Real-Time Rule-Based Monitoring Tool to Achieve Zero Defect Manufacturing

Julio Costa, Alexandre O. Júnior, Jose Barbosa, Gleifer Alves, Andre P. Borges, Gisela Garcia, Rui Pires, and Paulo Leitao

Abstract The demands of innovative production systems are shifting from mass production to the creation of smaller quantities with a focus on high quality. To achieve these evolving demands, Zero Defect Manufacturing has emerged as a key paradigm. This approach requires an innovative architectural monitoring tool where real-time data is continuously gathered and analysed to predict defects and assess their potential impacts. It also necessitates the seamless integration of diverse data sources, advanced processing algorithms, and Digital Twins to align with industrial requirements. In this paper we present a real-time, rule-based monitoring tool applied to a real-world car manufacturing use case. The tool successfully generated early alerts for quality deviations, enabling production engineers to shift from a reactive to a proactive approach by detecting potential quality issues early in the process.

Keywords Digital twin · Rule-based system · Monitoring · ZDM

J. Costa (✉) · A. O. Júnior · J. Barbosa · P. Leitao
Research Centre in Digitalization and Intelligent Robotics (CeDRI), Laboratório Associado para a Sustentabilidade e Tecnologia em Regiões de Montanha (SusTEC), Instituto Politécnico de Bragança, Bragança, Portugal
e-mail: juliocosta@ipb.pt

A. O. Júnior
e-mail: alexandrejunior@ipb.pt

J. Barbosa
e-mail: jbarbosa@ipb.pt

P. Leitao
e-mail: pleitao@ipb.pt

G. Alves · A. P. Borges
Universidade Tecnológica Federal do Paraná (UTFPR), Paraná, Ponta Grossa, Brazil
e-mail: gleifer@utfpr.edu.br

G. Garcia · R. Pires
Volkswagen AutoEuropa, Quinta do Anjo, Portugal
e-mail: apborges@utfpr.edu.br

T. Borangiu et al. (eds.), *Service Oriented, Holonic and Multi-agent Manufacturing Systems for Industry of the Future*, Studies in Computational Intelligence 1197,
https://doi.org/10.1007/978-3-031-85316-6_20

1 Introduction

Industrial production systems are continuously evolving, facing ongoing pressures to increase output while becoming more personalized and maintaining the highest quality [1]. To properly respond to this demand, industrial systems have shifted from centralized to distributed architectures, aiming to overcome the limitations of centralized systems, where data collection follows a strict bottom-up flow and control is top-down. In contrast, a distributed systems approach uses a flattened architecture, where information flow doesn't adhere to a hierarchical structure, as modules and components generate and consume information in a distributed manner from both data collection and control perspectives.

This architectural evolution introduces several challenges and opportunities, mainly due to the high availability of data. Particularly, an opportunity is to develop real-time data analytics that converts data into knowledge, generating insights for potential continuous improvements in production systems. This is particularly important to implement Zero Defect Manufacturing (ZDM) strategies [2], where the objectives include reducing production costs and adding value to the product by reducing rework. In this context, the openZDM (Open platform for realizing zero defects in cyber-physical manufacturing) project (https://www.openzdm.eu/) is proposing a micro-service architecture [3], following the state-of-the-art and industrial approaches, particularly by exploiting Asset Administration Shells (AAS) for data management and Digital Twins for data analytic to process collected data in real-time focusing on the zero defect production paradigm.

Having this in mind, the paper describes the development of a real-time rule-based monitoring tool, aligned with the Reference Architectural Model Industrie 4.0 (RAMI4.0) [4], to implement ZDM strategies, namely identifying early anomalies and patterns in the collected data. This tool assumes a critical step in day-to-day production and engineering operations since it allows the early detection of quality deviations, generating real-time alerts that enable proactive actions. The developed data analytics tool was tested and validated in an industrial automotive assembly line located in Portugal. A server based on AAS was implemented for real-time data collection and serialization with the rule-based monitoring tool through a micro-services architecture to ensure data interoperability between systems belonging to the assembly line. The achieved experimental results showed the developed solution's modularity, robustness, security, flexibility and scalability.

The rest of the paper is organized as follows: Sect. 2 briefly analyses the related work, namely the existing approaches to perform real-time monitoring. Section 3 presents the real-time monitoring system architecture, and Sect. 4 describes the experimental validation of the proposed approach, namely the description of the case study, the implementation of the real-time monitoring tool and the analysis of the achieved results. Finally, Sect. 5 rounds up the paper with conclusions and points out future work.

2 Related Work

Monitoring tools usually require collecting vast amounts of data from different data sources, and Internet of Things (IoT) and Machine-to-Machine technologies can be used for this purpose. The use of advanced data analytics, and particularly machine learning (ML) techniques allows performing real-time monitoring aiming to detect deviations and trends of the parameters's evolution over time. In particular numerous studies have explored the application of rule-based systems for real-time variable monitoring in manufacturing, highlighting its scientific and commercial importance. For instance, Yang et al. [5] conducted a longitudinal study that introduced a real-time rule-based engine integrated specifically for Key Performance Indicators (KPIs) for managerial insights. Conversely, Gunasegaram et al. [6] investigated the use of machine-learning models in an uncertain manufacturing environment, demonstrating their effectiveness in managing stochastic conditions. Liu et al. [7] introduced an intelligent quality prediction and autonomous decision system to improve quality management in natural product manufacturing by leveraging ZDM concepts and multi-agent with reinforcement learning. Similarly, Smith et al. [8] proposed a novel functional outlier detection method that leverages domain knowledge to identify defective products, especially those with latent defects, presenting a systematic framework for integrating domain knowledge into outlier detection methodologies. Ghansiyal et al. [9] applied adversarial neural networks that focus on discriminator training oriented towards detecting defects to achieve the ZDM environment. Still, besides all these innovative approaches and good results, there are some manufacturing limitations according to data collection, pre-processing, and a consistently changing environment, that could be explored. Finally, May and Kiritsis [10] proposed a methodology architecture for ZDM environments called Z-Factor. This approach, besides being robust in handling different use cases from the manufacturing environment has shown some limitations by using a holistic architecture this could hinder the integration process in older companies.

Additionally, Nelson rules are quality control statistical rules [11] that offer the means of verifying whether a sequential variable is within control by analysing a minimum set of data points. Figure 1 shows the graphical representation of implementing two Nelson rules to monitor the evolution of the variable behaviour.

The existing computational applications to monitor the operation parameters usually present difficulties in integrating data from multiple and heterogeneous sources, usually legacy systems with proprietary protocols for accessing data, and problems in expressing the rationale for identifying anomalies and tendencies for a multiplicity of scenarios. Additionally, most of the existing quality control is concentrated on single-stage processes such as statistical process control (SPC), design of experiments (DoE), and Six Sigma and Lean tools. However, in multi-stage manufacturing systems, a change in an upstream quality parameter may affect some downstream quality parameters in subsequent stages, which is known as cascade property [12].

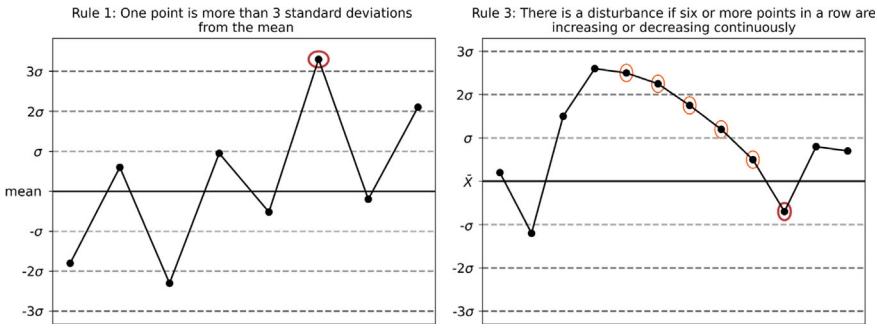


Fig. 1 Two examples of Nelson rules [11]

By revising the related work, it was noticed that most of the frameworks are strongly attached to the application context or solution technique. This opens the possibility for improvements in a more generalist and distributed monitoring approach, handling more heterogeneous data sources and simplifying the integration of newer technologies for context and non-context-aware rules monitoring.

3 System Architecture for the Rule-Based Monitoring Tool

The open, modular, and secure platform for industrial cyber-physical systems toward ZDM is aligned with the RAMI4.0 model and builds up the micro-services architecture, as illustrated in Fig. 2. The data collected from a physical asset, such as a welding station or a geometric measuring system, is sent in real-time to a folder of the ZDM platform monitored by a Python-based Filewatcher micro-service, which acts as a starting point for a type 2 AAS server implementation based on the FastAPI. An AAS can be defined as a digital representation of a physical asset through a collection of semantic models and submodels arranged in a static file (type 1 AAS) or in a server that allows communication between the asset and the AAS through an API that allows information to be accessed and updated (type 2 AAS) [13]. Therefore, integrated into the platform, the AAS server is responsible for the interoperability and translation of incoming data according to industrial standards defined through a type 1 AAS model of the physical asset, accessible through REST requests on an AAS Endpoint.

Once the data is in the specified format, it is published on the AAS Endpoint via a REST POST method and forwarded to a datalake, which is accessible to the data analytics tools via events (notification of changes in the system) or documents (structured data encapsulation units, such as JSON, XML or AVRO schemas). The analytics tools include different components based on micro-services, e.g., for performing real-time monitoring, what-if simulation, fault detection and diagnosis. The results of the analytical tools are also stored in the data lake. They can be viewed

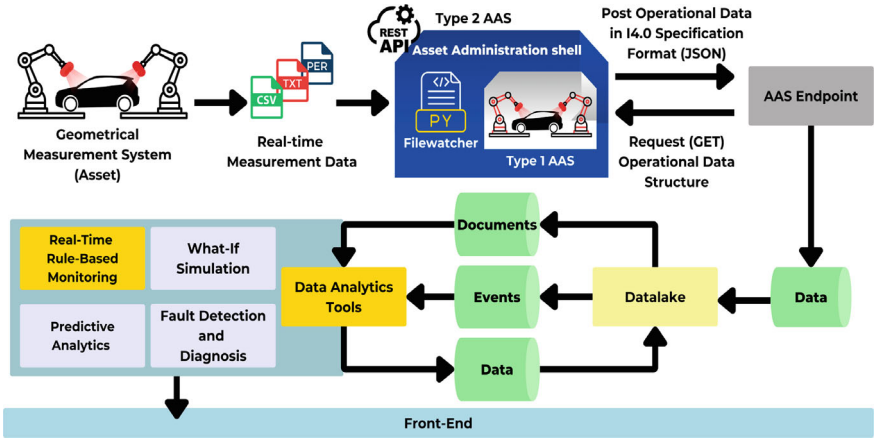


Fig. 2 Generic micro-services architecture for the open and modular industrial cyber-physical platform (based on [14])

via the platform's front end, which allows the easy visualization of the data history, analysis, and to identify anomalies.

3.1 Rule-Based Monitoring Tool Architecture

One of the data analytics tools included in the micro-service architecture is the rule-based monitoring tool that aims to detect earlier anomalies, defects and tendencies in the collected time series data. Figure 3 illustrates the architecture of this tool, which can subscribe to any data stream provided by any sensor data in real-time and file format. The monitoring tool processes the data and produces the triggering rule result alerts to other micro-services to consume. The dynamic programming approach aims to enable dynamic rules to be monitored in really different scenarios, and also generate a reliable output.

The step-by-step breakdown of the monitoring process can be detailed as follows.

- **Input Data:** Input data is received in any data format and from any data stream, being ingested into the micro-service where it will be processed and analyzed.
- **Data Handler:** Incoming data is firstly interpreted, analyzed, and processed by the Data Handler which is a configurable, use case-specific program designed to normalize and pre-process the data. The normalization ensures that the data is in a consistent format suitable for further processing by the monitoring engine.
- **Knowledge Base:** Stores the rules, rule configuration files, and domain-specific knowledge required for data evaluation. The Rule Monitoring Engine queries this repository to retrieve the rules needed for the current data set.

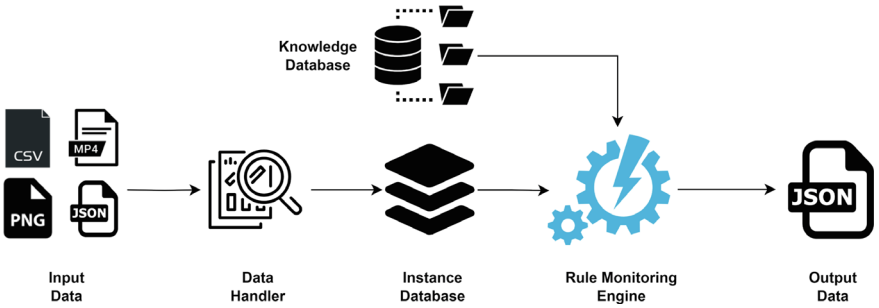


Fig. 3 Rule-based monitoring tool architecture

- **Instance Database:** Used to temporarily store data to be processed by the Rule Monitoring Engine, acting as an intermediary storage area to hold data until it is ready for further processing and final output generation.
- **Rule Monitoring Engine:** After normalization and buffering, data is retrieved by the Rule Monitoring Engine from the Instance Database, which is now responsible for evaluating the data against a set of predefined rules. These rules are sourced from the Knowledge Base, which contains the domain and not domain-specific logic and criteria for data evaluation. When a minimum amount of data is reached and all other conditions and criteria from the configuration file are matched, the Rule Monitoring Engine runs the rules, ensuring that the system processes data in manageable chunks and maintains efficiency.
- **Output Data:** Once the rules are executed, the system generates the output data in a normalized JSON format. This output is ready for downstream applications or systems requiring processed and evaluated data, such as the data visualization UI and other diagnosis systems.

Overall, this architecture provides a robust and scalable solution for leveraging configurable normalization, rule-based evaluation, and efficient data handling through buffering and knowledge management. It also addresses key challenges such as data normalization, rule execution, and efficient data management, thereby supporting the goal of achieving ZDM strategies.

3.2 Elaboration of Rules

Initially, the representation of rules relied on the Decision Model Notation (DMN) [15] standard from the OMG. While DMN was primarily designed for decision logic and rule evaluation using simple tabular data, it lacks inherent support for the intricate data processing tasks typically demanded by ML algorithms. Besides that, the Monitoring Tool rule definition protocol follows two primary steps: defining the business logic into a function and a configuration file that specifies how the

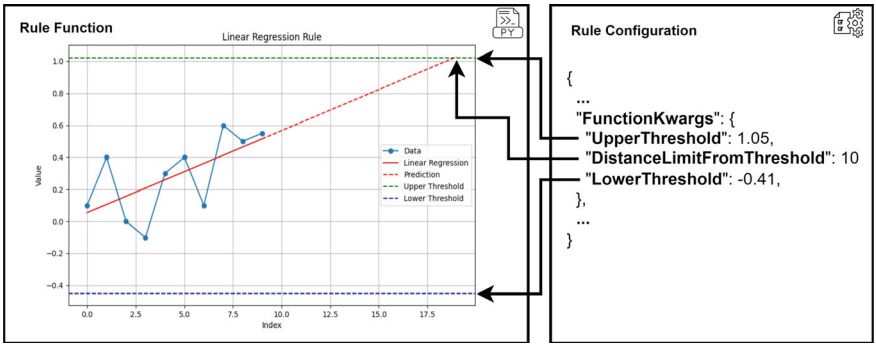


Fig. 4 A linear regression rule from the knowledge base

rule expects data to flow in order to work properly, and dynamically set the intern behaviour of the rule by adjusting internal variables without needing to make changes in the rule code. This approach has shown to be more efficient when leading with ML models that often require data in specific formats and pre-processing steps that may not fit neatly into DMN decision tables.

The Data Handler and Knowledge Base systems also played a crucial role in expanding these capabilities, not only normalizing the input data but also allowing more complex data interpretation and all steps required for ML pipelines. These possibilities, when designing statistical, probabilistic, contextualized, and non-contextualized rules to analyze variables behaviours on a large range of data kinds and environments, allow the early detection of defects and contiguous analysis following the ZDM principles. Figure 4 illustrate one non-context related ML rule that was defined using Linear Regression as a rule example in addition to a configuration file where the parameters can be adjusted. This rule triggers when the linear regression (Prediction) reaches the lower or upper thresholds within fewer than X predicted instances, indicating an extremely steep or shallow trend slope.

3.3 Nelson Rules Package

In the realm of ZDM, the ability to monitor real-time data and easily implement new rules can yield valuable insights to identify patterns and recurrences in data. This, in turn, can simplify the later analysis and early detection of defects, significantly enhancing the overall quality control process. The continuous and dynamic nature of this monitoring process allows for the immediate identification of potential issues, thereby enabling prompt corrective actions and minimizing the occurrence of defects.

To effectively demonstrate the application of the monitoring tool, the eight Nelson Rules were incorporated into the knowledge base as a non-context-specific rules package. The Nelson Rules, known for their statistical rigour, help in identifying

```
{
  "Name": "Nelson_Rule_1",
  "Description": "One point is more than 3 standard
  deviations from the mean.",
  "RelatedProblem": "One sample is grossly out of control.",
  "Function": "nelsonrule1",
  "FunctionKwargs": {...},
  "FunctionRequirements": {...}
}
```

Fig. 5 Nelson rule 1 configuration file example

out-of-control conditions in a process, thus ensuring that deviations are caught and addressed before they lead to significant quality issues. As an example, the implementation of Rule 1 (Fig. 1) states that if one point is more than three standard deviations from the mean, it indicates a potential process shift or outlier. This example shown in Fig. 5 and algorithm 1 below highlights the configuration and function files that respectively set the rule behaviour and the algorithm function. By configuring this rule within the monitoring tool, any such data points are flagged immediately, prompting further investigation.

Algorithm 1 Nelson Rule 1 - Function

Require: $data, AdditionalArguments$
Ensure: $data \leftarrow DataSeries$
  $upper\_limit \leftarrow data.mean() + data.std() * 3$
  $lower\_limit \leftarrow data.mean() - data.std() * 3$
  if $MeasurementsAreGreaterThan(data, upper\_limit)$ **then**
    return $\leftarrow True, AdditionalInformation$
  end if
  if $MeasurementsAreLowerThan(data, lower\_limit)$ **then**
    return $\leftarrow True, AdditionalInformation$
  end if
  return $\leftarrow False, AdditionalInformation$

4 Experimental Validation

4.1 Description of the Case Study

The case study involves an automotive assembly line, specifically the geometric measurement inspection stations in the body shop area, as shown in Fig. 6. The body shop workflow includes three automated process stages responsible for the bodywork, door assembly, and tailgate assembly. After each stage, the car is inspected at robotic

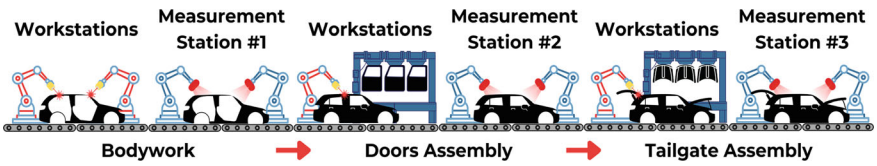


Fig. 6 Body shop stations sequence

measuring stations, to check for geometrical metrics. If nonconformities are found, the vehicle must go through a rework process, increasing production time and costs.

The assembly line is responsible for a daily production of approximately 700 cars divided into three shifts, totaling an average production of one car every 120 s. Therefore, any stoppage required to remove a vehicle for rework significantly impacts the line's processing time. The development of tools enabling the real-time monitoring of production trends allows for identifying potential issues in advance, enabling problems to be solved before they really occur by making adjustments in the workstations, thus preventing downtime.

4.2 Implementation

The micro-service implementation was exclusively conducted in Python, using two essential auxiliary libraries: NumPy for efficient numerical computations, particularly for array, matrix operations, and extensive mathematical functions, and Pandas for data manipulation and analysis, notably simplifying data filtering, grouping, and cleaning during the pre-processing steps. For persistent local data within the micro-service and also to ensure reliability and prevent system failure and data loss, a SQLite database was integrated as the instance database. This choice provided a dependable data persistence solution.

The docker technology was used to containerize the micro-service, ensuring that each process is isolated at the technology level; this means that each container has the system files within the pré-requisites to run, completely isolated from the host system preventing interference's with other micro-services. This approach enhanced the reliability and predictability of the application and facilitated deployment across the infrastructure environment, including local production machines in the factory.

4.3 Experiments

Two experiments were devised with the aim to test the micro-service performance in relation to execution time and reliability in a development local environment with the following specifications: (CPU) Intel Core I7 7700HQ (4/8 cores @ 2.8 Ghz 45W),

Table 1 Result of experiments: experiment 1 is related to the execution of 4 Nelson rules + the L. R. rule, and experiment 2 is related to the execution of 8 Nelson rules + the L. R. rule with both processing 50 cars

			Experiment 1		Experiment 2	
Station	No of features	Avg. rule triggering per validation	Total time in MS	Per instance time in MS	Total time in MS	Per instance time in MS
1	720	2	40.456	809	59.724	1.194
2	166	2	10.677	213	15.227	304
3	40	2	3.077	61	4.871	97

and (RAM) 16GB DDR4 2400 Mhz, similarly as it is being used in the production line to ensure the monitoring tool behaviour in each station before deployment. To achieve this, the first experiment implied the usage of 4 Nelson Rules plus the Linear Regression Rule. The second experiment otherwise implied the usage of 8 Nelson Rules plus the Linear Regression Rule, to establish fairness between the experiments. 50 cars were processed with the same system specification. Table 1 shows the results of both experiments.

4.4 Analysis of Results

The performance of the rule-based monitoring tool strongly depends on the number of rules and variables being considered and also on the rules complexity. As illustrated in Fig. 7 the execution time rises linearly, according to the number of variables (p) being analysed and the number of rules (r) being executed, leading to a complexity of $O(p * r)$. Also analysing the results shown in Fig. 7 one may conclude that from both factors, the number of rules is much more meaningful performance-wise than the number of variables. This happens because the execution time of every rule can vary in algorithm complexity accordingly with the business logic applied, and can require much more processing from the CPU than one additional variable that has a more pattern-type behaviour.

According to the use case description, data is received at an average rate of 700 instances (cars) per day which represents an arrival frequency of approximately one instance every 120.000 milliseconds. Considering this rate and the linear regression equation illustrated in Fig. 7, from the 8 Nelson rules experiment, the system can hypothetically handle a maximum of 74.512 variables, ensuring the real-time response for the use case.

Finally, the results of the continuous triggering of rules are showcased in a User Interface (UI) micro-service dashboard (See Fig. 8). For each rule being monitored, there is a UI component that tracks the rules triggering information produced by the monitoring tool, focusing on showing useful information to support the manager in making process-level decisions based on the early detection of defects.

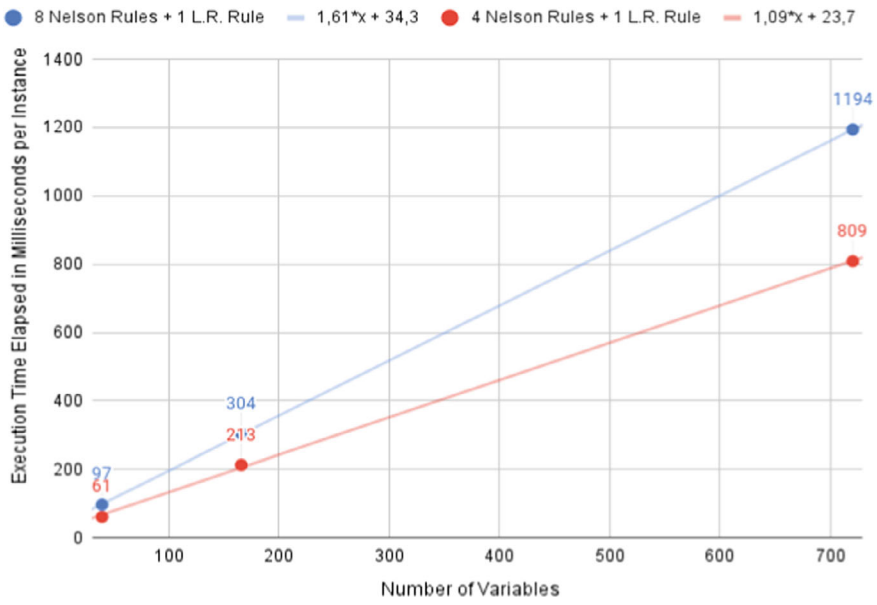
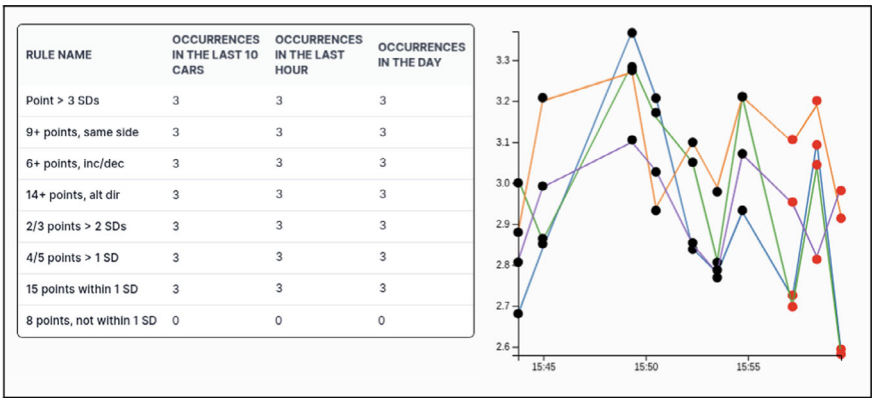


Fig. 7 Performance of the rule-based monitoring tool according to the variation of the number of variables and rules



RULE NAME	OCCURRENCES IN THE LAST 10 CARS	OCCURRENCES IN THE LAST HOUR	OCCURRENCES IN THE DAY
Point > 3 SDs	3	3	3
9+ points, same side	3	3	3
6+ points, inc/dec	3	3	3
14+ points, alt dir	3	3	3
2/3 points > 2 SDs	3	3	3
4/5 points > 1 SD	3	3	3
15 points within 1 SD	3	3	3
8 points, not within 1 SD	0	0	0

Fig. 8 Dashboard model that showcases the outputs of the rule-based monitoring tool

5 Conclusion

The proposed rule-based monitoring tool stands out for its innovative characteristics in adapting to new technologies and different use cases; this generalist approach expects to close the gaps in the analyzed related work. Applying the proposed approach to a real-world study case yielded promising results, demonstrating its

potential effectiveness in addressing complex data processing tasks. This approach also supports the definition and dynamic configuration of contextualized and multi-purpose rules, integrating ML, optimization, and statistical algorithms into a single tool. This monitoring tool also aims to allow an effective handling of various constraints on the manufacturing environment, such as high-frequency data, missing information, and non-normalized data, to achieve a ZDM environment.

Future work will focus on enhancing the system's capabilities by allowing the configuration of rule pipelines and a concurrency mode in which each rule will run in a different thread maximizing resource usage and performance. Additionally, efforts will be directed towards developing a configurator to simplify the rule definition, empowering high-level users to effortlessly create and manage rule sets. These improvements aim to further streamline the system's functionality and broaden its applicability across various domains.

Acknowledgements This work was partially supported by the HORIZON-CL4-2021-TWIN-TRANSITION-01 openZDM project, under Grant Agreement No. 101058673. Also, it was supported by national funds through FCT/MCTES (PIDDAC): CeDRI, UIDB/05757/2020 (DOI: 10.54499/UIDB/05757/2020) and UIDP/05757/2020 (DOI: 10.54499/UIDP/05757/2020); and SusTEC, LA/P/0007/2020 (DOI: 10.54499/LA/P/0007/2020).

References

1. Hoda ElMaraghy, Laszlo Monostori, Guenther Schuh, and Waguih ElMaraghy. Evolution and future of manufacturing systems. *CIRP Annals*, 70(2):635–658, 2021.
2. P.B. Crosby. *Quality is Free: The Art of Making Quality Certain*. Mentor book. 1979.
3. Rui Pedro Lopes, Ahmed Ibrahim, Jose Barbosa, and Paulo Leitao. Microservices architecture to enable an open platform for realising zero defects in cyber-physical manufacturing. *Submitted to Logic Journal of the IGPL*, 2024.
4. DIN SPEC 91345: Reference Architecture Model Industrie 4.0, 2016.
5. Pingle Yang, Yalei Yang, and Yunfeng Lou. A business activity real-time monitoring platform based on rule engine. *Procedia Engineering*, 15:3744–3748, 2011.
6. D.R. Gunasegaram, A.S. Barnard, M.J. Matthews, B.H. Jared, A.M. Andreaco, K. Bartsch, and A.B. Murphy. Machine learning-assisted in-situ adaptive strategies for the control of defects and anomalies in metal additive manufacturing. *Additive Manufacturing*, 81, 2024.
7. Xiaolin Liu, Guanghua Li, Qiang Gao, Shanshan Gao, and Yanan Yu. An intelligent quality prediction and autonomous decision system for zero defect manufacturing in natural product manufacturing. *Computers & Industrial Engineering*, 191, 2024.
8. John Smith, Jane Doe, and Emily Brown. Domain-knowledge-informed functional outlier detection for line quality control systems. *Computers & Industrial Engineering*, 189, 2024.
9. Shradha Ghansiyal, Li Yi, Peter M. Simon, Matthias Klar, Marius Marvin Müller, Moritz Glatt, and Jan C. Aurich. Anomaly detection towards zero defect manufacturing using generative adversarial networks. *Procedia CIRP*, 120:1457–1462, 2023.
10. Gökan May and Dimitris Kiritsis. Zero defect manufacturing strategies and platform for smart factories of industry 4.0. In Laszlo Monostori, Vidosav D. Majstorovic, S. Jack Hu, and Dragan Djurdjanovic, editors, *Proceedings of the 4th International Conference on the Industry 4.0 Model for Advanced Manufacturing*, pages 142–152, Cham, 2019. Springer International Publishing.

11. L. S. Nelson. The shewhart control chart—tests for special causes. *Journal of Quality Technology, 16(4), 237–239*, 1984.
12. S. Asadzadeh, A. Aghaie, H. Shahriari, and S. T. A. Niaki. Improving reliability in multistage processes with autocorrelated observations. *Quality Technology & Quantitative Management*, 12(2):143–157, 2015.
13. Xun Ye, Seung Ho Hong, Won Seok Song, Yu Chul Kim, and Xiongfeng Zhang. An industry 4.0 asset administration shell-enabled digital solution for robot-based manufacturing systems. *IEEE Access*, 9:154448–154459, 2021.
14. Iury Michel Treuk, Alexandre O. Júnior, Rui Pedro Lopes, Gonçalo Mota, J. Joaquim Mira, and Paulo Leitao. Digitalization of industrial inspection assets through the asset administration shell. In *2024 IEEE 7th International Conference on Industrial Cyber-Physical Systems (ICPS)*, pages 1–6, 2024.
15. Decision Model. Notation (dmn). version 1.1. *Object Management Group, Inc*, 2016.