A Highly Flexible, Distributed Data Analysis Framework for Industry 4.0 Manufacturing Systems

Ricardo Silva Peres, Andre Dionisio Rocha, Andre Coelho, and Jose Barata

CTS - UNINOVA, Faculdade de Ciencias e Tecnologia, Universidade Nova de Lisboa, 2829-516 Caparica, Portugal
{ricardo.peres,andre.rocha,jab}@uninova.pt,
am.coelho@campus.fct.unl.pt

Abstract. In modern manufacturing, high volumes of data are constantly being generated by the manufacturing processes. However, only a small percentage is actually used in a meaningful way.
As part of the H2020 PERFoRM project, which follows the Industry 4.0 vision and targets the seamless reconfiguration of robots and machinery, this paper proposes a framework for the implementation of a highly flexible, pluggable and distributed data acquisition and analysis system, which can be used for both supporting run-time decision making and triggering self-adjustment methods, allowing corrections to be made before failures actually occur, therefore reducing the impact of such events in production.

Keywords: Data Analysis, Industry 4.0, Manufacturing, Cyber-Physical Systems

1 Introduction

1.1 Interconnected Manufacturing Systems - Industry 4.0

Nowadays, the world is facing a huge and disruptive paradigm shift based on the IT developments of the last few decades. Several concepts such as Cloud Computing, Internet of Things, Big Data, etc. are creating one of the most challenging and drastic changes in the manufacturing world. The Industry 4.0 [1, 2] concept emerged as an idea for the perfect production environment based on the modern interconnected world.

More than the connectivity among the shop floor components and the diverse departments of each company, in Industry 4.0 all of the value chain activities as well as the entire product life cycle intervenients are connected and are capable of producing and sharing information among themselves [3].

Aligned with this vision, the PERFoRM project targets the conceptual transformation of existing industrial production systems towards the plug and produce paradigm, achieving flexible manufacturing environments based on full interoperability and seamless reconfiguration of harmonized machinery and robots as a response to operational or business variations.

Currently, a large amount of manufacturing information is already generated, however most of it is just stored and not processed in order to extract useful knowledge. Hence, with the recent advent of Data Mining it is possible to analyze this data and somehow generate relevant information for the production environment [4, 5].

As part of the PERFoRM project, this paper proposes a framework for the implementation of a data acquisition and processing system, capable of coping with the requirements imposed by the Industry 4.0 vision. The remainder of this paper is organized as follows. Subsection 1.2 provides an overview of related work on the topic of data analysis in manufacturing. Afterwards, Section 2 describes the proposed architecture, followed by Section 3 which provides the guidelines for a possible implementation of said architecture. Finally, some closing remarks and details regarding future work are provided in Section 4.

1.2 Data Analysis in Manufacturing

Data Analysis can be used to obtain meaningful knowledge or information from raw data, applying algorithms in order to achieve it. Typically, the concept of data mining is often discussed and used due to this purpose.

In a manufacturing context, several approaches and algorithms are used to gather knowledge. [6] reports about the development of an environment for providing knowledge based on search, evaluation and generalization, which addresses a clear challenge of suggesting good operating strategies for specific factory conditions at the proper time. Neural Networks (NN) and Generic Algorithms (GA) were used to identify the data structure, whereas the data extracted was in form of performance obtained. Finally, this type of knowledge can be used to increase the system accuracy. IGem was developed by [7] and its an artificial intelligent tool that uses fuzzy logic, GA algorithms and rule base knowledge which is applied to the diamond industry. The results presented demonstrated the reduction of processing time by 25% when analyzing data. In maintenance, EXPERT-MM was developed to work with historical failure data in order to suggest preventive maintenance schedules [8].

Regarding knowledge extraction, algorithms are applied based on what kind of knowledge it is aimed to retrieve. [9] makes an extensive review on data mining functions and applicability, which aims to identify several kinds of knowledge to be mined such as job shop scheduling, quality control, fault diagnostics, manufacturing process, maintenance, defect analysis, manufacturing systems, condition based monitoring, supply chain and others. For most of them, one could apply decision tree algorithms, regression functions, fuzzy clustering techniques, entropy based analysis methods, neural networks, generic algorithms, sampling methods, among others.

One interesting point to rethink is suggested by [10] which refers to how data is being organized during processing. As he clearly states, the processing model and the data model are often disjoint areas resulting in distributed computing systems to focus only on ordering tasks instead of including abstract models for

data processing. This type of models act as a schema for data processing which logically optimize the process.

The previously discussed techniques can be described as an aggregation of complex algorithms and methods that have been object of studies and improvements. However, [11] presents a different solution for the problem of data extraction and processing. In this case one specific type of raw data (data related to transitions) is processed to assert if a device needs maintenance based on the computation of moving averages and trends of key values. These types of methods are simpler but, nonetheless, provide recognition of abnormal behaviors or irreversible problems with the resources, which is a valuable asset for manufacturers. The example used was a clamp opening or closing time, providing real time information of important resources within a manufacturing process.

2 Layered Data Analysis Architecture

In this section an overview of the proposed real-time data analysis architecture is provided, along with its main goals, requirements and a description of its individual structural elements.

Being a critical part of the PERFoRM ecosystem, the proposed solution is responsible for not only performing the context-aware data analysis, thus generating predictive data that can be used to trigger the system's self-adjustment mechanisms (e.g. reconfiguration), but also for the acquisition of the data itself at both the manufacturing cell and component levels.

Additionally, a given number of requirements are imposed on the architecture's design. First and foremost, in line with PERFoRM's vision the architecture 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, thus facilitating its industrial integration and adoption. Moreover, it needs to be capable of adapting to changes to the process or its components in run-time, for instance in terms of both pluggability and changes to the Key Performance Indicators (KPI) to be analyzed. Furthermore, data and context representation should follow PERFoRM's common data model in order to enable the seamless interoperability and data exchange between the data analysis architecture and the remaining PERFoRM system elements and tools.

Another point to take into account is the aspect of scalability. In order to ensure that the approach is applicable to a varied number of different use cases, it needs to be capable of scaling according to each use case requirements. However, as a system scales its complexity tends to increase to higher levels as a consequence. Thus, in order to tackle this challenge, a layered architectural structure is proposed. An overview of this approach can be seen in Figure 1.

As depicted, the proposed architecture is divided into several layers in order to decrease the overall complexity, each operating according to a specific purpose on top of the shop floor, which stands as the base layer. Each of the subsequent layers is described in further detail in the remaining subsections of Section 2.

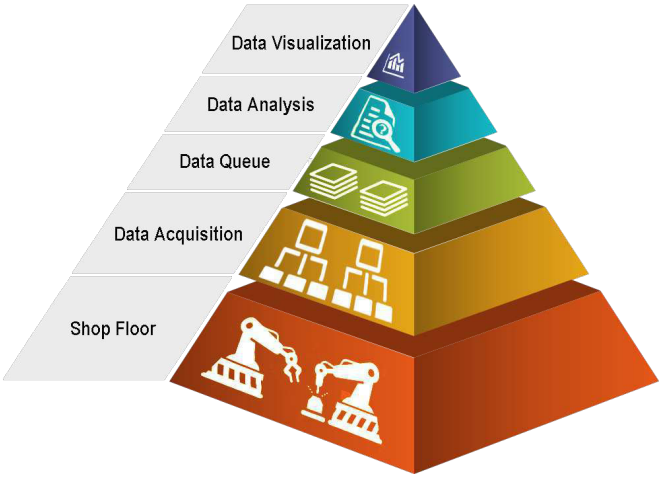

Fig. 1. Architecture Layered View.

2.1 Data Acquisition Layer (DAL)

Standing directly above the shop floor layer, the DAL is responsible not only for the acquisition of relevant data but also by its pre-processing in terms of the extraction of context-aware information.

In regards to the data acquisition, the DAL needs to be flexible in order to adapt to changes coming directly from its sources in the shop floor, be it in terms of new components being plugged or unplugged, or even changes to the KPIs that need to be collected and analyzed. Also, the communication with the shop floor needs to be specified in a generic way, thus allowing the consideration of different requirements from different potential use case. For instance, a specific case might present time constraints in the order of weeks or days, while a different one might require data to be collected and analyzed in near real-time, therefore requiring different approaches.

To this end, the DAL follows an approach similar to that presented in another successful European project, FP7 PRIME [12, 13, 11], in which a Cyber-Physical System (CPS) based approached was used. This approach is centered on a Multiagent System (MAS) architecture which abstracts both components and subsystems (e.g. cells, workstations) alike. The adoption of MAS technology confers additional flexibility and robustness to the DAL, allowing it to quickly adapt to changes in the shopfloor.

No less important is the existence of generic communication interfaces which allow the agents to interact with the environment in a "black-box" fashion, regardless of the underlying technology or communication standard. In PERFoRM's case, this means that the approach can be implemented in a way that the agents can communicate with the hardware via the harmonization middleware, or if required (e.g. specific time constraints), a different instantiation of these interfaces would allow an approach closer to edge computing. Upon col-

lecting the raw data, the agents can pre-process it in order to extract more meaningful information before passing it on to the upper layers, in this case the Data Queue Layer (DQL), which is described in further detail in Subsection 2.2.

2.2 Data Queue Layer (DQL)

The DQL's main purpose is to serve as a distributed continuous buffer for the data coming from the DAL. It should add another layer of robustness, allowing for high-volume streams of data to be transported from the DAL in order to be consumed by the data analysis network. As such, it should provide reliability in terms of message delivery, which can be achieved through the sequencing and replication of data messages.

More than a simple message queue, the DQL should be capable of not only handling a high throughput of data (in order for it to cope with the aforementioned varied time constraints), but also to enrich and filter or aggregate the buffered data as required in order to facilitate its consumption by the Data Processing Layer (DPL).

2.3 Data Processing Layer (DPL)

The last core layer is the DPL, responsible for the actual data analysis of the inputs coming from the lower layers. In the context of PERFoRM, this analysis is meant to generate predictive data related to the KPIs relevant for each use case, producing forecasts and identifying trends and correlations between these indicators.

As such, this layer enables the early detection of possible disturbances, degradation or KPI deviation from the expected boundaries in the shop floor. Hence, due to this capacity for predictive analysis, the DPL is 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. Additionally, the DPL is not limited to assisting in run-time decision making (e.g. by interfacing with external data visualization tools, which are however outside the scope of this work), being also capable of triggering self-adjustment methods (e.g. self-reconfiguration) which can promptly perform corrections in order to return the system to a state of normal operation.

3 Data Analysis Framework

This section aims at providing the guidelines for a possible implementation of the architecture described in Section 2. Each of the layers and related technologies is addressed in the coming subsections, namely the MAS-based CPS (DAL), the Apache Kafka data queue (DQL) and the Apache Storm stream processing network (DPL), detailed in Subsections 3.1 and 3.2, respectively.

3.1 MAS-based Cyber-Physical System in the DAL

Following the example set in [11, 13], the DAL's CPS can be implemented following a similar pluggable MAS-based approach. The Java Agent DEvelopment framework (JADE) [14] is indicated as it provides a robust infrastructure supporting the agent's core behavioral logic and communication, as well a wide array of auxiliary tools to further facilitate the development process. A representation of the comprised agents, as well as their respective interactions is shown in Figure 2.

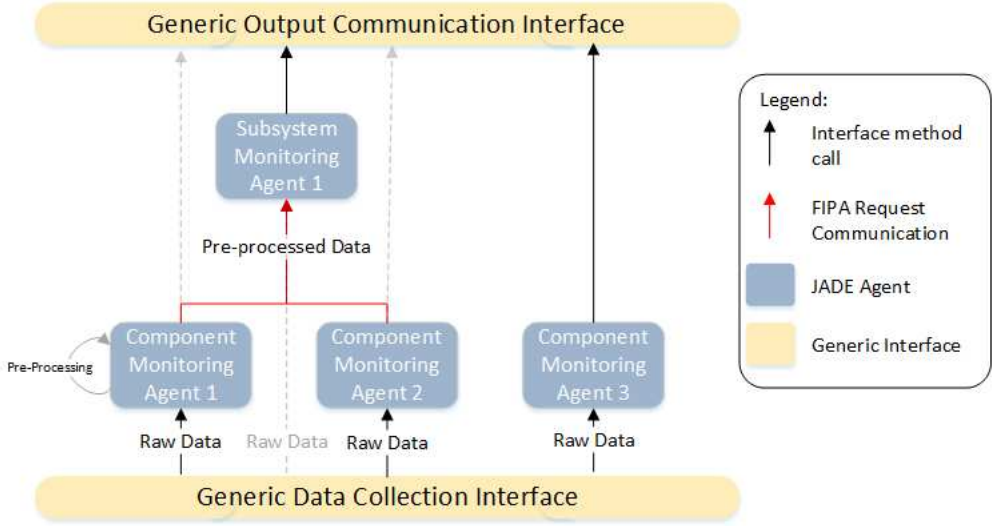

Fig. 2. JADE Data Acquisition and Pre-Processing MAS Overview.

The DAL implementation consists in a MAS network comprising two main different agent types. The Component Monitoring Agent (CMA) is responsible for abstracting individual components (e.g. clamp, robot) in the shop floor, collecting relevant raw data and pre-processing it according to a given set of rules. The data acquisition is performed through a generic Data Collection Interface (DCI), thus allowing this communication to be executed both through PERFoRM's middleware, or directly through the implementation of required communication protocols (e.g. instantiating an OPC UA connector), depending only on the DCI implementation.

Each CMA can then be associated to a Subsystem Monitoring Agent (SMA), to which it passes on its extracted and pre-process data, enabling the extraction of more complex information at a higher abstraction level. Hence, the SMA performs similarly to the CMA, being however responsible for abstracting a subsystem (e.g robotic cell, workstation), which can in turn be associated to

other SMAs. The agents' interactions are FIPA compliant, following the FIPA Request protocol [15].

Both agent types relay their data to the upper layers through a generic Output Communication Interface (OCI), allowing the CPS to be independent from the technology used to implement the remaining layers.

3.2 Data Message Queue and Processing Network

Any implementation of the DQL needs to take into consideration the requirements specified in 2.2, more specifically in terms of scalability, capacity to handle high volumes of data, low latency and reliability.

With this in mind, Apache Kafka [16, 17] is proposed as a possible framework to implement such a data queue. Kafka is a fault-tolerant, highly scalable, distributed messaging system. In essence, Kafka functions with a publish-subscribe approach, allowing *producers* (data sources, in this case the agents in the DAL) to publish data messages which are maintained in categories called *topics*. These can be subscribed by *consumers* (represented by the DPL nodes), being divided into ordered partitions supporting message persistence and replication. Kafka's message management is optimized for low latency and high throughput, with documented uses for even real-time applications [18].

Finally, for the DPL Apache Storm is considered, being a distributed stream processing system which easily integrates with both databases and queuing technologies (such as the aforementioned Apache Kafka). Storm's processing runs in *topologies*, which are essentially series of nodes each containing certain processing logic, with the associated links specifying the data flow. The framework integrating each of these technologies can be seen in Figure 3.

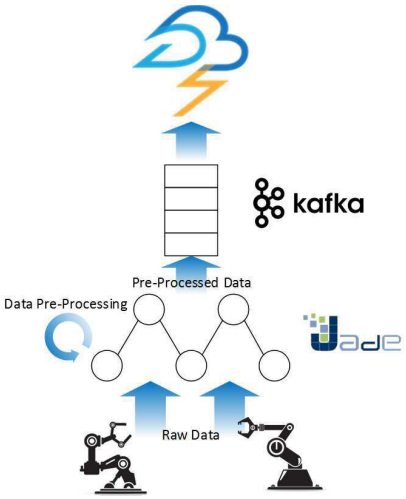

Fig. 3. Framework Overview.

With the whole framework integrated, data is collected by the CPS via the DCI, being pre-processed at both the component and subsystems levels. Afterwards, each agent publishes its respective data to the Kafka queue, where it is aggregated in topics according to its specific category (e.g. component origin or data group). Finally, data is continuously consumed by the Storm topology, which can in turn compute the trends and correlations necessary in order to generate meaningful predictive data which can be used for run-time decision making support or as a trigger for self-adjustment methods.

4 Conclusions and Future Work

This paper presents the framework of a highly flexible and distributed system for the acquisition and analysis of manufacturing data. A layered architecture is proposed, which is capable of coping with changes and disturbances in the shop floor, as well as with changing requirements in terms of monitored KPIs or time constraints.

Furthermore, guidelines regarding the implementation details and the related technologies were provided, illustrating a possible implementation of the proposed architecture.

As part of the H2020 PERFoRM project, future efforts will focus on the implementation of each of the proposed architecture's layers according to the described framework. This will be followed by the instantiation of the system in the project's different use cases, allowing for it to be validated across different application domains each with varying time constraints and requirements.

Acknowledgments .

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 680435.

References

1. Alasdair Gilchrist. Introducing industry 4.0. In *Industry 4.0*, pages 195–215. Springer, 2016.
2. Rainer Drath and Alexander Horch. Industrie 4.0: Hit or hype?[industry forum]. *IEEE industrial electronics magazine*, 8(2):56–58, 2014.
3. T Stock and G Seliger. Opportunities of sustainable manufacturing in industry 4.0. *Procedia CIRP*, 40:536–541, 2016.
4. Jay Lee, Hung-An Kao, and Shanhu Yang. Service innovation and smart analytics for industry 4.0 and big data environment. *Procedia CIRP*, 16:3–8, 2014.
5. Shiyong Wang, Jiafu Wan, Daqiang Zhang, Di Li, and Chunhua Zhang. Towards smart factory for industry 4.0: a self-organized multi-agent system with big data based feedback and coordination. *Computer Networks*, 101:158–168, 2016.
6. Kevin R Caskey. A manufacturing problem solving environment combining evaluation, search, and generalisation methods. *Computers in Industry*, 44(2):175–187, 2001.

7. Tony Holden and Matee Serearuno. A hybrid artificial intelligence approach for improving yield in precious stone manufacturing. *Journal of Intelligent manufacturing*, 16(1):21–38, 2005.
8. Dentcho Batanov, Nagen Nagarur, and Prapan Nitikhunkasem. Expert-mm: A knowledge-based system for maintenance management. *Artificial intelligence in engineering*, 8(4):283–291, 1993.
9. Alok Kumar Choudhary, Jenny A Harding, and Manoj Kumar Tiwari. Data mining in manufacturing: a review based on the kind of knowledge. *Journal of Intelligent Manufacturing*, 20(5):501–521, 2009.
10. Reginald Cushing, Adam Belloum, Marian Bubak, and Cees de Laat. Towards a data processing plane: An automata-based distributed dynamic data processing model. *Future Generation Computer Systems*, 59:21–32, 2016.
11. Andre Dionisio Rocha, Ricardo Peres, and Jose Barata. An agent based monitoring architecture for plug and produce based manufacturing systems. In *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, pages 1318–1323. IEEE, 2015.
12. André Dionísio Rocha, Diogo Barata, Giovanni Di Orio, Tiago Santos, and José Barata. Prime as a generic agent based framework to support pluggability and reconfigurability using different technologies. In *Doctoral Conference on Computing, Electrical and Industrial Systems*, pages 101–110. Springer International Publishing, 2015.
13. André Dionísio Rocha, Ricardo Silva Peres, Luis Flores, and Jose Barata. A multiagent based knowledge extraction framework to support plug and produce capabilities in manufacturing monitoring systems. In *Mechatronics and its Applications (ISMA), 2015 10th International Symposium on*, pages 1–5. IEEE, 2015.
14. Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. *Developing Multi-agent Systems with JADE*, pages 89–103. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.
15. Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. Developing multi-agent systems with a fipa-compliant agent framework. *Software-Practice and Experience*, 31(2):103–128, 2001.
16. Jay Kreps, Neha Narkhede, Jun Rao, et al. Kafka: A distributed messaging system for log processing. In *Proceedings of the NetDB*, pages 1–7, 2011.
17. Guozhang Wang, Joel Koshy, Sriram Subramanian, Kartik Paramasivam, Mammad Zadeh, Neha Narkhede, Jun Rao, Jay Kreps, and Joe Stein. Building a replicated logging system with apache kafka. *Proceedings of the VLDB Endowment*, 8(12):1654–1655, 2015.
18. Ken Goodhope, Joel Koshy, Jay Kreps, Neha Narkhede, Richard Park, Jun Rao, and Victor Yang Ye. Building linkedin's real-time activity data pipeline. *IEEE Data Eng. Bull.*, 35(2):33–45, 2012.