arrive to a bifurcation that can take different ways [see Fig. 3(b)], and depending on the path taken it will undergo a process, receive an inspection, or reach a destination. Another situation happens when some products converge at an intersection [see Fig. 3(a)], and it is necessary to evaluate which of the products should pass first. When products first converge at a junction, and then converge on multiple paths or conveyors, it is defined as crossing [see Fig. 3(c)]. Finally, as workcells offer different processes, products have to select which of them would be required to fulfill the task.

The presented types of decision making were modeled by means of PNs. PNs are a tool used to describe discrete event dynamic systems (DEDS) that abstract the logic of implementation of each function at the plant. PNs identify the state of resources at plant and analyze their availability to attend to different products or requests.

A PN is a directed graph, consisting of places (circles), transitions (bars), and arcs (connecting places and transitions). A place *p* is the input of a transition *t* if an arc from *p* to *t* exists. A place *p* is the output of a transition *t* if an arc from *t* to *p* exists. An integer value adjacent to the arc represents the weight of the arc. Places can contain marks or tokens, represented by dots. Tokens indicate the state of the net.

Formally, a PN can be defined by the five-tuple [34].

$$\overset{\mathrm{m}}{.}PN = (P, T, F, W, M_0) \tag{7}$$

where $P = \{p_1, p_2, \ldots, p_m\}$ is a finite set of places;
$T = \{t_1, t_2, \ldots, t_n\}$ is a finite set of transitions;
$P \cap T = \emptyset$;
$F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs (flow relation);
$W : F \to \{1,\ 2,\ 3,\ \ldots\}$ is a weight function;
$M_0 : P \to \{0, 1,\ 2,\ \ldots\}$ is the initial mark.

In the following sections, each of the kinds of decision-making arising from the shop floor control of a distribution control are modeled by place/transition PN graph. A PN is executed by firing transitions. All the models presented in this paper follow the transition rule: A transition is enabled if each input place *p* of a transition *t* contains as many tokens as the corresponding input arc weight indicates. When an enabled transition is fired, it removes from each input place *p* as many tokens as required for the weight of its input arc, and deposits in each output place *p* as many tokens as the output arc weights. However, when the weight of the arc is 1, as is the case in all our models, the integer value adjacent to the arc is omitted.

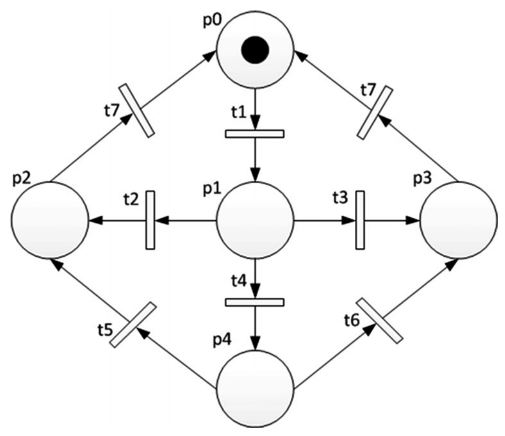


Fig. 4. PN of an intersection.

In the cases being analyzed here, all transitions are fired by events taking place in the physical world (e.g., the activation of proximity sensors or end-of-cycle signals in machines) or by conditions that are closely related to those events (e.g., the activation of negotiation flags). These PNs have binary inputs/outputs and their transitions are externally fired, therefore, they could be formalized as interpreted PNs [35] following the design process defined in [36]. But this whole process was not deemed necessary given the simplicity of the models below.

1) Intersection: An intersection, where products arrive from either conveyor belt 1 (c1) or conveyor belt 2 (c2), is modeled. As can be seen in Fig. 4, when a product approaches the intersection, transition t1 is triggered and the net is started (place p1). When two or more products arrive at the same time, it is necessary to evaluate which one will pass first, so transition t4 is triggered to initiate the evaluation (place p4). The evaluation is based on comparing the priority of each product. These priorities are specified by the physical BDI agents and sent as PLC messages. This evaluation concludes firing transition t5 or t6 to run places p2 or p3, respectively, granting the pass to the highest priority product. Then, the net is reset to p1 by transition t7. The model could be easily extrapolated to scenarios with n input conveyor belts.

Places are as follows:

1) p0: Initial state.
2) p1: Evaluation of the kind of negotiation according to the number of products.
3) p2: Actions required to the product arriving from c1.
4) p3: Actions required to the product arriving from c2.
5) p4: Evaluation of the priority level of the products arriving the intersection to give way to the highest priority one. If both products have the same preset priority, intersection asks the adjacent products about their priority level and the level of saturation of conveyor belts in order to weight the preset priorities.

Transitions are as follows:

1) t1: Is triggered when there is a product in any of the inputs of the intersection.

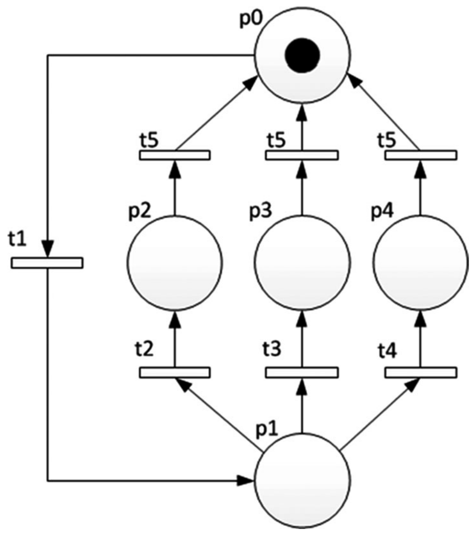


Fig. 5. PN of a diverter node.

2) t2: Is triggered when there is a product in c1 but none in c2.
3) t3: Is triggered when there is a product in c2 but none in c1.
4) t4: Is triggered when there is simultaneously a product in c1 and c2.
5) t5: Is triggered if after the evaluation carried out in p4 it is concluded that the product from c1 will pass first.
6) t6: Is triggered if after the negotiation carried out in p4 it is concluded that the product from c2 will pass first.
7) t7: Is triggered when the product leaves the intersection.

2) Diverter: A diverter is a node where a product can choose between different paths. As Fig. 5 shows, the PN is initialized in p0. When a product reaches the diverter (transition t1) the evaluation is fired. Remember that each product not only carries a priority level, but also a destination. The diverter contains the information about the offered paths: conveyor belts 1 (c1), 2 (c2), and 3 (c3). After evaluating the destination of the product, it is sent to one of the paths (places p2, p3, or p4) triggering the transitions t2, t3, or t4.

Places are as follows:
1) p0: Initial state.
2) p1: Diverter evaluates the optimal way to reach product's destination.
3) p2: Actions to guide the product to the conveyor belt c1.
4) p3: Actions to guide the product to the conveyor belt c2.
5) p4: Actions to guide the product to the conveyor belt c3.

Transitions are as follows:
1) t1: Is triggered when a product agent arrives at the diverter.
2) t2: Is triggered if the negotiation (p1) has concluded that product will continue on conveyor belt c1 (p2).
3) t3: Is triggered if the negotiation (p1) has concluded that product will continue on conveyor belt c2 (p3).
4) t4: Is triggered if the negotiation (p1) has concluded that product will continue on conveyor belt c3 (p4).
5) T5: Is triggered when the product leaves the diverter.

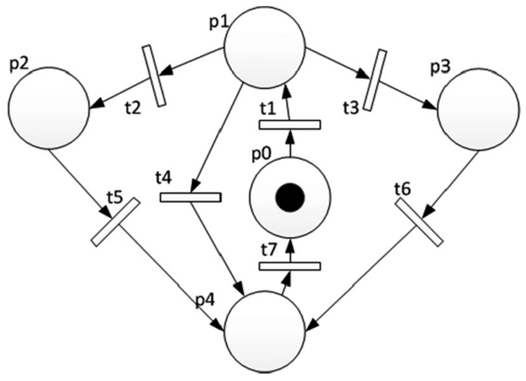


Fig. 6. PN of a machine.

3) Crossing: This kind of junction is a combination of the previous ones, where products converge in a node and then can take different paths.

4) Workcell: A machine is a node that offers multiple processes. A machine is controlled and represented by a resource agent. As Fig. 6 shows, when a product arrives at the machine (transition t1), the negotiation between the product and the machine begins (p1). After the negotiation, the machine launches the processes needed by the product, by firing transition t2 or t3. In cases where no process is needed, transition t4 is triggered. After the processes are done (p2 or p3), the product leaves the machine (p4).

Places are as follows:
1) p0: Initial state.
2) p1: Evaluates the processes the product will be subject to.
3) p2: Executes the process contained in p2.
4) p3: Executes the process contained in p3.
5) p4: Lets the product at the output of the machine.

Transitions are as follows:
1) t1: Is triggered when a product arrives at the machine.
2) t2: Is triggered if, after the evaluation is concluded, it is necessary to perform the processes contained in p2.
3) t3: Is triggered if, after the evaluation is concluded, it is necessary to perform the processes contained in p3.
4) t4: Is triggered if, after the evaluation is concluded, no process is needed by the product agent.
5) t5: Is triggered when the processes contained in p2 are completed.
6) t6: Is triggered when the processes contained in p3 are completed.

All the models have been duly analyzed to avoid deadlock problems. Being a PN *N*, characterized in the marked graph *G* [37].

1) *N* is bounded if there exists a number *b* such that each place contains at most *b* tokens in any reachable marking.
2) *N* is live if for each reachable making *M* and each transition *t*, there exists a marking *M'* that is reachable from *M* and enables *t*.
3) *N* is reversible if from each reachable marking, the initial marking can be reached.

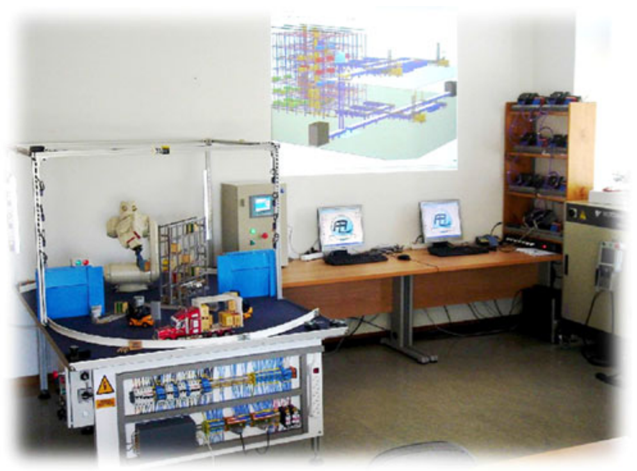


Fig. 7. Experimental platform at Autolog Labs.

As [38] states: “A PN is live if a deadlock cannot occur.” To check the models liveness test was carried out with the software TINA [39]. The output results of these tests showed how each net is bounded, live, and reversible. These results were the expected as the nets are simple. A more detailed discussion on deadlock avoidance for resource allocation systems can be found in [40].

Problems such as recursive request of resources are avoided through a consistent connection of the transitions with the physical world and the adequate marking of nets.

V. Implementation

A. Test Bench

The test bench where the physical BDI agents, MDP reasoning, and PN models were tested is shown in Fig. 7. The test bench represents the installations of a distribution center where all decision making modeled in the previous sections take place. There are conveyor belts, transelevators, elevators, shelves, a reception area, an expedition area, and check points.

The distribution center is divided between two floors named level 0 and level 1. While products get to the installation on level 0, the preparation of mixed pallets (picking) and the delivery of products take place on level 1. The test bench consists of a physical part and a virtual one, which are perfectly synchronized.

The test bench is controlled by the followings elements:

1) S7-300 CPU313C-2DP with CP-343-1 advance communication module by Siemens. It includes Profibus-DP, MPI, TCP/IP, and industrial ethernet communications.
2) S7-200 CPU224 with EM277 by Siemens. EM277 provides Profibus-DP communications. There are six units on the test bench, each of them controlling each of the part of the plant (level 0, level 1, three corridors, and the physical model).
3) S7-1200 CPU1212C with HMI KTP600 color display. The production area is controlled and simulated through this PLC. The PLC and HMI communicate by industrial Ethernet.

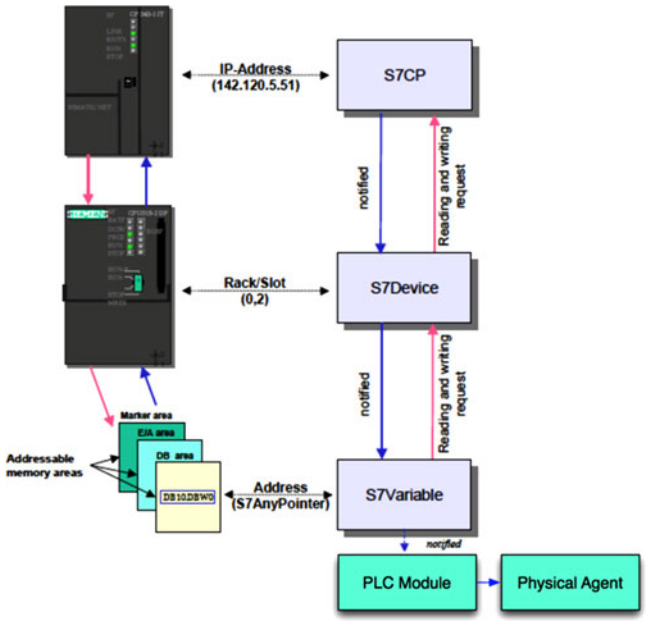


Fig. 8. PLC with S7 Java Beans. Siemens copyrights.

4) Motoman HP3 robot controller, NXP-1000. This is used to physically handle loads at the entrance to the installations. It offers a wide work area, thanks to its 6-axis arm.

B. Physical BDI Implementation

The Siemens S7 model of PLCs offers the option to directly access to the data via the S7-API beans of a Java application. The beans encapsulate Java component with functions that can be published. Beans can access to the addressable memory of the PLC by using the S7variable classes. These variables can include subscription and request methods about the state of specific data blocks and variables in the PLC. Therefore, a Java interface based on this library is implemented in the PLC module of the physical BDI agent to support the required communication (see Fig. 8).

In the PLC master, the dispatcher is in charge of collecting these data packets (instructions) that come from the physical BDI agents and sending them to the corresponding slave PLC. In this way, the master device controls and manages the outcoming traffic of instructions through the PLC network. It has the advantage that if any element of the system fails or there is an error, only the affected element would stop working, while the rest of the system would continue working with the received instructions. This is possible due to the automatic distributed control system between masters and slaves.

Each of the zones consisting the experimental platform are controlled by one of the S7-200 PLC. These PLCs and the S7-300 make up a Profibus-DP network. The master of the network (S7-300) is in charge of receiving instructions from management agents, shares them with the slave PLCs, and transmits acknowledgment of accomplishment. These tasks are performed using function blocks. Each PLC has reserved memory

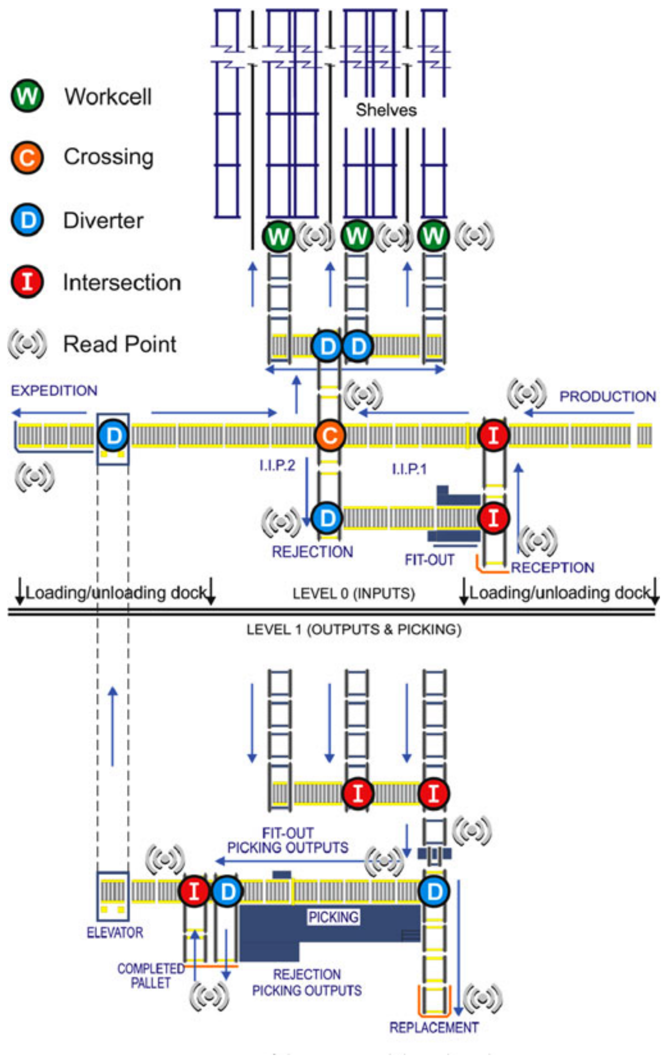


Fig. 9. Location of decision models and read points.

areas to communicate instructions. Slave PLCs only implement simple decision-making processes directly related to the specific bit of the physical system each of them controls.

All these simple decision-making processes (modeled with PNs in the previous section) have been implemented using IEC-61131-3 ladder logic diagrams (LLD) for the locations shown in Fig. 9. LLD is one of the most extended programing languages for PLCs. The previously proposed PNs come in handy to generate these LLDs. The way to convert places and transitions in PN to contacts in LLD has evolved from the approach proposed in [41], with the systematic conversion methods that can be found in [42] and the definition of strict formalization rules shown in [43].

To allow physical BDI agents to update their believes, and therefore Markov states, the RFID read points, also shown in Fig 9, were also included in the shop floor.

C. Discussion

The following facts constitute a great step forward for control systems, particularly when rescheduling is required after a disruption happens (i.e., rush orders, delays in raw materials, breakdowns).

1) The product scheduling, based on MDP reasoning in physical BDI agents, is continuously been updated with the information coming from the shop floor using EPCIS.
2) The shop floor decision making is led by the priority values set by the Physical Agent Platform via the instructions that are sent to the PLCs.
3) The capacity to modify these priority values in real time.

In traditional systems, a disruption conveys to manually introduce new instructions in the system, with the unavoidable deviation from the initial plan, or the impossibility to respond appropriately. With the proposed method, the system automatically generates new instructions, and rearranges the existing ones, modifying instructions (mainly priority or destination values), or reassigning instructions to new orders. Modifying instructions directly influence on the shop floor decision making, and therefore, in the shop floor.

In order to correctly modify the instructions in progress, it is vital to have accurate information on the state of the shop floor. RFID readers located at specific areas generate this visibility of information. This way, apart from efficiently modifying instructions, it is possible to avoid the saturation of the communication between controllers and physical BDI agents. Take as an example the case of an order with only one missing pallet. Physical BDI agents would increase the priority of the remaining instruction/pallet over and over with the aim of completing the order if they did not count with the feedback information provided by RFID. However, if the required pallet is near the exit, but blocked by other products, such encouragement instructions for fast completion would not have any reaction. The information coming from RFID generates a picture of the state of the plant avoiding those situations. In addition, it allows to include instructions to pallets in progress and pallet at docks in the rescheduling, decreasing the time of preparation of rush orders, with an agility and flexibility never seen before (i.e., a pallet near the exit can be reallocated to a rush order while other is being brought to replace it).

Finally, it is worth to mention that this paper deals with the movement of loads inside a distribution center, but it could also be implemented on a production line or a flexible manufacturing system making intensive use of the machine models, and adapting the Markov states, business step, and read points of the physical BDI agents.

VI. Conclusion

This paper addresses the problem of the implementation of agent technology within industrial environments through a new integration of physical BDI agents in PLC controlled systems. The motivation behind the choice to use agents lies on the need to generate real-time control systems that are able to face the unforeseen events that keep appearing in the everyday operation of companies while providing a distributed architecture that allows tackling complexity. To facilitate implementation, a two-level architecture has been defined in this paper: IS and physical. IS BDI agents are in charge of the tactical/strategic operations of the company. Physical BDI agents are in charge of planning, arrange tasks (called instructions), and define priorities; while, shop floor control (PLCs) carry out those instructions at shop floor level. This uncoupling between the physical details of the shop floor and the high level of the decision making can be justified by the need to get an optimal cohesion between the operational levels and the strategic levels of the shop floor.

PNs are used here to model the decision-making process dealing with the handling of materials in distribution centers. These decision-making processes are driven by the information contained in the instructions (as priority or destination) provided by physical BDI agents. All models proposed here have been satisfactorily implemented within the industrial controllers of an experimental platform representing the distribution center of a local company.

The proposed decision-making process is based on the information provided by physical BDI agents and provides capacity to modify ongoing instructions and their priority in real time; which constitutes a great step forward toward increasing control efficiency. In the significant case where rescheduling is required, i.e., when a disruption happens (e.g., rushed orders), it is necessary to perform a new scheduling in order to respond to the new requirements. With the proposed method, physical BDI agents automatically generate new instructions, rearrange the existing ones, modify ongoing instructions (mainly priority level or destination id), or reallocate instructions to new orders. This way of modifying instructions directly influences the behavior of the shop floor control, making it far more responsive and efficient in the face of dynamic changes.

References

[1] T. Chaari, S. Chaabane, N. Aissani, and D. Trentesaux, "Scheduling under uncertainty: Survey and research directions," in *Proc. Int. Conf. Adv. Logistics Transp.* Hammamet, Tunisia: IEEE Press, 2014, pp. 229–234.

[2] A. I. Pettersson and A. Segerstedt, "Measuring supply chain cost," *Int. J. Prod. Econ.*, vol. 143, no. 2, pp. 357–363, 2013.

[3] O. Shehory, A. Sturm, J. P. Müller, and K. Fischer, "Application impact of multi-agent systems and technologies: A survey," in *Agent-Oriented Software Engineering*, Berlin, Germany: Springer, 2014, pp. 27–53.

[4] P. Leitao, V. Marik, and P. Vrba, "Past, present, and future of industrial agent applications," *IEEE Trans. Ind. Informat.*, vol. 9, no. 4, pp. 2360–2372, Nov. 2013.

[5] K. Finkenzeller, *RFID Handbook: Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication*. New York, NY, USA: Wiley, 2010.

[6] M. Wooldridge and N. Jennings, "Intelligent agents: Theory and practice," *Knowl. Eng. Rev.*, vol. 10, no. 2, pp. 115–152, 1995.

[7] H. Van Brussel, "Holonic manufacturing systems," in *CIRP Encyclopedia of Production Engineering*, Berlin, Germany: Springer, 2014, pp. 654–659.

[8] P. Leitão and F. Restivo, "ADACOR: A holonic architecture for agile and adaptive manufacturing control," *Comput. Ind.*, vol. 57, no. 2, pp. 121–130, 2006.

[9] J. Barbosa, P. Leitão, E. Adam, and D. Trentesaux, "Dynamic self-organization in holonic multi-agent manufacturing systems: The ADACOR evolution," *Comput. Ind.*, vol. 66, pp. 99–111, 2015.

[10] P. García Ansola, J. de las Morenas, A. García, and J. Otamendi, "Distributed decision support system for airport ground handling management using WSN and MAS," *Eng. Appl. Artif. Intell.*, vol. 25, no. 3, pp. 544–553, 2011.

[11] S. Bussmann, N. Jennings, and M. J. Wooldridge, *Multiagent Systems for Manufacturing Control: A Design Methodology*. New York, NY, USA: Springer-Verlag, 2004.

[12] F. L. Bellifemine, G. Caire, and D. Greenwood, *Developing Multi-Agent Systems With JADE*, vol. 7. Hoboken, NJ, USA: Wiley, 2007.

[13] R. M. da Silva, F. Junqueira, D. J. S. Filho, and P. E. Miyagi, "Control architecture and design method of reconfigurable manufacturing systems," *Control Eng. Pract.*, vol. 49, pp. 87–100, Apr. 2016.

[14] P. Leitão, N. Rodrigues, J. Barbosa, C. Turrin, and A. Pagani, "Intelligent products: The GRACE experience," *Control Eng. Pract.*, vol. 42, pp. 95–105, 2015.

[15] K. Kruger and A. Basson, "Erlang-based control implementation for a holonic manufacturing cell," *Int. J. Comput. Integr. Manuf.*, vol. 30, pp. 1–12, 2016.

[16] A. Giret and D. Trentesaux, "Software engineering methods for intelligent manufacturing systems: A comparative survey," in *Proceedings of the 7th International Conference on Industrial Applications of Holonic and Multi-Agent Systems, Valencia, Spain, Sep. 2–3, 2015*, V. Mařík *et al.*, Eds., Cham, Switzerland: Springer, 2015, pp. 11–21.

[17] A. Soylemezoglu *et al.*, "A testbed architecture for Auto-ID technologies," *Assem. Autom.*, vol. 26, no. 2, pp. 127–136, 2006.

[18] T. Borangiu *et al.*, "Assessment of EPCIS standard for interoperable tracking in the supply chain," in *Service Orientation in Holonic and Multi Agent Manufacturing and Robotics*, vol. 472. Berlin, Germany: Springer, 2013, pp. 119–134.

[19] G. Zacharewicz, J.-C. Deschamps, and J. Francois, "Distributed simulation platform to design advanced RFID based freight transportation systems," *Comput. Ind.*, vol. 66, no. 6, pp. 597–612, 2011.

[20] M. Zawodniok, S. Jagannathan, C. Saygin, and A. Soylemezoglu, "RFID-based asset management of time and temperature sensitive materials," in *Engineering Asset Management 2011*. New York, NY, USA: Springer, 2014, pp. 549–561.

[21] P. Vrba and V. Mařík, "Capabilities of dynamic reconfiguration of multiagent-based industrial control systems," *IEEE Trans. Syst., Man. Cybern. A Syst., Humans*, vol. 40, no. 2, pp. 213–223, Mar. 2010.

[22] P. Vrba *et al.*, "Rockwell automation's holonic and multiagent control systems compendium," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 41, no. 1, pp. 14–30, Jan. 2011.

[23] P. Vrba, F. Macurek, and V. Mařík, "Using radio frequency identification in agent-based control systems for industrial applications," *Eng. Appl. Artif. Intell.*, vol. 21, no. 3, pp. 331–342, 2008.

[24] D. Patel, J. Bhatt, and S. Trivedi, "Programmable logic controller performance enhancement by field programmable gate array based design," *ISA Trans.*, vol. 54, pp. 156–168, 2015.

[25] V. Mařík, M. Fletcher, and M. Pěchouček, "Holons & agents: Recent developments and mutual impacts," in *Multi-Agent Systems and Applications II*. Berlin, Germany:Springer, 2002, pp. 89–106.

[26] M. Pechouček and V. Mařík, "Industrial deployment of multi-agent technologies: Review and selected case studies," *Auton. Agents Multi-Agent Syst.*, vol. 17, no. 3, pp. 397–431, 2008.

[27] Y. Sallez, T. Berger, S. Raileanu, S. Chaabane, and D. Trentesaux, "Semi-heterarchical control of FMS: From theory to application," *Eng. Appl. Artif. Intell.*, vol. 23, no. 8, pp. 1314–1326, 2010.

[28] FIPA-OS, "Foundation for intelligent physical agents. Specifications and documentation," 2014. [Online]. Available: http://www.fipa.org

[29] L. Yu, A. Schüller, and U. Epple, "On the engineering design for systematic integration of agent-orientation in industrial automation," *ISA Trans.*, vol. 53, no. 5, pp. 1404–1409, 2014.

[30] S. Theiss, V. Vasyutynskyy, and K. Kabitzsch, "Software agents in industry: A customized framework in theory and praxis," *IEEE Trans. Ind. Informat.*, vol. 5, no. 2, pp. 147–156, May 2009.

[31] P. García Ansola, A. García, and J. de las Morenas, "Aligning decision support with shop floor operations: A proposal of intelligent product based on BDI physical agents," in *Service Orientation in Holonic and Multi-Agent Manufacturing*, vol. 594. T. Borangiu *et al.*, Eds. New York, NY, USA: Springer, 2015, pp. 91–100.

[32] D. McFarlane, V. Giannikas, A. Y. Wong, and M. Harrison, "Intelligent products in the supply chain - 10 years on," in *Service Orientation in Holonic and Multi Agent Manufacturing and Robotics*, vol. 472. Berlin, Germany: Springer, 2013, pp. 103–117.

[33] C. Gerber, J. Siekmann, and G. Vierke, "Holonic multi-agent systems," Université- und Landesbibliothek, Stuttgart, Germany, Tech. Rep. no. RR-99-03, 1999.

[34] T. Murata, "Petri nets: Properties, analysis and applications," *Proc. IEEE*, vol. 77, no. 4, pp. 541–580, Apr. 1989.

[35] L. D. Murillo-Soto, "Redes de Petri: Modelado e implementación de algoritmos para autómatas programables," *Revista Tecnología en Marcha*, vol. 21, no. 4, 2008, pp. 102–125.

[36] G. Frey and L. Litz, "Verification and validation of control algorithms by coupling of interpreted Petri nets," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, 1998, vol. 1, pp. 7–12.

[37] W. Reisig, "Marking and covering graphs," in *Understanding Petri Nets: Modeling Techniques, Analysis Methods, Case Studies*. Berlin, Heidelberg: Springer, 2013, pp. 131–139.

[38] J. L. Peterson, "An introduction to petri nets," in *Proc. Nat. Electron. Conf.*, 1978, pp. 144–148.

[39] B. Berthomieu, P. O. Ribet, and F. Vernadat, "The tool TINA-Construction of abstract state spaces for petri nets and time petri nets," *Int. J. Prod. Res.*, vol. 42, no. 14, pp. 2741–2756, 2004.

[40] G. Liu, "Complexity of the deadlock problem for Petri nets modeling resource allocation systems," *Inf. Sci.*, vol. 363, pp. 190–197, 2016.

[41] M. Uzam, A. H. Jones, and N. Ajlouni, "Conversion of Petri net controllers for manufacturing systems into ladder logic diagrams," in *Proc. IEEE Conf. Emerging Technol. Factory Autom.*, 1996, vol. 2, pp. 649–655.

[42] J. Brusey, D. C. McFarlane, and A. Thorne, "Nonautonomous elementary net systems and their application to programmable logic control," *IEEE Trans. Syst., Man., Cybern. A Syst., Humans,* vol. 38, no. 2, pp. 397–409, Mar.2008.

[43] M. V. Moreira and J. C. Basilio, "Bridging the gap between design and implementation of discrete-event controllers," *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 1, pp. 48–65, Jan. 2011.

Javier de las Morenas received the M.Sc. degree in industrial engineering and the Ph.D. degree in mechatronics from the University of Castilla-La Mancha, Ciudad Real, Spain, in 2007 and 2012, respectively.

In 2007, he joined the AutoLog Research Group, Ciudad Real, Spain, where he is currently an Assistant Researcher in public and private projects related to the development of new RFID, WSN, and control applications. Since 2013, he has been an Assistant Professor in the Mining and Industrial Engineering School of Almadén, Almadén, Spain, where he is currently the Team Manager of the Electric Mobility Research Initiative for the development of electric prototypes. His research interests include automation for manufacturing and logistics focused on the application of distributed control systems and RFID.

Andrés García-Higuera received the Ph.D. degree in engineering in 1999, and the Eng. and M.Sc. degrees in electric/electronic engineering from the Polytechnic University of Madrid, Madrid, Spain, in 1991, with a previous degree in mechanical engineering in 1987.

Since January 2013, he has been a Full Professor of industrial automation, electronics, and control in the School of Engineering, University of Castilla-La Mancha, Ciudad Real, Spain. He is currently the Director of the research group AutoLog, Ciudad Real, Spain, and the Deputy Director/Dean of the Institute for Advanced Technologies ITAV. He was a Senior Researcher in the Institute for Manufacturing, University of Cambridge, at a position granted by the Massachusetts Institute of Technology and has a long experience of working in and for the industry. He supervises several nationally funded research projects with strong industrial participation and is the President of the Spanish Society for RFID and Tracking (Setra). He was appointed as an External Reviewer/Independent Expert for different projects funded by the European Commission and as Collaborator of the STOA-EPRS of the European Parliament.

Pablo García-Ansola received the M.Sc. degree in computer science in 2006 and the MBA degree in business administration, management and operations in 2007 from the University of Castilla-La Mancha (UCLM), Ciudad Real, Spain.

He is a currently a Research Scientist in the Autolog Labs, UCLM, with a stay at the University of Cambridge, Cambridge, U.K. He is also the Manager of the Autolog spin-off, called "Securware," which received several innovation awards such as the IDEA, Innovared, and Desafio 22. Before joining the UCLM, he was a Consultant involved in the business intelligent area of major Spanish consultancy INDRA. His research interests include integration of intelligent systems in operational decision-making support.