Proceedings of the 2025 Winter Simulation Conference
E. Azar, A. Djanatliev, A. Harper, C. Kogler, V. Ramamohan, A. Anagnostou, and S. J. E. Taylor, eds.

DATA-DRIVEN DIGITAL TWIN FOR THE PREDICTIVE MAINTENANCE OF BUSINESS PROCESSES

Paolo Bocciarelli and Andrea D'Ambrogio

Dept. of Enterprise Engineering, University of Rome "Tor Vergata", Rome, ITALY

ABSTRACT

This paper presents a data-driven framework for the predictive maintenance of Business Processes based on the Digital Twin (DT) paradigm. The proposed approach integrates process mining techniques and a low-code development approach to build reliability-aware simulation models from system logs. These models are used to automatically generate executable DTs capable of predicting resource failures and estimating the Remaining Useful Life (RUL) of system components. The predictions are then exploited to trigger preventive actions or automated reconfigurations. The framework is implemented using the PyBPMN/eBPMN framework and evaluated on a manufacturing case study. The results show that DT allows for timely interventions, minimizes system downtime, and ensures process continuity.

1 INTRODUCTION

In recent years, the ever-increasing adoption of innovative technologies, such as Artificial Intelligence (AI) and the Internet of Things (IoT), has radically changed the way organizations design and manage Business Processes (BPs). Specifically, a BP refers to the flow of activities executed by an organization through the orchestration of human or automated resources to achieve a well-defined objective, such as delivering services or producing goods (van der Aalst et al. 2003). This technological trend opens up new perspectives and challenges in the Business Process Management (BPM) context.

The execution of organizational and operational BPs often relies on an IoT-based infrastructure, including interconnected devices capable of detecting, storing, and processing large volumes of data. Data Science and AI offer effective methods to extract knowledge from such data and drive the optimization of BPs at the core of organizations' operational life.

The proper implementation of BPM practices is essential to achieve organizational goals. BPM supports stakeholders throughout the entire process lifecycle, offering tools to monitor BP execution and foster continuous improvement (Bocciarelli et al. 2017; Tiftik et al. 2022).

In this context, process optimization goes beyond performance metrics. A critical challenge for BPs that orchestrate complex infrastructures (e.g., production systems) lies in managing system reliability. BPs may involve execution platforms composed of various kinds of resources, such as equipment structured in multiple subsystems or with redundant components. Ensuring the reliability and availability of BPs over such platforms is a challenging objective that requires the adoption of innovative BPM approaches.

This paper addresses the *predictive maintenance* of BPs (Van Dinter et al. 2022; Aivaliotis et al. 2019) through a novel simulation approach aimed at analyzing BP reliability by predicting resource failures within the execution platform. In this respect, the proposed method leverages process mining (PM) techniques (van der Aalst 2016) and the Digital Twin (DT) paradigm (Singh et al. 2021).

Process Mining is a discipline that aims to discover, monitor, and improve BPs by extracting knowledge from event logs (van Der Aalst 2012). BP execution relies on process-aware information systems (PAIS), which capture and log process data (Martin et al. 2014). Furthermore, execution platform devices often detect and store their operational state in state logs (Friederich and Lazarova-Molnar 2022).

Models are central to the development of DT (Eramo et al. 2022). A systematic approach to model development and maintenance is crucial for an effective DT lifecycle. Thus, Model-Driven Engineering (MDE) approaches can significantly support DT-based systems (Michael et al. 2025).

This paper explores the use of DT to enable the predictive maintenance of BPs, introducing an architectural framework for the automated generation of data-driven, reliability-aware DTs. The framework adopts a low-code development paradigm (Bocciarelli and D'Ambrogio 2023) and relies on principles introduced in the MDE field. Specifically, it makes use of methods, standards and technologies provided by the Model Driven Architecture (MDA) (Ojbect Management Group 2003). Moreover, it adopts PM techniques to derive reliability-aware simulation models from process data. Such models are then used to automatically generate the required DT. The reliability-aware simulation system can predict resource failures and estimate the Remaining Useful Life (RUL) of devices within the execution platform. These predictions support BP Engine reconfiguration or the planning of timely maintenance interventions.

In order to achieve this objective, the proposed approach employs PyBPMN/eBPMN, a framework for performance- and reliability-aware BP modeling and simulation. PyBPMN (D'Ambrogio et al. 2016) extends BPMN to include performance and reliability properties. PyBPMN models can be easily simulated by using eBPMN, a Java-based domain-specific simulation infrastructure (Bocciarelli et al. 2014).

The paper is structured as follows. Section 2 outlines the background of this work. Section 3 presents the proposed framework for the DT-based predictive maintenance of BPs. Section 4 illustrates an example application. Finally, Section 5 provides conclusive remarks.

2 BACKGROUND

This section provides the necessary background for this work, illustrating the Digital Twin paradigm and briefly outlining the state of the art in predictive maintenance.

2.1 Digital Twin

The concept of the Digital Twin (DT) has gained significant attention in recent years, particularly in industrial and engineering domains. A Digital Twin (DT) is a virtual replica of a physical system that is continuously updated with real-time data to enable simulation, analysis, and optimization (Singh et al. 2021). DTs have been adopted in various sectors, including manufacturing, healthcare, and infrastructure management. One of the most promising applications of Digital Twin technology is in predictive maintenance, where real-time monitoring and simulation-based techniques are jointly employed to prevent failures, reduce downtime, and optimize maintenance schedules.

A key distinction must be made between a Digital Twin and a Digital Shadow (Kritzinger et al. 2018). Although both concepts involve digital representations of physical assets, they differ in their level of interaction and data exchange. A Digital Twin establishes a bidirectional link between the physical and digital systems, enabling not only passive data collection but also active decision-making and control. The continuous feedback loop allows for dynamic adjustments and real-time optimization of the physical asset. In contrast, a Digital Shadow is a more limited form of digital representation in which data flows only in one direction, from the physical object to its digital counterpart, without active feedback or influence over the physical entity. Although a Digital Shadow can still provide valuable insights by analyzing historical and real-time data, it lacks the ability to directly interact with or modify the physical system based on predictive analytics.

The adoption of DT has emerged as a promising approach to support predictive maintenance, where they can be used to prevent system failures, optimize maintenance schedules(Rossini et al. 2020), and also enabling proactive decision making (Friederich and Lazarova-Molnar 2022; Van Dinter et al. 2022). By integrating IoT sensor data, process mining algorithms and simulation-based techniques, DTs forecast potential failures and recommend optimal maintenance actions (Abdullahi, Longo, and Samie 2024), thus improving reliability and reducing operational costs.

2.2 Predictive Maintenance

Predictive maintenance (PdM) is an approach that leverages data analysis to predict the failure of system components and schedule timely interventions before breakdowns occur. Its relevance in many operational fields is witnessed by the available literature that underlines the benefits and challenges of this technique.

Various systematic literature reviews have been proposed that underline the potential of PdM to reduce downtime and improve operational efficiency (Zonta et al. 2020; Zhang et al. 2019). Such contributions also identified key challenges, such as data quality, interoperability, and the lack of standardized frameworks.

As underlined in (Aivaliotis et al. 2019) existing PdM approaches often rely on static models or require extensive manual calibration, which limits their adaptability and scalability. The adoption of data-driven techniques and DT constitutes a promising opportunity to enable proactive decision making (Friederich and Lazarova-Molnar 2022; Van Dinter et al. 2022). DTs, by leveraging IoT data and simulation models, enable accurate failure prediction and optimal maintenance planning (Abdullahi et al. 2024).

Further research is needed to develop integrated frameworks for DT-based predictive maintenance of BPs. This work contributes by proposing a model-based DT architecture specifically designed for reliability-aware BP monitoring and management.

3 DATA-DRIVEN FRAMEWORK FOR THE PREDICTIVE MAINTENANCE

The conceptual architecture of the proposed framework for the predictive maintenance of BPs is shown in Figure 1.

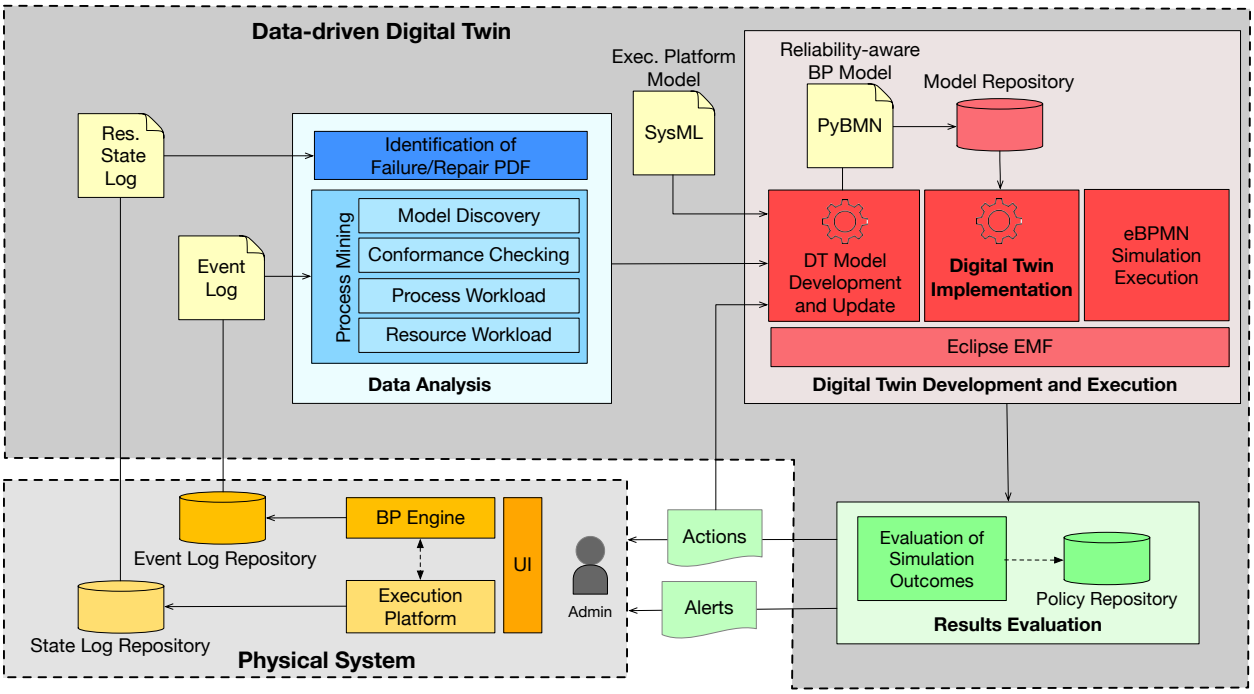


Figure 1: Architecture of the BPs Predictive Maintenance Framework.

The framework includes the Physical System (PS) and the related Data-driven DT. As noted in Sections 1 and 2.1, the proposed approach takes advantage of a complete DT paradigm. More specifically, on the one hand, real-time data gathered from the physical system are used to develop a simulation model of the PS. On the other hand, the predictions thus obtained are analyzed to generate appropriate actions that support the variation of the PS configuration, in order to address reliability issues. Furthermore, the DT configuration is continuously updated to remain aligned with its physical counterpart.

The PS is structured into three main subsystems:

- **BP Engine**, which is a *process-aware information system (PAIS)* responsible for executing the BP that models the operational logic of the system and orchestrates resources and equipment composing the underlying execution platform. It is also capable of tracking all events occurring during the BP execution.
- **Execution Platform**, which is composed of a set of hardware resources (i.e., sensors, actuators, equipment in a production line, etc.) and/or software resources (i.e., web services);
- **User Interface**, which enables the system administrator to deploy the BP and manage its execution. Through the user interface, the administrator is also able to monitor the state of the execution platform's devices;
- **Event Log Repository**, which stores the event data captured by the BP Engine. In the remainder of this work, we assume that the event log is compliant with the IEEE eXtensible Event Stream (XES) standard (XES Working Group 2023). It is worth noting that the use of a different formalism does not alter the overall structure of the framework. The format of the event log is discussed in Section 3.2;
- **State Log Repository**, which stores the operational state of the various device in the execution platform. The different kinds of information stored in the State Log are illustrated in Section 3.2;

The data-driven DT component consists of three subsystems:

- **Data Analysis**, the subsystem responsible for analyzing the data contained in the logs to discover the information needed for the initial development of a reliability-aware BP model and for its update to ensure that it remains aligned with changes in the physical system. It is described in detail in Section 3.2.
- **Digital Twin Development and Execution**, the subsystem that constitutes the actual DT. Specifically, the information produced as output by the Data Analysis component is used to create (and update when needed) the reliability-aware BP model. The latter is then used to generate the implementation of the simulation system, whose execution allows one to obtain predictions on the availability of the physical system. The *Digital Twin Development and Execution* subsystem is described in detail in Section 3.3.
- **Results Evaluation**, the subsystem responsible for analyzing the data produced by the simulator and providing the necessary feedback to the physical system. This component is described in Section 4.3.

An essential aspect of the proposed approach lies in the modeling of the execution platform, in order to effectively represent the structure of composed devices, and appropriately deal with the evaluation of their reliability. The next section outlines the resource modeling rationale.

3.1 Execution Platform and Resource Modeling

In many real-world scenarios, the various resources orchestrated by a BP could be complex systems themselves. An execution platform can be made up of equipment structured into multiple subsystems or include multiple redundant devices that ensure operability in the event of failures.

To enable the appropriate representation of such scenarios with the required degree of flexibility, and according to the design principles at the basis of PyBPMN, an Execution Platform EP is a set defined as $EP = \{R_0, R_i, \ldots, R_N\}$

Each resource $R_i \quad with \quad i = 1, \ldots, n$ represents a hardware or software system that is orchestrated by a BP engine to fulfill a well-defined service request. More specifically, each resource R_i represents one of the following entities:

- **Atomic Resource**: the structure of R_i does not require further specification. It is treated as an atomic entity capable of processing a service request coming from the BP engine. In the PyBPMN framework, this kind of resource is identified as a *Performer*;
- **Structured Resource**: the resource R_i is structured in various subsystems. Formally, $R_i = \{R^s_{i1}, R^s_{i2}, \ldots, R^s_{ij}, ..., R^s_{iM}\}$, where each resource R^s_{ij} represents a subsystem in which R_i is structured. In order to process a service request, all M subsystems R^s_{ij} must simultaneously be in an *operational state*, otherwise the whole subsystem R_i is in a failed state and the service request cannot be processed. In the PyBPMN framework, such resources are referred to as *Subsystems*;
- **Redundant Resource**: the resource R_i represents a collection of various redundant devices: $R_i = \{R^r_{i1}, R^r_{i2}, \ldots, R^r_{ij}, ..., R^r_{iK}\}$, where each resource R^r_{ij} is a redundant device that operates according to either a cold-standby or hot-standby paradigm: initially, one device, e.g., R^r_{i1}, is set to the *active state*, while the remaining $K-1$ redundant devices are in an *inactive state*. The active resource is responsible for executing the service requests directed to the redundant subsystem R_i. In case of failure, the active resource turns into a *failed state*, and a different inactive resource is selected and configured as the new active one. To process a service request, at least one of the K redundant devices R^r_{ij} must be in an operational state; otherwise, the redundant subsystem is considered to be in a failed state and the service request cannot be executed. In the PyBPMN framework, such resources are referred to as *Brokers*;

The metamodel that conceptually specifies the structure of the resources is represented in the class diagram shown in Figure 2.

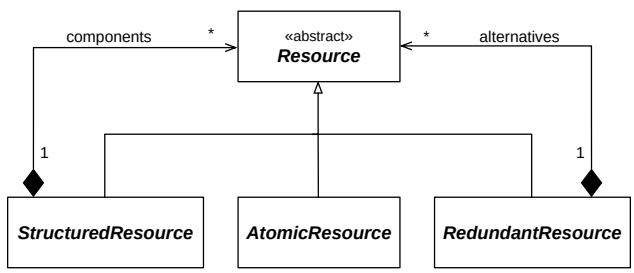


Figure 2: Conceptual model of resources.

The BP Engine is therefore configured so that each task of the BP is associated with one or more resources R_i among those that compose the execution platform EP. It should be underlined that if R_i (or one of its subsystems) is a redundant subsystem consisting of K device resources, the BP maintains its operability through a reconfiguration operation in which the failed resource is replaced at run-time with an available backup.

3.2 Data Analysis

In the proposed framework, the adoption of a data-driven approach is essential to ensure that the DT is based on real-time data collected from the PS. Therefore, the availability of appropriate data is crucial to develop an effective simulation model which is able to capture the reliability-oriented characteristics of the PS. As underlined in Section 3.1, in the addressed scenario, the resources that compose the execution platform are orchestrated by a BP Engine.

According to a widely shared assumption (Friederich and Lazarova-Molnar 2022; van der Aalst 2016; Martin et al. 2014):

- the BP engine (i.e., the PAIS) orchestrates the hardware and software resources capable of accomplishing the needed service requests, according to the execution flow specified by the BP model. The execution of each BP task generates appropriate events;

- the PAIS captures event-related data and record them into the Event Log Repository (see Figure 1);
- as generally each PAIS might capture various information, which vary depending on the specific operational context, we assume that the recorded data are compliant with the XES standard and include standard attributes that identify a *trace* and the *activity name*. In addition, there are attributes provided by three standard extensions (XES Working Group 2023): `Start Timestamp` and `Completion Timestamp` from the *life-cycle* and the*time* extensions, `Resources` and *Role* from the *organizational* extension. Figure 3a shows an example of an event log illustrating the various information recorded for each event;

We also assume that each resource can detect and record its own state in a State Log (Friederich and Lazarova-Molnar 2022). Although, in general, resources can track different states (e.g., operational, busy, idle, failed, etc.), for the purposes of this research, we assume that the State log is defined by the following attributes:

- **Resource Id**, which uniquely identifies a resource $R_i \in EP$;
- **Timestamp**, in the format `YYYY-MM-DD HH:mm:ss.sss`;
- **State**, which can take values `operative` and `failed`;

Figure 3b illustrates the structure of the State Log.

Case ID	Start Timestamp	Complete Timestamp	Activity	Resource	Role
1	2024/03/01 00:00:00.000	2024/03/01 00:37:00.000	Create Purchase Requisition	ResourceA	RequisitionManager
2	2024/03/01 00:16:00.000	2024/03/01 00:29:00.000	Create Purchase Requisition	ResouirceA	RequisitionManager
3	2024/03/01 02:23:00.000	2024/03/01 03:03:00.000	Create Purchase Requisition	ResourceB	RequisitionManager
1	2024/03/01 05:37:00.000	2024/03/01 05:45:00.000	Create Request for Quotation	ResourceC	QuotationManager
...	...	...	...	...	...

(a) Event log.

Start Timestamp	Resource ID	State
2024/03/01 00:00:00.000	R1	Operative
2024/03/01 00:00:00.000	R2	Operative
2024/03/10 00:08:50.000	R1	Failed
2024/03/10 00:08:50.000	R1	Operative
...	...	...

(b) State log

Figure 3: Structure of the Event and State Log recorded by the Physical System.

Data analysis is based on PM techniques. In particular, the two logs are analyzed using different approaches.

For the event log analysis, PM tools are employed to discover information such as: the structure of the BP model, the process workload, tasks' performance characterization (e.g., the average service time), routing probabilities, etc. The use of such approaches to derive a BP model has already been discussed in previous works (Bocciarelli and D'Ambrogio 2024) and will not be further detailed here. Interested readers are referred to the relevant literature.

The state log requires a dedicated component in charge of determining, for each resource, the mean values of failure (MTTF) and repair (MTTR) events, as well as their corresponding probability distribution functions (PDF). In this respect, various methods can be adopted; the prototypal implementation used to analyze the case study discussed in Section 4 makes use of the maximum likelihood estimation. This component is referred to as *Identification of Failure/Repair PDF* in Figure 1.

The information retrieved from the log repositories is then used by the *DT Development and Execution* component to build the digital counterpart of the physical system and obtain predictions about the resource failure. This aspect is discussed in the next Section.

3.3 Digital Twin Development, Execution and Results Evaluation

The development of the simulation system that constitutes the actual DT of the physical system is carried out in two steps: initially, the data extracted by the *Data Analysis* component are used to build a reliability-aware model of the BP. Subsequently, this model is used to generate the implementation of the executable simulation.

The simulation model, as described in Section 2.1, is a BPMN model annotated according to the PyBPMN extension. This model is automatically generated through a `model-to-text` transformation implemented by the *Model Development and Update* component. The transformation takes as input the data extracted by the *Data Analysis* component, a SysML model that describes the execution platform, and the allocation of BP tasks to resources.

The simulation system uses eBPMN. As underlined in Section 1, its implementation can be obtained straightforwardly from the PyBPMN model through a `model-to-text` transformation, as illustrated in (Bocciarelli et al. 2019).

Both the PyBPMN metamodel and the eBPMN framework are built on the Eclipse Modeling Framework, which provides a concrete implementation of the MDA standard.

The resulting eBPMN simulation is then executed to obtain predictions of resource failures. Finally, the simulation results are analyzed by the *Results Evaluation* component, which is responsible for determining the RUL of each resource that makes up the hardware platform (i.e., the time interval between the current instant and the next predicted failure).

Depending on the type of resource for which a failure is predicted, different actions might be taken:

- **Preventive maintenance notification:** atomic and structured resources do not have an available backup device that can be activated in case of failure. In this case, a message is shown in the Admin UI to notify the need for a preventive maintenance intervention. The administrator is responsible of scheduling the required maintenance intervention, coherently with the resources RUL, ensuring minimal impact on system operations;
- **Automated reconfiguration:** *Redundant Resources* are associated with a pool of backup resources. In this case, two actions are triggered:
  - if at least one backup resource is working and available (i.e., in an `inactive state`), the BP engine is automatically reconfigured so that the failed resource is replaced by its backup. The update is also notified to the *DT Model Development and update component*, to update the PyBPMN model accordingly and align the DT implementation;
  - A notification is sent to the Admin UI so that the administrator can assess the need for a preventive intervention.

The next section discusses an application example that has been analyzed through a prototypical implementation of the proposed framework.

4 EXAMPLE APPLICATION

To assess the feasibility and effectiveness of the proposed approach, this section presents its application to a case study that deals with a steel component production line.

The implementation of the prototypal framework includes the following open-source components:

- **Camunda Business Process Engine:** a PAIS which executes a BP avaiable in XML format in charge of orchestrating the Execution Platform's resources (https://camunda.com/);

- **ProM Tools:**a plugin-based open-source framework which implements various process mining algorithms (https://promtools.org/);
- a set of web services, which have been implemented to emulate the behavior of devices and actuators of a real-world production line.

Steel components are produced according to the following activity flow: initially, each component is obtained by appropriately cutting and bending a steel sheet. Depending on the product type, a specific insulating coating might optionally be applied. Finally, the finished product undergoes a quality inspection using a camera that captures images and sends them to an automated processing system for defect detection.

The execution platform consists of the following devices:

- **Forming Station**: a structured resource composed of two subsystems: a machine that cuts fixed-size pieces from a metal sheet (Cutter) and a subsequent hydraulic press for shaping (Bending Press). Both the Cutter and the Bending Press are redundant resources consisting of two device operating in cold-standby;
- **Coating Station**: a redundant resource consisting of two painters in a cold standby configuration;
- **Camera**: an atomic resource that provides the capability of inspecting the finished piece and detecting potential manufacturing defects through image analysis.

The *Event Log Repository* and the *State Log Repository* have been initialized with synthetic logs, generated through an additional feature of the eBPMN framework, as discussed in (Bocciarelli and D'Ambrogio 2025).

The following subsections illustrate the application of the framework depicted in Figure 1.

4.1 Data Analysis

The system log and the state log are analyzed to extract the necessary information to create the reliability-aware simulation model.

Table 1 shows the performance analysis results for the execution platform resources (derived from the event log), while Table 2 presents the reliability analysis results (from the state log). It should be underlined that while the Cutter and the Coating Station are both equipped with two identical redundant devices, the two Press devices have different reliability characteristics.

Table 1: Performance Analysis of execution platform's resources.

Atomic Resource	Service Time (Mean Value)	PDF
Blade 1-2	4 min	Exponential
Bending Press 1-2	90 sec	Exponential
Painter 1-2	3 min	Exponential
Camera	3 min	Exponential

Table 2: Reliability Analysis of execution platform's resources.

Atomic Resource	MTTF (mean)	MTTF (var)	Failure PDF	MTTR (mean)	MTTR (var)	Repair PDF
Blade 1-2	90 days	8 days	Normal	15 min	2 min	Normal
Bending Press 1	180 days	10 days	Normal	10 days	2 day	Normal
Bending Press 2	10 days	2 days	Normal	24 hours	1.5 hours	Normal
Painter 1-2	150 days	10 days	Normal	120 min	10 min	Normal
Camera	365 days	10 days	Normal	120 min	5 min	Normal

Regarding the process workload, the arrival time of production requests follows a Poisson distribution with a mean of 10 minutes.

4.2 Digital Twin Development

The automated specification of the PyBPMN model performed by the *DT Model Development and Update* component takes advantage of the information extracted from the log repositories and requires a SysML model that specifies the structure of the execution platform and the allocation of process activities to the needed resources.

Specifically, the Execution Platform specification consists of two distinct SysML diagrams: i) a Block Definition Diagram that illustrates the resource allocation, and an Internal Block Diagram that specifies the structure of non-atomic resources.

These models are shown in Figures 4 and Figure 5, respectively.

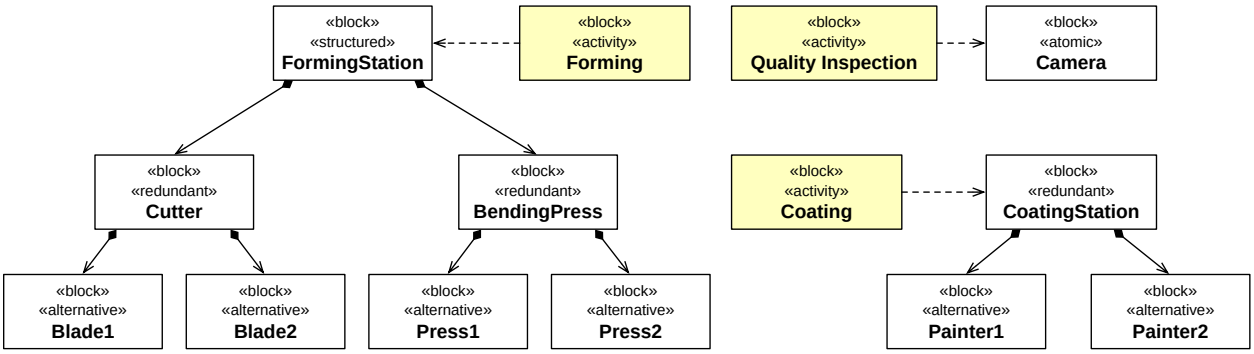


Figure 4: Block Definition Diagram specifying how the physical resources are used by the process activities.

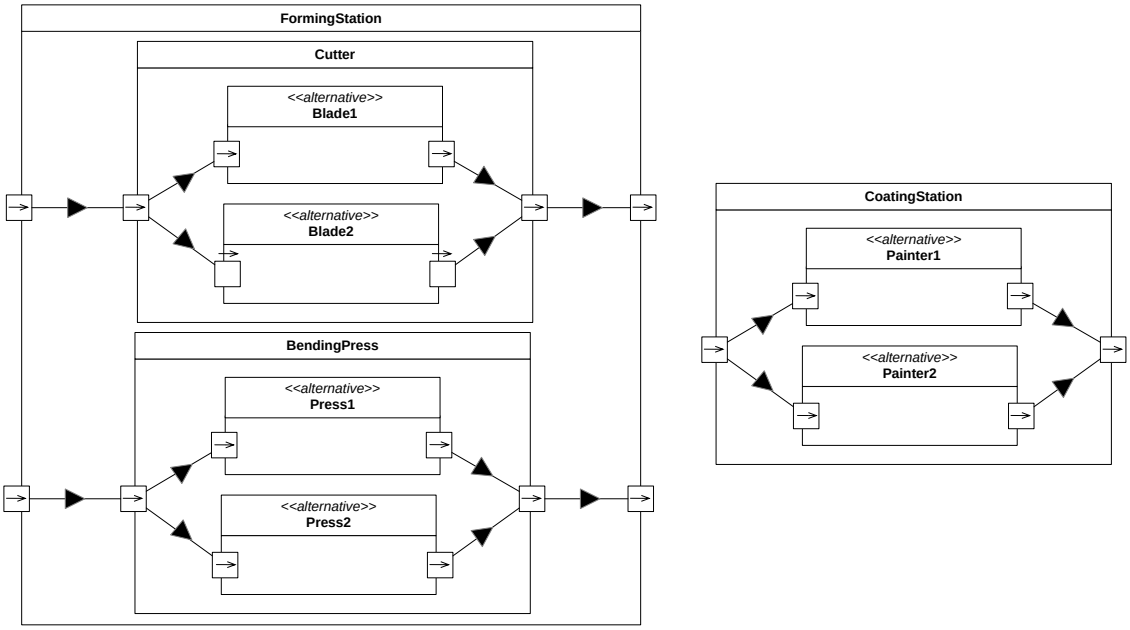


Figure 5: Internal Block Diagram showing the structure of the various complex resources composing the execution platform.

Figure 6 shows the PyBPMN model obtained.

For the sake of conciseness, Figure 6 only includes the «`PyPerformer`» annotation for one of the two redundant devices that equip each complex resource.

The PyBPMN model thus obtained is then used to generate the corresponding eBPMN code.

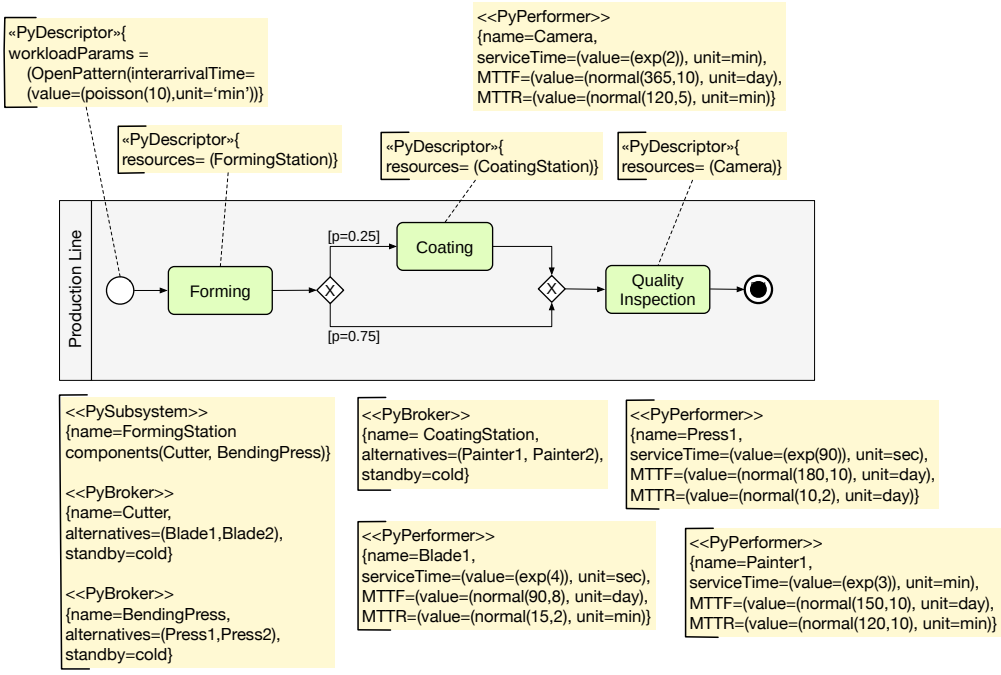


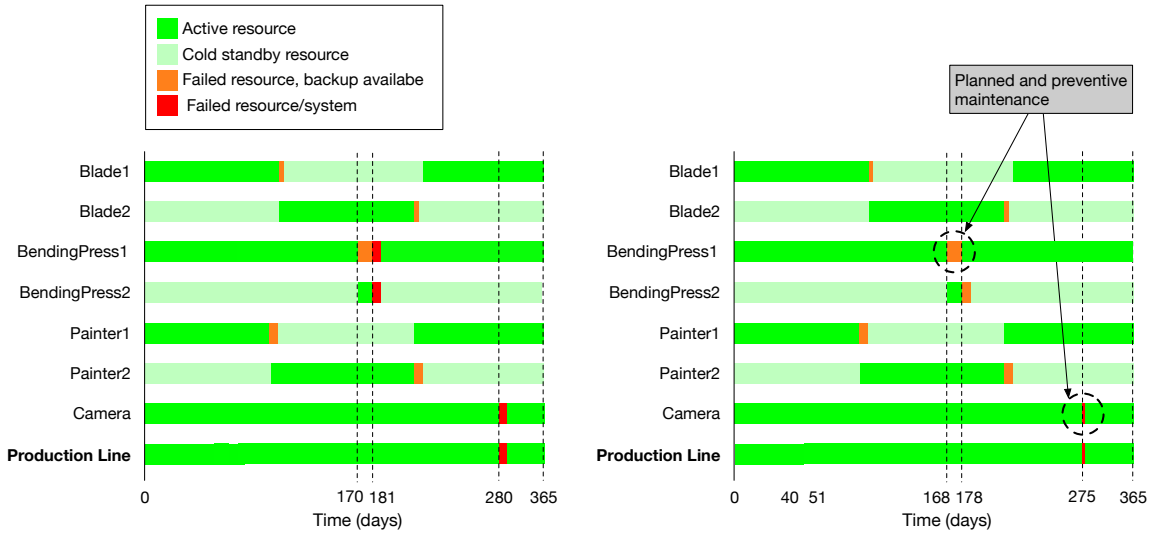
Figure 6: BPMN with performance and reliability PyBPMN annotations.

4.3 Results Evaluation and Actions on the Physical System

Figure7 illustrates the impact of the application of the framework on the production system. Specifically, Figure7a shows the results of the simulation analysis. At time $t = 170$, the Bending Press 1 resource is expected to fail. The backup resource will be activated as the repair of the first resource begins. Bending Press 2 is expected to fail before Bending Press 1 is repaired, causing an interruption of the production system at time $t = 181$. Similarly, at time $t = 280$, the failure of the Camera device is predicted. Since a backup resource is available, the production process will be interrupted again.

Figure7b shows how the results of the reliability prediction and the DT feedback impact the actual system:

- Failures affecting the Cutter (Blade 1–2) and Painter (1–2) can be managed through automatic reconfiguration, without impacting the state of the production process;
- The prediction of upcoming failures of the two Bending Press devices enables the administrator to schedule preventive maintenance interventions, minimizing recovery time and avoiding interruptions of the production system;
- Similarly, a preventive intervention can be planned on the Camera device. Since no backup is available, the administrator can schedule a planned downtime at an appropriate time to reduce the impact on productivity.



(a) Simulation outcomes: interruptions of the process operation are expected due to the failure of both the redundant Bending Press devices and the Camera device.

(b) Actual process execution: the reliability predictions are used to schedule timely interventions on the Bending Press (to prevent the failure of both devices) and on the Camera (minimizing the impact on the process operation).

Figure 7: Simulation outcomes and impact on actual process execution.

5 CONCLUSIONS

This paper presented a data-driven framework for the predictive maintenance of business processes, based on the integration of Digital Twin technology, process mining techniques, and model-driven engineering principles. The proposed method, which exploits a low-code paradigm, enables the automated development and update of reliability-aware DT from event and state logs, supporting both proactive maintenance interventions and real-time reconfiguration.

The feasibility of the framework was demonstrated through a case study involving a steel component production line, where resource failures were successfully predicted and mitigated.

The main limitations of the current work concern the use of synthetic datasets and the prototypal implementation of the framework. Future work will focus on the implementation of a full-fledged tool and its validation in more extensive scenarios with real-world data.

REFERENCES

Abdullahi, I., S. Longo, and M. Samie. 2024. "Towards a distributed digital twin framework for predictive maintenance in industrial internet of things (IIoT)". *Sensors* 24(8):2663.

Aivaliotis, P., K. Georgoulias, and G. Chryssolouris. 2019. "The use of Digital Twin for predictive maintenance in manufacturing". *International Journal of Computer Integrated Manufacturing* 32(11):1067–1080.

Bocciarelli, P., and A. D'Ambrogio. 2023. "A Low-code Approach for Simulation-based Analysis of Process Collaborations". In *Proceedings of the 2023 Winter Simulation Conference*, edited by C. G. Corlu, S. R. Hunter, H. Lam, B. S. Onggo, J. Shortle, and B. Biller, 2530–2541. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Bocciarelli, P., A. D'Ambrogio, A. Mastromattei, E. Paglia, and A. Giglio. 2017. "Business Process Modeling and Simulation: State of the Art and MSaaS Opportunities". In *Proceedings of the Summer Simulation Multi-Conference*, 1–12.

Bocciarelli, P., A. D'Ambrogio, and E. Paglia. 2014. "A Language for Enabling Model-Driven Analysis of Business Processes". In *Proceedings of the 2nd International Conference on Model-Driven Engineering and Software Development*, MODELSWARD'14. Lisbon, Portugal: SciTePress.

Bocciarelli, P., and A. D'Ambrogio. 2024. "Simulation-based Predictive Process Mining with eBPMN: methods, Challenges and Opportunities". In *2024 Annual Modeling and Simulation Conference (ANNSIM)*, 1–14. IEEE.

Bocciarelli, P., and A. D'Ambrogio. 2025. "Data-driven Simulation-based Analysis of Collaborative Business Processes in Distributed Environments". In *2025 Annual Modeling and Simulation Conference (ANNSIM)*. IEEE.

Bocciarelli, P., A. D'Ambrogio, A. Giglio, and E. Paglia. 2019. "BPMN-Based Business Process Modeling and Simulation". In *Proceedings of the 2019 Winter Simulation Conference*, edited by N. Mustafee, K.-H. G. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, and Y.-J. Son, 1439–1453. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc. https://doi.org/10.1109/WSC40007.2019.9004960.

D'Ambrogio, A., E. Paglia, P. Bocciarelli, and A. Giglio. 2016. "Towards performance-oriented perfective evolution of BPMN models". In *6th International Workshop on Model-Driven Approaches for Simulation Engineering*. Pasadena, CA, USA.

Eramo, R., F. Bordeleau, B. Combemale, M. v. d. Brand, M. Wimmer, and A. Wortmann. 2022. "Conceptualizing Digital Twins". *IEEE Software* 39(2):39–46 https://doi.org/10.1109/MS.2021.3130755.

Friederich, J., and S. Lazarova-Molnar. 2022. "Data-driven reliability modeling of smart manufacturing systems using process mining". In *Proceedings of the 2022 Winter Simulation Conference*, edited by B. Feng, G. Pedrielli, Y. Peng, S. Shashaani, E. Song, C. Corlu, L. Lee, E. Chew, T. Roeder, and P. Lendermann, 2534–2545. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Kritzinger, W., M. Karner, G. Traar, J. Henjes, and W. Sihn. 2018. "Digital Twin in manufacturing: A categorical literature review and classification". *Ifac-PapersOnline* 51(11):1016–1022.

Martin, N., B. Depaire, and A. Caris. 2014. "The use of process mining in a business process simulation context: Overview and challenges". In *2014 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, 381–388. IEEE.

Michael, J., L. Cleophas, S. Zschaler, T. Clark, B. Combemale, T. Godfrey, *et al*. 2025. "Model-Driven Engineering for Digital Twins: Opportunities and Challenges". *Systems Engineering*.

Ojbect Management Group 2003. "MDA Guide, Revision 2.0 (ormsc/14-06-01)". https://www.omg.org/cgi-bin/doc?ormsc/14-06-01.pdf, accessed 2022-04-15.

Rossini, R., D. Conzon, G. Prato, C. Pastrone, J. P. C. dos Reis, and G. Gonçalves. 2020. "REPLICA: A Solution for Next Generation IoT and Digital Twin Based Fault Diagnosis and Predictive Maintenance.". *SAM IoT* 2739:55–62.

Singh, M., E. Fuenmayor, E. P. Hinchy, Y. Qiao, N. Murray, and D. Devine. 2021. "Digital twin: Origin to future". *Applied System Innovation* 4(2):36.

Tiftik, M. N., T. G. Erdogan, and A. K. Tarhan. 2022. "A Framework for multi-perspective Process Mining into a BPMN Process Model". *Mathematical Biosciences and Engineering* 19(11):11800–11820.

van Der Aalst, W. 2012. "Process mining: Overview and opportunities". *ACM Transactions on Management Information Systems (TMIS)* 3(2):1–17.

van der Aalst, W. 2016. *Process Mining: Data Science in Action*. 2nd ed. Springer Publishing Company, Incorporated.

van der Aalst, W. M. P., A. H. M. ter Hofstede, and M. Weske. 2003. "Business process management: a survey". In *Proceedings of the 2003 international conference on Business process management*, BPM'03, 1–12. Berlin, Heidelberg: Springer-Verlag.

Van Dinter, R., B. Tekinerdogan, and C. Catal. 2022. "Predictive maintenance using digital twins: A systematic literature review". *Information and Software Technology* 151:107008.

XES Working Group 2023. "IEEE Standard for eXtensible Event Stream (xes) for Achieving Interoperability in Event Logs and Event Streams". *IEEE Std* 1849:1–50.

Zhang, W., D. Yang, and H. Wang. 2019. "Data-driven methods for predictive maintenance of industrial equipment: A survey". *IEEE systems journal* 13(3):2213–2227.

Zonta, T., C. A. Da Costa, R. da Rosa Righi, M. J. de Lima, E. S. Da Trindade, and G. P. Li. 2020. "Predictive maintenance in the Industry 4.0: A systematic literature review". *Computers & Industrial Engineering* 150:106889.

AUTHOR BIOGRAPHIES

PAOLO BOCCIARELLI is a postdoc researcher in the Department of Enterprise Engineering, University of Rome Tor Vergata, Italy. His email address is paolo.bocciarelli@uniroma2.it.

ANDREA D'AMBROGIO is associate professor of systems and software engineering in the Department of Enterprise Engineering, University of Rome Tor Vergata, Italy. His email address is dambro@uniroma2.it.