Reliability Engineering and System Safety 271 (2026) 112242

Contents lists available at ScienceDirect

Reliability Engineering and System Safety

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

Inferring failure processes via causality analysis: from event logs to predictive fault trees

Roberta De Fazio [a,*], Benoit Depaire [b], Stefano Marrone [a], Laura Verde [a]

[a] Dipartimento di Matematica e Fisica, Univ. della Campania-Luigi Vanvitelli, Italy
[b] Faculty of Business Economics, University of Hasselt, Belgium

ARTICLE INFO

PACS:
0000
1111

2000 MSC:
0000
1111

Keywords:
Predictive maintenance
Model-based approaches
Data-driven methods
Fault trees
Process mining
Causality analysis

ABSTRACT

In the current Artificial Intelligence era, the integration of the Industry 4.0 paradigm in real-world settings requires robust and scientific methods and tools. Two concrete aims are the exploitation of large datasets and the guarantee of a proper level of explainability, demanded by critical systems and applications. Focusing on the predictive maintenance problem, this work leverages causality analysis to elicit knowledge about system failure processes. The result is a model expressed according to a newly introduced formalism: the Predictive Fault Trees. This model is enriched by causal relationships inferred from dependability-related event logs. The proposed approach considers both fault-error-failure chains between system components and the impact of environmental variables (e.g., temperature, pressure) on the health status of the components. A proof of concept shows the effectiveness of the methodology, leveraging an event-based simulator.

1. Introduction

In the past decades, the adoption of Artificial Intelligence (AI) in industry has brought about significant changes in processes. AI has enabled the optimisation of production and control processes, and it is significantly contributing to Predictive Maintenance (PdM) spreading. The integration of AI into PdM strategies offers a more effective and flexible approach to the management of industrial equipment, early identifying operational inefficiencies and reducing downtime and maintenance expenses [1]. The availability of a significant amount of public data is limited because fault events occur occasionally, and they could require years and years of systems' operation [2]. In addition, sensors and monitoring systems can sometimes record inaccurate or incomplete data due to malfunction or noisy environments. Low-quality data can affect the accuracy of a Data-Driven (DD) predictive model. Model-Based (MB) approaches — where mathematical and physical models of equipment can be defined starting from explicit knowledge — traditionally guarantee a more explainable level of the analysis [3]. However, these methods suffer from a lack of flexibility in adapting the results of the modelling activities to the reality, captured by data.

The main objective of this paper is to provide a formalism for the integration of MB and DD approaches in PdM supported by a toolchain for the automatic refinement of top-down models via causality-based DD techniques. The methodology assumes the presence of a system log, where both discrete component failure events and continuous variable time series are captured. The model inference approach is based on the discovery of cause-effect relations among system events and environmental conditions. The formal foundation of the proposed approach is based on the work of Kleinberg [4] tailored to the PdM context.

The original contributions of this paper are:

- the introduction of the Predictive Fault Tree (PdFT) formalism, extending the classical Fault Trees (FTs), widespread MB formalism for reliability analysis, able to integrate DD approach, for the definition of system model;
- the extension of existing cause-effect discovery methodologies with Process Mining (PM) concepts in the dependability context;
- the prototyping of a tool, named Causality AnaLYsis for Prediction of System Operation (CALYPSO), supporting the automatic completion of PdFT models from data.

A preliminary version of the PdFT formalism has been published in [5]. In this paper, an enhancement of formalism and its integration into a data-fed improvement approach is presented. The performance of the proposed methodology is evaluated on a synthetic dataset build-up

* Corresponding author.

https://doi.org/10.1016/j.ress.2026.112242

Available online 17 January 2026

List of Acronyms

Abbreviation	Description
AI	Artificial Intelligence
ARM	Association Rule Mining
BDMP	Boolean Logic Driven Markov Processes
BN	Bayesian Network
CALYPSO	Causality AnaLYsis for Prediction of System Operation
CI	Critical Infrastructures
CM	Confusion Matrix
CNC	Computerized Numerical Control
DBN	Dynamic Bayesian Network
DD	Data-Driven
DFT	Dynamic Fault Tree
DRL	Deep Reinforcement Learning
DSE	Dependability Simulation Engine
DSS	Decision Support System
DT	Decision Tree
FMECA	Failure Mode, Effects, and Criticality Analysis
FN	False Negative
FP	False Positive
FT	Fault Tree
FTA	Fault Tree Analysis
ISM	Interpretive Structural Modeling
IoT	Internet of Things
KNN	K-Nearest Neighbor
LLM	Large Language Model
LSTM	Long-Short Term Memory
MB	Model-Based
ML	Machine Learning
MTBF	Mean Time Between Failures
MTTF	Mean Time to Failure
PM	Process Mining
PN	Petri Net
PTL	Probabilistic Temporal Logic
PdFT	Predictive Fault Tree
PdM	Predictive Maintenance
PoC	Proof of Concept
RCA	Root Cause Analysis
RF	Random Forest
RUL	Remaining Useful Life
SHyFTA	Stochastic Hybrid Fault Tree Automaton
SM	State Machine
SWRL	Semantic Web Rule Language
TN	True Negative
TP	True Positive

through a Python-based simulator, described in [6] and extended in this [7]. This choice provides a ground truth for comparing the results obtained with the expected one. CALYPSO tool, also, provides a prototype for the inference approach [8].

The rest of the paper is structured as follows: Section 2 describes related scientific work. Section 3 recalls some background information, specifically related to FTs, causality analysis and PM concepts. Section 4 introduces the PdFT formalism while Section 5 describes the core approach of the paper, i.e., the causality-based method for PdFT model inference. Section 6 demonstrates the approach and the related toolchain on an example. Section 7 discuss the possible impact of the proposed work in both academia and industrial settings, highlighting limitations and strengths. Section 8 ends the paper, drawing future research lines.

2. Related work

The main objective of the proposed work is to define a methodology for providing insights from DD techniques and leverage them in MB approach, assessing PdM tasks in critical systems.

2.1. Hybrid approaches

Scientific community addresses in hybrid MB and DD approaches as a key to improve reliability understanding and engineering capabilities in large and complex systems [9]. Despite being a very promising field of research, there are few works in the literature providing these hybrid strategies [10]. The definition of a combined approach requires a well-structured architecture in which all the parties can interact. Luo et. al, address this challenge framing the hybrid approach in a Digital Twin architecture for Remaining Useful Life (RUL) estimation [11]. This provides a playground for the combination of physical model-based, statistical and data-driven approaches, designed for the Computerized Numerical Control (CNC) machine tool outperforming the state of the art. Also in [12], a Decision Support System (DSS) for the design of a risk model is proposed. The system relies on the physics-based model for feature extraction, and DD approach providing the failure mode. Arena et al. [13], proposes a Decision Tree (DT)-based approach defining a DSS to improve the correct maintenance policy of a gearbox for roasting oilseeds. The results obtained confirmed a potential cost saving in maintenance actions compared to corrective maintenance.

In Wang et al. [14], Long-Short Term Memorys (LSTMs) are used in the domain of industrial robots PdM. The methodology relies on the prediction of machine running states, on the basis of historical data and knowledge of the actual state. K-Nearest Neighbors (KNNs)/LSTMs are combined with Knowledge Graphs for modelling the domain and comparing the output of the DD approach. Cao et al. [15], propose a methodology based on on ontology-based formalism combining results obtained from Semantic Web Rule Language (SWRL) and from the rules described by experts, for the prediction of failure occurrences. A causal aggregation loss is, instead, designed to separate the non-causal and causal factors in [16]. Five cross-machine vibrational fault diagnosis cases and three cross-environment acoustical anomaly detection cases were adopted in the experimental phase to evaluate the performance.

2.2. Model-based and data-driven aspects in system dependability

FT is a formalism for system failure analysis, based on a deductive approach and a graphical representation that improves usability and expressiveness. FTs are an industrial standard with impact on predictive maintenance [17,18]. Their success comes from object-oriented modelling, integration of domain knowledge, and accessibility to non-experts

In Gao et al. [19], Dynamic Fault Tree (DFT) is applied to Communication-Based Train Control using SimFIA for safety and reliability analysis. In Mandelli et al. [20], FT is integrated with sensor data, modelling component states. Ruijters' Fault Maintenance Tree [21] adds maintenance policies, inspections, and cost modelling, but focuses on maintenance policy evaluation rather than predictive modelling. Our work extends FTs Internet of Things (IoT) integration, introducing a dual-view component status propagation based on causal inference. Bayesian Networks (BNs) model causal structures without temporal data [4], relying on assumptions such as Markov condition, faithfulness, and causal sufficiency [4,22]. The PdFT formalism bridges FT and BN, separating structure from inference while retaining conditional probability evaluation. BNs [23,24], have been exteded by Dynamic Bayesian Networks (DBNs). However, the latter lacks flexibility [25].

Boolean Logic Driven Markov Processes (BDMP) integrates FT with Markov Chains for component interdependence [26]. Stochastic Hybrid Fault Tree Automaton (SHyFTA) combines DFT with Stochastic Hybrid

Automata to model multi-state systems under dynamic conditions [27]. Unlike BDMP and SHyFTA, PdFT models both component interactions and internal health, replaces gates with logical predicates, using Probabilistic Temporal Logic (PTL) for greater expressiveness. It also incorporates IoT sensors to link external behaviour to system states.

2.3. Process mining and causality analysis in system dependability

PM is an emerging field with impact in industry, enhancing analytics in control systems [28]. It discovers real processes by extracting models from data, often translated into Petri Nets (PNs) or State Machines (SMs) [29,30]. In Ruschel et al. [31], PM supports BN definition by extracting process models from data and integrating domain knowledge. in [32], an unsupervised method combines Association Rule Mining (ARM) and Large Language Model (LLM) to analyse textual maintenance data. While ARM extracts correlations [33], our methodology, built on Suppes and Kleinberg theories [4,34], leverages a probabilistic definition of causality.

Nadim et al. [35] integrate interpretable Machine Learning (ML) and PM by using DTs to detect sensor patterns and construct causal PNs, reducing reliance on expert knowledge. Their follow-up [36] uses Deep Reinforcement Learning (DRL) for supervisory control, combining simulation, causality analysis, and reinforcement learning to adapt policies from process data. However, these methods rely mainly on DD approaches, with domain knowledge added only in validation. Moreover, also in this cases, DTs rules capture correlations [33,37].

In [38], authors combine PM and DD approaches for Digital Twins. Offline, a process model is extracted and used to train a classifier that is used in the online phase. Van Houdt et al. [39,40] extend Suppes' and Kleinberg's causality theory to PM, quantifying causality between process activities. Our methodology adapts this framework to Critical Infrastructuress (CIs), PdM, and IoT, translating discovered relations into PdFT to better integrate domain knowledge.

2.4. Dependability related model completion

Chiacchio et al. propose the SHyFTA formalism [41], extending DFT with Hybrid Basic Events that couple physical process evolution with stochastic failure behaviour. Results highlight SHyFTA's ability to deliver accurate, dynamic dependability assessments for complex systems.

Arena et al. [42] extend the framework with a "Maintenance Box" to simulate corrective, preventive, and condition-based policies. Applied to a steam turbine benchmark, preventive strategies achieved higher availability and lower failure frequency over the mission time.

3. Background

This section provides some useful basic notions, allowing a deep understanding of the contribution of the paper. Causality theory represents a fundamental point for understanding the relationships between events, which can be effectively modelled using PTL. This connection is further exploited in Kleinberg's work, which introduces a metric for evaluating the causal significance between causes and effects [4]. She, according to Suppes' Theory [34], extended the notion of *prima facie* causes with the concept of PTL statement, providing a probabilistic definition of causal relations.

Let us introduce some notations: a and f be two generic events, assuming that a is a possible cause for an effect f; $\mathbf{X} = \{x_1, \ldots, x_n\}$ be the set of all the other possible causes for f; r, s, r', s' generic points in time; δ a generic upper bound of time unit and p a probability. Eq. 1 expresses the temporal condition requiring that the cause must occur before the effect.

$$a \rightsquigarrow_{\geq p}^{\geq 1, \leq \delta} f \tag{1}$$

Considering an infinite elapsed times $\delta = \infty$, we have that:

Table 1
Literature review of PdM approaches.

Ref.	Year	Domain	Main Goal	Models	Performance
Model-based Approaches					
[19]	2020	Communication-Based Train Control (Railway)	Reliability analysis	DFT	–
[21]	2018	Pneumatic compressor (Railway)	Maintenance Policy	Fault Maintenance Tree	–
[26]	2003	BUSBAR (Elettrica system)	System dependability	BDMP	–
Data-driven Approaches					
[43]	2024	Accidents (Railway)	Risk Factors	BN	Validation 72.8%
[32]	2024	Facility Management (Industrial)	Rules for decision-making	ARM + LLM	Similarity Rate 0.65
[38]	2024	Water Distribution (Industrial Systems)	Process Model and Anomaly Detection	PM + ML	Fitness 93% - F1-score 74.3%
[39]	2023	Road Traffic Fines Management (Business Processes)	Root Cause Analysis (RCA)	PM + causality analysis	–
[35]	2023	Kraft pulp mill (Industrial Systems)	Process model	PM + DT	Fitness 98.66%
[36]	2023	Paper and pulp mill (Industrial Systems)	Control policy	PM + DRL	Improvement 5.02%-15.6%
[44]	2016	Track circuit (Railway)	Latency Time and Repair Time	BN	Precision 85.38–89.76%
Hybrid Approaches					
[20]	2024	Circulating Water Pumps (Industrial Systems)	Diagnostic and Prognostic assessments and Health status	FT + ML	–
[14]	2023	Industrial robots (Automotive)	Machine state	LSTM, KNN, domain graphs	Accuracy 90.9%
[12]	2022	Turbomachinery	Probability of failure	Physics-based + Random Forest (RF)	Accuracy 83%
[13]	2022	Gearbox (roasting oilseeds)	Maintenance policy	Failure Mode, Effects, and Criticality Analysis (FMECA) + Decision Tree	Accuracy 75.4–91.6%
[15]	2022	Semiconductor (Manufacturing)	Failure time and constraint	Symbolic AI + ontologies	Precision 85.38- 89.76%
[42]	2021	Steam turbine (Industrial system)	System reliability	SHyFTA	–
[11]	2020	CNC machine tool (Mechanics)	RUL	Physical degradation model + DD	Error ratio 6.27%
[45]	2020	Goods transportation (Railway)	Risk Factors	Interpretive Structural Modeling (ISM) + BN	–
[41]	2020	Vessel feed pump (Industrial system)	System reliability	SHyFTA	–
[31]	2018	CNC (Automotive)	System availability and Maintenance costs	PM + BN + mathematical models	Improvement 1.40%–2.41%
[27]	2016	Decanting Unit (Chemical process plant)	System reliability	SHyFTA: DFT + Stochastic Hybrid Automata	Error: 0.012%

Definition 1. Given a and f two events, a is *prima facie* cause of f if there is a probability p such that:

1. $F_{>0}^{\leq\infty} a$
2. $a \rightsquigarrow_{\geq p}^{\geq 1, \leq\infty} f$
3. $F_{<p}^{\leq\infty} f$

In other words, Kleinberg stated that to be *prima facie* for an effect f, (1) the probability of a has to be different from zero; (2) a has a no null probability to occur *before* f (*time priority condition*); (3) the marginal probability of f has to be lower than the conditional one (i.e., p). Kleinberg reported this definition for identifying *prima facie* causes of an effect, and she proposed a metric for measuring the causal significance. To establish the significance of a possible cause, the impact of the other *prima facie* causes must be considered. She also discarded *"factors that are independent or negatively correlated with f or which never co-occur with a and $\neg a$"*. The significance of a cause a for an effect f, considering the influence of another cause x, is quantified by the Eq. (2).

$$\varepsilon_x(a, f) = \mathbf{P}(f|a \wedge x) - \mathbf{P}(f|\neg a \wedge x) \tag{2}$$

Then, the Eq. (3) computes, on average, how much a increase the probability of f considering the impact of all the possible other causes $\mathbf{X} = \{x_1, \ldots, x_n\}$, as reported in the Eq. (3)

$$\varepsilon_{avg}(a, f) = \frac{\sum_{x \in \mathbf{X}} \mathbf{P}(f|a \wedge x) - \mathbf{P}(f|\neg a \wedge x)}{|\mathbf{X} - a|} \tag{3}$$

3.1. Addendum to Kleinberg's work

Let us define the concept of *non-exclusive causes* as in Eq. (2)

Definition 2. Let a and x be two prima facie causes for f, Defination 1 implies that:

- $a \rightsquigarrow^{\geq r, \leq s} f$
- $x \rightsquigarrow^{\geq r', \leq s'} f$

They are *non-excluding causes* for the effect f if $[r, s] \cap [r', s'] \neq \emptyset$ and f occurs in that intersection.

As Kleinberg stated, omitting the subscripts for ease *"$a \wedge x$ refers to a and x being true such that f could be caused in appropriate intervals"*. However, there are two cases in which this theory needs further explanation. The first case happens when for all x *prima facie* cause of f, $a \wedge x = \emptyset$; in other words, a — *prima facie* cause of f — has not any *non-excluding causes* among the other *prima facie* causes of f. This is the case when all the other possible *prima facie* causes of f always occur in time windows not intersecting any of the ones associated with the occurrence of a (see Eq. (4) as a refinement of Eq. (3)).

$$\varepsilon_{avg}(a, f) = \mathbf{P}(f|a) - \mathbf{P}(f|\neg a) \tag{4}$$

The second case occurs when there is at least one x — *prima facie* cause of f — such that $a \wedge x \neq \emptyset$ but $\neg a \wedge x = \emptyset$. In this case, Eq. 2 leads to an undetermined situation (see Eq. (5)).

$$\varepsilon_x(a, f) = \mathbf{P}(f|a \wedge x) - \mathbf{P}(f|\neg a \wedge x) = \mathbf{P}(f|a \wedge x) - \mathbf{P}(f|\emptyset) \tag{5}$$

where $\mathbf{P}(f|\emptyset)$ it is mathematically not defined. Hence, Defination 3 refines the definition of *non-excluding causes* reported in Eq. (2), mitigating the problem induced by Kleinberg's discussion about "factors that are independent or negatively correlated".

Definition 3. Let a, x be two *prima facie* causes for the effect f, a and f are defined *non-concordant causes* if and only if $\neg a \wedge x \neq \emptyset$

In the rest of this paper, two causes x and a are required to be *non-excluding* and *non-concordant* to compute the Eq. (2).

4. Predictive fault trees

The idea under the definition of PdFT is to combine FTs-based formalism with some aspects involving causality analysis. The proposed formalism is responsible for modelling complex and dynamic behaviour, providing a double view of the system — inside the components and outside within their interaction — and analysing the maintenance actions and their impact. With respect to a previous work [5], here a new version of PdFT is introduced.

4.1. PdFT formalism

From a theoretical point of view, a PdFT model is represented by a tuple: $< C, D, \mathcal{E}, \mu >$:

- $C = \{c_1, c_2, \ldots, c_n\}$, a set of *components*, describing the system;
- D, a set of the *dynamics*, which model external processes — such as environmental changes, variable conditions, and systematic or random external events — influencing the behaviour of the components;
- $\mathcal{E}$, a set of *events*, which model the relationships between components;
- the *evaluation function* μ that computes the global state of the system given the values measured by dynamics and propagated throughout the events.

One of the main innovation in PdFT is the introduction of the dynamics. More formally, a dynamic is a real function over time $d \in D \mid d : \mathbb{R} \longrightarrow \mathbb{R}$. The set of all the possible values assumed by all the dynamics is $\mathcal{V} = \bigcup_{d \in D} \mathcal{V}_d$ where $\mathcal{V}_d = d(\mathbb{R}) \subseteq \mathbb{R}$. The dynamic conditions, instead, are boolean predicates defined on the values recorded by the dynamics function: each condition models how a component reacts to those values. The relationship between components and dynamics is described by the functional $\theta : C \times D \rightarrow \{true, false\}^{\mathbb{R}}$. It assigns to the couple (c, d) a function $\mathcal{F}_{c,d} : \mathcal{V}_d \subseteq \mathbb{R} \rightarrow \{true, false\}$. The latter models the behaviour of the component c concerning the value measured by the dynamic d: $\mathcal{F}_{c,d}$ is a boolean expression computed on y, the value recorded by d at a certain point in time t (i.e., $y = d(t)$). If the recorded value y satisfies the expression, $\mathcal{F}_{c,d}(y)$ returns *true* otherwise it returns $false$. In the present work, only a fixed group of boolean expressions are considered, as in Eq. (6): the ones based on the definition of a threshold. This threshold, $\theta_T(c, d) \in \mathbb{R}$ is a real number, fixed for each couple (c, d), and expresses the upper bound for the normal functioning of the component c under the condition recorded by the dynamic d.

$$\mathcal{F}_{c,d}(y) = y \geq \theta_T(c, d) \tag{6}$$

The PdFT provides an external view of the components' relationships and the interaction between them and the external environment, introducing new elements: let $P^I = \bigcup_{c_i \in C} P_i^I$ be the union of all the *input ports* and $P^O = \bigcup_{c_i \in C} p_i$ the union of all the *output ports*, where P_i^I is the set of *input ports* for the component c_i and p_i its single *output port*. The latter describes how a component propagates inner changes to the outside, while the former is involved in the representation of how a component reacts to external changes.

The connections between the components are modelled by a set of **events** $\mathcal{E} \subseteq P^O \times P^I$ which relate the output port of a component to the input port of another. For the sake of clarity, let us introduce the running example in Fig. 1a.[1] Let C_0 be the top component, the train, and let C_1 be the main engine, while C_2 is the spare engine, they are all depicted using squared boxes. Each component holds a single output port, depicted as little squared boxes on the top of the component, connected to input ports by arrows, representing the events. The dynamic $f(t)$ represents the temperature sensor. It is depicted with an elliptical shape. The inner behaviour of the components is hidden at this level, since this layer contributes to a top-down view of the system.

[1] For a detailed description of the running example, refer to A.

Running Example
Considering the example depicted in Fig. 1a:

- $C = \{C_0, C_1, C_2\}$
- $D = \{d(T) = temperature\}$
- $\mathcal{V}_d = (24, 71)$

The system is based on three components and one dynamic. The dynamic d models how the temperature changes in time. The values assumed by this function range between the 24° and over 70°. The temperature affects the behaviour of the two engines C_1 and C_2, leading to a failure when it overcomes 70°. Due to this, the functional θ is defined as follows:

- $\theta_T(C_1, d) = \theta_T(C_2, d) = 70$
- $\theta(C_1, d) = \mathcal{F}_{C_1,d}(y) = y \geq \theta_T(C_1, d)$
- $\theta(C_2, d) = \mathcal{F}_{C_2,d}(y) = y \geq \theta_T(C_2, d)$

The port sets are:

- $P^O = \{p_0, p_1, p_2\}$
- $P^I = \{p_{0,1}, p_{0,2}, p_{2,1}\}$

This inter-component view is completed by an inner definition of the behaviour of each component, which is provided with the introduction of some notions. Each component $c_i \in C$ is a tuple $< S^i, \pi^i, T^i, p_i, P_i^I >$:

- a finite set of **states**, S^i;
- a **state priority** function assigning a priority to each state of the component, $\pi^i : S^i \to \mathbb{N}$[2], which is represented by a natural number where $max(S^i) := s_j \in S^i$ such that $\forall s_k \in S^i, s_k \neq s_j \implies \pi^i(s_k) \leq \pi^i(s_j)$ is defined *critical state* for the component c_i (i.e. the state with the highest priority);
- a set of oriented **transitions** $T^i \subseteq S^i \times S^i$ as a relation over the Cartesian product of components' states, where $t = (s_k, \max(S^i))$ is defined *critical transition*, for all $s_k \in S^i$;
- p_i is the **output port** of the component, that is a variable, whose values range in the set $\mathcal{V}(\mathcal{P}^{\mathcal{O}}) := \{True, False, Neutral\}$. The port assumes the *Neutral* value if the inner behaviour of the component does not impact outside. The *True* value, instead, is assumed when it switches to the critical state. On the contrary, the *False* value is acquired by the output port when the component switches from the critical state to another one with lower priority;
- P_i^I is the set of **input ports** (which could also be empty), whose values range in $\mathcal{V}(\mathcal{P}^{\mathcal{I}}) := \{True, False, Neutral\}$, relying on the value assumed by the output port of the connected components.

The intra-component view, modelled on the proposed running example is depicted in Fig. 1b. The states are represented by circles, while transitions are represented by arrows. It is possible to distinguish the transitions by the events because the former links two states, while the latter links two ports.

The inner condition of a component, due to a transition from one state to another, can impact the behaviour of other components. Before explaining this concept in detail, let us consider:

- $\mathbb{P}(P_i^I \cup D)$, the set of the boolean predicates over both the dynamic values and input port values;
- $T = \bigcup_{c_i \in C} T^i$, the set of all the possible transitions;
- $S = \bigcup_{c_i \in C} S^i$, the set of all the possible states.

Each component's transition is also characterised by the following functions:

- **trigger**: $\tau : C \times T \longrightarrow \mathbb{P}(P^I \cup D)$ associate to each transition a boolean predicate defined over ports as well as dynamics values;
- **action**: $\alpha : P^O \times T \to \mathcal{V}(P^O)$, representing the value assigned to the output port of a component when the transition is fired.

Each transition models an inner behaviour of the component, which reacts to some external internal degradation. These definitions can also be adopted for modelling the "intermediate" behaviour of the component that gradually moves from normal operating conditions to an unhealthy state. Let us introduce some properties of a transition, which will enhance the definition of *impacting transitions*:

- given $t \in T$ a transition on a component $c \in C$, it is defined *impacting* if $\alpha(p_c, t) \neq Neutral$. If this property is not held, the transition is defined *dormant* for the component c;
- given $t = (s_i, s_j) \in T$, the *inverse* of t is a transition such that the states involved are the same of t but the direction is the opposite: $\bar{t} = (s_j, s_i)$.

Summing up, from the previously introduced definitions, it is possible to state that: the only two kinds of transitions, in a component $c_i \in C$ that could propagate their state outside are the critical ones and their inverse.[3] If t is an *impacting transition*, moving the component c in the state with the highest priority, then the α *function* turns value *True*; while the same function, computed on its $\bar{t}$ inverse, will turn the value *False*[4]. The function α assigns to the *impacting* transitions, a value of False or True according to the direction of the arc that connects the initial state to the final state.[5]

The function, described in Eq. (7), evaluates the predicate value.

$$\mu : \mathbb{P}(P_i^I \cup D)^{C \times T} \times \prod_{i=1}^{n} V(P^I) \times \prod_{d \in D} V_d \to \{true, false\} \tag{7}$$

The evaluation function works on the set of the trigger functions (i.e., $\mathbb{P}(P_i^I \cup D)^{C \times T}$) and on all the value sets of the component's output ports (i.e., $V(P^O)$) as well as on the value sets of the dynamics (i.e., V_d). Its implementation relies on the substitution of each port with its current value (at T time) and of each dynamic with its value (at T time, also). Eq. (8) represents such a mechanism.

$$\mu(\tau(c_i, t), \bar{p}, \bar{d}) = \tau(c_i, t)\Big|_{\substack{d=d(T)\ \forall d \in \bar{d}, \\ p_{i,j}=\beta(p_{i,j})\ \forall p_{i,j} \in \bar{p}}} \tag{8}$$

where $\bar{p}_i$ is the sequence of all the input ports and $\bar{d}$ is the sequence of all the system dynamics. $d(T)$ is the value of the dynamic d at the time T, and $\beta(p_{i,j})$ is the (boolean) value of $p_{i,j}$.

One of the possible scenarios described by the formalism has been depicted in Fig. 2, which puts together the inter and intra component views to model a specific behaviour of the system. In this case, the main engine C_1 is in the down state, due to natural degradation, which triggered two events, propagating the effects on component C_2 and C_0. C_2 moved from its "dormant" condition to the up state, to ensure the continuity of the service provided. C_0 moved to "failing" state, which means that the train is still able to function, but it needs some maintenance action to avoid failures. The complete scenario, described using the PdFT formalism, is reported in Appendix B.4.

5. The model inference methodology

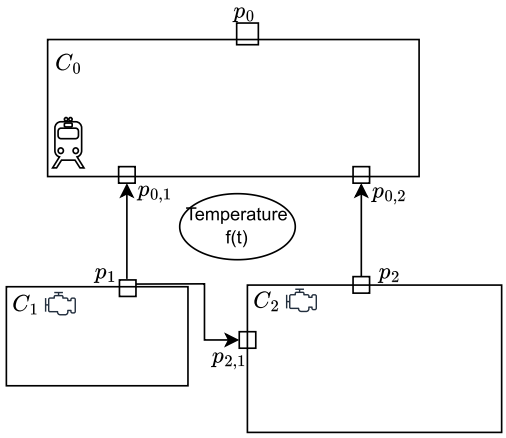
PdFT models can be built according a-priori knowledge as domain experts' knowledge, the structure of the systems, requirement specification documents. Some details could be hidden or unpredictable since, as

[2] where $\pi : C \times S \to \mathbb{N}$ and for all $c_i \in C$, $\pi(c_i, s_j) = \pi^i(s_j)$.

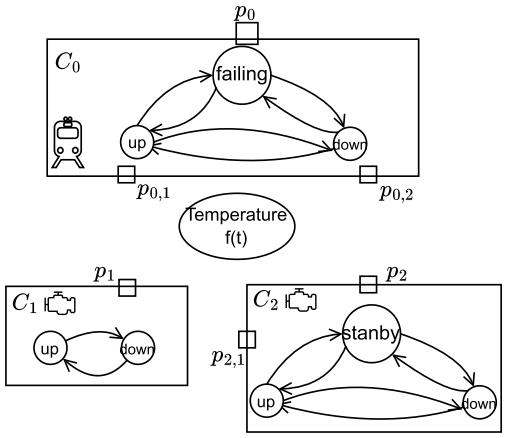
[3] More formally: $t = (s_k, \max(S^i))$ and $\bar{t} = (max(S^i), s_k)$ for all $s_k \in S^i \setminus \{max(S^i)\}$.

[4] More formally: If $t = (s_k, \max(S^i))$ is an *impacting transition* for a component c, i.e. $\alpha(p_c, t) \neq Neutral$ then $\alpha(p_c, t) = True$ and $\alpha(p_c, \bar{t}) = False$

[5] Some clarification about the meaning of these values are provided in Appendix B.2



(a) Graphical representation of inter-component view on a running example.



(b) Graphical representation of intra-component view on a running example.

Fig. 1. Graphical representation of PdFT model.

Running Example

The state sets are:

- $S^0 = \{up, down, failing\}$
- $S^1 = \{up, down\}$.
- $S^2 = \{up, standby, down\}$ The following table represents the value assumed by the priority function.

Component	State			
	up	**failing**	**down**	**standby**
C_0	0	1	2	—
C_1	0	—	1	—
C_2	1	—	2	0

So, given the definition of the priority function, the critical states, for each component are:

- $max(S^0) = max(S^1) = max(S^2) = down$

The transition sets:

- $T^{C_0} = \{(up, down), (down, up), (up, failing), (failing, up), (down, failing), (failing, down)\}$
- $T^{C_1} = \{(up, down), (down, up)\}$
- $T^{C_2} = \{(up, down), (down, up), (up, standby), (standby, up), (down, standby), (standby, down)\}$ and the critical transitions are:
- $(up, down), (failing, down)$ for C_0
- $(up, down)$ for C_1
- $(up, down), (standby, down)$ for C_2

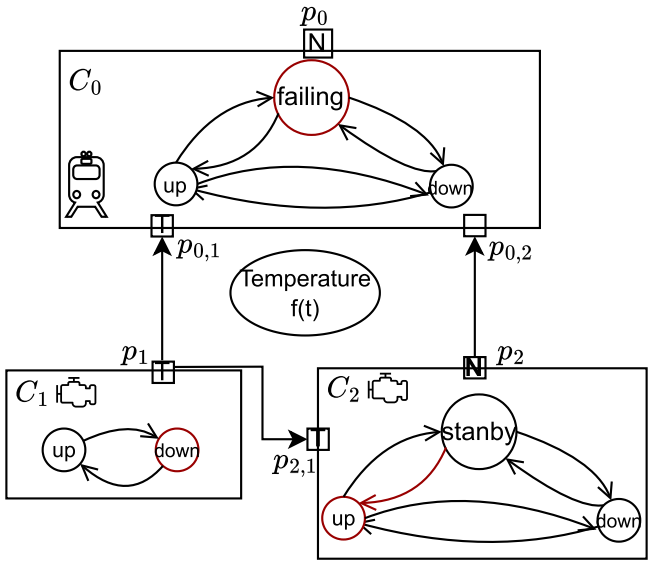


Fig. 2. A possible scenario for the running example.

an example, some conditions can depend on the environment the system operates.

The methodology here introduced leverages causality analysis and Kleinberg's causality metrics, defined in the PM context, to infer a PdFT model from data, making it more adherent to the system's real behaviour and operating conditions. The approach proposed is inspired by the work of Van Houdt et al., in which they provided an analysis tool that allows the estimation of the potential causes of a given effect in PM context [39]. An overall description of the workflow is shown in Fig. 3.

The figure highlights three steps.

As the model discovery is based on PM concepts, it is worthy to be clear on the meaning of a "process instance" in the context of this work. Hence, due to the centrality of the presence of a *CaseID* in PM approaches, some issues could be present in real-world cases: multiple definitions of *CaseIDs* for the same process, particularly when considered in an unconventional context like PdM [46]; coarse-grained *CaseIDs*, when data does not separate between different process instances in a clear way [47]. To avoid ambiguities, here a process instance is defined as the set of events that starts from the normal operating functioning of the system, proceeds through a fault-error-failure series of events, continues with maintenance action performance, and ends with the system recovery.

5.1. Problem formalisation

This methodology step aims to adapt such a setting to the causality analysis approach. Considering the presence of an IoT sensing network able to monitor the system under the study, two are types of data are acquired: time-series records, where continuous system and environmental variables are captured, and event logs, storing meaningful (discrete) events, e.g., that are composed of the loss of communications, errors, hw/sw faults, etc. In this context, the notions of hypotheses and effects are introduced. The observed effects encompass all possible com-

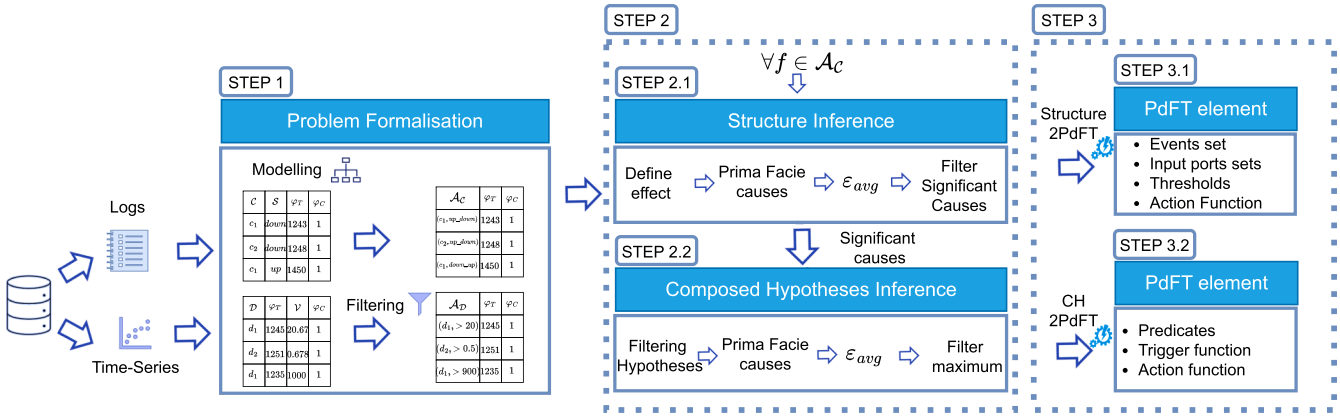


Fig. 3. Methodology workflow.

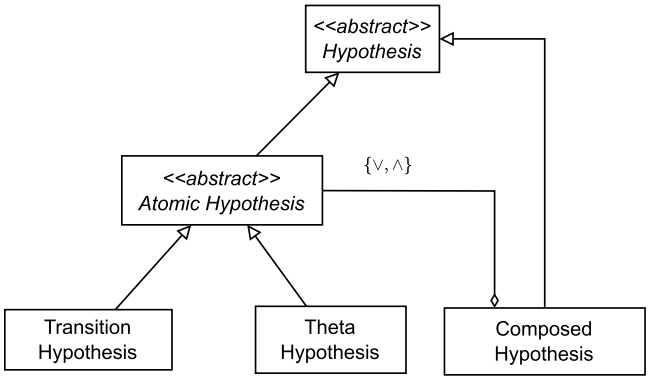


Fig. 4. Hypotheses definition metamodel.

ponents' transitions, while the hypotheses represent events that can be either atomic or composed. Atomic hypotheses are further classified into Transition Hypotheses and Theta Hypotheses: *Transition Hypotheses* denote state changes within a component, whereas *Theta Hypotheses* represent the recording of a value that exceeds a threshold defined by the dynamic on a component. *Composed Hypotheses* are formulated, considering all the possible combinations among Atomic Hypotheses using "AND" and "OR" operators.

Fig. 4 depicts the hypothesis definition metamodel, which is formalised by the following definitions.

Definition 4. Let C be the set of the components and T the set of the transitions, where $T^c \subseteq T$ is the subset of transitions involving a component $c \in C$, the components' transition set is:

$$\mathcal{A}_C = \{(c,t) \mid c \in C, t \in T^c\}$$

The components' transition set is composed of couples (c,t) that define the transition t observed on the component c.

Definition 5. Let D be the set of the dynamics and $\bigcup_{c\in C, d\in D} \theta_T(c,d)$ be the set of the thresholds that connect value recorded by dynamics and components, a Theta Hypotheses set is:

$$\mathcal{A}_D = \{(d,v) \mid d \in \mathcal{D}, v \in \bigcup_{c\in C} \theta_T(c,d) \in \mathcal{V}_d\}$$

The set of Theta Hypotheses is composed of couples (d,v) where d is a dynamic and v is one of its recorded values, that has been associated with a threshold on a component c.

Definition 6. Let $\mathcal{A}_C$ be the Transition Hypotheses set and $\mathcal{A}_D$ be the Theta Hypotheses set, the Composed Hypotheses set is defined by hypotheses $a = a_{i_1} * \cdots * a_{i_j}$ are expression defined by logical operators computed on both Transition and Theta Hypotheses.

Running Example

For clarifying the concepts of occurrences and related functions, an excerpt of the dataset, associated with the running example introduced in Section 4, is provided in the Appendix A, Table A.10. Each occurrence is represented by a row in the dataset. Considering for example the occurrences number 1737 and number 1761, the functions assign:

- $\varphi_R(1737) = sensor$; $\varphi_R(1761) = C_2$
- $\varphi_M(1737) = 27.25944$; $\varphi_M(1761) = standby$
- $\varphi_T(1737) = 27/04/2021\text{-}17{:}45{:}00$; $\varphi_T(1761) = 27/04/2021\text{-}18{:}30{:}00$
- $\varphi_{CI}(1737) = 1$; $\varphi_{CI}(1761) = 1$

The *Composed Hypotheses* are obtained by combining *Transition* and *Theta Hypotheses* by means of logical operators.[6] According to these definitions, the set of hypotheses is $\mathcal{H} = \mathcal{A} \cup \bar{\mathcal{A}}$, where the subset of atomic hypotheses is $\mathcal{A} = \mathcal{A}_D \cup \mathcal{A}_C$, and the set of the effect is $\mathcal{A}_C$. The goal is to determine whether the causes of a component transition arise from transitions in other components or from dynamic variable values. To support this analysis, a preliminary preprocessing step performs a twofold transformation: component states in the event logs are converted into transitions, while continuous time-series variables are discretised into event-like representations. The set $\mathcal{O} = \{o_1, \ldots, o_l\}$ defines possible system occurrences (i.e. records of the dataset). Each occurrence $o \in \mathcal{O}$ is a tuple $o = (\varphi_R(o), \varphi_M(o), \varphi_T(o), \varphi_{CI}(o))$ where:

- $\varphi_R : \mathcal{O} \to C \cup D$ which assigns to each occurrence the component/-dynamic involved
- $\varphi_M : \mathcal{O} \to S \cup \mathcal{V}$ which assigns to each occurrence the state/value recorded
- $\varphi_T : \mathcal{O} \to \mathbb{R}$ which assigns to each occurrence the timestamp in which it has been performed
- $\varphi_{CI} : \mathcal{O} \to \mathbb{N}$ which assigns to each occurrence the *CaseID*

Definition 7. A Transition Hypothesis $a = (c,t)$ where $t = (s_i, s_j)$ is observed on a component c in a process instance $N \in \varphi_{CI}(\mathcal{O})$, if there are two consecutive occurrences, involving the component c, where the first records the state s_i and the subsequent one records the state s_j, with no other occurrence involving the same component in between.[7]

[6] For a more formal definition, it is possible to refer to Defination 14 in the Appendix C

[7] Refers to C Defination 15

Running Example

To better clarify these concepts, an example is illustrated here, referring to the complete dataset shown in Table A.10 in the Appendix A. The sets of the Atomic Hypothesis are:

- $\mathcal{A}_C = \{(C_2, \texttt{standby\_down}), (C_2, \texttt{down\_standby}), (C_0, \texttt{up\_failing}), (C_0, \texttt{failing\_up}), (C_0, \texttt{up\_down}), (C_0, \texttt{down\_failing}), (C_1, \texttt{up\_down})\}$
- $\mathcal{A}_D = \{(\texttt{sensor}, 70)\}$

Considering the Transition Hypothesis $a_1 = (C_2, \texttt{standby\_down})$ it is observed in the CaseID number $1, 10$ since:

- The occurrences 1001 and 6021 record the state "standby" for C_2; the occurrences 1739 and 6791 record the state "down" for C_2;
- There is no other occurrence involving C_2 in between.

It is possible to define two new occurrences belonging to $\bar{\mathcal{O}}_C$:

- $(a_1, \varphi_T(1739), \varphi_{CI}(1739)) = ((C_2, \texttt{standby\_down}), 27/04/2021\ 17{:}50{:}00, 1)$
- $(a_1, \varphi_T(6791), \varphi_{CI}(6791)) = ((C_2, \texttt{standby\_down}), 29/06/2021\ 07{:}34{:}30, 10)$

Considering the Theta Hypothesis $\bar{a} = (d, 70)$ is observed in the CaseID number 10 since:

- The occurrence 6789 records a value $\varphi_M(6789) = 70.00023$ such that $\mathcal{F}_{C_1,d}(\varphi_M(6789)) = (\varphi_M(6789) > 70) = true$ and also $\mathcal{F}_{C_2,d}(\varphi_M(6789)) = (\varphi_M(6789) > 70) = true$.

It is possible to define a new occurrence belonging to $\bar{\mathcal{O}}_D$:

- $(\bar{a}, \varphi_T(6789), \varphi_{CI}(6789)) = \big((\texttt{sensor}, 70), 29/06/2021\ 07:34:00, 10\big)$

Definition 8. A Theta Hypothesis $a = (d, \theta_T(c, d))$ triggered by a dynamic d on a component c is observed in a process instance $N \in \varphi_{CI}(\mathcal{O})$, if there is at least one occurrence $o \in \mathcal{O}$, which involves the dynamic d and the value recorded satisfied the expression $\mathcal{F}_{c,d}$[8].

From this point onward, the set of occurrences considered is the union $\bar{\mathcal{O}}_C \cup \bar{\mathcal{O}}_D$, respectively representing the occurrences of Transition and Theta Hypotheses. For simplicity, $\mathcal{O}$ is redefined as $\mathcal{O} := \bar{\mathcal{O}}_C \cup \bar{\mathcal{O}}_D$, with the function φ_T and φ_{CI} adjusted to this new set. Additionally, a new function is introduced $\varphi_{\mathcal{A}} : \mathcal{O} \to \mathcal{A}$, assigning the corresponding hypothesis to each occurrence.

Finally, the definition of Composed Hypothesis is introduced.

Definition 9. A composed hypothesis $a = a_{i_1} \wedge \cdots \wedge a_{i_j} \in \bar{\mathcal{A}}$ is observed in a process instance $N \in \varphi_{CI}(\mathcal{O})$, if **all** the atomic hypotheses a_{i_k} are observed in the same process instance and its observation $\bar{o}$ is associated with the occurrence of the last atomic hypothesis observed.[9]

Definition 10. A composed hypothesis $a = a_{i_1} \vee \cdots \vee a_{i_j} \in \bar{\mathcal{A}}$ is observed in a process instance $N \in \varphi_{CI}(\mathcal{O})$, if **at least one** atomic hypothesis a_{i_k} is observed in that process instance and its observation $\bar{o}$ is associated with the occurrence of the first atomic hypothesis observed.[10]

[8] Refers to Appendix C Defination 16
[9] Refers to Defination 17
[10] Refers to Defination 18

Running Example

Let $a_1 = (C_2, standby\_down)$, $a_2 = (C_1, up\_down)$ be two Atomic Hypotheses and $a_3 = a_1\_AND\_a_2 \in \bar{\mathcal{A}}$ a Composed Hypothesis. Considering the occurrences reported in Table A.10, the Composed Hypothesis a is observed in CaseID 10 since:

- $((C_2, standby\_down), 29/06/2021\ 07{:}34{:}30, 10) \in \mathcal{O}$ and $((C_1, standby\_down), 29/06/2021\ 07{:}34{:}30, 10) \in \mathcal{O}$

The occurrence associated with the composed hypothesis is the last one performed:

- $((C_2, standby\_down), 29/06/2021\ 07{:}34{:}30, 10)$

Considering another Composed Hypothesis $a_3 = a_1\_OR\_a_2 \in \bar{\mathcal{A}}$, is observed in both CaseID 1 and 10 since:

- $((C_2, standby\_down), 27/04/2021\ 17{:}50{:}00, 1) \in \mathcal{O}$
- $((C_2, standby\_down), 29/06/2021\ 07{:}34{:}30, 10) \in \mathcal{O}$ and $((C_1, standby\_down), 29/06/2021\ 07{:}34{:}30, 10) \in \mathcal{O}$

The occurrence associated with the Composed Hypothesis, in both cases, is the first one performed:

- $((C_2, standby\_down), 27/04/2021\ 17{:}50{:}00, 1)$
- $((C_1, standby\_down), 29/06/2021\ 07{:}34{:}30, 10)$

5.2. Inference model discovery

The Inference Model Discovery step is performed in a two-step approach: *Structure Inference* and *Composed Hypothesis Inference*. Moreover, it is an iterative approach, since each possible transition in the system's components is considered an effect. For each given effect, the significant causes are selected by computing Kleinberg's metrics on all the possible events occurring in the system. Only once all the significant causes of a transition have been discovered, all the possible Composed Hypothesis are formulated, in terms of boolean predicates, and tested again with Kleinberg's metric. To avoid the explosion in the state space, the *Composed Hypothesis Inference* step relies on the results of the previous step for reducing the hypothesis that constitutes a possible Composed Hypothesis for each considered effect.

5.2.1. Structure inference

As already remarked, this step aims to discover the interactions between components and the impacts of values recorded by the dynamics. Considering all the Atomic Hypotheses defined by the set $\mathcal{A}$, the proposed approach evaluates whether they can cause states' transition in other components. Algorithm 1 implements the *Structure Inference* step, iteratively repeating the procedure for each considered effect (*line 1*).

Algorithm 1 Structure inference algorithm.

Require: $\mathcal{A}$, $\mathcal{O}$,
1: **for each** $f \in \mathcal{A}_C$ **do**
2: $PF(f) \leftarrow$ TPFC$(f, \mathcal{A})$ ▷ Testing Prima Facie Causes using Algorithm 5
3: **for each** $a \in PF(f)$ **do**
4: $\varepsilon_a \leftarrow \varepsilon_{avg}(f, a)$ ▷ Computing Epsilons applying Eq. (2)
5: **Filtering Significant Causes** $SC(f)$ applying p-value algorithm
6: **return** $\bigcup_{f \in \mathcal{A}_C} SC(f)$

For the sake of clarity, a description of each step of the algorithm is following provided.

Running Example
Referring to the dataset in Table A.10, it is possible to state that:

- $a_1 = (C_2, standby\_down)$ verifies the Definition 11 for $f_1 = (C_0, up\_down)$

since there are two occurrences, associated respectively to a and f such that:

- $o_{a_1} = ((C_2, standby\_down), 29/06/2021\ 07{:}34{:}30, 10)$
- $o_{f_1} = ((C_0, up\_down), 29/06/2021\ 07{:}34{:}31, 10)$

that belongs to the same process instance: $\varphi_{CI}(o_{a_1}) = \varphi_{CI}(o_{f_1}) = 10$ and hold the time property:

- $max(\{\mathcal{T}(o) | o \in \mathcal{O} \wedge \varphi_{CI}(o) = 10\}) = 29/06/2021\ 10{:}20{:}31$
- $min(\{\mathcal{T}(o) | o \in \mathcal{O} \wedge \varphi_{CI}(o) = 10\}) = 22/06/2021\ 16{:}40{:}00$
- $\delta = max - min = 582000s$
- $\Delta = \varphi_{\mathcal{T}}(o_{a_1}) - \varphi_{\mathcal{T}}(o_{f_1}) = 1 \implies 0 < \Delta \leq 582000$

This implies $\varphi_{CI}(o_{a_1} \in \mathcal{W}(f_1, a_1)$, but two more considerations are due:

- the value of δ can be arbitrarily chosen and in this case has been set to $10s$;
- the hypothesis a_1 can hold Definition 11 for other effects, for example, $f_2 = (C_0, up\_failing)$, since in the process instance number 1 two occurrences hold the definition and thus implies that $1 \in \mathcal{W}(f_2, a_1)$.

Testing Prima Facie Causes

In order to compute the *prima facie* causes, according to Definition 1, provided by Suppes theory, some concepts are introduced.

Definition 11. A hypothesis $a \in \mathcal{H}$, fits the *prima facie* first condition for an effect $f \in \mathcal{A}_C$ if and only if there is **at least one** process instance —i.e. *CaseID*— in which they both occur and a occurs before f.[11]
The set of the *CaseIDs* that holds these two properties is defined $\mathcal{W}(f, a)$.

Since from the first condition, a is a *prima facie* cause for f, this implies that there exist at least two occurrences respectively associated with a and f in the same process instance. The elapsed time between the two events, then, is at maximum the elapsed time of the entire process instance.
$TPFC : \mathcal{A}_C \times \mathcal{H} \to \mathcal{P}(\mathcal{H})$ function assigns to each effect the set of all the hypotheses that verify the property of *prima facie* causes to each selected effect.

Computing Epsilon

Kleinberg's metric, described in Eq. (2), can be computed introducing the concept of non-exclusive causes and non-concordant causes, as respectively defined in Defination 2 and Defination 3. In the PM context, Defination 2, 3 are related to the properties held by a specific process instance, as illustrated by Defination 12.

Definition 12. Let $a \in PF(f)$, and $x \in PF(f)$ with $x \neq a$, they are non-excluding for f and non-concordant if and only if:

- exists at least one process instance in which both the causes a, x and the effect f occur.
- exists at least one process instance in which x occurs and a not.

[11] For a formal definition and property proofs, refer to Appendix C Defination 19.

Let $\mathcal{L}_{a,f}$ be the set of the x that holds these properties.[12]

For all $a \in PF(f)$ and for all $x \in \mathcal{L}_{a,f}$ it is possible to compute the metric defined in Eq. (2). In the proposed context $\mathbf{P}(f|a \wedge x)$ is given by the process instances in which f, a, x occur over the process instances in which x and a occur. $\mathbf{P}(f|\neg a \wedge x)$ is given by the process instances in which f and x occur but a does not, over those in which x occurs and a does not. A complete discussion of this formula in the proposed context and the proof of the well-posed definition is provided by Eq. (C.4) in the Appendix C.

Filtering Significant Causes

After computing the epsilon values, multiple causes $a \in \mathcal{A}$ are assigned to a single effect $f \in \mathcal{A}_C$. A one-sample t-test is applied to identify the causes a that are *statistically significant*. The t-test evaluates the null hypothesis[13] that the true population $\varepsilon_{avg}(a, f)$ equals 0, which means that "*a is not a cause for the effect f*". If this null hypothesis is true, any $\varepsilon_{avg}(a, f)$ computed for a given sample being different from 0 is a mere sampling artifact. The t-test computes a p-value which expresses the probability of observing a sample $\varepsilon_{avg}(a, f)$ equal or greater to 0 under the assumption that the null hypothesis is true [48]. If this p-value (probability) is less than a considered threshold (in such case 0.05), the observed $\varepsilon_{avg}(a, f)$ is considered to be such unlikely to occur under the null hypothesis that the null hypothesis is rejected or a is considered a statistically significant cause for f.

In the details, assuming the null hypothesis all the prima facie causes of f are considered. For each of those causes, if the p-value, computed on all the $\varepsilon_x(a, f)$ with $x \in \mathcal{L}_{a,f}$, is lower than threshold, the null hypothesis is rejected, and a considered a *significant cause* for the effect f. A formal definition of the algorithm for filtering significant causes is provided in the Appendix D by Algorithm 6.

5.2.2. Composed hypothesis inference

Once the *significant causes* are discovered for each effect, the focus is shifted to the internal behaviours of the component involved, discovering the *predicates* that fire its *output ports*. These predicates are represented in the metamodel of Fig. 4 by the *Composed Hypothesis* concept (named C-Hypothesis). Algorithm 2 proposes a way to compute the C-Hypotheses, implementing the *Composed Hypothesis Inference* step in Fig. 3. More in detail, this algorithm is constituted by three main parts.

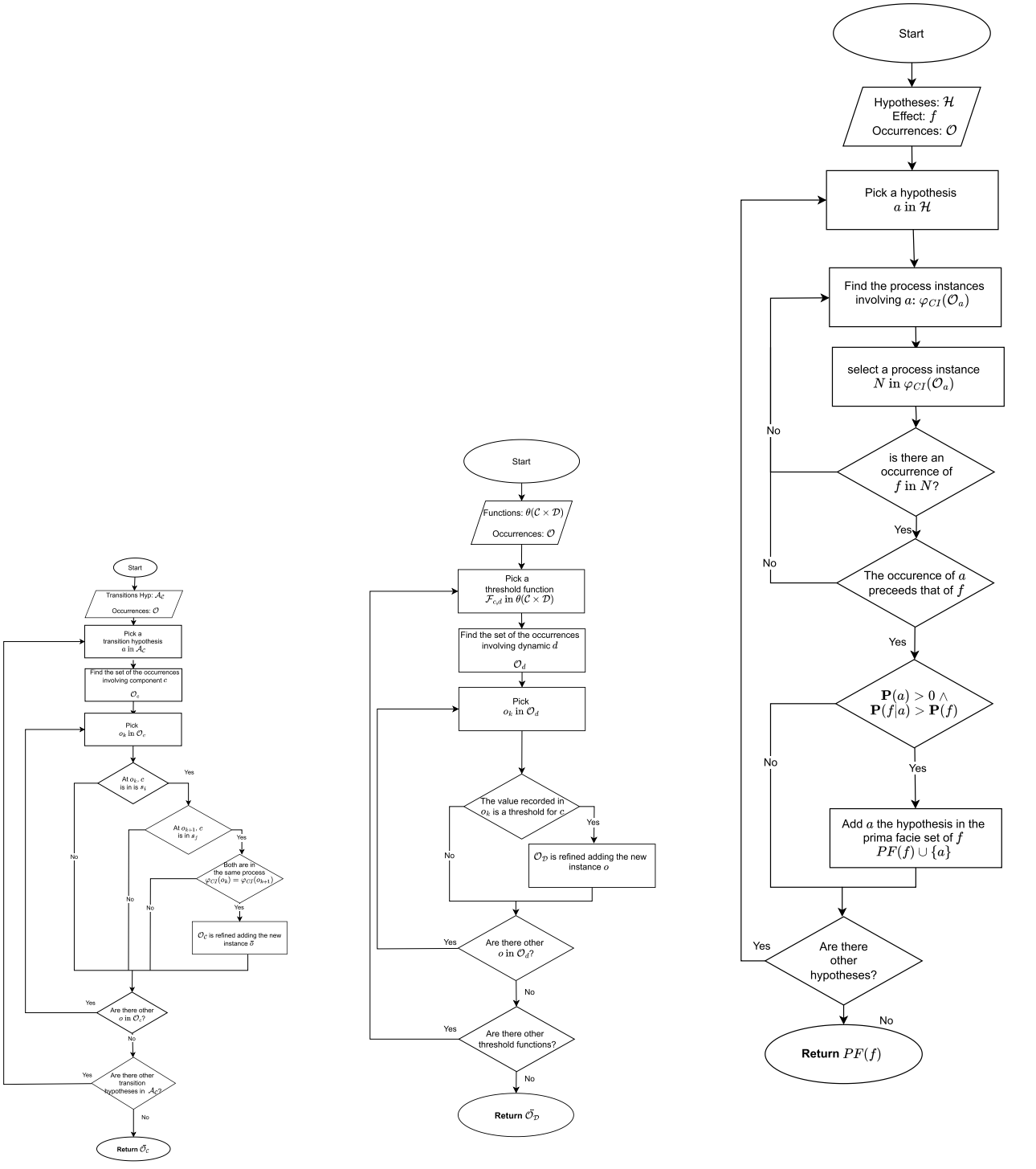
Algorithm 2 C-Hypothesis inference algorithm.

Require: $\bar{\mathcal{A}}\ \mathcal{A}_C, \bigcup_{f \in \mathcal{A}_C} SC(f)$
1: **for each** $f \in \mathcal{A}_C$ **do**
2: ▷ filtering the composed hypothesis set
3: $CH(f) = \{a_1 * \cdots * a_{I(f)}$[14] $\in \bar{\mathcal{A}}$ with $a_i \in SC(f)$ and $* \in \{AND, OR\}\}$
4: $PF(f) \leftarrow$ TPFC$(f, CH(f))$ ▷ Testing Prima Facie Causes using Algorithm 5
5: **for each** $a \in PF(f)$ **do**
6: $\varepsilon_a \leftarrow \varepsilon_{avg}(f, a)$ ▷ computing epsilons applying Eq. 2
7: $Pr(f) \leftarrow \max_{a \in PF(f)} \varepsilon_a$
8: **return** $\bigcup_{f \in \mathcal{A}_C} Pr(f)$

Filtering the composed Hypothesis set

The Algorithm 2 requires as input the set of the Composed Hypotheses $\bar{\mathcal{A}}$, the set of the effects $\mathcal{A}_C$ and for each $f \in \mathcal{A}_C$, the *significant causes* set $SC(f)$, obtained from the previous step. For each effect $f \in \mathcal{A}_C$ (line

[12] A more formal definition is presented in Appendix C, Def 22.
[13] The null hypothesis (often denoted as H_0) is a statement used in statistics that proposes the absence of effects, differences, or relationships between variables in a study. It serves as a baseline or default position for hypothesis testing.



(a) Transition Hypotheses set construction.

(b) Theta Hypotheses set construction.

(c) Prima Facie set construction.

Fig. 5. Flowcharts for the construction of Transition, Prima Facie, and Theta Hypotheses sets.

1), $\bar{\mathcal{A}}$ is filtered, considering only the Composed Hypotheses that involve all the *significant causes* of f (line 2). The CH set contains the Composed Hypotheses to test with Kleinberg's metric. It is built by combining all the *significant causes* with the logical operators (line 3).

Testing Prima Facie and Computing Epsilon

The computation of the prima facie causes and epsilon average values recalls the same procedure explained in *Structure Inference* step. In this case, the function $TPFC$ requires as input the effect f and the Composed Hypotheses $CH(f) \subseteq \bar{\mathcal{A}}$, returning the subset of them that hold prima facie property (line 4). After that, for each prima facie cause, the epsilon value is computed (lines 5-6).

Running Example
Considering the two hypotheses $a_1 = (C_2, standby\_down)$ and $a_2 = (C_1, up\_down)$, they verify the Defination 11 for $f_1 = (C_0, up\_down)$, but more in general they both hold the property to be a prima facie causes for f_1:

- $\{a_1, a_2\} \subseteq PF(f_1)$

Moreover, they also hold the property described by Definition 12:

- $10 \in \mathcal{W}(f_1, a_1)$ and $10 \in \mathcal{W}(f_1, a_2) \implies \mathcal{W}(f_1, a_1) \cap \mathcal{W}(f_1, a_2) \neq \emptyset$
- $1 \in \varphi_{CI}(\mathcal{O}_{a_1})$ but $1 \notin \varphi_{CI}(\mathcal{O}_{a_2}) \implies \varphi_{CI}(\mathcal{O}_{a_1}) \setminus \varphi_{CI}(\mathcal{O}_{a_2}) \neq \emptyset$.

So it is possible to state that a_1 and a_2 are *confounding* and not *concordant* causes for f_1 and

- $a_1 \in \mathcal{L}_{a_2}$

Maximum Calculation
Finally, the Composed Hypothesis with the highest epsilon value is selected (line 7) for each effect. This choice is due to the semantic meaning assumed by the composed hypothesis in the PdFT formalism. PdFT's *Triggers* are functions assigning to each transition on each component one and only one predicate.

5.3. PdFT generation

The last step of the methodology, depicted in Fig. 3, is responsible for generating the proper PdFT elements from the results obtained in the Inference Model Discovery step. The translation involves both the *Structure Inference* and the *Composed Hypothesis Inference* steps.

5.3.1. Structure to PdFT

For each $f \in \mathcal{A}_C$, a set of significant causes $SC(f) \subseteq \mathcal{A}_C \cup \mathcal{A}_D$ obtained from the previous step, is considered.

Transition Hypotheses
Let the effect $f = (c_i, t_i)$ be a transition involving the component c_i and $a = (c_j, t_j)$ the discovered significant cause involving the component c_j. From the discovered relation between the effect and the hypothesis some PdFT's elements can be constructed. The component c_i is refined by adding an *input port* $p_{i,j} \in P^I_{c_i} \subseteq P^I$, modelling the connection with c_j. An event is generated, starting from the output port of the component $c_j \in C$. Finally, the relation between a and the effect f determines the generation of the action function α of the t_j transition on the c_j component. Hence, t_j becomes an impacting transition for c_j and $\alpha(p_j, t_j) \neq Neutral$.

Theta Hypothesis. Let $a = (d_j, v_j)$ be the discovered cause and f the effect, involving the dynamic d_j, this means that the latter records a value v_j influencing the component c_i. Hence, a predicate based on the $\mathcal{F}_{c_i,d_j}$ function is discovered, implying the discovery of the $\theta_T(c_i, d_j)$ threshold.

A possible implementation of this procedure is provided in the Appendix D, reported in Algorithm 7.

5.3.2. Composed hypothesis to PdFT

A similar approach is performed on the results of the *Composed Hypothesis Inference* step. At this step, for each effect $f = (c_i, t_i)$, the Composed Hypothesis $Pr(f)$ is a boolean predicate composed of Atomic Hypotheses connected by means of logical operators. If $a_j = (c_j, t_j) \in \mathcal{A}_C$ involves a component, the predicate, associated to f, is updated with the input port $p_{i,j}$ that connects c_i with c_j. The input port value depends on those assigned to p_j from the function $\alpha(p_j, t_j)$. Otherwise, if $a_j = (d_j, v_j) \in \mathcal{A}_D$ involves a sensor, the predicate, associated to f, is updated with the expression of the function $\mathcal{F}_{c_i,d_j}$. In the PdFT model,

Running Example
Let consider the hypothesis $a_1 = (C_2, standby\_down)$ and the effect $f_1 = (C_0, up\_down)$, given that:

- $C_2 \in C$
- $t = (standby, down) \in T^{c_2}$
- p_2 is the output port of the component C_2

the relationship $a_1 \in SC(f_1)$ is translated into:

- the definition of the input port for the component C_0 that connects the component with C_2: $p_{0,2}$
- the definition of the value to assign to the transition $t = (standby, down)$ that is an *impacting* transition for the component C_2, since it reflects some event outside: $\alpha(C_2, t) = 1$
- the definition of the event that connects the output port p_2 with the input $p_{0,2}$: $e = (p_2, p_{0,2})$

Running Example
Let consider the effect $a_1 = (C_2, standby\_down)$, and the hypothesis $\tilde{a}_2 = (sensor, 70)$, according to the results of the previous step, given that:

- $d = sensor \in D$ with $70 \in \mathcal{V}_d$

the relationship $\tilde{a}_2 \in SC(a_1)$ is translated into:

- the definition of the threshold value for the parameter monitored by d on the component C_2: $\theta_T(C_2, d) = 70$

the predicate is a boolean expression that triggers a transition when the value of the expression is true. While the function *alpha* assigns to the impacting transitions, a value of *True* or *False* according to the direction of the arc that connects the initial state to the final state. This allows modelling a direction rather than assigning a boolean meaning. For this reason, if the value of the function $\alpha(p_j, s_j)$ is *False*, this does not mean that the condition is not satisfied, but this means that the condition is satisfied in the opposite direction. The hypothesis is translated into the predicate according to the function γ in Eq. 9.

$$\gamma(p_{j,i}) := \begin{cases} p_{j,i} \iff \alpha(p_i, t) = True \\ \neg p_{j,i} \iff \alpha(p_i, t) = False \end{cases} \tag{9}$$

A possible implementation of this procedure is provided in the Appendix D, reported in Algorithm 8.

6. Proof of concept

This chapter is devoted to demonstrating the effectiveness of the proposed approach using a concrete example. A CALYPSO tool, implementing the methodology, is available at GitHub repository [8].

6.1. Simulator

Another supporting tool is Dependability Simulation Engine (DSE), which implements a simulator for the proposed example [6]. The extension of DSE for the CALYPSO tool is available at GitHub repository [7]. Overall, the DSE provides a structured and extensible framework for simulating and evaluating the dependability of complex systems, incorporating both model-based and data-driven approaches.

The choice of testing the methodology on a simulated scenario is motivated by several issues: (1) data availability: especially in the case of

Running Example

Let consider the effect $f_1 = (C_0, up\_down)$ and the Composed Hypothesis $a_4 = (C_2, standby\_down)\_AND\_(C_1, up\_down)$, according to the results of the *Composed Hypothesis Inference* step:

- $Pr(f_1) = a_4$

Given that

- $\alpha(C_2, up\_down) = True \implies \gamma(p_{0,2}) = p_{0,2}$
- $\alpha(C_1, up\_down) = True \implies \gamma(p_{0,1}) = p_{0,1}$

the relationship $Pr(f_1) = a_4$ is translated into:

- the definition of the trigger function: $\tau(C_0, up\_down) = p_{0,1} \wedge p_{0,2}$

Running Example

Let consider the effect $a_1 = (C_2, standby\_down)$ and the hypothesis $\tilde{a}_2 = (sensor, 70.00023)$, according to the results of the *Inference* step:

- $Pr(a_1) = \tilde{a}_2$

Given that

- $\theta_T(C_2, d) = 70$
- $\mathcal{F}_{C_2,d} = d(x) > 70$

the relationship $Pr(a_1) = \tilde{a}_2$ is translated into:

- the definition of the trigger function: $\tau(C_2, standby\_down) = \mathcal{F}_{C_2,d}$

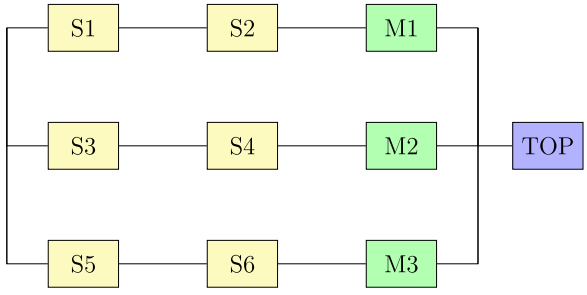


Fig. 6. System configurations (Reliability block diagram).

complex infrastructure, the proposed approach requires a huge amount of data to be trained, which is usually not freely available [49]; (2) evaluation purposes: the possibility of defining the system architecture provides complete knowledge about the simulated process; in such a way, a precise evaluation of the methodology can be achieved, comparing the results with the ground truth.

The first activity of this Proof of Concept (PoC) is to define a simulation model in the DSE framework. The example, shown in Fig. 6, is structured into ten components: six *subcomponents*, which are instances of the *Component* class and do not have any "child" component; three *middle components*, each of these considers two *subcomponents*. The failure of both *subcomponents* leads to a failure of the *middle components*; one *top component*, connected to the three *middle components*. A failure of at least one *middle component* leads to the break of the *top component* provoking the entire system failure.

The hierarchical structure of the simulator implies that, when a fault arises in the lower layers, it is propagated toward the top. The system is

Table 2
Faults' description within the system.

Component	Internal	Other components	Signals
X_TOP	$t > MTBF_{TOP}$	X_C1s down ∨ X_C2s down ∨ X_C3s down	—
X_C1s	$t > MTBF_{CX_s}$	X_C10 down ∧ X_C11 down	—
X_C2s	$t > MTBF_{CX_s}$	X_C20 down ∧ X_C21 down	—
X_C3s	$t > MTBF_{CX_s}$	X_C30 down ∧ X_C31 down	—
X_C10, X_C11	$t > MTBF_{C1X}$	—	$sigA > 70$
X_C20, X_C21	$t > MTBF_{C2X}$	—	$sigB > 3.99$
X_C30, X_C31	$t > MTBF_{C3X}$	—	$sigC > 85$

Table 3
Excerpt of the dataset.

Element_ID	time:timestamp	case:concept:name	Message_description	Value
sigB	129,384	462	—	4
sigC	129,387	463	—	27.26
X_C20	129,400	462	is down	—
X_C21	129,400	462	is down	—
X_C2s	129,401	462	is down	—
X_top	129,402	462	is down	—
sigA	129,410	463	—	10.93
sigC	129,430	463	—	27.29
sigA	129,500	463	—	10.96
sigC	129,530	463	—	27.32

also equipped with three sensors: *sigA*, *sigB* and *sigC*, which are instances of the class *Signal*. Two components are assigned to each signal, with a corresponding condition. Table 2 summarises all the possible faults that could arise in the system.

Once the simulation model is defined, running DSE produces two kinds of data:

- **Time-series**: the records produced by the sensors *sigA*, *sigB* and *sigC*. The attributes acquired are the ID of the sensor, the timestamp in seconds, the sensed value and the *CaseID*;
- **Event logs**: the messages sent by the components in case of state changes. The acquired attributes are the ID of the component, the timestamp in seconds, the state change and the *CaseID*.

Table 3 shows an excerpt of the dataset [8]. The simulated environment reproduces 100 years of the system's activity. Sensors' data is recorded every 90 minutes of simulated time, resulting in approximately 584,411 records of time-series data. Regarding the component states, a total of 16 failure modes can be injected, as specified in Table 2. The dataset is organised into process instances, each representing a scenario generated through fault injection and subsequent recovery actions. These fault chains are not deterministic; instead, they are statistically generated based on probability distributions for the Mean Time Between Failures (MTBF) and sensors' behaviours. In total, 425 process instances were processed. The produced dataset comprises more than one million rows and 5 columns.

6.2. Applying the methodology

The application of the method and the algorithms presented in Section 5 to this example is here reported in the three subsections: problem formalisation (Section 6.2.1), inference model discovery (Section 6.2.2), and PdFT model generation (Section 6.2.3). As mentioned above, the CALYPSO tool implements and automates the steps.

6.2.1. Problem formalisation

According to the content of Section 5.1, Table 4 maps the elements of the causality formalisation, the features in the dataset and the PdFT syntactic elements. Other parameters to consider are: the number of the components ($n = 9$), the number of dynamics ($z = 2$), and the total number of the rows ($l = 1048576$).

Starting from this, Algorithms 3 and 4 are applied.

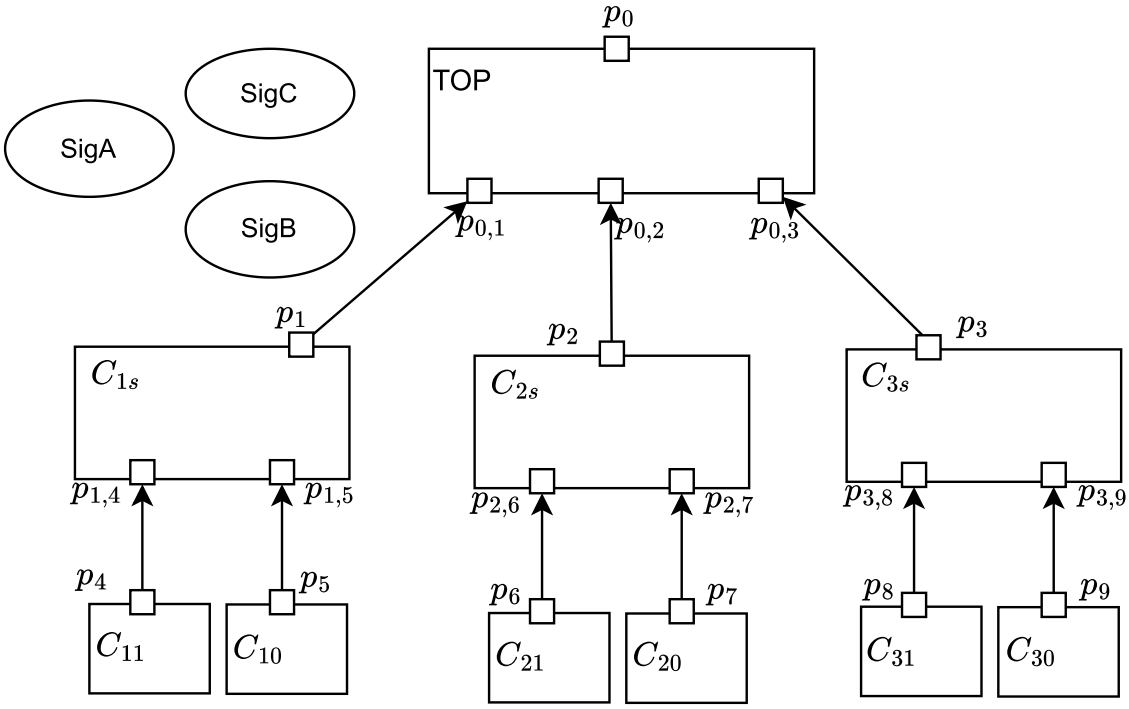


Fig. 7. Discovered structure of the PdFT model.

Table 4
Causality-PdFT elements mapping.

Causality Formalisation	Feature in the Dataset	PdFT Element
$\mathcal{C} = \{c_0, \dots, c_n\}$	Element_ID(Component)	Components $\mathcal{C}$
$\mathcal{S} = \bigcup_{c \in \mathcal{C}} \mathcal{S}^c$	Messages_description	States $\mathcal{S}$
$\mathcal{D} = \{d_0, \dots, d_z\}$	Element_ID(Signals)	Dynamics $\mathcal{D}$
$\mathcal{V} = \bigcup_{d \in \mathcal{D}} \mathcal{V}_d$	Value	Dynamics values $\bigcup_{d \in \mathcal{D}} \mathcal{D}(\mathbb{R})$
$\mathcal{O} = \{o_1, \dots, o_l\}$	Dataset rows	—
φ_R	Element_ID	$\mathcal{C} \cup \mathcal{D}$
φ_M	Message_description, Value	$\mathcal{S} \cup \mathcal{V}$
φ_T	time:timestamp	—
φ_C	case:concept:name	—

6.2.2. Inference model discovery

The second step is the *Inference Model Discovery*, whose two stages generate the model. Concerning the **Structure Inference**, Algorithm 1 is applied generating the results shown in Table 5. The table assigns to each effect considered f, i.e. each possible transition observed on each component, its discovered cause a, which could be either a transition on another component or a value measured by a dynamic. To each cause-effect couple is also assigned the computed metric measuring the causal significance.

The second stage of this step — i.e., the **Composed Hypothesis Inference** step — consists of the discovery of hypotheses to generate PdFT's *trigger* functions. Starting from all the *significant causes* represented in Table 5, Algorithm 2 is applied. Table 6 shows the results. As in the previous step, in the table each effect f is associated with the boolean predicate that enables the transition considered. Last column reports the value of statistical significance computed with the Kleinberg's metric.

6.2.3. Model generation

This last step oversees the translation of the results into the PdFT formalism. From the result shown in Table 5, the application of the Algorithm 7 enables the **Structure to PdFT** stage. The PdFT structure is then defined and graphically represented in Fig. 7.

The last phase is devoted to populating the model with the discovered trigger functions, according to the **Composed Hypothesis to PdFT** stage. The results obtained, enrich the model reported in Fig. 7. The graphical representation of the whole PdFT model of the considered example, is reported in Fig. 8. For the sake of clarity, the *trigger function*, described on each considered effect, is reported in Table 7.

6.3. Evaluation

This subsection is devoted to the evaluation of the methodology proposed from two different points of view: the trustworthiness of the inference process and robustness to data noise.

6.3.1. Inference trustworthiness

Here, the term *trustworthiness* means the degree of adherence of the inferred model to reality and, hence, the ability to infer the right relationship between the causes and effects. As already remarked, the proposed example is based on simulated data, also to let a more straightforward evaluation of the trustworthiness. In this setting, cause-effect relationships can be derived, as well as the logical rules that enable the transitions. The validation of the model inference is based on widespread ML metrics.

Definition 13. A *possible cause* $x \in \mathcal{A}$ is:

- True Positive (TP) for an effect f if it is a real cause for f and it is labelled as a *significant causes* $x \in SC(f)$ for f;
- False Positive (FP) for an effect f if it is **not** a real cause for f but it is labelled as a *significant causes* $x \in SC(f)$ for f;
- True Negative (TN) for an effect f if it is **not** a real cause for f and it is **not** labelled as a *significant causes* $x \notin SC(f)$ for f;
- False Negative (FN) for an effect f if it is a real cause for f but it is **not** labelled as a *significant causes* $x \notin SC(f)$ for f.

TNs are all the possible combinations of (fake) causes and/or predicates that are not considered in the inferred model. Theoretically, this set can have a huge dimensionality. Even considering a finite number of

Table 5
Structure inference results.

$f \in \mathcal{A}_C$	$a \in \mathcal{A}$	$\epsilon_{avg}(a, f)$	$f \in \mathcal{A}_C$	$a \in \mathcal{A}$	$\epsilon_{avg}(a, f)$
(X_top,up_down)	(X_C1s,up_down)	1	(X_top,up_down)	(X_C2s,up_down)	1
(X_top,up_down)	(X_C2s,up_down)	1	(X_C1s,up_down)	(X_C11,up_down)	1
(X_C1s,up_down)	(X_C10,up_down)	1	(X_C2s,up_down)	(X_C20,up_down)	1
(X_C3s,up_down)	(X_C31,up_down)	1	(X_C3s,up_down)	(X_C30,up_down)	1
(X_C3s,up_down)	(sigC,85)	0.37	(X_C11,up_down)	(sigA,70)	0.74
(X_C10,up_down)	(sigA,70)	0.76	(X_C21,up_down)	(sigB,3.99)	0.67
(X_C20,up_down)	(sigB,3.99)	0.74	(X_C31,up_down)	(sigC,85)	0.76
(X_C30,up_down)	(sigC,85)	0.77	(X_C1s,up_failing)	(X_C11,up_down)	0.2
(X_C1s,up_failing)	(X_C10,up_down)	0.12	(X_C2s,up_failing)	(X_C21,up_down)	0.13
(X_C2s,up_failing)	(X_C20,up_down)	0.20	(X_C3s,up_failing)	(X_C31,up_down)	0.27
(X_C3s,up_failing)	(X_C30,up_down)	0.21	(X_top,down_up)	(X_C1s,down_up)	1
(X_top,down_up)	(X_C2s,down_up)	1	(X_top,down_up)	(X_C3s,down_up)	1
(X_C1s,down_up)	(X_C11,down_up)	0.33	(X_C1s,down_up)	(X_C10,down_up)	0.43
(X_C2s,down_up)	(X_C21,down_up)	0.41	(X_C2s,down_up)	(X_C20,down_up)	0.35
(X_C3s,down_up)	(X_C31,down_up)	0.32	(X_C3s,down_up)	(X_C30,down_up)	0.32
(X_C1s,failing_up)	(X_C11,down_up)	0.27	(X_C1s,failing_up)	(X_C10,down_up)	0.21
(X_C2s,failing_up)	(X_C21,down_up)	0.22	(X_C2s,failing_up)	(X_C20,down_up)	0.27
(X_C3s,failing_up)	(X_C31,down_up)	0.38	(X_C3s,failing_up)	(X_C30,down_up)	0.33

Table 6
Composed hypothesis inference results.

$f \in \mathcal{A}_C$	$Pr(f)$	$\epsilon_{avg}(a, f)$
(X_top, up_down)	$(X\_C1s, up\_down)\_OR\_(X\_C2s, up\_down)\_OR\_(X\_C3s, up\_down)$	1
(X_C1s, up_down)	$(X\_C11, up\_down)\_AND\_(X\_C10, up\_down)$	1
(X_C2s, up_down)	(X_C20, up_down)	0.73
(X_C3s, up_down)	$(X\_C31, up\_down)\_AND\_(X\_C30, up\_down)\_AND\_(sigC, > 85)$	0.54
(X_C11, up_down)	$(sigA, 70)$	0.74
(X_C10, up_down)	$(sigA, 70)$	0.76
(X_C21, up_down)	$(sigB, 3.99)$	0.67
(X_C20, up_down)	$(sigB, 3.99999)$	0.74
(X_C31, up_down)	$(sigC, 85)$	0.76
(X_C30, up_down)	$(sigC, 85)$	0.77
$(X\_C1s, up\_failing)$	$(X\_C11, up\_down)\_OR\_(X\_C10, up\_down)$	0.39
$(X\_C2s, up\_failing)$	$(X\_C21, up\_down)\_OR\_(X\_C20, up\_down)$	0.40
$(X\_C3s, up\_failing)$	$(X\_C31, up\_down)\_OR\_(X\_C30, up\_down)$	0.53
$(X\_top, down\_up)$	$(X\_C1s, down\_up)\_OR\_(X\_C2s, down\_up)\_OR\_(X\_C3s, down\_up)$	1
$(X\_C1s, down\_up)$	$(X\_C11, down\_up)\_OR\_(X\_C10, down\_up)$	0.61
$(X\_C2s, down\_up)$	$(X\_C21, down\_up)\_OR\_(X\_C20, down\_up)$	0.60
$(X\_C3s, down\_up)$	$(X\_C31, down\_up)\_OR\_(X\_C30, down\_up)$	0.48
$(X\_C1s, failing\_up)$	$(X\_C11, down\_up)\_OR\_(X\_C10, down\_up)$	0.39
$(X\_C2s, failing\_up)$	$(X\_C21, down\_up)\_OR\_(X\_C20, down\_up)$	0.40
$(X\_C3s, failing\_up)$	$(X\_C31, down\_up)\_OR\_(X\_C30, down\_up)$	0.52

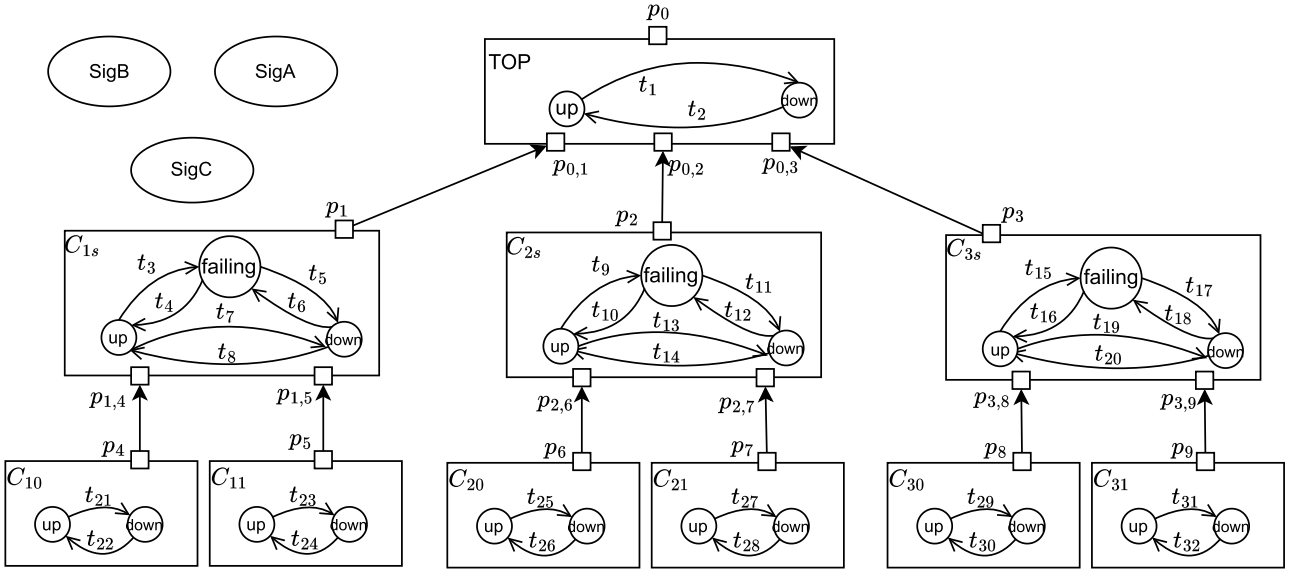
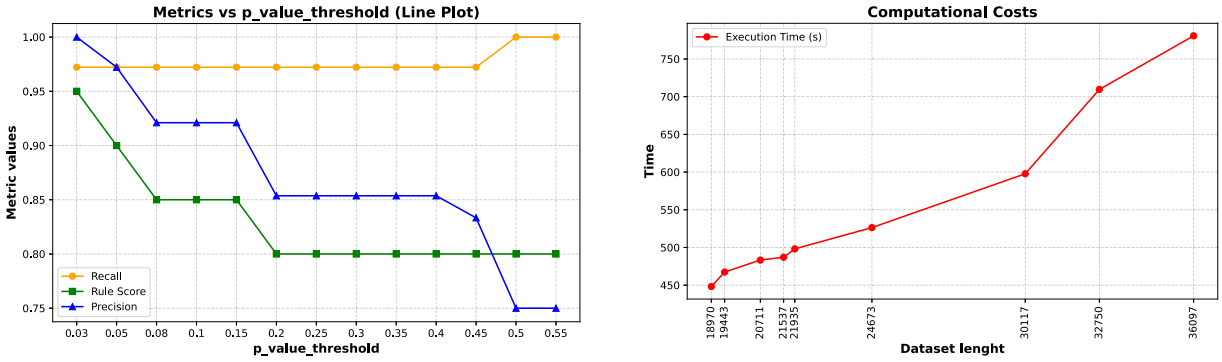


Fig. 8. PdFT inferred model.



(a) Influence of p-value threshold on metrics score. (b) Computational costs.

Fig. 9. Comparison of results: (a) Impact of significance threshold on performance metrics; (b) Computational costs across different dataset sizes.

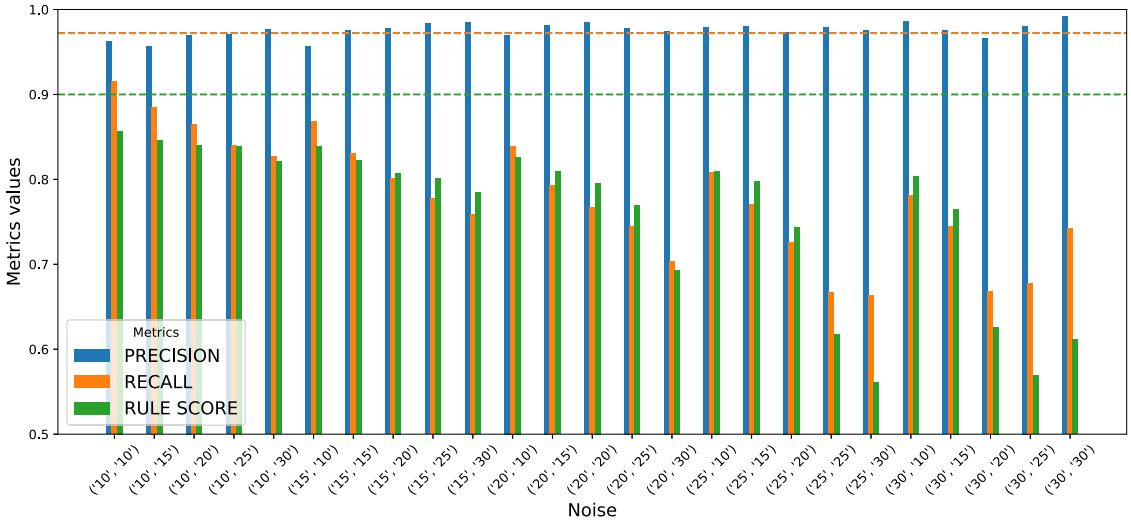


Fig. 10. Metric overall evaluation.

Table 7
PdFT inferred predicates.

$f=(c,t)$	$\tau(c,t)$	$f=(c,t)$	$\tau(c,t)$
(X_top,up_down)	$p_{0,1} \vee p_{0,2} \vee p_{0,3}$	(X_C1s,up_failing)	$p_{1,4} \vee p_{1,5}$
(X_C1s,up_down)	$p_{1,4} \wedge p_{1,5}$	(X_C2s,up_failing)	$p_{2,6} \vee p_{2,7}$
(X_C2s,up_down)	$p_{2,7}$	(X_C3s,up_failing)	$p_{3,8} \vee p_{3,9}$
(X_C3s,up_down)	$p_{3,8} \wedge p_{3,9} \wedge sigC > 85$	(X_top,down_up)	$\neg p_{0,1} \vee \neg p_{0,2} \vee \neg p_{0,3}$
(X_C11,up_down)	$sigA > 70$	(X_C1s,down_up)	$\neg p_{1,4} \vee \neg p_{1,5}$
(X_C10,up_down)	$sigA > 70$	(X_C2s,down_up)	$\neg p_{2,6} \vee \neg p_{2,7}$
(X_C21,up_down)	$sigB > 3.99999$	(X_C3s,down_up)	$\neg p_{3,8} \vee \neg p_{3,9}$
(X_C20,up_down)	$sigB > 3.99999$	(X_C1s,failing_up)	$\neg p_{1,4} \vee \neg p_{1,5}$
(X_C31,up_down)	$sigC > 85$	(X_C2s,failing_up)	$\neg p_{2,6} \vee \neg p_{2,7}$
(X_C30,up_down)	$sigC > 85$	(X_C3s,failing_up)	$\neg p_{3,8} \vee \neg p_{3,9}$

hypotheses, the TNs is huge compared with the TPs, affecting the computation of the classical accuracy metric, for this reason not taken into account.

The results of the *Rules Inference* are validated by comparing obtained predicates to the real ones using the function defined in Eq. (10).

$$\mathcal{H} := \begin{cases} \mathcal{H}(p,f)=1 \iff p=\bar{Pr}(f) \\ \mathcal{H}(p,f)=0 \iff p\neq\bar{Pr}(f) \end{cases} \tag{10}$$

where $\bar{Pr}(f)$ is the true predicate of the effect f and $p=Pr(f)$ is the inferred rule. Hence, the function $\mathcal{H}$ assigns a score to each prediction.

Table 8
Metrics for methodology evaluation.

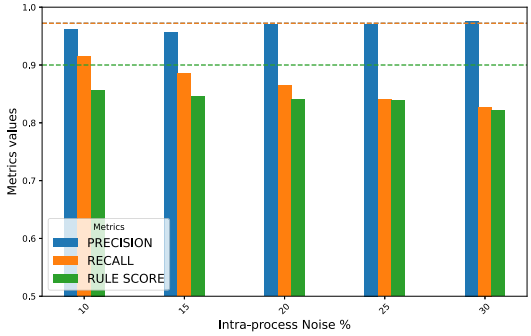
Recall	Precision	Rule Score	Time
0.97	0.97	0.90	382.59 s

The rule-score metric is defined in Eq. (11).

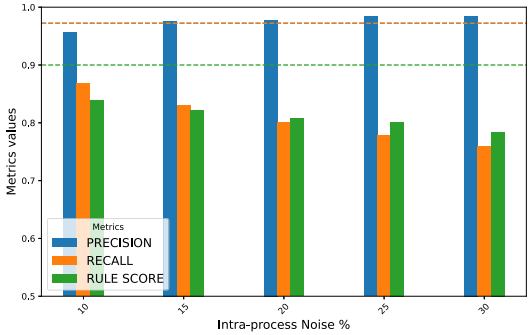
$$Rule\ Score = \frac{\sum_{f\in\mathcal{A}_C}\mathcal{H}(p,f)}{|\mathcal{A}_C|} \tag{11}$$

The metrics and the results on the example used in the PoC are shown in Table 8.

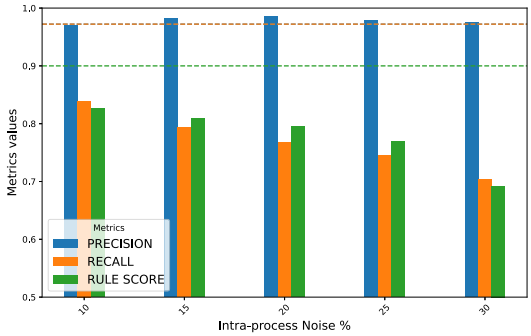
The values obtained for the recall and precision, over the 90%, provide information about how the methodology classifies the hypotheses, showing a promising result in terms of recognition of TPs. This means that the methodology is able to discover the real causes in the proposed PoC. Additionally, the rule score shows that the composed hypotheses discovered match, in 90% of the cases, the real boolean condition that triggers the transitions. Further experiments have been conducted to investigate the influence of the significant causes threshold adopted in p-value test. Such a threshold is adopted to distinguish spurious causes from significant ones: if the p-value, computed on a discovered cause, is lower than the threshold, the null hypothesis is rejected, and it is considered a *significant cause* for the effect. Fig. 9a reports the metric scores obtained for different values of the significance threshold. It can be observed that both Precision and Rule Score decrease as the signifi-



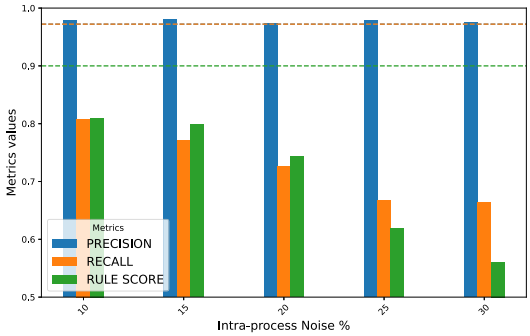
(a) Metrics vs $interaNL$ with $interNL = 10\%$.



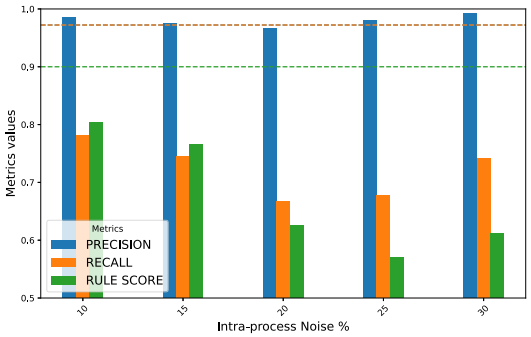
(b) Metrics vs $intraNL$ with $interNL = 15\%$.



(c) Metrics vs $intraNL$ with $interNL = 20\%$.



(d) Metrics vs $intraNL$ with $interNL = 25\%$.



(e) Metrics vs $intraNL$ with $interNL = 30\%$.

Fig. 11. Metrics vs $intraNL$ w.r.t. different $interNLs$.

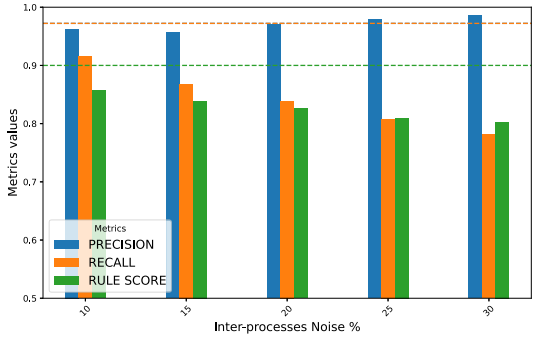
cance level increases, due to the inclusion of additional causes that are in fact spurious FPs. These experiments confirm the theoretical rationale behind the significance level in the p-value test: while the filtering step effectively separates spurious from significant causes, raising the threshold leads to a higher number of FPs, which negatively affects Precision and Rule Score. Conversely, Recall–being related to FNs–remains stable across different thresholds.

Considering computational metrics, the time required to train the model, with over 1 million records filtered in 18,500 activities, is around six minutes. Fig. 9b illustrates that the execution time grows with the dataset size, confirming the scalability trend of the proposed approach. Nevertheless, the increase appears approximately linear, suggesting that the method preserves computational efficiency even when applied to larger inputs. This behaviour highlights the practical feasibility of the algorithm for real-world scenarios where datasets can vary significantly in size. However, the computational costs could increase considering a more complex architecture of the system. In such cases, the explosion in the space of states can be mitigated by filtering the hypothesis sets, considering domain knowledge, and reducing the number of possible causes to test.

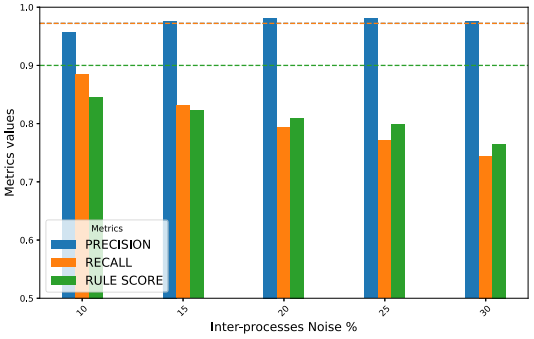
6.3.2. Robustness

Two injection mechanisms are considered:

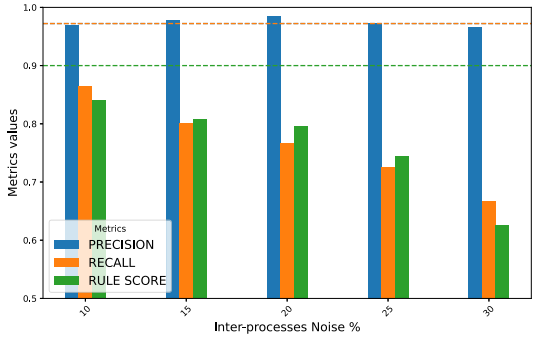
- inter-process noise: a subset of process instances is impacted by noise. The noise injection is measured by the *inter-process noise level*, of the noisy processes over the total number of processes;



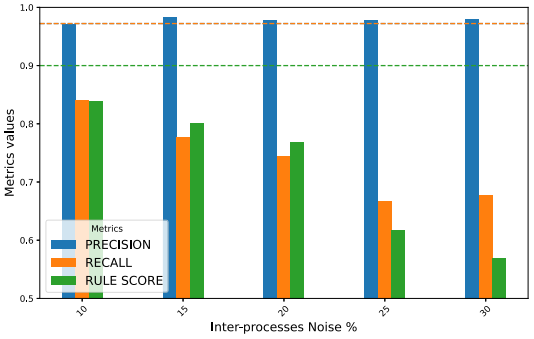
(a) Metrics vs $interNL$ with $interNL = 10\%$.



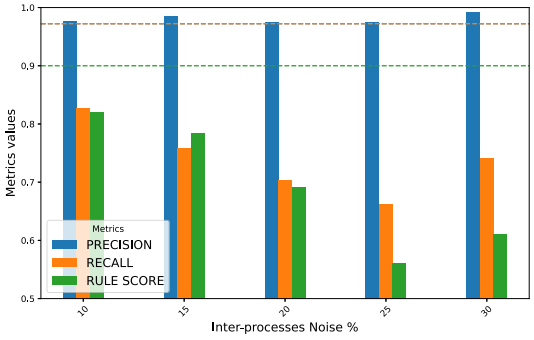
(b) Metrics vs $interNL$ with $interNL = 15\%$.



(c) Metrics vs $interNL$ with $interNL = 20\%$.



(d) Metrics vs $interNL$ with $interNL = 25\%$.



(e) Metrics vs $interNL$ with $interNL = 30\%$.

Fig. 12. Metrics vs $interNL$ w.r.t. different $intraNLs$.

- intra-process noise: considering a process instance, the subset of the lines that are affected by noise. The noise injection is measured by the *intra-process noise level* as the of noisy events injected in the process instances over the total number of events.[15]

Briefly, the couple $(interNL, intraNL)$ characterises the experiments; the noisy couple (20,30) means that the 20% of the total number of the process instances has been modified. Every time a single process instance has been randomly selected, the 30% of the total events belonging to that process instance has been added.

[15] Injecting the noise in a process is meant in this work as "adding" some lines that are not previously present.

The noisy occurrence is a new row belonging to the same process instance — i.e., same *CaseID* — recording an activity randomly chosen among all the possible ones performed by the system. Moreover, the timestamps associated with the new rows are again randomly chosen within the timeframe associated with the selected process instance. All the attributes needed to perform the methodology — i.e. *CaseID*, timestamp and activity — have been randomly selected but ensuring coherence with the other occurrences belonging to the same process instance. Table 9 shows an example of a noisy log, with the new rows in blue, added to the original one proposed in Table 3.

Twenty-five different couples of noise percentages have been selected and, for each of them, 80 datasets were randomly generated, for a total of 2000 tests. The choice to repeat 80 times the test for each group is due to the random selection of the noise to inject in the dataset. Fig. 10

Table 9
Example of a noisy dataset.

Element_ID	time:timestamp	case:concept:name	Message_description	Value
sigB	129,384	462	—	4
X_C10	129387	462	is up	—
sigC	129,387	463	—	27.26
X_C3s	129389	462	is down	—
X_C20	129,400	462	is down	—
X_C21	129,400	462	is down	—
sigB	129401	462	—	3.79
X_C2s	129,401	462	is down	—
X_top	129,402	462	is down	—
sigA	129,410	463	—	10.93
sigC	129,410	463	—	27.29
sigA	129,500	463	—	10.96
sigC	129,530	463	—	27.32

depicts the overall behaviour of the metrics, according to the percentage of noise added to the data, and considering the average values for each group. The dotted lines in the graphs represent the values assumed by the corresponding metric in the case of the original dataset, i.e., the one without noise, whose values are reported in Table 8. This view enables a quick and intuitive evaluation of the distances between the values achieved by the metrics on a noisy dataset — the high of the bars — and the reference value achieved by the metrics on the real dataset.

From the plot, it is clear that the values of *Precision* are stable, this means that the methodology is conservative since the number of FPs is close to zero. *Recall* decreases rapidly as noise levels increase. This suggests that the proposed approach is more likely to overlook a real cause than to incorrectly identify a non-real cause as significant. Finally, *Rule Score* is the metric most affected by the noise. This is reasonable, since the introduced metric can not consider how much inferred rules are different from real rules. The inference process can, indeed, determine correctly at least part of the real rule, but the *Rule Score* does not take it into account, considering the predicate completely wrong. However, in the overall evaluation, it is possible to state that the methodology is quite stable to the noise and robust enough since the metrics results range on average around 80 – 70%.

7. Discussion

This section provides a discussion of the main results of this work, also highlighting current limitations. The first claimed point the definition of a formalism, based on MB approaches, capable of integrating knowledge extracted from data. PdFT indeed, starts from FT baseline, inheriting the tree-based structure and the concept of events and logical conditions, but at the same time moves toward an object-oriented view. One of the main innovation is the "layered structure" with the inter-component and intra-component view, conjugating the top-down view, focusing on relations with a bottom-up strategy that enhances the study of the inner behaviour of each single component. This object-oriented framework suits the logic of industrial systems, helping domain experts grasp the model, even if they are not familiar with this formalism.

Domain experts, indeed, play a central role, since the model seeks to conjugate domain knowledge with data. In an ideal application in an industrial scenario, PdFT model template should be produced by experts based on the topology of the system. From the template, it is possible to formulate hypotheses to be tested with the CALYPSO tool, obtaining the refined version of PdFT that integrates knowledge extracted from data. CALYPSO is in charge of implementing a methodology for extracting causal relationships from an event log in the context of PM. PM has emerged as a suitable candidate for integration as a DD, based on several factors: its relevance to industrial contexts, the process model's pivotal role in the analytical approach, and the format of data generated by industrial systems, specifically event logs. The proposed methodology has been evaluated on a proof of concept framed in CI context.

However, it is worth underlining some limitations. The metrics reported in Figs. 12 and 11 evaluate the quality of the inference process; in other words, they measure the ability to correctly identify causal relationships. the computational cost of such an approach could explode according to a growth of state space, considering all the possible hypotheses that can be formulated in a real-world context. In this paper, a sensitivity analysis of execution times is done according to the variation of dataset size, while an analysis following the variation of the complexity of the model has not been accomplished. Both the above mentioned limitations can be mitigated by the support of expert knowledge to ensure higher performance and lower execution times.

8. Conclusions

This paper presents a novel methodology for inferring failure processes and generating PdFTs from event logs, causality analysis to extract meaningful insights from real-world data.

More formally, $\Phi : S^1 \times S^2 \times \ldots \times S^n \longrightarrow \mathbb{N}$ where $\bar{s} = (\bar{s_1}, \ldots, \bar{s_n})$ is the n-uple in which $\bar{s_j} \in S^j$ represents the state considered for the component c_j. So that $\Phi(\bar{s})$ depends on the priority function $\pi^j(\bar{s}_j)$ computed on the actual state of the component and a minimal cost established by maintenance policies.

All these future work proposals can be framed into a single research effort, devoted to ease the representation and the automatic usage of expert knowledge. This would enable the overcoming of the limitations highlighted in Section 7.

CRediT authorship contribution statement

Roberta De Fazio: Writing – original draft, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization; **Benoit Depaire:** Writing – original draft, Validation, Supervision, Project administration, Methodology, Investigation, Conceptualization; **Stefano Marrone:** Writing – original draft, Validation, Supervision, Software, Methodology, Formal analysis; **Laura Verde:** Writing – original draft, Validation, Supervision, Investigation, Conceptualization.

Declaration of interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Roberta De Fazio reports financial support was provided by PON. Laura Verde reports financial support was provided by PON. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

The work of Roberta De Fazio is granted by PON Ricerca e Innovazione 2014/2020 MUR – Ministero dell'Università e della Ricerca (Italy) – with the PhD program XXXVII cycle D.M. N.1061 "Dottorati e contratti di ricerca su tematiche dell'Innovazione".

The work of Laura Verde is granted by the "Predictive Maintenance Multidominio (Multidomain predictive maintenance)" project, PON "Ricerca e Innovazione" 2014-2020, Asse IV "Istruzione e ricerca per il recupero"-Azione IV.4-"Dottorati e contratti di ricerca su tematiche dell'innovazione" programme CUP: B61B21005470007.

Appendix A. Running example

This section introduces some details about the running example reported in Section 4. The described scenario is composed of four elements:

The main engine C_1 is associated with a failure rate. It provides information on the average time after that C_1 is going to break, i.e. MTBF.

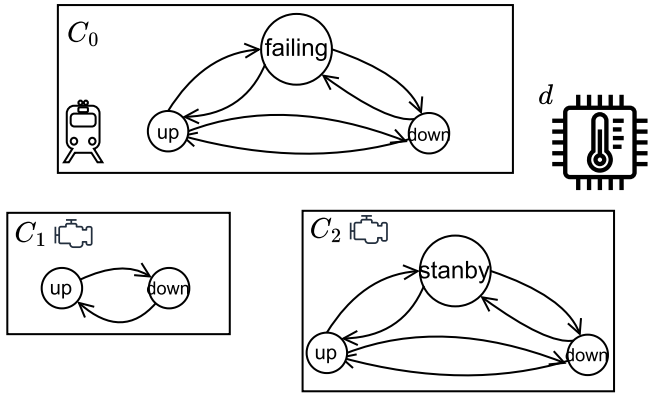


Fig. A.13. Train system running example.

The failure rate quantifies the likelihood of having a failure due to the operating time. C_2, instead, is a warm spare component. This kind of component is characterized by a "standby" state in which is not working, but ready to replace the main component. The failure rate $\alpha\lambda$ is characterized by a dormancy factor, that is $\alpha = 1$ when C_2 substitutes the main component C_1 [50]. Fig. A.13 depicts a simple model of the system.

Summing up:

- The component C_1 can assume two possible states: "up" and "down". It switches from "up" to "down" due to internal degradation, ruled by the failure rate. It switches from "down" to "up" due to maintenance action according to the repair rate.
- The component C_2 can assume three states, "up", "standby" and "down". It switches from "standby" to "up" if C_1 is in "down" state, and it switches back again to "standby" when C_1 is repaired. It switches from "up"/"standby" to "down" according to its failure rate and switches back according to its repair rate.
- The component C_0 can assume three states, "up", "failing" and "down". It switches from "up" to "failing" when only one of two engines is in the "down" state. In "failing" state C_0 continues its normal functioning, but it is in a possibly risky condition. If also the other engine goes down, indeed, C_0 switches from "failing" to "down". It switches back to "failing" state only once one of the two engines is repaired, and back to "up" state when both engines are repaired.

However, the degradation of the component could also be affected by other factors. In this scenario, for example, if the environmental temperature overcomes 70°, both the engines break at the same time. This leads to an instantaneous system failure, since both the engines are down. Fig. A.14a shows some trends in the data recorded by the sensor compared to the transition events on the components C_1 and C_2. Just focusing on the first peak, depicted in Fig. A.14b, it is possible to draw some conclusions: every time the temperature recorded by the sensor (represented by the blue line) reaches 70° the monitoring centre records the transition of both C_1 and C_2 (represented by the dotted red line) from "up"/"standby" to "down" state.

Then, after a brief period of deadlock, which ends only when external resources operate to repair the components, they return to "up" standby" state and suddenly the temperature decreases until it reaches the normal range between 24° and 27°. It is possible to state that, after 7 days of intensive working, the temperature increases until it reaches a threshold, after that both the engines go down at the same time and this provokes the train's system failure.

A.1. Dataset

The table here reported stores an excerpt of the dataset associated with the example introduced in Section 4.

Appendix B. PdFT

This section reports some details regarding the application of PdFT formalism to the running example introduced in Appendix B.

B.1. Trigger and action functions

In this version of the formalism, the only *impacting* transitions are the critical ones and their inverse, this enables the definition of the α function. For the sake of clarity, Table B.11 reports the value assigned by the *action* and *trigger* functions to the element of the running example:

B.2. Impacting transitions

In the proposed framework, an *Impacting Transition* propagates the internal behaviour of a component to the external environment, thus influencing the overall system. This effect is regulated by the *trigger* and *action* functions. In particular, when an impacting transition t moves the component c to its highest-priority state, the function α evaluates to *True*; conversely, when computed on its inverse transition $\bar{t}$, α evaluates to *False*. It is worth underlining the difference between the true (false) and *True* (*False*) values, used in the definitions reported above. *True* and *False* are the values returned by the α function: their meaning — as well as the *Neutral* value — has to be intended not in the boolean sense, but as simple labels. On the other hand, true and false are boolean values used as the resulting set for the trigger functions, as an example. To overcome this difference, a simple function, named β, is defined on the ports set $P^O \cup P^I$, as in Eq B.1. It is worth reminding that the β function is defined only for *impacting* transitions and, hence, it is not possible to have a *Neutral* value of the ports. Future work will extend this function also to the general case.

$$\beta(p) := \begin{cases} true \iff p = True \\ false \iff p = False \end{cases} \tag{B.1}$$

B.3. Evaluation function

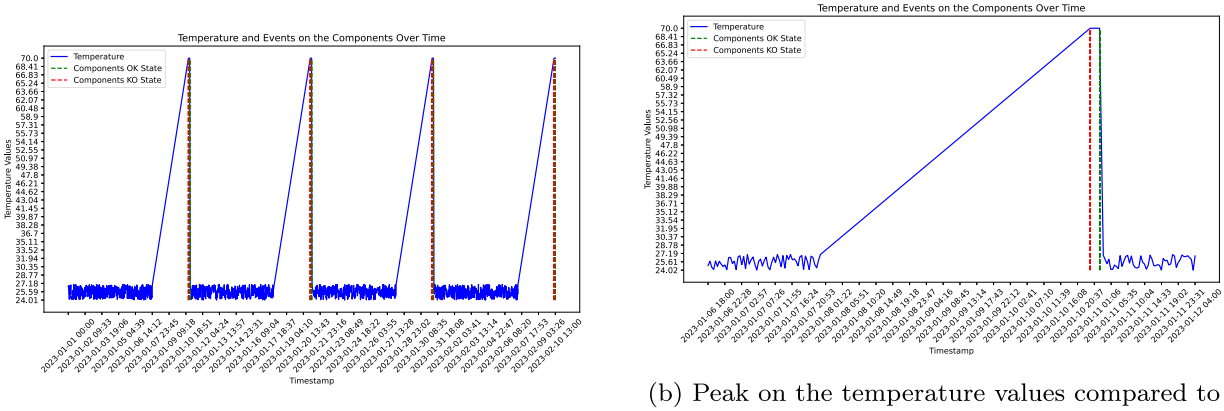
To better explain the logic of the *Evaluation function*, let us consider, as a possible example: the predicate associated with the transition $(down, up)$ on the component C_0. According to Table B.11, $\tau(C_0, (down, up)) = (\neg p_{0,1}) \wedge (\neg p_{0,2})$. The following schema evaluates the expression according to all the possible values assumed by the involved variables. It is worth noticing that we assumed a fixed value for the variable associated with the dynamic. This is without loss of generality, since the predicate does not depend on the value of the dynamic: whatever value it assumes, the function is not affected.

- $\tau(C_0, (down, up)) = (\neg p_{0,1}) \vee (\neg p_{0,2})$

The evaluation function computes the value of the predicates as follows:

- $\mu(\tau(C_0, (down, up)), (True, True), (24.007)) = (\neg p_{0,1}) \vee (\neg p_{0,2})\Big|_{\substack{p_{0,1}=true,\\ p_{0,1}=false}} = false$
- $\mu(\tau(C_0, (down, up)), (True, False), (24.007)) = (\neg p_{0,1}) \vee (\neg p_{0,2})\Big|_{\substack{p_{0,1}=true,\\ p_{0,1}=false}} = true$
- $\mu(\tau(C_0, (down, up)), (False, True), (24.007)) = (\neg p_{0,1}) \vee (\neg p_{0,2})\Big|_{\substack{p_{0,1}=false,\\ p_{0,1}=true}} = true$
- $\mu(\tau(C_0, (down, up)), (False, False), (24.007)) = (\neg p_{0,1}) \vee (\neg p_{0,2})\Big|_{\substack{p_{0,1}=false,\\ p_{0,1}=false}} = true$

The predicate $\tau(C_0, (down, up))$ can be associated with the following truth Table B.12:



(a) Trends in time-series temperature data.

(b) Peak on the temperature values compared to events on the components.

Fig. A.14. Temperature analysis in two time periods.

Table A.10
Excerpt of the dataset for the running example.

Occurence_ID	Element_ID	time:timestamp	Message_description	Value	CaseID
1000	C_1	20/04/2021 09:45:00	up		1
1001	C_2	20/04/2021 09:45:00	standby		1
1002	C_0	20/04/2021 09:45:00	up		1
…	…	…	…	…	…
1737	sensor	27/04/2021 17:45:00		27.25944	1
1738	sensor	27/04/2021 17:49:00		27.25971	1
1739	C_2	27/04/2021 17:50:00	down		1
1740	C_0	27/04/2021 17:50:01	failing		1
1741	sensor	27/04/2021 17:51:00		27.26789	1
1742	sensor	27/04/2021 17:53:00		27.24567	1
…	…	…	…	…	…
1761	C_2	27/04/2021 18:30:00	standby		1
1762	C_0	27/04/2021 18:30:01	up		1
…	…	…	…	…	…
6020	C_1	22/06/2021 16:40:00	up		10
6021	C_2	22/06/2021 16:40:00	standby		10
6022	C_0	22/06/2021 16:40:00	up		10
…	…	…	…	…	…
6788	sensor	29/06/2021 07:32:00		69.99088	10
6789	sensor	29/06/2021 07:34:00		70.00023	10
6790	C_1	29/06/2021 07:34:30	down		10
6791	C_2	29/06/2021 07:34:30	down		10
6792	C_0	29/06/2021 07:34:31	down		10
6793	sensor	29/06/2021 07:36:00		70.30001	10
…	…	…	…	…	…
6860	C_1	29/06/2021 09:50:30	up		10
6861	C_0	29/06/2021 09:50:31	failing		10
6862	C_2	29/06/2021 10:20:30	standby		10
6863	C_0	29/06/2021 10:20:31	up		10

B.4. A possible scenario

In the scenario depicted in Fig. 2, the main engine C_1 is in the down state, due to natural degradation, which triggered two events, propagating the effects on components C_2 and C_0. Let us analyse it through PdFT formalism using the notion of *trigger*, *alpha* and *evaluation* function introduced in Section 4. τ, i.e. the *trigger function*, associates with each couple component-transition, the boolean predicate that enables the transition. C_1, moving from "up" state to the highest priority "down" state, enables an impacting transition which, according to Table B.11, is described by the predicate $\tau(C_1,(up,down)) = (T > MTBF) \vee (d(T) > 70°)$. The first OR-clause is verified and thus implies turning the value of the predicate into `true` value. The α function sets the value on the port p_1 to `True`, according to Table B.11. In its turn, this enables two events: $e_0 = (p_1, p_{2,1})$ and $e_1 = (p_1, p_{0,1})$.

Impact of C_1 on C_2

One the value `True` is copied by the event e_0 on the input port $p_{2,1}$, the predicate $\tau(C_2,(standby,up)) = p_{2,1}$ is verified. This enables the transition in the component C_2 from "standby" state to "up" state, commissioning the spare engine and avoiding the system failure. Moreover, since the transition $(standby, up)$ is not a critical transition for C_2 — it does not involve the state with the highest priority—, it does not impact the overall system in terms of fault propagation. The output port of C_2, p_2 will be set on `Neutral` value.

Impact of C_1 on C_0

One the value `True` is copied by the event e_1 on the input port $p_{0,1}$, however the predicate $\tau(C_0,(up,down)) = p_{0,1} \wedge p_{0,1}$ is not verified, given that $p_{0,2}$ has not been set to `True`. This ensures that the system is still able to work; however, the main component turns in the "failing" state. Indeed, the predicate $\tau(C_0,(up,failing)) = p_{0,1} \vee p_{0,2}$ is verified (note that $\vee$ conditions need just one clause true to be verified). Again, $(up, failing)$ is not a critical transition for C_0 so the output port p_0 is set to `Neutral`. In this case, some warnings can be sent to maintainers for proceeding with some actions, avoiding a complete system failure.

The formal description of the scenario is the following:

Table B.11
Trigger and action function.

Component $c_i \in C$	Transition $t \in T^c$	Output Port P_c^O	Trigger $\tau(C, T^c)$	Action $\alpha(P_c^O, T^c)$
C_0	$(up, failing)$	p_0	$\tau(C_0,(up,failing)) = p_{0,1} \vee p_{0,2}$	$\alpha(p_0,(up,failing)) = Neutral$
C_0	$(failing, down)$	p_0	$\tau(C_0,(failing,down)) = p_{0,1} \vee p_{0,2}$	$\alpha(p_0,(failing,down)) = True$
C_0	$(up, down)$	p_0	$\tau(C_0,(up,down)) = p_{0,1} \wedge p_{0,2}$	$\alpha(p_0,(up,down)) = True$
C_0	$(down, failing)$	p_0	$\tau(C_0,(down,failing)) = (\neg p_{0,1}) \vee (\neg p_{0,2})$	$\alpha(p_0,(down,failing)) = False$
C_0	$(down, up)$	p_0	$\tau(C_0,(down,up)) = (\neg p_{0,1}) \wedge (\neg p_{0,2})$	$\alpha(p_0,(down,up)) = False$
C_0	$(failing, up)$	p_0	$\tau(C_0,(failing,up)) = (\neg p_{0,1}) \vee (\neg p_{0,2})$	$\alpha(p_0,(failing,up)) = Neutral$
C_1	$(up, down)$	p_1	$\tau(C_1,(up,down)) = (T > MTBF) \vee (d(T) > 70°)$	$\alpha(p_1,(up,down)) = True$
C_1	$(down, up)$	p_1	$\tau(C_1,(down,up)) = (T > MTTR)$	$\alpha(p_1,(down,up)) = False$
C_2	$(up, down)$	p_2	$\tau(C_2,(up,down)) = (T > MTBF) \vee (d(T) > 70°)$	$\alpha(p_2,(up,down)) = True$
C_2	$(down, up)$	p_2	$\tau(C_2,(down,up)) = (T > MTTR) \wedge p_{2,1}$	$\alpha(p_2,(down,up)) = False$
C_2	$(up, standby)$	p_2	$\tau(C_2,(up,standby)) = \neg p_{2,1}$	$\alpha(p_2,(up,standby)) = Neutral$
C_2	$(standby, up)$	p_2	$\tau(C_2,(standby,up)) = p_{2,1}$	$\alpha(p_2,(standby,up)) = Neutral$
C_2	$(down, standby)$	p_2	$\tau(C_2,(down,standby)) = (\neg p_{2,1}) \wedge (T > MTTR)$	$\alpha(p_2,(down,standby)) = False$
C_2	$(standby, down)$	p_2	$\tau(C_2,(standby,down)) = T > MTBF \vee d(T) > 70$	$\alpha(p_2,(standby,down)) = True$

Table B.12
Truth table defined by μ function on $\tau(C_0,(down,up))$ predicate.

$p_{0,1}$	$p_{0,2}$	$(\neg p_{0,1}) \vee (\neg p_{0,2})$
True	True	false
True	False	true
False	True	true
False	False	true

- $\tau(C_1,(up,down)) = (T > MTBF) \vee (d(T) > 70°) = true$
- $\alpha(p_1,(up,down)) = \texttt{True}$
- $e_0 = (p_1, p_{2,1}) \implies p_{2,1} = \texttt{True}$; $e_1 = (p_1, p_{0,1}) \implies p_{0,1} = \texttt{True}$
- $\tau(C_2,(standby,up)) = p_{2,1} = true$
- $\alpha(p_2,(standby,up)) = \texttt{Neutral}$
- $\tau(C_0,(up,failing)) = p_{0,1} \vee p_{0,2} = true$
- $\alpha(p_0,(up,failing)) = \texttt{Neutral}$

Appendix C. Definitions

This section contains additional material that provides formal definitions of concepts introduced in Section 5.

Definition 14. Let $\mathcal{A}_C$ be the Transition Hypotheses set and $\mathcal{A}_D$ be the Theta Hypotheses set, the Composed Hypotheses set is:

$$\bar{\mathcal{A}} = \{a = a_{i_1} * \cdots * a_{i_j} \mid a_{i_k} \in \mathcal{A}_C \cup \mathcal{A}_D \text{ for all } k \in \{1,\dots j\} \text{ and for all } j \in \{2,\dots,|\mathcal{A}_C \cup \mathcal{A}_D|\} \text{ with } * \in \{\wedge,\vee\}\}$$

Definition 15. A Transition Hypothesis $a = (c,t)$ where $t = (s_i, s_j)$ is observed on a component c in a process instance $N \in \varphi_{CI}(\mathcal{O})$, if:

$\exists\, o, \tilde{o} \in \mathcal{O}$ such that $(\varphi_{CI}(o) = \varphi_{CI}(\tilde{o}) = N) \wedge (\varphi_R(o) = \varphi_R(\tilde{o}) = c) \wedge$
$(\varphi_M(o) = s_i \wedge \varphi_M(\tilde{o}) = s_j) \wedge (\varphi_T(o) < \varphi_T(\tilde{o}))$
$\exists\, \bar{o} \in \mathcal{O}$ such that $(\varphi_{CI}(\bar{o}) = N) \wedge\ (\varphi_T(o) < \varphi_T(\bar{o}) < \varphi_T(\tilde{o})) \implies (\varphi_R(\bar{o}) \neq c)$

Definition 16. A Theta Hypothesis $a = (d, \theta_T(c,d))$ triggered by a dynamic d on a component c is observed in a process instance $N \in \varphi_{CI}(\mathcal{O})$:

$\exists\, o \in \mathcal{O}$ such that $(\varphi_{CI}(o) = N) \wedge (\varphi_R(o) = d) \wedge (\mathcal{F}_{c,d}(\varphi_M(o)) = true)$

Definition 17. A composed hypothesis $a = a_{i_1} \wedge \cdots \wedge a_{i_j} \in \bar{\mathcal{A}}$ is observed in a process instance $N \in \varphi_{CI}(\mathcal{O})$, if:

- $\exists\, o_{i_1}, \dots o_{i_j} \in \mathcal{O}$ such that $(\varphi_{CI}(o_{i_k}) = N) \wedge (\varphi_{\mathcal{A}}(o_{i_k}) = a_{i_k})\ \forall i_k \in \{i_1 \dots i_j\}$
- $\tilde{o} \in \{o_{i_1}, \dots o_{i_j}\}$ such that $\varphi_T(\tilde{o}) \geq \varphi_T(o_{i_k})\ \forall i_k \in \{i_1 \dots i_j\}$

Definition 18. A composed hypothesis $a = a_{i_1} \vee \cdots \vee a_{i_j} \in \bar{\mathcal{A}}$ is observed in a process instance $N \in \varphi_{CI}(\mathcal{O})$, if:

- $\exists\, \bar{o} \in \mathcal{O}$ such that $(\varphi_{CI}(\bar{o}) = N) \wedge (\varphi_{\mathcal{A}}(\bar{o}) = a_{i_k})$ with $i_k \in \{i_1 \dots i_j\}$
- if $\exists\, \tilde{o} \neq \bar{o} \in \mathcal{O}$ such that $(\varphi_{CI}(\tilde{o}) = N) \wedge (\varphi_{\mathcal{A}}(\tilde{o}) = a_{i_s})$ for some $i_s \in \{i_1 \dots i_j\} \implies \varphi_T(\tilde{o}) \geq \varphi_T(\bar{o})$

Definition 19. A hypothesis $a \in \mathcal{H}$, fits the *prima facie* first condition for an effect $f \in \mathcal{A}_C$ if and only if: $\exists\, n \in \varphi_{CI}(\mathcal{O})$ such that:

- $\exists\, o, \tilde{o} \in \mathcal{O}$ where $(\varphi_{CI}(o) = \varphi_{CI}(\tilde{o}) = n) \wedge (\varphi_{\mathcal{A}}(o) = a \wedge \varphi_{\mathcal{A}}(\tilde{o}) = f)$
- $\varphi_{\mathcal{T}}(\tilde{o}) - \varphi_{\mathcal{T}}(o) > 0 \wedge \varphi_{\mathcal{T}}(\tilde{o}) - \varphi_{\mathcal{T}}(o) < \delta$

This definition adapts Suppes's theory within the PM paradigm. However, the second condition is directly claimed from the first one:

$$Given\ \varphi_{CI}(o) = \varphi_{CI}(\tilde{o}) = n \implies \varphi_{\mathcal{T}}(\tilde{o}) - \varphi_{\mathcal{T}}(o) < \delta$$
$$\text{where } \delta \leq \max(\{\mathcal{T}(o) | o \in \mathcal{O} \wedge \varphi_{CI}(o) = n\}) - \min(\{\mathcal{T}(o) | o \in \mathcal{O} \wedge \varphi_{CI}(o) = n\}) \tag{C.1}$$

Moreover, Eq. 1 expresses the causal relation to discovering, introducing the parameter δ. It is possible to limit the value considered for the parameter, resulting in further filtering of the number of hypotheses that hold the property in Eq. (19) and consequently reducing the computational costs.

Definition 20. For each $a \in \mathcal{H}$, the hypothesis probability of a is given by the number of process instances in which this activity is performed, divided by the total number of process instances:

$$\mathbf{P}(a) = \frac{|\{n |\ \exists o \in \mathcal{O} \text{ such that } (\varphi_{\mathcal{A}}(o) = a \wedge \varphi_{CI}(o) = n)\}|}{|\varphi_{CI}(\mathcal{O})|} = \frac{|\varphi_{CI}(\mathcal{O}_a)|}{|\varphi_{CI}(\mathcal{O})|} \tag{C.2}$$

where $\mathcal{O}_a = \{o \in \mathcal{O} \mid \varphi_{\mathcal{A}}(o) = a\}$.

Definition 21. For each $a \in \mathcal{H}$, that holds the Definition 19 for an effect $f \in \mathcal{A}_C$, the hypothesis-effect probability of f given the occurrence of a, is given by the number of cases that holds *prima facie* first condition property, divided by the total number of process instances in which a occurs:

$$\mathbf{P}(f|a) = \frac{\mathbf{P}(a \wedge f)}{\mathbf{P}(a)} = \frac{|\mathcal{W}(f,a)|}{|\varphi_{CI}(\mathcal{O})|} \frac{|\varphi_{CI}(\mathcal{O})|}{|\varphi_{CI}(\mathcal{O}_a)|} = \frac{|\mathcal{W}(f,a)|}{|\varphi_{CI}(\mathcal{O}_a)|} \tag{C.3}$$

Definition 22. Let $a \in PF(f)$, and $x \in PF(f)$ with $x \neq a$, they are non-excluding for f and non-concordant if and only if:

- $\mathcal{W}(f,a) \cap \mathcal{W}(f,x) \neq \emptyset$

- $\varphi_{CI}(\mathcal{O}_x) \setminus \varphi_{CI}(\mathcal{O}_a) \neq \emptyset$ [16]

Let $\mathcal{L}_{a,f}$ be the set of the x that holds these properties.

Based on these considerations, Eq. (3) becomes Eq. (C.4).

$$\varepsilon_{avg}(a,f) = \frac{\sum_{x \in \mathcal{L}_{a,f}} \frac{|\mathcal{W}(f,a) \cap \mathcal{W}(f,x)|}{|\varphi_{CI}(\mathcal{O}_a) \cap \varphi_{CI}(\mathcal{O}_x)|} - \frac{|\mathcal{W}(f,x) \setminus \mathcal{W}(f,a)|}{|\varphi_{CI}(\mathcal{O}_x) \setminus \varphi_{CI}(\mathcal{O}_a)|}}{|\mathcal{L}_{a,f}|} \tag{C.4}$$

Eq. C.4 is well-posed since:

- if $\mathcal{L}_{a,f} = \emptyset$, Eq. (4) guarantees that Eq. (C.4) can be simplified in $\frac{|\mathcal{W}(f,a)|}{|\varphi_{CI}(\mathcal{O}_a)|} - \frac{|\varphi_{CI}(\mathcal{O}_f) \setminus \varphi_{CI}(\mathcal{O}_a)|}{|\varphi_{CI}(\mathcal{O}) \setminus \varphi_{CI}(\mathcal{O}_a)|}$;
- $\varphi_{CI}(\mathcal{O}_a) \cap \varphi_{CI}(\mathcal{O}_x) \neq \emptyset$ since $\mathcal{W}(f,a) \cap \mathcal{W}(f,x) \neq \emptyset$ and $\mathcal{W}(f,a) \cap \mathcal{W}(f,x) \subseteq \varphi_{CI}(\mathcal{O}_a) \cap \varphi_{CI}(\mathcal{O}_x)$ (see Defination 22);
- $\varphi_{CI}(\mathcal{O}_x) \setminus \varphi_{CI}(\mathcal{O}_a) \neq \emptyset$ by definition (see Defination 22).

Appendix D. Algorithms

In this section, a formal description of the Algorithms implemented by CALYPSO tool is provided:

- Algorithm 3 implements the transition hypotheses set construction.
- Algorithm 4 implements the theta hypotheses set construction.
- Algorithm 5 implements prima facie set construction.
- Algorithm 6 implements the filtering of significant causes[17].
- Algorithm 7 implements the generation of PdFT structure from the discovered cause-effect relations.
- Algorithm 8 implements the construction of *trigger* function from the discovered cause-effect relations.

Algorithm 3 $\bar{\mathcal{O}}_C$ set construction.

Require: $\mathcal{O}$, $\mathcal{A}_C$
1: $\bar{\mathcal{O}}_C \leftarrow \emptyset$
2: **for each** $a = (c, (s_i, s_j)) \in \mathcal{A}_C$ **do**
3: $\mathcal{O}_c := \{o \in \mathcal{O} \mid \varphi_R(o) = c\}$
4: **for each** $k \in \{1, \ldots, |\mathcal{O}_c| - 1\}$ **do**
5: **if** $(\varphi_M(o_k) = s_i) \wedge (\varphi_M(o_{k+1}) = s_j) \wedge (\varphi_{CI}(o_k) = \varphi_{CI}(o_{k+1}))$ **then** ▷ where o_k is the k-st value in $\bar{O}$ set
6: $\bar{o} \leftarrow (a, \varphi_T(o_{k+1}), \varphi_{CI}(o_{k+1}))$
7: $\bar{\mathcal{O}}_C \leftarrow \bar{\mathcal{O}}_C \cup \{\bar{o}\}$
8: **Return** $\bar{\mathcal{O}}_C$

Algorithm 4 $\bar{\mathcal{O}}_D$ set construction.

Require: $\mathcal{O}$, $\theta((\mathcal{C} \times D))$
1: $\bar{\mathcal{O}}_D \leftarrow \emptyset$
2: **for each** $\mathcal{F}_{c,d} \in \theta((\mathcal{C} \times D))$ **do**
3: $a \leftarrow (d, \theta_T(c,d))$
4: $\mathcal{O}_d := \{o \in \mathcal{O} \mid \varphi_R(o) = d\}$
5: **for each** $k \in \{1, \ldots, |\mathcal{O}_d| - 1\}$ **do**
6: **if** $\mathcal{F}_{c,d}(\varphi_M(o_k)) = true$ **then** ▷ where o_k is the k-st value in $\bar{O}$ set
7: $\bar{o} \leftarrow (a, \varphi_T(o_k), \varphi_{CI}(o_k))$
8: $\bar{\mathcal{O}}_D \leftarrow \bar{\mathcal{O}}_D \cup \{\bar{o}\}$
9: **Return** $\bar{\mathcal{O}}_D$

Algorithm 5 Prima facie causes of a given effect f.

Require: f, $\mathcal{H}$, $\mathcal{O}$, δ
1: $PF(f) \leftarrow \{\emptyset\}$
2: **for each** a in $\mathcal{H} \setminus \{f\}$ **do**
3: ▷ Computing prima facie 1st condition according to the Definition 19
4: $condition1 \leftarrow False$
5: **for each** o in $\mathcal{O}_a$ and $\tilde{o}$ in $\mathcal{O}_f$ where $\varphi_{CI}(o) = \varphi_{CI}(\tilde{o})$ **do**
6: **if** $(\varphi_{\mathcal{T}}(\tilde{o}) - \varphi_{\mathcal{T}}(o) > 0 \wedge \varphi_{\mathcal{T}}(\tilde{o}) - \varphi_{\mathcal{T}}(o) < \delta)$ **then**
7: $condition1 \leftarrow True$
8: **break**
9: ▷ Computing prima facie 2nd condition
10: **if** condition1 **then**
11: $p \leftarrow \mathbf{P}(a)$ ▷ according to the Definition 20
12: $condition2 \leftarrow (p > 0)$
13: ▷ Computing prima facie 3rd condition
14: **if** (condition1 $\wedge$ condition2) **then**
15: $p_m \leftarrow \mathbf{P}(f)$ ▷ according to the Definition 20
16: $p_c \leftarrow \mathbf{P}(f \mid a)$ ▷ according to the Definition 21
17: $condition3 \leftarrow (p_c > p_m)$
18: **if** (condition1 $\wedge$ condition2 $\wedge$ condition3) **then**
19: $PF(f) \leftarrow PF(f) \cup \{a\}$
20: **return** $PF(f)$

Algorithm 6 Algorithm for filtering *significant causes*.

Require: f, $PF(f)$, $\bigcup_{a \in PF(f)} \mathcal{L}_{a,f}$,
1: $SC(f) \leftarrow \{\emptyset\}$
2: $\nu \leftarrow 0$
3: **for each** a in $PF(f)$ **do**
4: $\varepsilon(a) = \emptyset$
5: **for each** x in $\mathcal{L}_{a,f}$ **do**
6: $\varepsilon(a) \leftarrow \varepsilon(a) \cup \{\varepsilon_x(a,f)\}$
7: $p\_value(a) \leftarrow ttest(\varepsilon(a), \nu)$
8: **if** $p\_value(a) < 0.05$ **then**
9: $SC(f) \rightarrow SC(f) \cup \{a\}$
10: **return** $SC(f)$

Algorithm 7 PdFT structure generation.

Require: $\mathcal{A}_C$, $SC := \{(a,f) \mid f \in \mathcal{A}_C \wedge a \in SC(f)\}$
1: $\mathcal{P}^I \leftarrow \emptyset$, $\mathcal{E} \leftarrow \emptyset$, $\alpha(\mathcal{P}^O \times T) \leftarrow \emptyset$
2: **for each** $f \in \mathcal{A}_C$ **do** ▷ assuming that $f = (c_i, t_i)$ with $c_i \in C$
3: $\mathcal{P}^{c_i}_I \leftarrow \emptyset$
4: **for each** $a \in SC(f)$ **do**
5: **if** $a \in \mathcal{A}_C$ **then** ▷ assuming that $a = (c_j, t_j)$ with $c_j \in C$
6: $\mathcal{P}^I_{c_i} \leftarrow \mathcal{P}^I_{c_i} \cup \{p_{i,j}\}$
7: $\mathcal{E} \leftarrow \mathcal{E} \cup (p_j, p_{i,j})$
8: $\alpha(p_j, t_j) \leftarrow \bar{v}$
9: $\alpha(\mathcal{P}^O \times T) \leftarrow \alpha(T \times \mathcal{P}^O) \cup \{\alpha(p_j, t_j)\}$
10: **else** ▷ assuming that $a = (d_j, v_j)$ with $d_j \in D$
11: $\theta_T(c_i, d_j) \leftarrow v_j$
12: $\mathcal{P}^I \leftarrow \mathcal{P}^I \cup \mathcal{P}^I_{c_i}$
13: **return** $\mathcal{P}^I$, $\mathcal{E}$, $\alpha(\mathcal{P}^O \times T)$

[16] where "\" stands for the sets subtraction operator.

[17] In a practical implementation of this algorithm, Line 7 could be supported by external libraries (e.g., the `stats.ttest_1samp` is from `scipy` Python library https://scipy.org/

Algorithm 8 PdFT trigger functions definition.

Require: $\mathcal{A}_C$, $Pr := \{Pr(f) = a_1 * \ldots * a_{\mathcal{I}(f)} \mid f \in \mathcal{A}_C\}$

$\tau(C \times T) \leftarrow \emptyset$
for each $f \in \mathcal{A}_C$ **do** ▷ assuming that $f = (c_i, t_i)$ with $c_i \in C$
  for each $j \in \{1, \ldots, \mathcal{I}(f)\}$ **do**
    if $a_j \in \mathcal{A}_C$ **then** ▷ assuming that $a = (c_j, t_j)$ with $c_j \in C$
      $\tau(c_i, t) \leftarrow \tau(c_i, t) * \gamma(p_{i,j})$
    else ▷ assuming that $a = (d_j, v_j)$ with $d_j \in \mathcal{D}$
      $\tau(c_i, t) \leftarrow \tau(c_i, t) * \mathcal{F}_{c_i,d_j}$
  $\tau(C \times T) \leftarrow \tau(C \times T) \cup \{\tau(c_i, t)\}$
return $\tau(C \times T)$

References

[1] Compare M, Baraldi P, Zio E. Challenges to IoT-enabled predictive maintenance for industry 4.0. IEEE Internet Things J 2020;7(5):4585–97. https://doi.org/10.1109/JIOT.2019.2957029

[2] Jung D, Ng KY, Frisk E, Krysander M. Combining model-based diagnosis and data-driven anomaly classifiers for fault isolation. Control Eng Pract 2018;80:146–56. https://doi.org/10.1016/j.conengprac.2018.08.013

[3] Jacky J, Veanes M, Campbell C, Schulte W. Model-Based Software Testing and Analysis with C#. 1 ed.; 2007. ISBN 9780521687614.

[4] Kleinberg S. Defining Causality. Cambridge University Press; 2012, p. 65–110.

[5] De Fazio R, Marrone S, Verde L, Reccia V, Valletta P. Towards an extension of Fault Trees in the Predictive Maintenance Scenario. arXiv e-prints 2024;. https://doi.org/10.48550/arXiv.2403.13785

[6] Abate C, Campanile L, Marrone S. A flexible simulation-based framework for model-based/data-driven dependability evaluation. 2020, p. 261–6. https://doi.org/10.1109/ISSREW51248.2020.00083

[7] Stefano Marrone. Dependability Simulation Engine. https://github.com/stefanomarrone/dse/tree/dse4calypso; 2024. Online; accessed 01 September 2025.

[8] Roberta De Fazio. Causality AnaLYsis for Prediction of System Operation. https://github.com/robidfz/CALYPSO; 2024. Online; accessed 01 September 2025.

[9] Arias Chao M, Kulkarni C, Goebel K, Fink O. Fusing physics-based and deep learning models for prognostics. Reliab Eng Syst Saf 2022;217:107961. https://doi.org/https://doi.org/10.1016/j.ress.2021.107961

[10] Tidriri K, Chatti N, Verron S, Tiplica T. Bridging data-driven and model-based approaches for process fault diagnosis and health monitoring: a review of researches and future challenges. Annu Rev Control 2016;42:63–81. https://doi.org/10.1016/j.arcontrol.2016.09.008

[11] Luo W, Hu T, Ye Y, Zhang C, Wei Y. A hybrid predictive maintenance approach for CNC machine tool driven by digital twin. Robot Comput Integr Manuf 2020;65:101974. https://doi.org/10.1016/j.rcim.2020.101974

[12] Sepe M, Graziano A, Badora M, Di Stazio A, Bellani L, Compare M, et al. A physics-informed machine learning framework for predictive maintenance applied to turbomachinery assets. J Global Power Propul Soc 2021;(May):1–15. https://doi.org/10.33737/jgpps/134845

[13] Arena S, Florian E, Zennaro I, Orrù PF, Sgarbossa F. A novel decision support system for managing predictive maintenance strategies based on machine learning approaches. Saf Sci 2022;146:105529. https://doi.org/10.1016/j.ssci.2021.105529

[14] Wang X, Liu M, Liu C, Ling L, Zhang X. Data-driven and knowledge-based predictive maintenance method for industrial robots for the production stability of intelligent manufacturing. Expert Syst Appl 2023;234:121136. https://doi.org/10.1016/j.eswa.2023.121136

[15] Cao Q, Zanni-Merk C, Samet A, Reich C, de Bertrand de BF, Beckmann A, et al. KSPMI: a knowledge-based system for predictive maintenance in industry 4.0. Robot Comput Integr Manuf 2022;74:102281. https://doi.org/10.1016/j.rcim.2021.102281

[16] Jia L, Chow T WS, Yuan Y. Causal disentanglement domain generalization for time-series signal fault diagnosis. Neural Netw 2024;172:106099. https://doi.org/10.1016/j.neunet.2024.106099

[17] Haasl DF, Roberts NH, Vesely WE, Goldberg FF. Fault tree handbook 1981;.

[18] Geymayr JAB, Ebecken NFF. Fault-tree analysis: a knowledge-engineering approach. IEEE Trans Reliab 1995;44(1):37–45. https://doi.org/10.1109/24.376519

[19] Gao P, Liu C, Dong H, Zheng W. A dynamic fault tree based CBTC onboard ATP system safety analysis method. In: 2020 IEEE 23rd international conference on intelligent transportation systems (ITSC). 2020, p. 1–7. https://doi.org/10.1109/ITSC45102.2020.9294605

[20] Mandelli D, Wang C, Agarwal V, Lin L, Manjunatha KA. Reliability modeling in a predictive maintenance context: a margin-based approach. Reliab Eng Syst Saf 2024;243:109861. https://doi.org/10.1016/j.ress.2023.109861

[21] Ruijters E. Zen and the art of railway maintenance. Ph.D. thesis; University Library/University of Twente; 2025 https://doi.org/10.3990/1.9789036545228

[22] Kleinberg S, Mishra B. The temporal logic of causal structures. In: Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence. UAI '09; Arlington, Virginia, USA: AUAI Press. ISBN 9780974903958; 2009, p. 303–12.

[23] Kabir S, Papadopoulos Y. Applications of bayesian networks and petri nets in safety, reliability, and risk assessments: a review. Saf Sci 2019;115:154–75. https://doi.org/10.1016/j.ssci.2019.02.009

[24] McGeachie MJ, Sordillo JE, Gibson T, Weinstock GM, Liu Y-Y, Gold DR, et al. Longitudinal prediction of the infant gut microbiome with dynamic bayesian networks. Sci Rep 2016;6(1). https://doi.org/10.1038/srep20359

[25] Ganian R, Korchemna V. The complexity of bayesian network learning: revisiting the superstructure. vol. 1. 2021, p. 430–42.

[26] Bouissou M, Bon J-L. A new formalism that combines advantages of fault-trees and markov models: boolean logic driven markov processes. Reliab Eng Syst Saf 2003;82(2):149–63. https://doi.org/10.1016/s0951-8320(03)00143-1

[27] Chiacchio F, D'Urso D, Compagno L, Pennisi M, Pappalardo F, Manno G. SHyFTA, a stochastic hybrid fault tree automaton for the modelling and simulation of dynamic reliability problems. Expert Syst Appl 2016;47:42–57. https://doi.org/10.1016/j.eswa.2015.10.046

[28] van der Aalst W, et al. Process mining manifesto. In: Business process management workshops. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-642-28108-2; 2012, p. 169–94.

[29] Rozinat A, van der Aalst W MP. Conformance checking of processes based on monitoring real behavior. Inf Syst 2008;33(1):64–95. https://doi.org/10.1016/j.is.2007.07.001

[30] Van Eck ML, Sidorova N, Van Der Aalst W MP. Composite state machine miner: discovering and exploring multi-perspective processes. vol. 1789. 2016, p. 73–7.

[31] Ruschel E, Santos E AP, de Freitas Rocha LE. Establishment of maintenance inspection intervals: an application of process mining techniques in manufacturing. J Intell Manuf 2018;31(1):53–72. https://doi.org/10.1007/s10845-018-1434-7

[32] Lowin M. A text-based predictive maintenance approach for facility management requests utilizing association rule mining and large language models. Mach Learn Knowl Extrac 2024;6(1):233–58. https://doi.org/10.3390/make6010013

[33] Bowes J, Neufeld E, Greer JE, Cooke J. A Comparison of Association Rule Discovery and Bayesian Network Causal Inference Algorithms to Discover Relationships in Discrete Data. Springer Berlin Heidelberg. ISBN 9783540454861; 2000, p. 326–36. https://doi.org/10.1007/3-540-45486-1_27

[34] Suppes P. A Probabilistic Theory of Causality. Amsterdam: North-Holland Pub. Co.; 1970.

[35] Nadim K, Ragab A, Ouali M-S. Data-driven dynamic causality analysis of industrial systems using interpretable machine learning and process mining. J Intell Manuf 2023;34(1):57–83. https://doi.org/10.1007/s10845-021-01903-y

[36] Nadim K, Ouali M-S, Ghezzaz H, Ragab A. Learn-to-supervise: causal reinforcement learning for high-level control in industrial processes. Eng Appl Artif Intell 2023;126. https://doi.org/10.1016/j.engappai.2023.106853

[37] Li J, Ma S, Le T, Liu L, Liu J. Causal decision trees. IEEE Trans Knowl Data Eng 2017;29(2):257–71. https://doi.org/10.1109/tkde.2016.2619350

[38] Vitale F, Guarino S, Flammini F, Faramondi L, Mazzocca N, Setola R. Process mining for digital twin development of industrial cyber-physical systems. IEEE Trans Ind Inf 2024;:1–10 https://doi.org/10.1109/tii.2024.3465600

[39] Van Houdt G, Martin N, Depaire B. Aitia-PM: discovering the true causes of events in a process mining context. Eng Appl Artif Intell 2023;126. https://doi.org/10.1016/j.engappai.2023.107145

[40] Van Houdt G, Depaire B, Martin N. Root Cause Analysis in Process Mining with Probabilistic Temporal Logic. Springer International Publishing. ISBN 9783030985813; 2022, p. 73–84. https://doi.org/10.1007/978-3-030-98581-3_6

[41] Chiacchio F, Iacono A, Compagno L, D'Urso D. A general framework for dependability modelling coupling discrete-event and time-driven simulation. Reliab Eng Syst Saf 2020;199:106904. https://doi.org/10.1016/j.ress.2020.106904

[42] Arena S, Roda I, Chiacchio F. Integrating modelling of maintenance policies within a stochastic hybrid automaton framework of dynamic reliability. Appl Sci 2021;11(5):2300. https://doi.org/10.3390/app11052300

[43] Shi L, Liu Y, Zhang Y, Liang J. Data-driven bayesian network analysis of railway accident risk. IEEE Access 2024;12:38631–45. https://doi.org/10.1109/ACCESS.2024.3376590

[44] Zilko AA, Kurowicka D, Goverde R MP. Modeling railway disruption lengths with copula bayesian networks. Transp Res Part C Emerg Technol 2016;68:350–68. https://doi.org/10.1016/j.trc.2016.04.018

[45] Huang W, Zhang Y, Kou X, Yin D, Mi R, Li L. Railway dangerous goods transportation system risk analysis: an interpretive structural modeling and bayesian network combining approach. Reliab Eng Syst Saf 2020;204:107220. https://doi.org/10.1016/j.ress.2020.107220

[46] Bayomie D, Awad A, Ezat E. Correlating unlabeled events from cyclic business processes execution. In: Nurcan S, Soffer P, Bajec M, Eder J, editors. Advanced information systems engineering. Springer International Publishing; 2016, https://doi.org/10.1007/978-3-319-39696-5_17

[47] De Fazio R, Balzanella A, Marrone S, Marulli F, Verde L, Reccia V, et al. CaseID detection for process mining: a heuristic-based methodology. Lecture Notes in Bus Inf Process 2024;503 LNBIP:45–57. https://doi.org/10.1007/978-3-031-56107-8_4

[48] Fisher RA. Statistical Methods for Research Workers. Springer New York. ISBN 9781461243809; 1992, p. 66–70. https://doi.org/10.1007/978-1-4612-4380-9_6

[49] Pappaterra MJ, Flammini F, Vittorini V, Bešinović N. A systematic review of artificial intelligence public datasets for railway applications. Infrastructures 2021;6(10). https://doi.org/10.3390/infrastructures6100136

[50] Bobbio A, Codetta R D. Parametric fault trees with dynamic gates and repair boxes. In: Annual symposium reliability and maintainability, 2004 - RAMS. IEEE; 2004, p. 459–65. https://doi.org/10.1109/rams.2004.1285491