Technology-Independent Demonstrator for Testing Industry 4.0 Solutions

Alejandro Lopez*, Lucas Sakurada†, Paulo Leitao†, Oskar Casquero*, Elisabet Estevez‡, Fernando De la Prieta§, and Marga Marcos*

* Systems Engineering and Automatic Control Department, University of the Basque Country (UPV/EHU), Bilbao, Spain
Email: {alejandro.lopez, oskar.casquero, marga.marcos}@ehu.eus

† Research Centre in Digitalization and Intelligent Robotics (CeDRI), Polytechnic Institute of Bragança, Campus de Santa Apolónia, 5300-253 Bragança, Portugal
Email: {lsakurada, pleitao}@ipb.pt

‡ Electronics and Automation Engineering Department, University of Jaén, Jaén, Spain
Email: eestevez@ujaen.es

§ BISITE Digital Innovation Hub, University of Salamanca, Edificio Multiusos I+D+i, 37007, Salamanca, Spain
Email: fer@usal.es

***Abstract*—Cyber-Physical Systems (CPS) are devoted to be the main participants in Industry 4.0 (I4.0) solutions. In recent years, many authors have focused their efforts on making proposals for the design and implementation of CPS based on different digital technologies. However, the comparative evaluation of these I4.0 solutions is complex, since there is no uniform criterion when it comes to defining the test scenarios and the metrics to assess them. This paper presents a technology-independent CPS demonstrator for benchmarking I4.0 solutions. To that end, a set of testing scenarios, Key Performance Indicators and services were defined considering the available automation cells setup. The proposed demonstrator has been used to test an I4.0 solution based on a Multi-agent Systems (MAS) approach.**

Keywords: Benchmarking, Cyber-Physical Systems, Demonstrator, Industry 4.0.

I. Introduction

Over the last decade, academic and industry interest in Cyber-Physical Systems (CPS) has increased exponentially. CPS participants are components consisting of a physical part (e.g. any relevant asset in a manufacturing process, such as a robot or a machining station), and a virtual part, which is responsible for giving visibility and connectivity to the CPS in the system [1]. CPS play a fundamental role in the Industry 4.0 (I4.0) paradigm, as they provide autonomy and adaptability to the systems of which they are part, contributing to increase the efficiency of manufacturing processes [2].

For this reason, it is not difficult to find a large number of research papers devoted to the design and development of CPS [3]–[5]. An important part of these efforts focus on the problem of integration between the physical and virtual parts of CPS, and some of these works propose ad-hoc solutions oriented to the integration of a specific asset (or type of asset) in a specific case study [6], [7]. Although these solutions may be illustrative, they lack scalability and reusability, so other works focus on proposing more generic solutions based on some of the different technologies enabling the realization of CPS [8], [9]. In this regard, when it comes to integrating physical assets, two approaches stand out above the rest:

- Plattform Industrie 4.0 proposes using Internet of Things (IoT) protocols, such as Message Queuing Telemetry Transport (MQTT), and specially Open Platform Communications Unified Architecture (OPC UA), for the integration of assets, as well as for the interoperability between I4.0 components [10].
- The use of industrial agents is another consolidated alternative for the integration of physical assets, since it is supported by international standards such as the IEEE 2660.1-2020 on recommended practices for industrial agents [11].

In addition to the problem of integrating the physical and virtual counterparts of a CPS, researchers working in this area face an additional problem when presenting and highlighting their contributions: the difficulty in comparing the results obtained by their deployed solutions. One of the main causes of this problem is the use of distinct industrial testing scenarios, including different metrics and Key Performance Indicators (KPIs), which makes the comparative analysis of results difficult. Some authors have focused on comparing different proposals in order to define a common set of metrics to assess CPS. However, these approaches focus on very specific areas, such as security [12] and communication protocols [13]. Besides, even in cases where the metrics used are identical, or at least comparable, it is often difficult to make valid comparisons because the tests have been performed on different demonstrators, and therefore under different conditions.

Thus, a possible solution to avoid these problems would be to compare two or more I4.0 solutions by testing them on the same demonstrator, under the same conditions, and quantifying the same metrics. However, this is not a trivial task because, as a general rule, the demonstrators in which I4.0 solutions are validated are ad-hoc. This is due to the fact that, in many cases, demonstrators are developed or adapted to fit the I4.0 solutions they are intended to test, becoming part of the solution itself [14], [15]. These works conceive the I4.0

solution and the demonstrator as a whole, in an indivisible way. This dependency prevents both the I4.0 solution from being tested on other demonstrators, and the demonstrator to be used for validating other I4.0 solutions. There are other cases in which the same demonstrator has been used in several works, as in the cases of the AIP-PRIMECA [16], [17] and myJoghurt demonstrators [18], [19]. Nevertheless, despite these works use the same demonstrator for similar uses, none of them refer to common testing scenarios and KPIs: they have been used for performing different tests, but not with the aim of benchmarking different I4.0 solutions. Finally, there are other works whose contribution is exclusively focused on proposing a reference for the development of demonstrators suitable for validating CPS [20], [21]. However, although these papers propose what hardware and software elements should compose a demonstrator, and what functionalities it should offer, once again they do not propose common scenarios or metrics for benchmarking I4.0 solutions.

Having this in mind, this paper describes the development of a technology-independent CPS demonstrator for benchmarking different I4.0 solutions, by enabling their integration in a quick and easy way. For this purpose, a set of testing scenarios, KPIs and services were defined according to the available automation cells setup. The applicability of the developed demonstrator is presented through an integration test with an I4.0 solution based on Multi-agent Systems (MAS).

The rest of the paper is organized as follows. Section II overviews the demonstrator, describing: a) the hardware elements that make it up and how their low-level control has been implemented; b) the testing scenarios that have been considered to evaluate I4.0 solutions and the KPIs associated with them; and c) the services that have been developed to integrate each asset within the demonstrator. Section III validates the generic nature of the demonstrator by presenting an integration example through an I4.0 solution based on MAS. Finally, Section IV rounds up the paper with the conclusions and future work.

II. Description of the Demonstrator

As aforementioned, the comparative evaluation of I4.0 solutions is complex, since most of the demonstrators on which these solutions are validated are technology-dependent and tested under different conditions. In this context, this paper focuses on developing a demonstrator, as illustrated in Figure 1, which allows the research community to test and compare different I4.0 solutions using the same metrics and conditions. This demonstrator comprises a small scale production system in which the I4.0 solutions are tested, a set of scenarios to measure the solution's performance, KPIs that allow comparing different I4.0 solutions, and the services that encapsulate the asset functionalities providing a technology-independent interface for any I4.0 solution.

A. Small-Scale Production System

The small-scale production system based on a Fischertechnik infrastructure, illustrated in Figure 1, is comprised of several assets, namely two punching machines, two indexed line machines, one industrial robot and warehouses. These assets offer several functionalities that simulate real machines' operation on a small scale prototype, e.g., transportation, punching and drilling of workpieces, allowing developers to test their solutions physically.

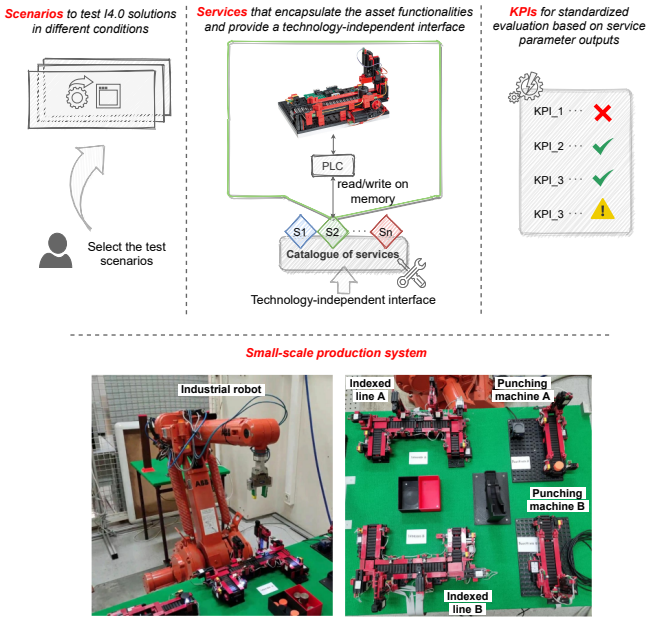

Figure 1. Overview of the CPS demonstrator for benchmarking I4.0 solutions.

Regarding the hardware configuration of each asset, the punching machines are composed of a conveyor belt with a punching device of workpieces, photoelectric sensors to detect the workpieces in the initial position of the conveyor and the position of punching, limit sensors to detect the movement of the punching device, and motors to move the conveyor and the punching device. The indexed line machines are composed of a conveyor line with a milling and drilling station in a U-shaped arrangement, photoelectric sensors to detect the workpieces in the conveyors and the processings positions, limit sensors to detect the movement of the embolus, and motors to actuate the conveyors, embolus and processing stations. Lastly, the robot is a 6-axis industrial IRB 1400 ABB robot designed for manufacturing purposes, which has been equipped with a gripper to transport the workpieces between the stations and the warehouses.

The low-level logic control of these machines is implemented as IEC 61313-3 programs running in a Modicon M340 Programmable Logic Controller (PLC), which allows reading and writing on memory addresses using the Modbus communication protocol. On the other hand, the ABB robot executes its operations using proper programs developed in RAPID language programming.

B. Testing Scenarios and KPIs

As described in Table I, the established testing scenarios aim to verify fundamental behaviours required for I4.0 solutions,

Table I
OVERVIEW OF TESTING SCENARIOS.

Scenario	Description	Highlighted Behaviour	KPI
#0	Reference scenario where everything works correctly	Normal operating capacity of the system. The goal is to calculate benchmark metrics for comparison with other scenarios	OEE, MLT
#1	At a given time, a rush order appears	Capacity of the system to handle the unpredictable rush orders and redistribute the tasks	TDT
#2	At a given time, a new machine is added to the manufacturing cell and is immediately available for production	Capacity of the system to adapt to evolution of the cell topology	TTO
#3	At a given time, one of the redundant machines will go down in a given time window	Capacity of the system to manage a redundant machine's breakdown	OEE, MLT, TTDE
#4	At a given time, the machine processing time increases for all its operations in a given time window	Capacity of the system to adapt to evolution of the machine processing time	OEE, MLT, TTDE
#5	At a given time, the network supporting the communication among decisional entities breaks down in a given time window	Capacity of the system to manage the loss of the decisional network	OEE, MLT, TTDE

OEE - Overall Equipment Effectiveness; MLT - Manufacturing Lead Time; TDT - Task Designation Time; TTO - Time to Operation; TTDE - Time to Detect the Event.

Table II
DESCRIPTION OF KPIS.

KPI	Description	Formula
OEE	Measures the effectiveness of a resource, considering whether the resource is operative or not (Availability, A), the number of products manufactured in a period (Performance, P), and their quality (Quality, Q)	$OEE = A * P * Q$ $OEE <= 1$
MLT	Time indicator to trace the process between the moment of the order request (Order time, OT) until the end of the process (Completion time, CT)	$MLT = CT - OT$
TDT	Time to redistribute tasks (Redistribution Time, RT) when some unexpected event occurs (Event Time, ET) and prevents the execution of the current task plan	$TDT = RT - ET$
TTO	Time for a new resource introduced in the cell topology (Pluging Time, PT) to be able to operate (Resource Ready Time, RRT)	$TTO = RRT - PT$
TTDE	Time to detect an event (Detection Time, DT), e.g., a failure, increase in operating time and loss of network, since it actually happened (Event Time, ET)	$TTDE = DT - ET$

namely the capacity of the system to handle the introduction of a rush order, adapt to the evolution of the cell topology or processing time, and manage a machine's breakdown or the loss of the network. Each scenario has associated KPIs, shown in Table II, to help assess the performance of the tested solution. Moreover, it is important to highlight that the test scenarios and KPIs defined in this subsection are attached to the demonstrator to measure the performance of the I4.0 solution in terms of flexibility. However, new scenarios and KPIs can be easily defined to test other relevant characteristics for I4.0 solutions.

C. Services

Aiming to facilitate the integration of any I4.0 solution with the demonstrator, a set of services was developed since each station is pre-configured only to read and write on the PLC memory registers using the Modbus protocol, which can be complex, error-prone and time-consuming to realize the integration with each asset in the demonstrator. Basically, the services encapsulate the particular form to operate the stations, such as requesting the start or state of a machine operation. In this case, the services were implemented using Node-RED[1] and offer a generic technology-independent interface that can be invoked externally. The services can be invoked using MQTT and RESTful HTTP, and are responsible for sending the commands to the PLC via Modbus to read and write in the memory registers. In this sense, the service requester does not need to be concerned about how to configure the memory registers to execute some manufacturing process, and thus only requests the service, e.g., using a POST request, intended to write data, on an URI representing a resource, e.g., `/Request/ManufacturingStation/{station}`.

The set of implemented services, described in Table III, has been designed to facilitate the realization of the proposed scenarios. To that end, two types of services can be distinguished: on the one hand, the so-called ordinary services allow the interaction with the stations to manage them under normal conditions (i.e., these services are the base for implementing the reference scenario #0 from Table I). On the other hand, some auxiliary services have been considered necessary to provide the conditions to test some scenarios. In this case, the services *forceStop()* and *forceDelay()*, aims to provide the necessary conditions for testing scenarios #3 and #4 (see Table I), respectively. Moreover, each service may require input parameters and provide outputs at the end of its execution. These parameters intend to provide helpful information, e.g., the timestamp values, which can be used to calculate some previous defined KPIs, e.g., "Time to Detect the Event".

III. EXAMPLE OF INTEGRATION USING THE DEMONSTRATOR

Based on the developed demonstrator, this section shows the simplicity of using the demonstrator with any I4.0 solution, in this case, by means of a MAS-based approach, exemplifying how the established services can be invoked using the proposed solution. MAS [22] allow decentralizing intelligence in I4.0 systems through a set of intelligent, autonomous and cooperative entities, named agents, which can make decisions, negotiate and interact with each other to achieve their goals aligned with the business purposes.

This example shows the integration into the demonstrator of a manufacturing platform developed with the Java Agent

[1]https://nodered.org/

Table III
SERVICES AVAILABLE TO INTERFACE WITH THE DEMONSTRATOR.

Ordinary Services				
Service Name	**Service Description**	**Parameters**		**Parameter Description**
stationProcess()	Starts the station process	Input	refSubproductType	Identifier of the type of product
		Output	itemNumber	Identifier of the type of product
			initialTimeStamp	Timestamp that shows the moment the operation on the item starts
			finalTimeStamp	Timestamp that shows the moment the operation on the item ends
stationStatus()	Returns the state of the station	Output	state	State of the manufacturing station
transportProcess()	Starts the transport operation	Input	targetPositions	Initial position and final position
		Output	initialTimeStamp	Timestamp that shows the moment the operation on the item starts
			finalTimeStamp	Timestamp that shows the moment the operation on the item ends
transportStatus()	Returns the state of the transport device	Output	stateTransport	State of the transport robot
resetDemonstrator()	Resets the demonstrator to its normal operating condition	Output	resetExecuted	Returns true if the service was performed or false otherwise
Auxiliary Services				
Service Name	**Service Description**	**Parameters**		**Parameter Description**
forceStop()	Stop running a specific manufacturing station	Input	refStation	Identifier of the station
		Output	stopExecuted	Returns true if the service was performed or false otherwise
forceDelay()	Applies a delay to a given manufacturing station	Input	refStation	Identifier of the station
		Output	delayExecuted	Returns true if the service was performed or false otherwise

DEvelopment (JADE) Framework[2], which is subsequently used to perform the test scenario #0 in Table I. Specifically, we focus on the machine agent, which is responsible for managing assets with the capacity to offer manufacturing services, and how it is integrated with the assets in the demonstrator described above.

As aforementioned, the integration with the demonstrator is facilitated through the implemented services. In this context, the agents are in charge of the high-level control functions, acting at a different layer than the low-level control, which is executed by the PLC, which executes IEC 61131-3 programs to fulfill the invoked services. Thus, the machine agent needs a "Machine" submodel, which represents the state of the asset (i.e., the services it can offer, the execution times required to accomplish each service, the type and quantity of consumable materials in case that machine requires of consumable materials because of the services it offers, etc.). Besides the "Machine" submodel, the machine agent manages a "MachinePlan" submodel, which contains the list of services that have been requested from the machine agent, including information about the requester and the expected start and finish times for the execution of the service.

Thus, when a machine agent instance is started, it is provided with information regarding the status of the machine and the list of tasks assigned to it at the time of starting the instance. From that moment on, this information is kept up to date by the agent itself. During this startup process, the machine agent must also verify that it can successfully interact with its asset, as this is a precondition for offering its services on the system. To that end, the machine agent invokes the *stationStatus()* service (see Table III). If it receives a reply, it continues with the startup process; on the other hand, if there is no response from the asset, it assumes there is a problem and interrupts the process.

Figure 2 shows a sequence diagram to illustrate how the *stationStatus()* service has been invoked. As it can be seen, the machine agent (MAgent_PA in Figure 2) interacts with its asset through a gateway developed using the JadeGateway class of the JADE library. This class allows generating a gateway (Gateway in Figure 2) that acts as a bridge between a JADE based MAS, in which communications are based on Agent Communication Language (ACL) messages, and some non-JADE envirionment (in this case, the interfaces offered by each asset through RESTful HTTP). In this way,

[2]https://jade.tilab.com/

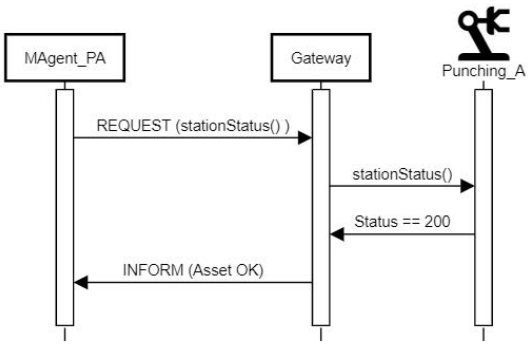

Figure 2. Machine agent requesting stationStatus() service.

the machine agent is decoupled from the communication capabilities of each asset: thus, when integrating two assets with different communication capabilities (or two identical assets through different communication protocols), it would only be necessary to modify the gateway, but not the machine agent. Hence, the machine agent sends an ACL message to the gateway, requesting the *stationStatus()* service, so the gateway transmits this request to the asset. Finally, when the gateway receives the response from the asset, it forwards it to the machine agent.

Once all the demonstrator assets have been integrated, the scenario #0 from Table I can be executed. To do so, a planner agent has been used to load a series of manufacturing orders into the system. These orders required the same type of product, but in different quantities (from 1 item to a maximum of 20), planning one item every 90 seconds. In this example, the desired product is obtained by introducing a raw material in a punching machine, and moving the resulting unfinished product to an indexed line. The movements between machines, as well as from/to the warehouses are carried out by the robot. Table IV summarizes the services involved in the process and their estimated execution times.

The results achieved from the execution of these manufacturing orders for the two KPIs considered for the scenario #0 (OEE and MLT, see Table II) are presented in Figure 3 and Figure 4, respectively. More specifically, Figure 3 presents the evolution of the OEE and its three components (Availability, Performance and Quality), depending on the number of items, for the set of machines involved in the manufacturing process (i.e., the robot, a punching machine and an indexed line) together. As it can be seen, the availability component shows a slight downward trend as the number of items manufactured increases. This is due to the fact that the mean time required to manufacture each item is about 86 seconds (the sum of all the execution times listed in Table IV), slightly less than the 90 seconds planned in the production orders. Thus, as more parts are produced, the availability tends to a stable value ($86s/90s = 95.56\%$). This is the factor that has had the greatest influence on OEE, since the performance component has not been affected by the increase in the number of items (changes are driven by a small variability in the cycle time of the indexed line) and the quality component is assumed to be 1 (as the small-scale production system is not equipped with a quality control station).

With respect to the MLT, Figure 4 shows the mean value (in blue) and the overall value (in orange), depending on the number of items. In the mean values, it can be seen how there is a gap between the first point of the graph (results for a single item), and the rest. The reason is that the first test shows the actual cycle time for the manufacturing process (around 86 seconds), while the following tests include the wait between the end of one item and the start of the next explained in the previous paragraph, tending to 90 seconds. On the other hand, this is also visible in the overall values, which, after a slight variation in the first section, show a linear trend.

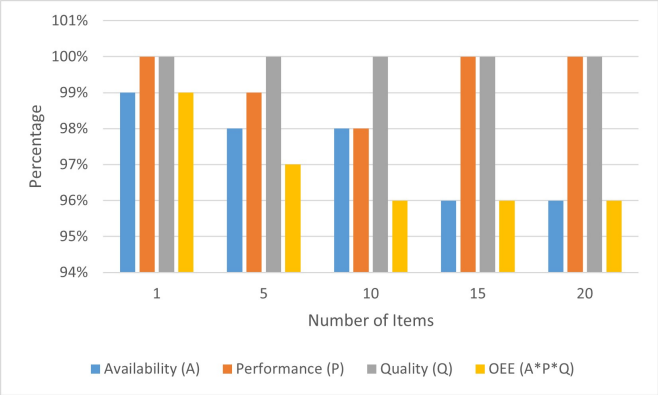

Figure 3. Evolution of the OEE with respect to the number of items.

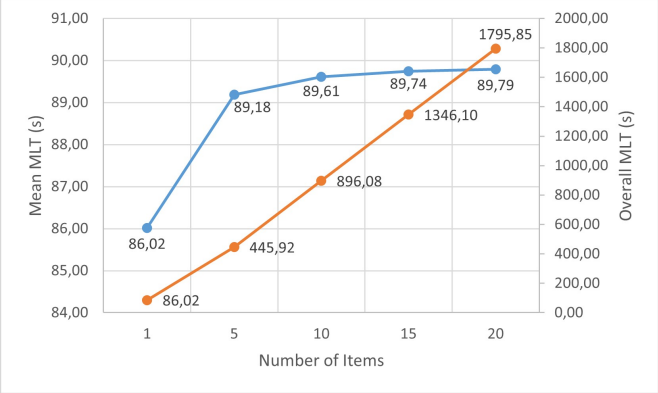

Figure 4. Evolution of the MLT with respect to the number of items.

IV. Conclusions and Future Work

CPS assume a crucial relevance in implementing I4.0 solutions, integrating cyber and physical counterparts to develop large-scale systems that perform tasks in a decentralized and intelligent manner. However, researchers in this area face an additional problem when presenting and highlighting their contributions. In general, the demonstrators on which I4.0 solutions are validated are technology-dependent and present

Table IV
DESCRIPTION OF SERVICES INVOLVED IN SCENARIO #0.

Step	Asset Name	Service Name	Input Parameters		Execution Time
#1	Industrial Robot (Robot)	transportProcess()	targetPositions	Initial=warehouse_2, Final=PA	15 seconds
#2	Punching Machine A (PA)	stationProcess()	refSubproductType	1	17 seconds
#3	Industrial Robot (Robot)	transportProcess()	targetPositions	Initial=PA, Final=IA	16 seconds
#4	Indexed Line A (IA)	stationProcess()	refSubproductType	1	26 seconds
#5	Industrial Robot (Robot)	transportProcess()	targetPositions	Initial=IA, Final=warehouse_3	12 seconds

different conditions and metrics, making it difficult to compare one approach with others.

This paper presented the development of a technology-independent CPS demonstrator, which allows to efficiently test and compare different I4.0 solutions using the same metrics and test conditions. For this purpose, a set of test scenarios have been proposed, in which the flexibility provided by these solutions is assessed through different KPIs. Besides, a list of services have been defined to facilitate the interaction with the demonstrator and support the execution of the test scenarios. The applicability of the demonstrator is illustrated by integrating an I4.0 solution based on MAS with its manufacturing assets, and the later execution the reference testing scenario.

Future work will be devoted to extending the testing scenarios, services and KPIs, as well as developing a report and recommendation system for the demonstrator to provide automatic feedback related to performed tests, e.g., based on a score to compare with other approaches already tested and implemented.

ACKNOWLEDGMENT

This work has been supported by FCT - Fundação para a Ciência e Tecnologia within the Project Scope: UIDB/05757/2020 and also by MCIU/AEI/FEDER, UE under grant number RTI2018-096116-B-I00.

REFERENCES

[1] L. Monostori, "Cyber-Physical Systems," in *CIRP Encyclopedia of Production Engineering*. Springer, 2018.

[2] S. Zanero, "Cyber-Physical Systems," *Computer*, vol. 50, no. 4, pp. 14–16, 2017.

[3] S. Karnouskos, P. Leitão, L. Ribeiro, and A. W. Colombo, "Industrial Agents as a Key Enabler for Realizing Industrial Cyber-Physical Systems," *IEEE Industrial Electronics Magazine*, vol. 14, no. 3, pp. 18–32, 2020.

[4] S. Cavalieri and M. G. Salafia, "Asset Administration Shell for PLC Representation Based on IEC 61131–3," *IEEE Access*, vol. 8, pp. 142 606—142 621, 2020.

[5] R. S. Peres, A. D. Rocha, P. Leitão, and J. Barata, "IDARTS – Towards intelligent data analysis and real-time supervision for industry 4.0," *Computers in Industry*, vol. 101, pp. 138—146, 2018.

[6] A. D. Neal, R. G. Sharpe, K. van Lopik, J. Tribe, P. Goodall, T. W. Jackson, and A. A. West, "The potential of industry 4.0 Cyber Physical System to improve quality assurance: An automotive case study for wash monitoring of returnable transit items," *CIRP Journal of Manufacturing Science and Technology*, vol. 32, pp. 461–475, 2021.

[7] J. de las Morenas, A. García-Higuera, and P. García-Ansola, "Shop Floor Control: A Physical Agents Approach for PLC-Controlled Systems," *IEEE Trans. Ind. Inf.*, vol. 13, no. 5, pp. 2417–2427, 2017.

[8] J. Arm, T. Benesl, P. Marcon, Z. Bradak, T. Schröder, A. Belayev, T. Werner, V. Braun, P. Kamensky, F. Zezulka, C. Diedrich, and P. Dohnal, "Automated Design and Integration of Asset Administration Shells in Components of Industry 4.0," *sensors*, vol. 21, no. 6, 2021.

[9] B. Vogel-Heuser, M. Seitz, L. A. Cruz Salazar, F. Gehlhoff, A. Dogan, and A. Fay, "Multi-agent systems to enable Industry 4.0," *at - Automatisierungstechnik*, vol. 68, no. 6, pp. 445—458, 2020.

[10] Plattform Industrie 4.0, "Details of the Asset Administration Shell - Part 1: The exchange of information between partners in the value chain of Industrie 4.0," 2020.

[11] IEEE Standards Association, "IEEE Recommended Practice for Industrial Agents: Integration of Software Agents and Low-Level Automation Functions," 2021.

[12] A. Aigner and A. Khelil, "A Benchmark of Security Metrics in Cyber-Physical Systems," in *Proc. of the IEEE International Conf. on Sensing, Communication and Networking (SECON Workshops)*, 2020, pp. 1–6.

[13] M. Iglesias-Urkia, A. Orive, M. Barcelo, A. Moran, J. Bilbao, and A. Urbieta, "Towards a lightweight protocol for Industry 4.0: An implementation based benchmark," in *Proc. of the IEEE Intl. Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM)*, San Sebastian, Spain, 2017.

[14] B. Karaduman, I. David, and M. Challenger, "Modeling the Engineering Process of an Agent-based Production System: An Exemplar Study," in *Proceedings of the ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, Fukuoka, Japan, 2021, pp. 296–305.

[15] T. Tomiyama and F. Moyen, "Resilient architecture for cyber-physical production systems," *CIRP Annals*, vol. 67, no. 1, pp. 161—164, 2018.

[16] J. Barbosa, P. Leitão, E. Adam, and D. Trentesaux, "Self-organized Holonic Multi-agent Manufacturing System: the Behavioural Perspective," in *Proceedings of the 2013 IEEE International Conference on Systems, Man, and Cybernetics*, Manchester, United Kingdom, 2013, pp. 3829–3834.

[17] B. Mihoubi, M. Gaham, B. Bouzouia, and A. Bekrar, "A Rule-Based Harmony Search Simulation-Optimization Approach for Intelligent Control of a Robotic Assembly cell," in *Proceedings of the 3rd International Conference on Control, Engineering Information Technology (CEIT)*, Tlemcen, Algeria, 2015.

[18] I. Kovalenko, D. Ryashentseva, D. Tilbury, and K. Barton, "Dynamic Resource Task Negotiation to Enable Product Agent Exploration in Multi-Agent Manufacturing Systems," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2854—2861, 2019.

[19] B. Vogel-Heuser, C. Diedrich, D. Pantförder, and P. Göhner, "Coupling heterogeneous production systems by a multi-agent based cyber-physical production system," in *Proceedings of the 12th IEEE International Conference on Industrial Informatics (INDIN)*, Porto Alegre, Brazil, 2014, pp. 713–719.

[20] J. S. Oks, M. Jalowski, A. Fritzsche, and K. M. Möslein, "Cyber-physical modeling and simulation: A reference architecture for designing demonstrators for industrial cyber-physical systems," in *Procedia CIRP*, Póvoa de Varzim, Portugal, 2019, pp. 257–264.

[21] J. Wermann, A. W. Colombo, A. Pechmann, and M. Zarte, "Using an interdisciplinary demonstration platform for teaching Industry 4.0," in *Procedia Manufacturing*, Braunschweig, Germany, 2019, pp. 302–308.

[22] M. Wooldridge, *An Introduction to MultiAgent Systems*. John Wiley and Sons, 2002.