Computers in Industry 101 (2018) 138–146

Contents lists available at ScienceDirect

Computers in Industry

journal homepage: www.elsevier.com/locate/compind

IDARTS – Towards intelligent data analysis and real-time supervision for industry 4.0

Ricardo Silva Peres[a,b,*], Andre Dionisio Rocha[a,b], Paulo Leitao[c,d], Jose Barata[a,b]

[a] UNINOVA - Centre of Technology and Systems (CTS), FCT Campus, Monte de Caparica, 2829-516, Caparica, Portugal
[b] Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, Departamento de Engenharia Electrotécnica, Monte de Caparica, 2829-516, Caparica, Portugal
[c] Polytechnic Institute of Bragança, Campus Sta Apolónia, Apartado 1134, 5301-857, Bragança, Portugal
[d] LIACC -Artificial Intelligence and Computer Science Laboratory, R. Campo Alegre 102, 4169-007, Porto, Portugal

ARTICLE INFO

Keywords:
Predictive manufacturing systems
Cyber-physical systems
Industry 4.0
Multi- agent systems
Data analytics

ABSTRACT

The manufacturing industry represents a data rich environment, in which larger and larger volumes of data are constantly being generated by its processes. However, only a relatively small portion of it is actually taken advantage of by manufacturers. As such, the proposed Intelligent Data Analysis and Real-Time Supervision (IDARTS) framework presents the guidelines for the implementation of scalable, flexible and pluggable data analysis and real-time supervision systems for manufacturing environments. IDARTS is aligned with the current Industry 4.0 trend, being aimed at allowing manufacturers to translate their data into a business advantage through the integration of a Cyber-Physical System at the edge with cloud computing. It combines distributed data acquisition, machine learning and run-time reasoning to assist in fields such as predictive maintenance and quality control, reducing the impact of disruptive events in production.

1. Introduction

With the advancements in the Information and Communication Technology field and the exponentially increasing volumes of data being generated every day, a new set of possibilities has been presented to improve the efficiency and the characteristics of production processes. Adding to this is the transformation from a saturated seller's market into a customer-driven one, with its growing demand for highly customized products accompanied by decreasing product lifecycles and smaller lot sizes, pushing companies towards a paradigm shift in order to leverage their data to attain a business advantage in such a competitive and dynamic market [1].

As such, the currently ongoing 4th industrial revolution, usually referred to as Industry 4.0 in Europe [2–4] and Industrial Internet in the US [5], aims to introduce and take advantage of the interconnected world along the entire value chain, allowing the sharing and processing of the data available in all of the its actors to generate relevant knowledge and optimize the overall process. The adoption of the Industry 4.0 paradigm encompasses the following three characteristics [6]:

- ○ "***Horizontal integration across the entire value network***": By integrating the overall value chain it is possible to optimize it beginning with the suppliers, materials, logistics, etc. In this sense, all of the value chain's actors must be connected and coordinated among each other based on their individual requirements, creating a very dynamic ecosystem.
- ○ "***End-to-end engineering across the entire product life-cycle***": The integration and digitalization across all phases of the product's life-cycle is crucial to ensure that data can be collected, stored and processed to generate new knowledge from the product's inception to its end of life. This knowledge can be particularly relevant for the product's improvement, not only regarding its production, but also the for instance its design or material suppliers.
- ○ "***Vertical integration and networked manufacturing systems***": At the shop-floor, the integration among the different components and actors (such as resources, humans and Manufacturing Execution Systems) should be done through a Cyber-Physical System (CPS). This system will allow not only the internal integration and optimization but also a harmonized and smooth integration with the two previously presented functionalities.

With the vertical integration emerges the concept of Smart Factory (SF). According to [7] these factories must have some characteristics

* Corresponding author at: UNINOVA - Centre of Technology and Systems (CTS), FCT Campus, Monte de Caparica, 2829-516, Caparica, Portugal.
E-mail addresses: ricardo.peres@uninova.pt (R.S. Peres), andre.rocha@uninova.pt (A. Dionisio Rocha), pleitao@ipb.pt (P. Leitao), jab@uninova.pt (J. Barata).

https://doi.org/10.1016/j.compind.2018.07.004
Received 30 October 2017; Received in revised form 5 June 2018; Accepted 10 July 2018

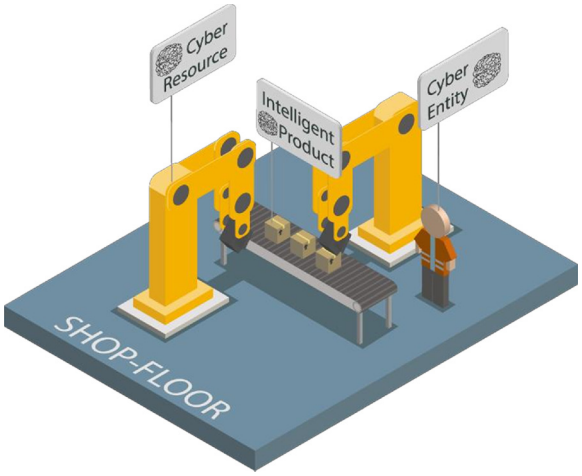

Fig. 1. Cyber-Physical Production Systems.

including among others the capacity to deal with mass customization [4], flexibility [8] and new maintenance strategies [9]. However, one of the main characteristics is the ability to generate new and relevant knowledge based on data processing [10]. As previously stated, SFs are implemented and deployed via CPS. The concept of Cyber-Physical Production System (CPPS) [11] merges the functionalities and benefits of CPS applied to the industrial context. The main objective in a CPPS is to create an abstraction layer where each of the shop-floor's actors is represented by a cyber entity, as shown in Fig. 1.

The communication between the heterogeneous components is now made at the cyber level, allowing a smooth and effective integration of all the components and actors avoiding the usual problems related to vendors' specifications and standards. In [12] the authors present a comparison between today's factories and the now emerging Industry 4.0 based SFs, implemented through CPPS. With all the resources integrated, sharing information and their behaviours among each other, the shop-floor can adapt and organize in runtime to optimize at different levels (production, maintenance, energy consumption, etc). Moreover, with the advances in the Industrial Internet of Things (IIoT) and the increasingly large number of sensors and other data sources available on the shop-floor, the amount of extracted data is growing and the traditional algorithms are no longer able to process these volumes of data. Hence, the big data analysis field is becoming more and more important in several areas as a way to tackle this challenge [13].

This is often coupled with the usage of Machine Learning (ML), allowing manufacturers to obtain insights regarding their factory which would have been otherwise missed. ML can be defined as a system's capacity to improve its performance on a given task or set of tasks over time based on previous results [14]. Therefore, ML algorithms can be used to predict a system's behaviour and/or improve its overall performance, enabling the development of tools capable of analysing data and perceive what are the underlying trends and correlations. Thus, ML-based approaches can be used to predict abnormal events (failures, degradation, energy consumption, etc), generate warnings and advise the system and/or the operator regarding which course of action to take, assisting in diagnosis and maintenance tasks [15].

In line with this, the work detailed in the coming sections aims to provide an integration of this real-time and historical analysis with self-improvement mechanisms under a single framework for Predictive Manufacturing Systems (PMS), along with an initial implementation of its core functionalities. It is structured as follows: Section 2 presents a summary of related work that can be found in current literature pertaining to CPPS-based PMS. Section 3 introduces the proposed framework and formalizes some core concepts, requirements and functionalities. Section 4 details the initial implementation of the framework and the integration of its modules. Afterwards, Section 5 describes the tests performed to validate the implemented solution, with the results being discussed in Section 6. This is followed by Section 7 in which a brief summary of the developments is presented, along with some conclusions and remarks regarding future work.

2. Related work

Over the last few years, significant efforts have been put into the research of the various facets of predictive manufacturing in Industry 4.0. In [16] the authors overview the recent advances and trends regarding CPPS and big data analysis, identifying self-predictiveness and self-awareness as key characteristics to gain insight into Industry 4.0 factories. Also, the authors mention that several sources of information in current prognostics methods remain untapped, such as peer-to-peer evaluation and historical life-cycle information from identical assets. Insightful discussions and guidelines regarding solutions for PMS can also be found in the current literature [17,18], with some common denominators including the employment of CPPS for virtualization, ML models for data analysis (e.g. early fault detection, quality control), decentralization and self-adjustment. However, the discussions are often on the conceptual or architectural level, with a lack of deployable implementations or results.

Regardless, the growing importance of PMS in the current information age is evident from its multitude of research venues. In [19], the authors survey several articles pertaining to the applications of PMS and propose to group them into four main application fields, namely system control [20], quality control [21], fault diagnosis [22], and predictive maintenance [23]. Other recent examples include an architecture for predictive maintenance as a service based on the cloud computing paradigm [24], the prediction of power consumption levels in machining processes through big data analytics [25] and a distributed multi-agent oriented framework for failure prediction from real-time sensor data [26].

Some frameworks for the application of predictive analytics in manufacturing environments have also been proposed. In [23], a CPPS-based framework for intelligent predictive maintenance is presented, using mostly the processing and feature extraction of real-time signal information as enablers of fault diagnosis and prognosis. Zhiqiang Ge [27] presents a distributed framework for the prediction and diagnosis of key performance indices in plant-wide industrial processes. One key aspect is the division of the entire process into several smaller blocks, later enabling data to be extracted more efficiently whilst greatly reducing its dimensionality. A fog computing-based framework for data-driven machine health and process monitoring in CPPS is introduced in [28], highlighting the importance of scalable, high-performance approaches and the usage of cloud-based machine learning algorithms for predictive analytics. Zhong et. al [29]. introduce a big data analytics framework for radio-frequency identification-enabled shop-floor logistics, which presents a considerable challenge in terms of complexity not only due to the large amount of assets involved, but also the intrinsic dynamic logic of the logistics domain.

Overall, it can be said that there is still a clear need to further combine real-time streams of data from the shop-floor with historical data at both the resource and system levels, as well as closing the loop to autonomously act on the results of the predictive analytics. Furthermore, in the context of Industry 4.0 systems, new solutions should be flexible to cope with topological (i.e. Plug-and-Produce) or requirement changes on the shop-floor, as well as scalable and high-performing in order to deal with the growing volumes of data. These solutions should also be highly adaptable, being capable of changing even after deployment by learning from newly generated knowledge [30], adapting at both the analysis and action fronts. This implies a continuous adaptation and dynamic improvement of their self-adjustment mechanisms during execution, avoiding unnecessary downtime for redeployment and additional programming effort on the deployed system. Finally, the generalization of the solutions should also be taken

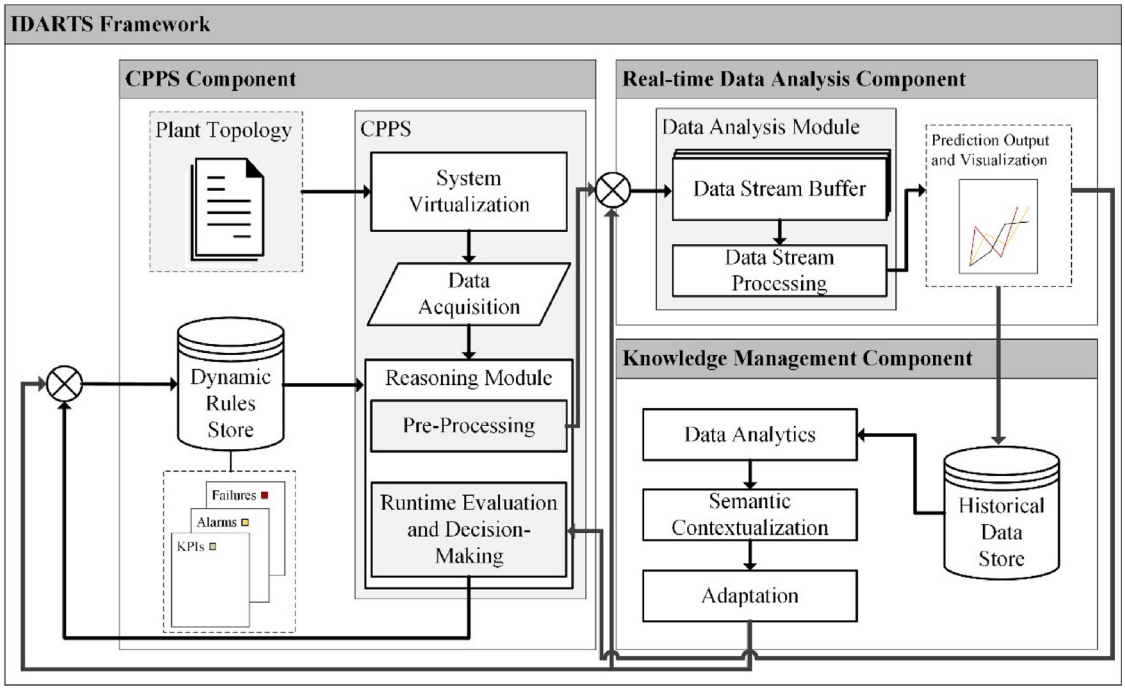

Fig. 2. Intelligent Data Analysis and Real-Time Supervision (IDARTS) Framework Overview.

into account, so that they can be easily migrated and applied to wide array of manufacturing scenarios and domains. These are the main differentiating characteristics of the framework proposed in this article, which will be detailed in Section 3.

3. The IDARTS framework

In this section, an overview of the proposed Intelligent Data Analysis and Real-Time Supervision (IDARTS) framework is provided along with its main design goals, requirements and a thorough description of each of its individual structural elements.

IDARTS targets not only the acquisition of data at different granularity levels, but also the realization of context-aware data analysis and evaluation based on both real-time and historical data. This analysis outputs predictive data, which in this context can be defined as probable future values or states forecasted based on models representing a given process, with prediction referring to the act of making statements about data that is yet to be observed [31]. Predictive data can then be used to trigger the system's self-adjustment mechanisms (e.g. reconfiguration) or alert operators in the shop-floor, thus assisting in returning a deviating or unstable manufacturing system to normal operation conditions, before critical breakdown events occur.

To this end, IDARTS' foundations lay on top of three core principles, namely:

- **The Integration of the Physical and Software Elements** – Through the application of a CPPS, IDARTS' real-time computation module should be capable of extracting data from the shop-floor and reason on it in order to assess possible deviations, acting accordingly. This should assist in preventing the propagation of anomalies downstream and returning the system to its normal operation conditions, either via self-adjustment mechanisms or alerts to trigger human intervention.
- **Seamless Data Exchange between Heterogeneous Components** – The employment of a common data representation and exchange format in crucial to ensure the interoperability of the heterogeneous components comprising an IDARTS-based platform.
- **Knowledge Management and Data Analytics** – Despite the exponential growth in the volume and velocity at which data is generated in manufacturing environments (e.g. embedded sensors), a large portion of it remains untapped. IDARTS' approach aims to translate this data into a business advantage by employing advanced data analysis and knowledge management methods on semantically enriched data acquired by the CPPS. The generated knowledge can be then used to improve the CPPS' reasoning system and the real-time analysis, hence further mitigating the occurrence of breakdown events during production. The approach encompasses the combination of real-time and historical data throughout the entire production, allowing the adaptation of the analysis and monitoring algorithms after deployment for a truly adaptable and flexible approach to predictive manufacturing.

Additionally, a number of non-functional requirements are imposed by the framework's design. First and foremost, it should be generic enough to be applicable to various different scenarios, being open so as to not depend on the existence of a single communication protocol or standard on the shop-floor, hence facilitating its industrial integration and adoption. Moreover, it needs to be flexible, capable of adapting to changes to the process or its assets in runtime, for instance in terms of pluggability, changes to the Key Performance Indicators (KPI) to be analysed or to improvements in the analysis process.

Another point to take into account is the aspect of scalability. To ensure that the approach is applicable to varied scenarios, it needs to be capable of scaling according to the requirements of the use case at hand, which might also translate in an increase in its complexity. These include for instance the number of assets to be virtualized in the shop-floor, as well as the volume, velocity and variety of data to be extracted and analysed. This reinforces a critical point, which is the need to connect, harmonize and transform heterogeneous data received from different sources [32].

Hence, to tackle these challenges, a layered, modular design approach is suggested. An overview of the resulting IDARTS framework can be seen in Fig. 2.

As it can be observed, the IDARTS framework is comprised of several modular components, each operating at its own abstraction level and with particular internal goals in order to decrease the overall

system complexity.

Firstly, the CPPS Component entails all the activities pertaining to the acquisition and processing of production data. This interacts directly with the Real-time Data Analysis (RDA) Component, which is responsible for analysing said data during the system's execution, providing relevant information relating to KPI trends, deviations and alarms. Finally, the Knowledge Management Component deals with the higher-level data analytics and knowledge generation based on the collected historical data, which can then provide feedback and updates to the previous modules. Each will be further detailed in the remainder of this section.

3.1. Cyber-physical production system component

The CPPS Component is composed of three main elements, namely the CPPS itself, the Plant Topology information and the Dynamic Rules Store. As the name suggests, the CPPS acts as the core element at the shop-floor level, playing the role of the centrepiece that glues the different constituents of the framework together, integrating both production and quality control processes.

The Plant Topology data should be an integral part of the system's data model, representing its existing resources, their organizational structure and other relevant information such as connection interfaces and existing data sources. Through it, the CPPS can be instantiated in a way that is capable of virtualizing each of the system's elements and initiate the data acquisition process. This System Virtualization creates a logical one-to-one relationship between each element of the shop-floor and its cyber representation, enabling a non-invasive application of the framework's capabilities. It also represents a way to break down the system into smaller, more manageable building blocks and thus reducing the complexity of the problem.

This data acquisition process is responsible for feeding new incoming data from the shop-floor not only to the reasoning module, but also to the RDA component and to an historical data store to be used by the Knowledge Management layer. To do so, this system needs to be both flexible and adaptable in order to deal with unforeseen disturbances at the shop-floor level in a robust and efficient manner.

Also, the communication with the shop-floor needs to be specified in a generic way, thus allowing the consideration of different requirements from potentially heterogeneous use cases. For instance, a specific case might present time constraints in the order of weeks or days, while another might require data to be collected and analysed in near real-time, allowing only the consideration of relatively small delays in the communication and processing and therefore requiring different approaches.

Finally, the CPPS is also responsible for the local processing of the collected data. This is done in two stages, the first of which deals with the pre-processing of raw shop-floor data and generation of more complex knowledge. The other refers to being capable of reasoning and following through with rule-based decision-making processes, providing an earlier identification of faults, potential deviations or other critical events. The basis of this behaviour is depicted in Fig. 3, in which the arrows indicate the general data flow throughout the process.

These rules are contained in a Dynamic Rule Store, and should be modelled using the system's common data representation format. The store can be updated dynamically during runtime by the Knowledge Management layer, if either as a result of the data analysis performed on the historical data it is found that certain changes are required to improve overall quality control, or if the CPPS requests an update from Knowledge Management due to having insufficient or outdated rules.

3.2. Real-time data analysis component

Concerning the runtime domain, the RDA Component encompasses the elements necessary to perform the analysis of relevant production-related data during the system's execution.

The first of these consists in the Data Stream Buffer, which should act as a robust data queue capable of handling high volumes of data while ensuring its reliable delivery. Through it streams of data collected by the CPPS can be then passed on to the Data Stream Processing. This, in turn, is responsible for the actual data analysis, focusing on the early detection of deviating patterns and trends that might lead to breakdown or failure events on the shop-floor. Hence, due to this capacity for predictive analysis in runtime, the RDA component acts a key-enabler of condition- based maintenance, allowing manufacturers to schedule maintenance operations before a failure actually occurs, thus diminishing the direct impact on production.

The output of this module should then be visually represented in order to facilitate its comprehension by human operators, as well as being passed back to the CPPS so that its runtime decision-making component can trigger a self-adjustment response or suggest possible maintenance actions that might be required to return the system to its normal operation conditions.

3.3. Knowledge management component

Contrastingly, the Knowledge Management Component operates outside of the constraints imposed by the real-time execution and monitoring of the production system. It consists in combination of a Historical Data Store and 3 processing modules, namely Data Analytics, Semantic Contextualization and Adaptation.

Each of these modules is responsible for a different step of the knowledge management pipeline. While the Data Analytics component refers to the actual data analysis process, Semantic Contextualization deals with capturing domain-expert knowledge and enriching the results with meaningful, easily understandable context. This is extremely important because it assists not only human operators, but also the CPPS in interpreting the analysis results. Lastly, the Adaptation component handles the management and refinement of the decision-making rules and runtime analysis processes.

While the analysis performed at runtime focuses solely on the constantly incoming streams of raw data, the one done at the higher-level additionally takes into account historical data, not only raw but also the more complex knowledge generated by the CPPS. This makes it possible to generate new knowledge from correlations and patterns that might be harder or impossible to discover in the RDA alone. This can then be used to update the rules that govern the CPPS' reasoning mechanisms or models used in the RDA, either periodically or on request, therefore improving the overall quality of the manufacturing processes.

4. Implementation

An initial implementation was developed focusing on the integration of the IDARTS' core elements. The main goal at this stage is to showcase an initial implementation of the data acquisition and pre-processing CPPS, the real-time processing module and its integration with a Knowledge Management tool, as well as all the data flow in between and its required interfaces. Each of these elements is described in the following subsections.

4.1. MAS-based cyber-physical production system

As mentioned in Section 3, the CPPS needs to be capable of extracting data from the shop-floor during execution in a flexible and robust manner. To this end, a Multi-Agent System (MAS) was implemented using the Java Agent DEvelopment framework (JADE) [33] based on the monitoring approach developed in [34], previously validated in an automotive industry cell under the FP7 PRIME project [35]. JADE provides a robust infrastructure which supports the agents' core behavioural logic and communication, as well a wide array of auxiliary tools to further facilitate the development process.

The MAS-based CPPS abstracts both components (e.g. robots,

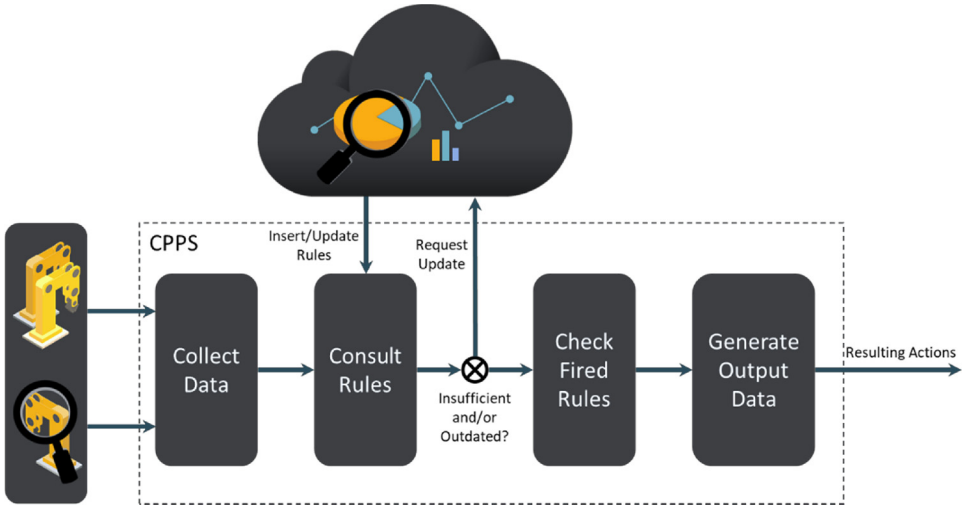

Fig. 3. CPPS's Rule-based Reasoning Flow.

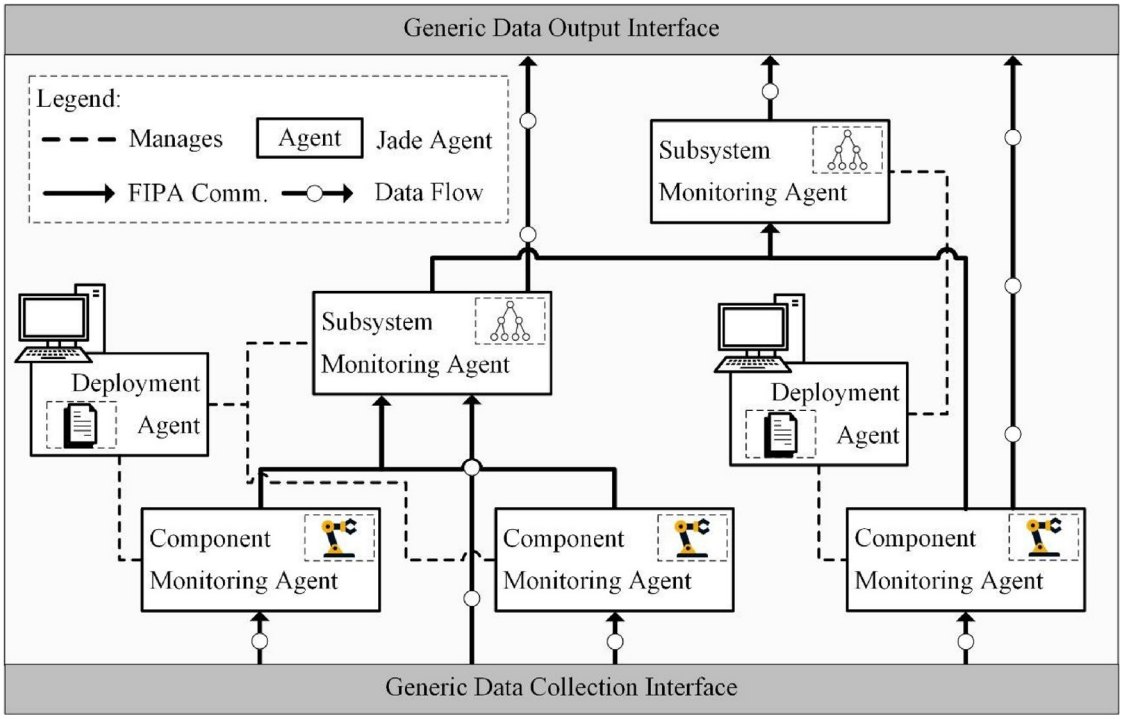

Fig. 4. CPPS Multi-Agent System Overview.

conveyors, sensors) and subsystems (e.g. cells, workstations), taking care of the acquisition of their respective data as well as its pre-processing, preparing it to be further analysed by the other framework modules. The adoption of MAS technology confers additional flexibility and robustness to the CPPS, allowing it to quickly adapt to changes in the shop-floor as imposed by the framework's requirements. The MAS' support for pluggability, combined with the framework's modular design, allow for the components it abstracts to be plugged/unplugged in runtime without requiring additional reprogramming effort in the overall system. An overview of the MAS architecture is provided in Fig. 4.

The Component Monitoring Agent (CMA) presents a one-to-one relation with a component on the shop-floor (e.g. clamp, robot), being the sole responsible for extracting and pre-processing the data from the resource it is abstracting, as explained in [35]. This extraction can be event-based (i.e. triggered by the low-level, for instance an on-change event from an OPC-UA server) or time-based, with the agent periodically polling the corresponding asset in a cyclic behaviour.

Similarly, the Subsystem Monitoring Agent (SMA) fulfils a comparable role, albeit on a higher- level of abstraction. However, not only does the SMA extract raw data from its associated subsystem (set of components and/or other subsystems) but it also receives the data computed by other components or lower-level subsystems associated to it. As such, due to its broader view of the system, it is able to generate more complex knowledge that would have remained untapped otherwise. This is an example of the data fusion performed by the CPPS, through which different data sources are combined to generate new meaningful and useful data. The agents' interactions are FIPA compliant, following the FIPA Request protocol [36].

Although the aforementioned monitoring approach employed an additional type of agent to handle the output of data, in this context this could result in a bottleneck when scaling the number of deployed

agents, mainly due to the unnecessary load imposed on JADE's messaging system during output negotiation. As such, in the current implementation this agent type was scraped off, with each CMA and SMA handling its output independently instead. This not only avoids the bottleneck, but also improves robustness, since even if a certain monitoring agent fails, only the data pertaining to its particular resource will be affected, as opposed to crippling the output of the entire CPPS.

For each machine in the system, a Deployment Agent (DA) must be present to manage the remaining agents associated with a particular area running in the same machine. Each DA periodically checks the system's online topology, launching or killing agents as assets are plugged or unplugged from the system.

Lastly, the employment of generic communication interfaces for the shop-floor data acquisition as well as the data exchange between modules allows the approach to remain independent from the underlying protocols or technologies present at each different application scenario. This means that even within the same CPPS instance, different agents can interface with their physical shop- floor counterparts through different protocols without requiring additional programming effort on the agents' side, thus greatly reducing deployment effort.

4.2. Real-time data analysis

The implementation of the RDA module was split into two components, namely a data message queue based on Apache Kafka, and a stream processing network developed in Apache Storm. This development approach builds on the work and guidelines laid out in [26] with the instantiation of the Kafka broker and the implementation of the Storm topology. The representation of the data analysis topology can be seen on Fig. 5.

Upon collecting or generating new data, the CPPS publishes it to Kafka, which acts as a reliable, highly-scalable, high-throughput real time data broker. This broker handles all the communications between the distributed modules of the IDARTS framework, enabling the seamless exchange of data between them. Once data has been published to a Kafka topic, it can be easily consumed by the Storm topology via a Kafka Spout. In Apache Storm, spouts represent data stream sources, typically reading tuples from external sources and emitting them into the topology.

The topology's actual processing is performed by its bolts, each containing its own specific logic. The proposed topology is multi-layered, with each bolt performing a specific task on the incoming tuples, as it can be seen in the pipeline represented in Fig. 5.

First and foremost, tuples are filtered by the Split Stream bolt according to their respective topic of origin, as they can be related to either data or updates to the running system (the latter is further addressed in Subsection 4.3. Tuples are then aggregated in window slots of a given size and emitted to the Data Analysis Bolt (DAB), enabling a sliding window approach.

The DAB is responsible for the actual processing task, using ML models to generate predictions for the shop-floor resources based on the incoming real-time data (e.g. likelihood of failure within a certain number of cycles). These models can vary from resource to resource, and can be provided in Predictive Model Markup Language (PMML) [37] by either their respective agent during deployment, or by the Knowledge Management tool as an update in runtime, being then stored in memory by each of the bolt's workers.

Once the processing is concluded, the results are relayed to the Context Bolt (CB), which enriches them with their significance in the physical world so that the CPPS can later on interpret them and act accordingly. Depending on the adopted approach, this can mean for instance triggering self-reconfiguration, scheduling predictive maintenance or simply alerting an operator by providing said context via a human-machine interface.

Finally, the enriched results are sent to the Logger Bolt to be stored. In the current implementation these are simply logged to the local disk, but should ultimately be stored in a database like MongoDB or Cassandra, hence being available to be used by the Knowledge Management tool to further analyse and improve the runtime process.

4.3. Knowledge management tool

For this initial implementation, the Knowledge Management tool focuses on empowering the RDA with additional flexibility, by providing the means to adapt the way data analysis is performed in runtime, without requiring any stoppages or re-redeployment of the running system. This process occurs in two distinct stages.

Firstly, ML models are implemented in Python using the Scikit-Learn library [38]. The models are created offline, trained using historical data and then serialized into PMML. This constitutes the application-specific stage of the solution, however, the usage of PMML ensures that developers are not restrained to using Python-based models, as long as they can be serialized to the adopted format.

Afterwards, the resulting model description is then loaded by a Java-based application, which also allows the user to input additional model information such as the model ID, the resource or resources it is related to, and its context. This context can be used to capture domain knowledge and enriching model outputs with their meaning in the physical world, allowing the CPPS to autonomously interpret the results of the RDA later on.

All this information is then serialized and pushed to a dedicated Kafka topic to be consumed by the running Storm topology. Once there, the Split Stream bolt takes care of dividing the message into the updates to be processed by the Data Analysis Bolt and the Context Bolt, thus effectively adapting the runtime system without required any additional effort or time.

5. Testing and validation

This section details the steps taken during the testing and validation of the initial implementation described in Section 4. The tests were conducted with the goal of validating the main three non- functional requirements identified in the specification of the framework, namely scalability, flexibility and pluggability.

The testing environment consisted in the four-node cluster shown in Fig. 6, consisting on four machines running Core i7-4770 processors

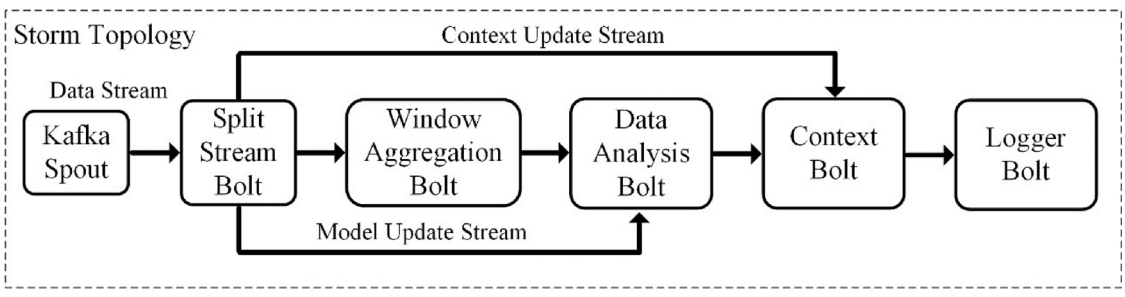

Fig. 5. Real-time Data Analysis Pipeline.

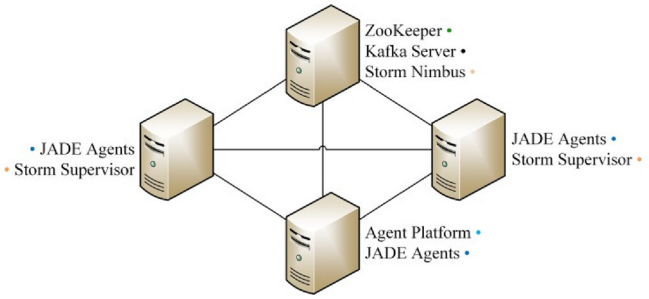

Fig. 6. IDARTS Testing 4-Node Cluster.

with 12GB of memory each.

One machine was dedicated to running Zookeeper (a dependency of both Kafka and Storm), the Kafka server and Storm Nimbus, which is responsible for assigning tasks and monitoring other nodes. Two others act as Storm supervisor nodes. These are the nodes that host and govern several worker processes to complete the tasks previously assigned by the Nimbus node. Finally, the remaining machines were allocated to the hosting of the agents running in each test (evenly divided between each of the three machines). The testing configuration consisted in two main data flows, as illustrated in Fig. 7.

Regarding the CPPS, the agents were instantiated with a dummy hardware communication interface, which for each agent simulated the generation of data from its respective emulated resource. This was achieved by developing a small Java graphical application which enabled the user to plug and unplug virtual resources that, once plugged, generated a new random value between 100 and 200 every 100 ms. Additionally, every time a new value was generated the resource had a three percent chance to enter failure mode, which forced it to generate data of increasingly larger values in small increments for 30 cycles.

The actual shop-floor data flow is depicted with a solid line in Fig. 7. For each resource a CMA was deployed, collecting all its data and publishing it to a Kafka topic (1). This topic was consumed by the Kafka Spout in the Storm topology (2) in order to enable the processing of the emulated shop-floor data. Once this process was concluded, the result was once more published to Kafka (3) and made available to be consumed by the CPPS one more (4).

For the scalability tests, three different deployment configurations were used, each doubling the number of resources/agents deployed in the previous one and running for 30 min. A summary of the data

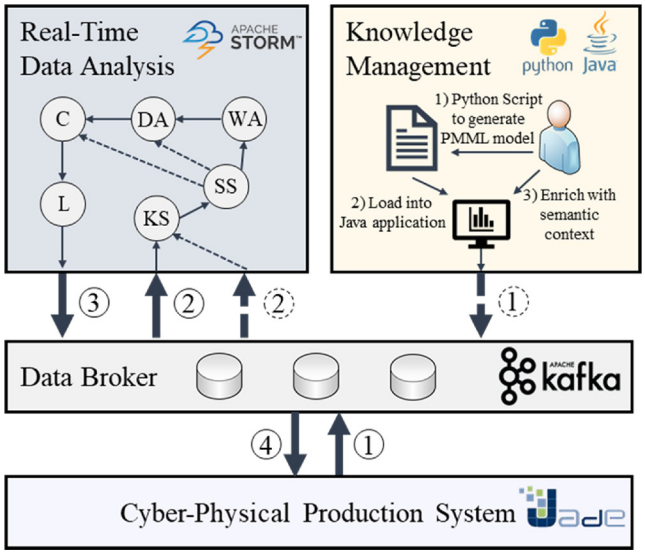

Fig. 7. Data Flow for the Test Case (Solid line: Data, Dashed line: Model Updates).

Table 1
Apache Storm Topology Metrics.

# Agents	Bolt	Capacity	Execution Latency (ms)
75	*Split Stream*	0.026	0.145
	Window Agg.	0.034	0.062
	Data Analysis	0.032	0.059
	Context	0.016	0.023
150	*Split Stream*	0.034	0.146
	Window Agg.	0.055	0.077
	Data Analysis	0.048	0.060
	Context	0.048	0.035
300	*Split Stream*	0.100	0.224
	Window Agg.	0.207	0.123
	Data Analysis	0.189	0.119
	Context	0.181	0.067

collected during these runs can be seen in Table 1.

The metrics observed in Table 1 are capacity and execution latency, extracted using Storm's UI daemon. The former is referent to the processing capacity of the bolts deployed in the Storm topology. The closer the value is to "1.0", the closer the bolt is to be running as fast as it can. This is useful to verify if the parallelism of the topology needs to be adjusted. The latter refers to how long each tuple takes on average (in milliseconds) to be executed by the respective bolt. The Logger Bolt was excluded because during testing it was only quickly logging the model updates, thus its metrics were very close to zero and not relevant for comparison. An additional metric that was extracted was the complete topology latency, meaning the time it takes each tuple, on average, to be fully processed and acknowledged by the entire topology. Its values were 11.452 ms, 13.441 ms and 16.222 ms, for 75, 150 and 300 agents respectively. These results will be discussed in Section 6.

On the other hand, the second data flow, represented by the dashed line, pertains to the Knowledge Management updates used to demonstrate the flexibility of the initial implementation regarding the analysis process. The data generated by the virtual resources was used to train two different ML models beforehand, more specifically K-Means Clustering and Logistic Regression classifiers. An example of the output from the latter can be seen in Fig. 8.

As it can be observed, the model outputs the probability of failure within a given number of cycles. For testing purposes, the training data was labelled based on a 5-cycle period, and it was considered that a probability above 60% would result in a model classification of "1" as a representation of impending resource failure.

Upon deployment, every agent is initialized being associated with the aforementioned model, passing this information to the Storm topology via Kafka. During runtime, this configuration was then changed by the Knowledge Management tool, individually shifting certain resources to the clustering model instead. As shown in Fig. 7, this process is initiated once the user has finished introducing all the information pertaining to the new model in the Java application. The serialized update is published to a Kafka topic (1), being then consumed by the Kafka Spout and emitted into the Storm topology (2). Internally, it is then split into the model and context updates, and sent to the respective bolts. The latency associated with these updates was measured over 100 iterations, consisting in the average timespan in milliseconds between the user publishing the update to Kafka, and each bolt completely updating its internal execution process accordingly.

Finally, the pluggability was tested using the deployment tool mentioned in the beginning of this section. For this purpose, timestamps were extracted from two specific moments for both the plugging and unplugging of resources/agents. Once when pressing the button to launch/remove a

new resource, mimicking the detection of a new/removed shop-floor asset by the Deployment Agent, and then again when the respective agent is deployed and ready to start publishing data, or when it

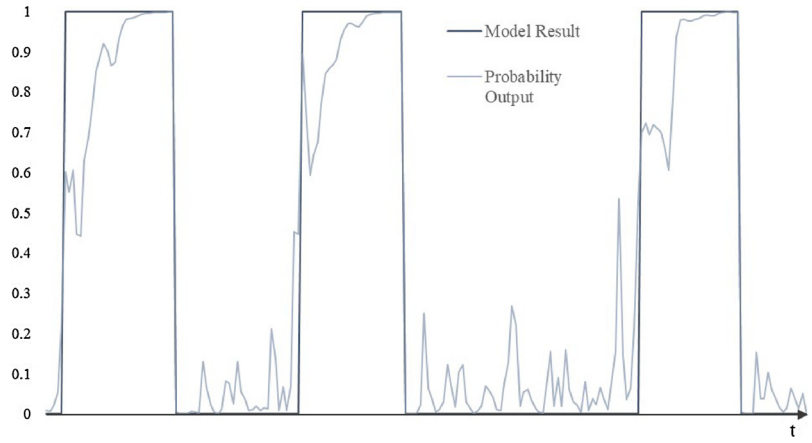

Fig. 8. Example of the output from the Logistic Regression Model.

Table 2
CPPS Pluggability Test Results.

# Agents	Plug Latency (ms)	Unplug Latency (ms)
75	248.889	3.556
150	252.000	5.050
300	249.560	11.300

is removed from the agent platform. The results are summarized in Table 2, representing the average latency measured when unplugging and plugging randomly chosen agents 100 times during execution, one at a time, with arbitrary intervals in between each action and with a varying number of active agents for each run.

6. Discussion of results

The tests described in Section 5 aimed to verify the three main non-functional requirements described in the IDARTS specification, namely scalability, flexibility and pluggability.

Regarding the first set of tests pertaining to the scalability of the solution, the implementation was deployed on a four-node cluster and tested with varying data throughput rates. This rate started at around 750, then 1500, and finally 3000 data points per second, corresponding to 75, 150 and 300 virtual resources/agents, respectively. The comparison of the results for the RDA module can be seen in Fig. 9.

As it can be observed, even with 300 concurrent resources/agents each tuple was spending slightly less than 0.2 ms at most per bolt. The average capacity of each bolt increases significantly for the last set of the test, but is still considerably less than "1.0" meaning that the bolts are not being forced to run faster than they can, which would result in unwanted queueing and delays. Regardless, once the capacity reaches higher values as the system scales further, this issue can be tackled by increasing the parallelism of the topology.

Regarding the knowledge management flow, the latency associated with each update to the running system was also measured under different throughput rates. The goal was to verify the impact that increasing volumes of data could have in the system's capacity to adapt the RDA process.

The results for the average update latency can be visualized in Fig. 10, along with a comparison with the overall complete latency for the tuples pertaining to the shop-floor data stream.

The latency associated with the knowledge management updates is shown to increase as the system scales, albeit very slightly. These results show that not only is the Knowledge Management tool capable of adapting the RDA during execution, but also that the system's scale has very little impact in the performance of this updates. As far as the complete latency for the data stream is concerned, the overall impact of the system's scale is also relatively small in comparison. This is considering that while the scale doubles for each run, the latency only increases by a factor of around 1.174 and 1.217 between each of them. This can be attributed not only to the increase in the processing time, but also in the increased network load and respective associated delays.

Lastly, for the pluggability tests the latency was measured between the moment a new/removed resource is detected and the instant the agent is fully initialized/removed from the platform. The comparison of the results is illustrated in Fig. 11.

It is interesting to note the difference between the results for unplugging and plugging events. At first glance, it might seem that the results suggest the agent platform simply takes a lot longer to handle the deployment of new agents. However, this difference lies mainly in the monitoring agent's ramp-up time, since the measurement is taken only after it has fully completed its setup process, which includes initializing its data collection and output communication libraries.

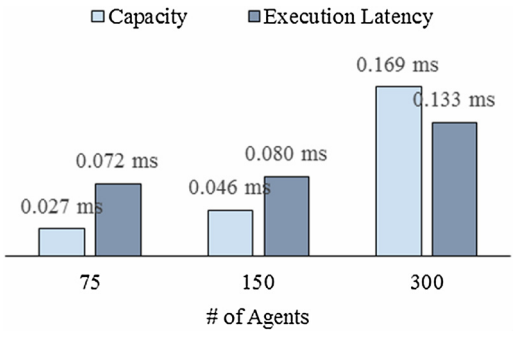

Fig. 9. Apache Storm Topology's Average Capacity and Execution Latency.

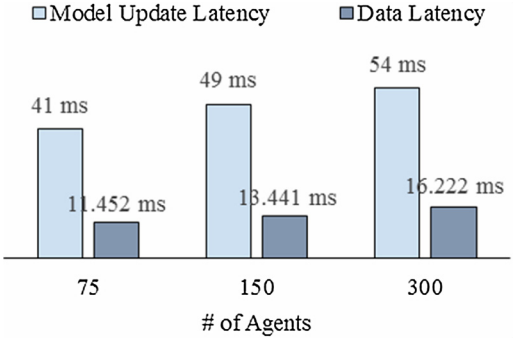

Fig. 10. Shop-floor Data and Model Update Latency.

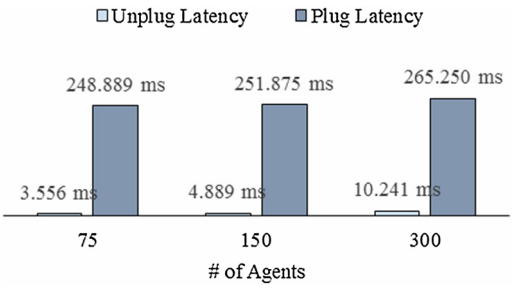

Fig. 11. Pluggability Latency.

7. Conclusion and future work

A generic framework was proposed, focused on the aspects of data analysis and real-time supervision of manufacturing systems. Being aligned with the Industry 4.0 vision, the framework aims to take advantage of the ongoing data explosion, presenting a scalable and flexible solution for predictive manufacturing, being as little invasive as possible. Its efficacy is however dependent on the availability, volume and quality of the data from the underlying production system.

The main contributions of the framework are thus:

- Capacity to support a plug-and-produce paradigm in the context of predictive manufacturing through dynamic system virtualization via a MAS-based CPPS, coping with changes and disturbances at the shop-floor;
- Integration of real-time and historical data at both the component and system levels, enabling the adaptation of the real-time analysis and rule-based supervision algorithms after deployment;
- Support for context-aware self-adjustment coupled with human-machine interaction, allowing the system to either adjust its operation parameters through self-reconfiguration, or suggest corrective actions to an operator in order to return to normal production conditions and product quality.

The initial implementation of the IDARTS framework focused on the integration of the CPPS data acquisition, the RDA module, as well as a prototype knowledge management application to adapt the runtime analysis during execution. The solution was deployed in a four-node cluster and tested in order to validate three main non-functional requirements, namely scalability, flexibility and pluggability.

From these tests it was possible to observe that the solution was scalable and capable of adapting to the changes in production, be it in terms of the shop-floor layout or capturing domain knowledge to adapt the data analysis being performed in runtime, without requiring additional programming effort, stoppages or re-deployment.

Finally, future work shall also include the implementation and integration of the CPPS' runtime decision module, as well as the updates to the decision-making rules that govern it.

References

[1] M. Brettel, N. Friederichsen, M. Keller, M. Rosenberg, How virtualization, decentralization and network building change the manufacturing landscape: an industry 4.0 perspective, Int. J. Inf. Commun. Eng. 8 (1) (2014) 37–44.
[2] Deloitte, Industry 4.0–challenges and Solutions for the Digital Transformation and Use of Exponential Technologies, (2014).
[3] A. Gilchrist, Introducing industry 4.0, Industry 4.0, Apress, 2016, pp. 195–215.
[4] H. Kagermann, J. Helbig, A. Hellinger, W. Wahlster, Recommendations for implementing the strategic initiative INDUSTRIE 4.0: securing the future of German manufacturing industry; Final report of the industrie 4.0 working group, Forschungsunion, Technical Report, (2013).
[5] I.I. Consortium, Industrial internet reference architecture, Industrial Internet Consortium, Technical Report, (2015).
[6] T. Stock, G. Seliger, Opportunities of sustainable manufacturing in industry 4.0, Procedia CIRP 40 (January) (2016) 536–541.
[7] F. Shrouf, J. Ordieres, G. Miragliotta, Smart factories in industry 4.0: a review of the concept and of energy management approached in production based on the internet of things paradigm, 2014 IEEE International Conference on Industrial Engineering and Engineering Management, (2014), pp. 697–701.
[8] D. Zuehlke, SmartFactory—towards a factory-of-things, Annu. Rev. Control 34 (April (1)) (2010) 129–138.
[9] R.K. Mobley, An Introduction to Predictive Maintenance, 2nd editio, Elsevier, 2002.
[10] J. Lee, E. Lapira, B. Bagheri, H. Kao, Recent advances and trends in predictive manufacturing systems in big data environment, Manuf. Lett. 1 (1) (2013) 38–41.
[11] L. Ribeiro, Cyber-physical production systems' design challenges, 2017 IEEE 26th International Symposium on Industrial Electronics (ISIE), (2017), pp. 1189–1194.
[12] J. Lee, B. Bagheri, H.A. Kao, A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems, Manuf. Lett. 3 (December) (2015) 18–23.
[13] R.F. Babiceanu, R. Seker, Big Data and virtualization for manufacturing cyber-physical systems: a survey of the current status and future outlook, Comput. Ind. 81 (2016) 128–137.
[14] A. Patcha, J.M. Park, An overview of anomaly detection techniques: existing solutions and latest technological trends,", Comput. Networks 51 (12) (2007) 3448–3470.
[15] A. Marvuglia, A. Messineo, Monitoring of wind farms' power curves using machine learning techniques, Appl. Energy 98 (October (Supplement C)) (2012) 574–583.
[16] J. Lee, B. Bagheri, H.-A. Kao, Recent advances and trends of cyber-physical systems and Big data analytics in industrial informatics, Int. Conf. Ind. Informatics 2014, November 2015, 2014.
[17] L.C. Siafara, H.A. Kholerdi, A. Bratukhin, N. Taherinejad, A. Wendt, SAMBA : a self-aware health monitoring architecture for distributed industrial systems, Industrial Electronics Society, IECON 2017-43rd Annual Conference of the IEEE, (2017), pp. 3512–3517.
[18] A. Canito, M. Fernandes, I. Pra, An architecture for proactive maintenance in the machinery industry, 8th International Symposium on Ambient Intelligence (ISAml 2017) 615 (2017) ISAmI.
[19] D. Lechevalier, A. Narayanan, S. Rachuri, Towards a domain-specific framework for predictive analytics in manufacturing, Proc. - 2014 IEEE Int. Conf. Big Data, IEEE Big Data 2014, February 2015, 2015, pp. 987–995.
[20] J. Li, J. Shi, Knowledge discovery from observational data for process control using causal Bayesian networks, IIE Trans. 39 (6) (2007) 681–690.
[21] Y.K. Penya, P.G. Bringas, A. Zabala, Advanced fault prediction in high-precision foundry production, IEEE Int. Conf. Ind. Informatics, (2008), pp. 1672–1677.
[22] A. Chan, K.R. McNaught, Using Bayesian networks to improve fault diagnosis during manufacturing tests of mobile telephone infrastructure, J. Oper. Res. Soc. 59 (4) (2008) 423–430.
[23] K. Wang, Intelligent Predictive Maintenance (IPdM) system – Industry 4.0 scenario, WIT Trans. Eng. Sci. 113 (2016) 259–268.
[24] L.S. Terrissa, S. Meraghni, Z. Bouzidi, N. Zerhouni, A New approach of PHM as a service in Cloud computing, 2016 4th IEEE Int. Colloq. Inf. Sci. Technol. (CiSt). Proc. (2016), pp. 610–614.
[25] S.-J. Shin, J. Woo, S. Rachuri, Predictive analytics model for power consumption in manufacturing, Procedia CIRP 15 (2014) 153–158.
[26] R.S. Peres, A. Rocha, A. Coelho, J. Barata, A highly flexible, distributed data analysis framework for industry 4.0 manufacturing systems, in: T. Borangiu, D. Trentesaux, A. Thomas, P. Leitão, J. Barata (Eds.), Service Orientation in Holonic and Multi-Agent Manufacturing. SOHOMA 2017, March, Springer, Cham, 2017, pp. 373–381.
[27] Z. Ge, Distributed predictive modeling framework for prediction and diagnosis of key performance index in plant-wide processes, J. Process Control 65 (2018) 107–117.
[28] D. Wu, et al., A fog computing-based framework for process monitoring and prognosis in cyber-manufacturing, Int. J. Ind. Manuf. Syst. Eng. 43 (2017) 25–34.
[29] R.Y. Zhong, C. Xu, C. Chen, G.Q. Huang, Big data analytics for physical internet-based intelligent manufacturing shop floors, Int. J. Prod. Res. 55 (9) (2017) 2610–2621.
[30] J. Lee, H.A. Kao, S. Yang, Service innovation and smart analytics for Industry 4.0 and big data environment, Procedia CIRP 16 (2014) 3–8.
[31] O. Kanjanatarakul, T. Denœux, S. Sriboonchitta, Prediction of future observations using belief functions: a likelihood-based approach, Int. J. Approx. Reason. 72 (2016) 71–94.
[32] A. Gandomi, M. Haider, Beyond the hype: big data concepts, methods, and analytics, Int. J. Inf. Manage. 35 (2) (2015) 137–144.
[33] F. Bellifemine, G. Caire, D. Greenwood, Developing Multi-agent Systems With JADE, (2007).
[34] A.D. Rocha, R. Peres, J. Barata, An agent based monitoring architecture for plug and produce based manufacturing systems, Industrial Informatics (INDIN), 2015 IEEE 13th International Conference, (2015), pp. 1318–1323.
[35] A.D. Rocha, R.S. Peres, L. Flores, J. Barata, A multiagent based knowledge extraction framework to support plug and produce capabilities in manufacturing monitoring systems, ISMA 2015 - 10th Int. Symp. Mechatronics Its Appl. (2016).
[36] FIPA, FIPA Request Interaction Protocol Specification, [Online] Available (2002) http://www.fipa.org/specs/fipa00026/SC00026H.pdf.
[37] A. Guazzelli, M. Zeller, W. Lin, G. Williams, PMML: an open standard for sharing models, R J. 1 (1) (2009) 60–65.
[38] F. Pedregosa, et al., Scikit-learn: Machine Learning in Python, J. Mach. Learn. Res. 12 (2012) 2825–2830.