Computers in Industry 166 (2025) 104253

Contents lists available at ScienceDirect

Computers in Industry

journal homepage: www.sciencedirect.com/journal/computers-in-industry

Collaborative fault tolerance for cyber–physical systems: The detection stage

Luis Piardi [a,c,d,*], André Schneider de Oliveira [c], Pedro Costa [d,e], Paulo Leitão [a,b]

[a] Research Centre in Digitalization and Intelligent Robotics (CeDRI), Instituto Politécnico de Bragança, Campus de Santa Apolónia, Bragança, 5300-253, Portugal
[b] Laboratório Associado para a Sustentabilidade e Tecnologia em Regiões de Montanha (SusTEC), Instituto Politécnico de Bragança, Campus de Santa Apolónia, Bragança, 5300-253, Portugal
[c] Universidade Tecnologica Federal do Paraná (UTFPR), Av. Sete de Setembro, Curitiba, 80230-901, Paraná, Brazil
[d] Faculty of Engineering University of Porto (FEUP), Porto, 4200-465, Portugal
[e] Institute for Systems and Computer Engineering, Technology and Science (INESC TEC), Porto, 4200-465, Portugal

ARTICLE INFO

Keywords:
Fault tolerance
Cyber–physical system
Multi-agent system
Fault detection

ABSTRACT

In the era of Industry 4.0, fault tolerance is essential for maintaining the robustness and resilience of industrial systems facing unforeseen or undesirable disturbances. Current methodologies for fault tolerance stages namely, detection, diagnosis, and recovery, do not correspond with the accelerated technological evolution pace over the past two decades. Driven by the advent of digital technologies such as Internet of Things, cloud and edge computing, and artificial intelligence, associated with enhanced computational processing and communication capabilities, local or monolithic centralized fault tolerance methodologies are out of sync with contemporary and future systems. Consequently, these methodologies are limited in achieving the maximum benefits enabled by the integration of these technologies, such as accuracy and performance improvements. Accordingly, in this paper, a collaborative fault tolerance methodology for cyber–physical systems, named Collaborative Fault * (CF*), is proposed. The proposed methodology takes advantage of the inherent data analysis and communication capabilities of cyber–physical components. The proposed methodology is based on multi-agent system principles, where key components are self-fault tolerant, and adopts collaborative and distributed intelligence behavior when necessary to improve its fault tolerance capabilities. Experiments were conducted focusing on the fault detection stage for temperature and humidity sensors in warehouse racks. The experimental results confirmed the accuracy and performance improvements under CF* compared with the local methodology and competitiveness when compared with a centralized approach.

1. Introduction

Cyber–physical systems (CPS) represent the core of Industry 4.0 (Lee et al., 2015) as a network of systems and comprise cyber counterparts with digitization, processing, and control, and physical counterparts containing elements such as sensors and actuators. Furthermore, the ability to exchange information among cyber components enables collaboration and represents an essential factor in the distribution of systems, providing scalability, flexibility, and robustness.

Owing to the inherent characteristics of CPS, such as the tight coupling of hardware and software capabilities and system heterogeneity (Boi-Ukeme et al., 2020), an associated complexity emerges in applications due to the increasing requirements for autonomy, coordination, and synchronization among different devices (Chaterji et al., 2019). Therefore, the development of theoretical and practical methods for deploying CPS, as well as integrating and maximizing the benefits of new technologies, such as Internet of Things (IoT), cloud and edge computing, and artificial intelligence (AI), are essential tasks that pave the way for systems in the future.

More sophisticated applications, such as zero-defect manufacturing and condition health monitoring (Caiazzo et al., 2022), require fault tolerance (FT) to ensure the robustness and resilience of the system. The fault tolerance refers to the system's ability to continue operating even in the presence of faults (Avizienis et al., 2004). This implies that the system can detect, diagnose, and recover from faults during the production process, thereby minimizing waste and defects in products resulting from disturbances or malfunctions. Because CPS are inherently complex owing to their distributed and cooperative characteristics, where they share information to execute operations, integrating fault tolerance in these systems is a challenging task that requires effort, and sensitivity and consumes significant computational resources, thereby increasing the challenges related to the FT (Piardi

* Corresponding author at: Research Centre in Digitalization and Intelligent Robotics (CeDRI), Instituto Politécnico de Bragança, Campus de Santa Apolónia, Bragança, 5300-253, Portugal.
E-mail address: piardi@ipb.pt (L. Piardi).

https://doi.org/10.1016/j.compind.2025.104253
Received 13 June 2024; Received in revised form 13 December 2024; Accepted 20 January 2025
Available online 30 January 2025

et al., 2020). Additionally, in a CPS scenario, a fault episode in one component can affect other components and even harm the overall system operation.

The main FT methodologies are typically deployed locally, i.e., each component locally integrates FT techniques, or is monolithic centralized, i.e., a central node is responsible for the fault tolerance of the CPS. Such practices are out of sync with the current technological reality, and even with the paradigm addressed by the CPS, which is based on distribution, autonomy, and collaboration. Although CPS adopt technologies that emerged with the fourth industrial revolution, implementing FT methodologies locally or centrally introduces a single fault point and is consequently limited in achieving the maximum benefits enabled by the integration of these technologies, such as improvements in accuracy, performance, and flexibility.

Accordingly, in this paper, a collaborative methodology for fault tolerance, entitled Collaborative Fault* (CF*), is presented, which introduces the self-fault tolerance capability in cyber components and adopts collaborative and distributed intelligence behavior when necessary to improve the asset FT. The CF* is organized into stages, namely, collaborative fault detection, collaborative fault diagnosis, and collaborative fault recovery, which represent the behaviors implemented in CPS components. These stages take advantage of the CPS's ability to analyze data and components communication to exchange information between the same or between different stages to improve the FT capabilities in the CPS compared to traditional methodologies, i.e., local or monolithic centralized.

For this purpose, the contribution of this study relied on the specification of the CF* methodology, highlighting the system components, behavior, collaboration parameters, functionalities, and interactions for fault tolerance in a decentralized and collaborative manner in a CPS based on the concepts of multi-agent systems (MAS), such as autonomy, intelligence, and interaction. The proposed methodology can be instantiate for different CPS applications and supports different detection, diagnosis and recovery models. This study emphasized the collaborative fault detection stage, detailing the dynamic behavior of Petri nets and agent unified modeling language (AUML) for interaction protocols. The proposed approach was experimentally tested and validated via a case study based on the racks of a mock-up warehouse, conducting fault detection experiments on the racks' temperature and humidity sensors, and comparing the detection performance in terms of precision, recall, and F1-score of the proposed CF* with the local and monolithic centralized methodologies. The CF* offers substantial benefits by addressing precision and performance improvements in CPS applications through collaboration between agents.

The remainder of this paper is organized as follows: Section 2 contextualizes fault tolerance in CPS. Section 3 introduces the CF* meta-model, and Section 4 presents the formal specification of the detection stage of CF* using Petri nets. In Section 5, the experimental results are presented and discussed. Finally, Section 6 presents conclusions and outlines future research work.

2. Fault tolerance in cyber–physical systems

CPS are promising because of their substantial long-term social, economic, and environmental benefits (Zhang et al., 2022). Faced with these benefits and rapid advances in information and communications technology (ICT), the industry has been adopting CPS-based solutions, where the component has intelligence and collaborative capabilities (Hastbacka et al., 2021; Lee and Kundu, 2022). CPS can cover a broad spectrum of applications, such as the automation of industrial production, energy generation and distribution systems, healthcare, autonomous vehicles, logistics systems, and Internet of Things (IoT) sensor networks (Pivoto et al., 2021).

As highlighted in several surveys (Bayar et al., 2015; Piardi et al., 2020), there is a lack of formalized architecture that integrates and adopts collaboration between cyber–physical components as a key factor in improving fault detection, diagnosis, and recovery at the local (i.e., in each component) and global (i.e., system-wide) levels. CPS can be affected by both cyber and physical faults. The cyber part can be affected by logical faults (e.g., processing problems), communication (e.g., network disconnection), and project design (e.g., software bugs). The physical part can be affected by the calibration or breakage of sensors, defects, actuator deterioration or electrical discontinuities (Jan et al., 2021).

Currently, the academic community is dedicated to developing new techniques or algorithms for FT deployment in CPS applications. This effort addresses the main techniques related to data-driven, knowledge -based, and model-based fault detection and diagnosis applied to industrial CPS, specifically in the automotive, aerospace, and industrial control fields, highlighting the main challenges presented in the survey (Dowdeswell et al., 2020). Recent advancements in data-driven techniques have demonstrated significant potential for improving fault tolerance in CPS. For example, the data-driven total measurable fault information residual (ToMFIR) approach has shown significant promise for incipient fault detection and diagnosis. For instance, it has been successfully evaluated in high-speed rail train suspension systems, showcasing its effectiveness in identifying faults and enhancing system robustness (Wu et al., 2024a,b). Additionally, recent research related to FT adopts intelligent algorithms to enhance the system's robustness, for example, using artificial neural networks with an in situ methodology to detect CNC faults with real industrial datasets (He et al., 2023), approaches using the support vector machine (SVM) algorithm with online training as a fault tolerant control to adapt to the operating conditions of an industrial plant in the presence of a fault (He et al., 2023), or implementing a deep-learning algorithm to detect and diagnose the degradation of rotating machines (Li et al., 2019). These intelligent algorithms are integrated into CPS based on traditional methodologies, namely, monolithic centralized or local methodologies. This scenario raises a question related to the appropriateness of these algorithms or techniques relying on traditional methodologies for detection, diagnosis, and recovery, or instead taking advantage of the collaboration presented by CPS.

Three different FT methodologies are widely used in CPS, namely local fault tolerance (LFT), monolithic centralized fault tolerance (MCFT), and decentralized fault tolerance (DFT), as illustrated in Fig. 1.

LFT is a FT methodology for embedded systems that integrates independent detection and diagnostic techniques for each device (Safari et al., 2022). Therefore, each component is locally responsible for detecting and diagnosing faults based on integrated techniques that analyze the individual device context and generate alerts for the maintenance team to conduct the recovery stage. There is no information exchange or interaction to mitigate faults. Several applications adopt this methodology due to its simple development, for example, the development of machine-condition monitoring embedded in low-cost micro-controllers (Grethler et al., 2021). Currently, LFT can integrate recent technologies, such as AI and machine learning (ML) for accuracy improvement. This methodology is responsive quickly because it deals with small data volumes and is suitable for scalability. However, it has drawbacks, primarily when a global or generalized fault occurs, that is, a fault with same nature is detected, diagnosed, and reported by several system components, or presents difficulties in adapting to situations in which operating conditions change.

With the advent of CPS, centralized industrial systems have been enhanced, and the MCFT methodology has gained prominence and become popular for fault tolerance in CPS (Moussa et al., 2019; Reijnen et al., 2021). Each physical component has its data digitized to the cyber part of the system through sensors that adopt IoT technology. Equipped with high computational power, advanced techniques, and AI algorithms, the cyber layer processes and analyzes these data to control their physical components. In addition, the cyber layer sends raw data to a central node or server, which stores data from all components and uses data-driven, knowledge-based, and model-based techniques

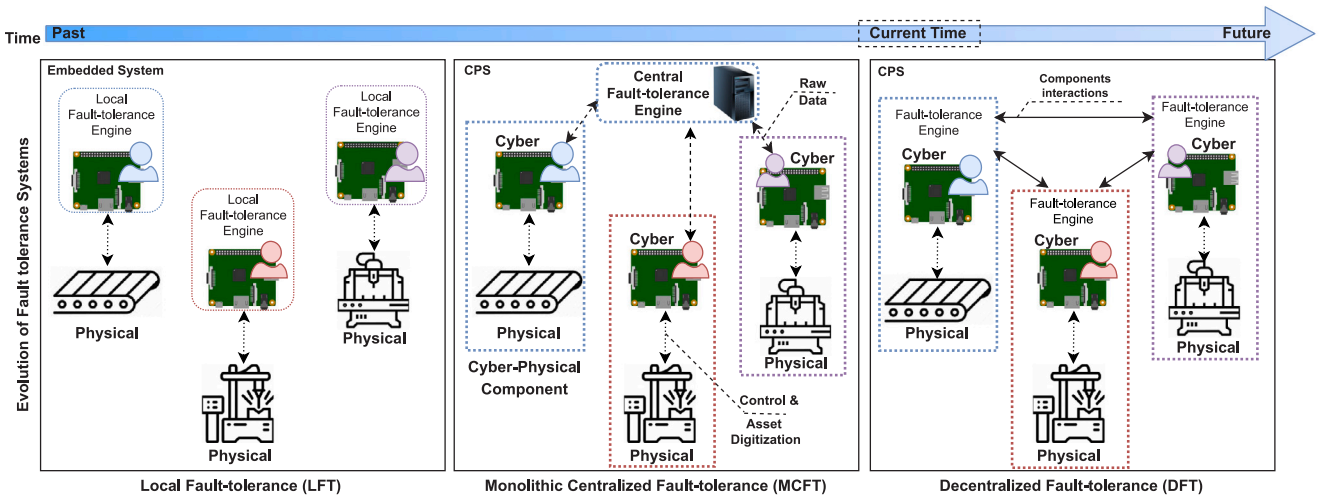

Fig. 1. Timeline of different methodologies for fault tolerance in the context of cyber–physical systems.

to detect and diagnose faults globally. When a fault is diagnosed, the recovery stage starts; for example, executing self-calibration of control parameters or sending a report to the maintenance team to make the necessary corrections, adjustments, repairs, or replace damaged equipment (Ribeiro and Barata, 2011). This approach has the advantage of simple deployment and provides support easily, because all stages run in a unique logical unit or server, simplifying algorithm updates and bug fixes. However, its main drawbacks include the following: (i) the central node must deal with a large amount of raw data, which can cause data loss and reduced responsiveness when detecting and diagnosing faults; (ii) difficulties in system scalability due to processing and memory limitations, which directly affect the reactivity of detection and diagnosis, as well as fault recovery; (iii) monolithic policies can compromise the accuracy and confidence of fault detection and diagnosis because of the adoption of uniform classification rules across diverse equipment, disregarding the operational context.

DFT integrates the collaboration between distributed components with FT models. This proposal implements FT methods locally for each cyber component and sends information regarding the status and context to the other components. Component resilience can be improved based on the local data received from sensors in the physical part and contextualized with data from other components. For example, multiple mobile robots share their status and context to avoid congestion in navigation during the transport of products in logistics warehouses and to avoid planning routes with waypoints where robots have stopped (i.e., failed), interrupting the path (Piardi et al., 2022). Another example is consensus-based voting to detect faults, in which edge devices with ML algorithms perform consensus voting on each data observation to detect faults in temperature sensors, thereby improving the detection accuracy compared to local approaches (Fidelis et al., 2022).

FT design in CPS is not limited to improving detection, diagnosis, and recovery models with intelligent algorithms. FT techniques in CPS must consider the ecosystem in which collaboration between different devices translates into local and global benefits, thereby improving the individual capacity of components as well as that of the system as a whole. DFT is still in its embryonic phase and requires a detailed and comprehensive specification of how a CPS can take advantage of the benefits of recent technological advances in collaborative behavior.

3. Collaborative fault tolerance for CPS

In this section, a methodology for distributed and collaborative fault tolerance in CPS, CF*, is proposed, which uses a multi-agent system as the infrastructure to provide distributed intelligence and collaboration capabilities to the CPS key components. These key components, which perform vital system functions and may include interconnected subsystems, are essential for maintaining the CPS's overall fault tolerance and robustness while possessing adequate computational capacity to implement FT algorithms. The CF* approach structures the fault tolerance in stages, namely, detection, diagnosis, and recovery, as illustrated in Fig. 2, and presents a strategy for collaboration and information exchange among agents to improve the FT robustness, flexibility, and accuracy performance in CPS.

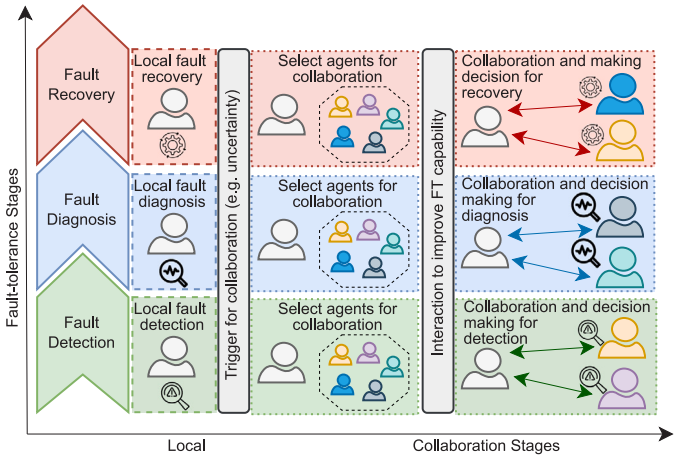

Fig. 2. Fault tolerance stages related to collaborations.

3.1. CF* meta-model description

The CF* approach considers a CPS comprising multiple components (i.e., agents), each possessing autonomous, intelligent, and collaborative characteristics for fault tolerance. While capable of operating independently, these agents can collaborate when needed to enhance their FT capabilities through agent interaction protocols, culminating in collective intelligence. The techniques and algorithms inherent in each agent are adaptable, can be tailored according to the application, and may encompass, for example, statistical or AI models. The proposed methodology is designed to maintain scalability, adaptability, and robustness, providing a comprehensive and resilient solution for CPS applications where reliability and accuracy are paramount.

As illustrated in Fig. 3, each agent can autonomously detect, diagnose, and recover from a fault. However, under certain conditions, such as uncertainty, each agent can request collaboration from a set

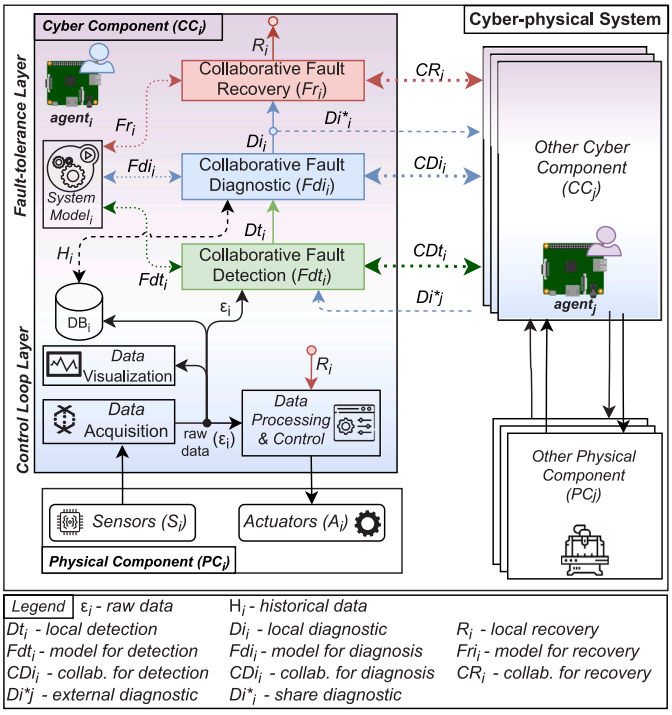

Fig. 3. CF* meta-model with the different FT stages and the possibility of collaboration between stages.

of selected agents. For the detection and diagnosis stages, this collaboration occurs by sending data, where a set of agents perform their analysis based on their local models and respond with their evaluation. Considering that each agent has its own intelligence capacity, such as an algorithm trained with different parameters or specific datasets, the collaboration will result in a set of evaluations. Through decision-making based on the evaluations, the agent that initiates the collaboration determines whether the shared data is a fault (in the case of detection) or the type of fault (in the case of diagnosis). During the recovery stage, through collaboration, the agent can define strategies to overcome faults that have already been detected and diagnosed.

The $agent_i$ is formed by the union of a physical component (PC_i) and a cyber component (CC_i). PC_i may have a set of sensors S_i that digitize the information to be processed and a set of actuators A_i to interact with the environment. Therefore, a CPS is a set of n components that can be represented by the union of the cyber component (CC) and physical component (PC) using the following notation:

$$CPS = \bigcup_{i=1}^{n} CC_i \cup \bigcup_{i=1}^{n} PC_i \quad \forall \quad n \in \mathbb{N} \tag{1}$$

Fig. 3 shows that individual CC_i comprises different modules. The *Data Acquisition*, *Data Processing & Control*, and *Data Visualization* modules are part of the control loop layer, representing the digitization of data to execute local PC_i control, sending commands to the A_i actuators to reproduce the expected behavior. In the FT layer, every CC_i has a set of models for detection (Fdt_i), diagnosis (Fdi_i), and recovery (Fr_i) provided by the *System Model*$_i$. These models can be unique for each agent and can be distinguished by, for example, different AI algorithms, parameters, and training datasets. Furthermore, *System Model*$_i$ receives feedback from the different FT stages (hence, the bidirectional arrows in Fig. 3), which is useful for retraining or updating the detection, diagnosis, and recovery models as new data are processed and collaborations are executed. All raw data ϵ_i, produced from digitization by sensors installed in the PC_i, is stored in database DB_i, which is useful for diagnosis to determine the classification or for the system model to complement the retraining and updating.

Table 1
Description of the collaboration parameters.

Parameter	Symbol	Description
Trigger	τ	Determines ***when*** the agent should collaborate
Selection	σ	Determines ***who*** the agent should collaborate with
Decision make	ρ	Determine ***how*** evaluation of others agents should be considered

For each agent to be self-fault tolerant and collaborate to improve its tolerance when needed, CF* defines collaboration parameters τ, σ, and ρ as described in Table 1. These parameters are integrated in each agent and are useful for directing when, who, and how the collaboration should be carried out. According to the representation of each parameter, it is possible to adjust the collaborative behavior that directly reflects on the collective intelligence of the CPS. For example, the collaboration trigger τ could be based on the degree of uncertainty in the inference. Selection parameter σ could be used to select the most similar or closest agents. Decision-making parameter ρ could be based on the most voted decision or weighted according to the precision of each agent's detection model participating in the collaboration.

In complex CPS environments with numerous agents, the parameter σ governs the selection of the most appropriate agents for collaboration. This dynamic selection process ensures that the initiating agent identifies the most relevant peers and the number of participants that can contribute to collaborative fault detection, diagnosis, or recovery, thereby preventing unnecessary overload and optimizing resource utilization. Additionally, the ρ parameter is crucial in guiding decision-making, enabling each agent to assess the context and peer feedback and adapt its strategy to enhance collaborative fault tolerance.

3.2. Detection, diagnosis, and recovery

In the detection stage, fault occurrences in a CPS are identified. Faults should be detected as soon as they occur to avoid considerable impact; therefore, the models responsible for detection should not be computationally demanding. Precise fault detection is pivotal because it sets in motion the diagnostic and recovery stages that are critical to FT in CPS. Without an efficient fault detection stage, faults can propagate through the system, leading to more severe disruptions and complicating recovery efforts. Considering these aspects, detection must minimize false alarms and undetected faults. As shown in Fig. 3 the fault detection (Dt_i) output of $agent_i$ is generated when the detection model (Fdt_i) identifies a fault in the raw data (ϵ_i) obtained from the $DataAcquisition_i$ module or when receiving a diagnosis (Di_j^*) from another CC_j. If model Fdt_i has detection uncertainties, a collaboration process is conducted with other agents to detect the fault (CDt_i) initiated by $agent_i$. Therefore, it is possible to express Dt_i as follows:

$$Dt_i = Fdt_i(\epsilon_i, CDt_i(\tau, \sigma, \rho), Di_j^*) \tag{2}$$

where ϵ_i represents an observation from sensor S_i and CDt_i denotes the collaboration model used when $agent_i$ is uncertain about classifying the observation ϵ_i as a fault. CDt_i depends on the collaboration parameters. Finally, fault detection considers the fault diagnosis Di_j^* from another $agent_j$, which can impact $agent_i$.

After detecting a fault, the $agent_i$ starts the diagnosis stage, whose functionality is to mitigate the cause and nature of the fault, such as determining which sensor or actuator failed and classifying the type of fault that occurred. Furthermore, it is essential to minimize fault misclassifications. Compared with the detection stage, this stage has more complex models that require more computational resources. The fault diagnosis by $agent_i$ is denoted as follows:

$$Di_i = Di_i^* = Fdi_i(Dt_i, H_i, CDi_i(\tau, \sigma, \rho)) \tag{3}$$

where the fault diagnosis output (Di_i) consists of a set of fault types and characteristics. The fault diagnosis model (FDi_i) considers the fault detection Dt_i information and the previous fault history (H_i), for example, a table of possible faults, with causes and nature. If $agent_i$ cannot classify the fault due to uncertainties, the collaboration process for detecting CDi_i is initiated, where parameters τ, σ, and ρ are used according to the idea presented in the detection stage. After diagnosing the fault (Di_i), a report is sent to the recovery stage of local agent and to the detection stages of other $agent_j$, as represented by Di_i^*, to assess whether the diagnosed fault impacts them. For example, when a mobile robot fails and congests a waypoint, it can alert other mobile robots that it failed at said waypoint. Consequently, the robots that receive this information can evaluate whether they are impacted by the fault and, if needed, execute a new path-planning procedure to avoid congestion.

The recovery stage begins after diagnosis. Recovery must provide a recommendation or action plan so that $agent_i$ can perform its operation even in the presence of a fault episode. The recovery action R_i can be described as follows:

$$R_i = Fr_i(Di_i, CR_i(\tau, \sigma, \rho)) \tag{4}$$

The recovery model Fr_i consider the fault diagnosis Di_i and possible collaboration with other agents CR_i. Examples of recovery actions include operational parameter reconfiguration, system rollback, restart and rejuvenation, requesting an idle agent to perform the operation, or requesting human maintenance and specifying the part to be replaced.

Based on the agent concept, the proposed methodology specifies the component for the CPS FT in a generic manner. This methodology should be represented in applications that use CPS as a paradigm for distributed and intelligent systems via the detection, diagnosis, and recovery models, as well as the collaboration parameters τ, σ, and ρ defined by the designer according to the application. The ability to collaborate at different stages reproduces the collective intelligence resulting from different detection, diagnosis, and recovery models and collaboration parameters. The agents adopts a different model and provides a collaborative dynamic to the system. Furthermore, because there is no centralization or hierarchy, the system becomes naturally scalable with plug-and-play features and removes a single fault point.

4. Agent dynamic behavior specification for CF*

The design and conceptualization of distributed CPS, considering control, fault tolerance, and collaboration are complex tasks that require exhaustive validation to detect design errors before implementation. To avoid design errors in these systems, it is recommended to adopt formal modeling methods whose objective is to formalize the structure and behavior of each agent, aiming to simplify their understanding and synthesis as a reference for implementation. Formal modeling should capture characteristics, such as deadlocks, concurrency, asynchronous tasks, parallelism, and conflicts inherent in distributed systems. This section addresses the specification of the generic dynamic behavior and collaboration protocols for agents, focusing on the fault detection stage, using Petri nets (PN) formalism and an agent unified modeling language (AUML) sequence diagram.

4.1. Generic agent behavior with CF*

Fig. 4 shows the PN models that describe the agent dynamic behavior, which comprises a thread for the control loop and a thread for the FT loop, processed in parallel to avoid mutual dependence or delay.

As shown in Fig. 4, the agent is ready to start its operation at p_1, waiting for the start to trigger transition t_1. After starting, the agent waits at p_2 for the sensor data to control the physical component. In parallel, two tokens are created, the first is created in p_5 and is responsible for receiving and processing messages from other agents regarding collaboration in fault detection, diagnosis, and recovery. The second token is created in p_8 to manage the processing resources

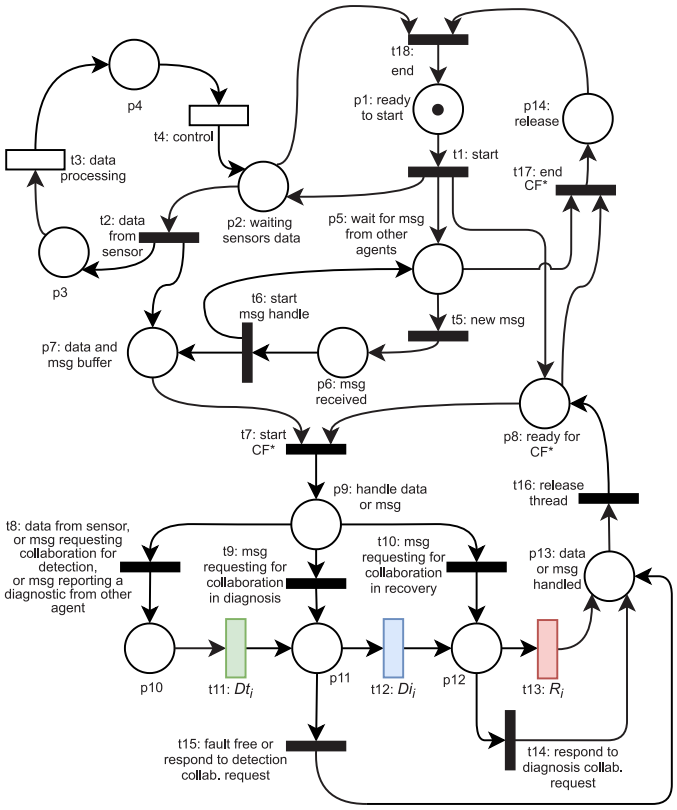

Fig. 4. Agent modeled via Petri nets integrating CF*.

available to execute CF*. To prevent loss of sensor data or collaborative messages caused by CF* processing, a buffer in p_7 is used to store data and messages to be processed when p_8 has an available token.

When tokens are present in $p7$ and $p8$, at $t7$ transition is triggered. The place $p9$ is specific for handling the transitions of the different CF* stages, indicating which stage the agent will execute. In this step, three transition possibilities exist, and the decision is made based on the token context. For example, suppose that the token at $p7$ originates from the data received from the sensor. In this case, performing an inference in the fault detection stage is necessary; therefore transition $t8$ is triggered. After $p10$, transition $t11$ is triggered to detect the fault (this transition is expanded to explain the detection in detail). If a fault is identified at $p11$ a diagnosis is made with transition $t12$ and recovery at $t13$. However if ($t11$) not detect a fault, there is no need to diagnosis and recovery; therefore, the token in $p11$ will perform the sequence $[t15, p13, t16, p8]$, where the agent will be able to process another token that is in $p7$. The same situation occurs when another agent reports a diagnosis and the local agent must check whether this diagnosed fault affects it (see Fig. 3, specifically Di_j^*).

However, if the context of the token in $t7$ originates in a message from another agent requiring collaboration for any of the CF* stages, an appropriate transition is triggered when the token is in $p9$. For example, if the agent receives a message requesting collaboration for the diagnosis (as shown in Fig. 3 by CDi_i), on this occasion, transition $t9$ is triggered after $p11$ executes transition action $t12$ responding to the collaboration request through the sequence $[p12, t14, p13, t16, p8]$. If, for example, the context of the token in $p9$ is a request for collaboration in detection, the sequence will be $[t8, p10, t11, p11, t15, p13, t16, p8]$; in case of a collaboration request for recovery, the sequence will be $[t10, p12, t13, p13, t16, p8]$.

The agent's dynamic behavior ends when transition $t17$ is triggered, ending the CF* procedure; then, transition $t18$ is triggered, ending the agent's control thread.

4.2. Collaborative fault detection details

Fig. 5 shows the PN refinement resulting from the expansion of transition $t11$. The detection has four main capabilities: (i) the local ability to detect faults when possible; (ii) the ability to request collaboration when fault uncertainty occurs (transitions and places highlighted in solid green); (iii) the ability to respond to a collaboration request (transitions and places highlighted with dashed green) by analyzing the data with your local model; and (iv) the ability to check whether the fault diagnosed and reported by another agent can affect it (transitions and places highlighted in dashed blue).

Initially, in transition $t11.t1$, the request for fault detection is managed according to the token context. Suppose the context refers to new sensor data, the $t11.t2$ transition prepares the data; then, transition $t11.t3$ performs the inference according to the defined Fdt_i fault detection model. In place $t11.p3$, there are three different sequence fault detection sequence possibilities. The first occurs when a fault is detected outside the τ collaboration criteria. In this situation, transition $t11.t4$ is triggered, resulting in a fault state in place $t11.p4$, to later report the detected fault to the diagnosis stage according to the sequence $[t11.t21, t11.p10, t11.t24, p11]$. The second possibility occurs when no fault outside the τ collaboration criteria is detected. Transition $t11.t9$ is triggered, resulting in a normal operating state ($t11.p10$), followed by the sequence $[t11.t22, t11.p10, t11.t24, p11]$. The third possibility begins when the collaboration trigger (τ) is fired on transition $t11.t5$. Then, in transition $t11.t6$, the agents for collaboration are selected according to parameter σ. After exchanging data

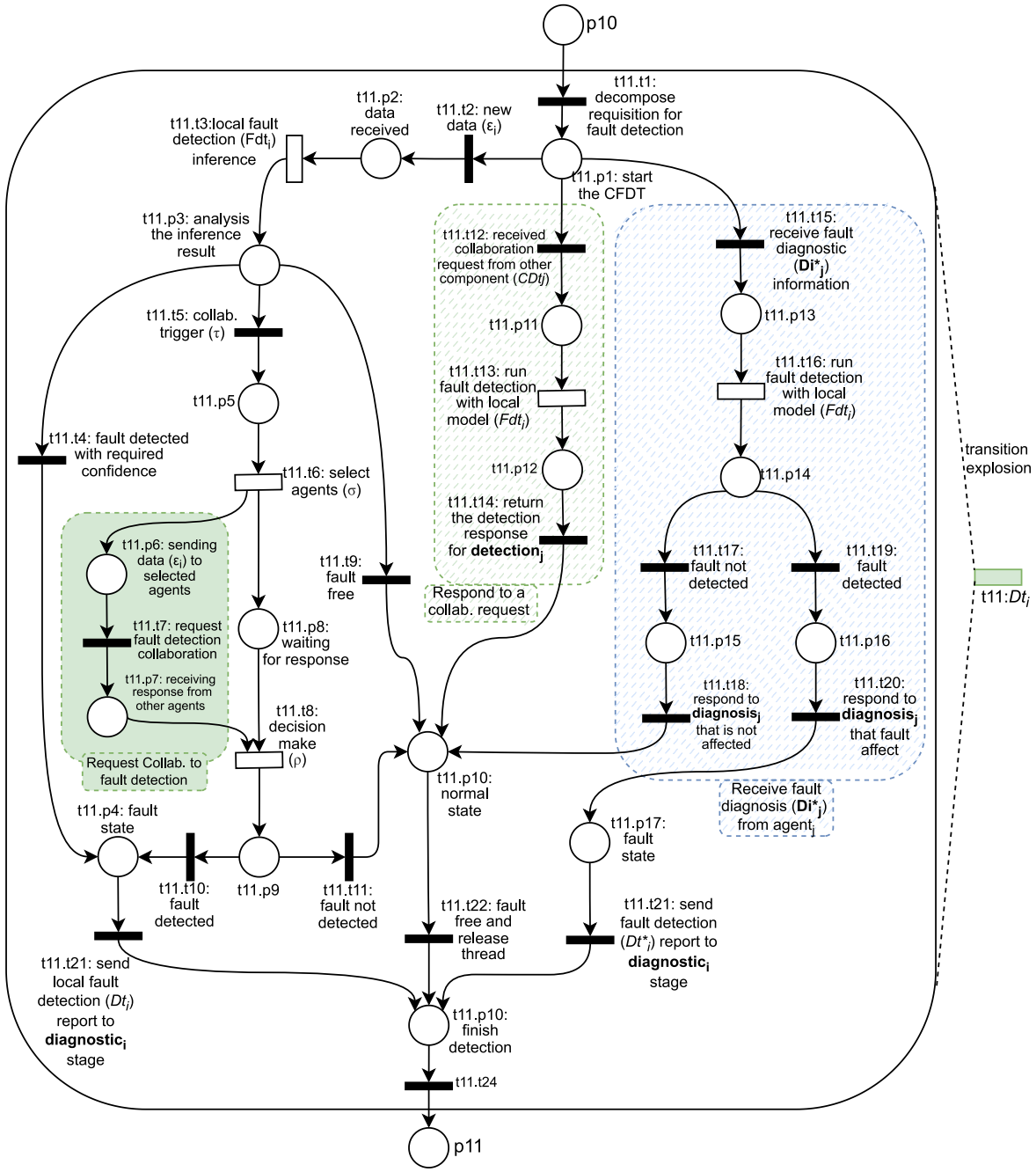

Fig. 5. Dynamic model refinement for the detection stage in the proposed CF* methodology. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

for the collaboration, $[t11.p6, t11.t7, t11.p7]$ the local agent, in transition $t11.t8$, evaluates the collaborations and decides according to parameter ρ. If the decision results in fault detection, the sequence $[t11.t10, t11.p4, t11.t21, t11.p10, t11.t24]$ follows, updating the agent status to the fault state and reporting the fault detection to the diagnostic stage.

However, the trigger on $t11.t1$ can be performed by a token with a context requiring collaboration, in which case, after triggering $t11.t1$, the token in place $t11.p1$ triggers transition $t11.t12$ transition, starting the sequence to respond to a collaboration request. Transition $t11.t13$ executes the fault detection model of local agent Fdt_i using the data from the agent that initiated the collaboration. In the transition $t11.t14$, the inference result is returned along with the model accuracy and precision, ending with the sequence $[t11.p10, t11.t22, t11.p10, t11.t24, p11]$.

Finally, the token that triggers on $t11.t1$ may have a fault report context generated by the diagnostic stage of another agent. In this case, after triggering $t11.t1$, token in place $t11.p1$ triggers transition $t11.15$, which receives a diagnosis from another agent. If no fault is detected, the agent's operating state remains normal. Otherwise, a fault state is assigned before proceeding to the diagnosis stage.

4.3. Interaction protocols for fault detection

The collaboration between agents for fault detection follows interaction protocols based on FIPA-ACL (Poslad, 2007), e.g., FIPA-request and FIPA-query. For instance, if an agent activates the communication trigger τ, it can use the FIPA-request to interact with the selected agents to confirm whether these agents also observed any disturbance for complementing the fault detection decision. In addition to these standard interaction protocols, CF* defines a collaboration protocol to achieve collaborative fault detection, as illustrated in Fig. 6, which extends the FIPA-request capability for a more complex collaboration.

Briefly, after executing detection model Fdt_i and activating collaboration trigger τ, the initiator agent requests collaboration for the selected agents, i.e., the participants. Considering that the participant fault detection model is compatible with the initiator model, the initiator sends the observation data for the participants to infer.

Since the participants' fault detection models may have been trained with different datasets or even with different detection algorithms, this interaction provides a valuable complement for decision-making. After each participant responds to the initiator, the decision is made according parameter ρ. The outcome of the decision-making process is communicated to all participating agents to ensure transparency and collective awareness within the MAS. It also provides valuable data for updating each agent's detection model and collaboration parameters, thereby improving the fault detection over time. The possibility of using collaboration between the system actors (with different algorithms, datasets and experiences) makes possible the detection models adapting to address the challenge of the difficulty of detecting faults.

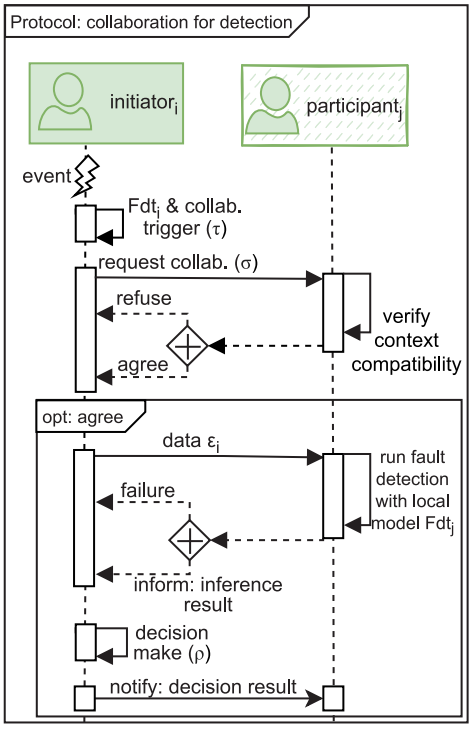

Fig. 6. Collaborative fault detection protocol.

5. Experimental validation

The proposed methodology, particularly focusing on fault detection stage, was validated with a case study and compared with traditional methodologies, namely, LFT and MCFT, in terms of their precision, recall, and F1-score for fault detection.

5.1. Case study description

The case study is a warehouse mock-up, as illustrated in Fig. 7. The warehouse comprises five racks that store different products under controlled temperature and humidity conditions and a set of mobile robots that transport the parts within the warehouse (see [blind] for more details).

The case study focuses on the rack condition, with temperatures ranging between 17 and 20 °C and the air humidity between 35% and 40%. Deviation from expected sensor output in terms of temperature and humidity can compromise the storage quality and integrity of the stored product. Therefore, it is essential to ensure the fault detection efficiency in terms of temperature and humidity anomalies, paving the way for the next fault tolerance stages, namely diagnosis and recovery, and minimizing the fault impact on storage quality.

Fig. 7 shows the five cyber–physical components of the rack class. Each rack's physical part comprises a wooden structure to store the products, a DHT22 sensor connected to an ESP32 microcontroller to acquire the temperature and humidity data and communicate this information via Wi-FI to the agent.

Each rack component is represented by an agent operated on a dedicated Raspberry Pi 3B, using a proprietary framework programmed in Python. The agents behave according to the models previously specified and communicate by following the designed interaction protocols encoding messages in JSON format, which is a simple and computationally efficient method for structuring information.

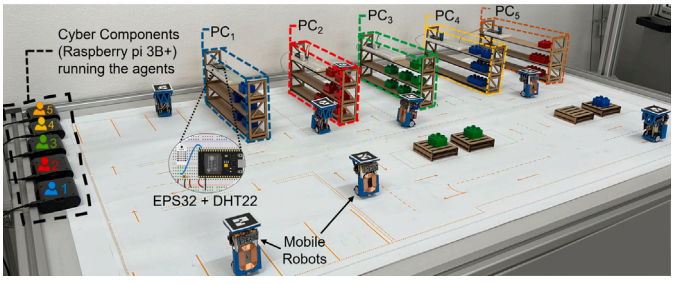

Fig. 7. Warehouse mock-up highlighting the five component racks used in the experiments.

5.2. Dataset elaboration

Seven healthy DHT22 sensors were used to acquire a dataset containing seven samples with 1440 temperature and humidity observations for the experiments, five of which were used for training and two for testing. Each observation was obtained within a 10-s time frame, totaling 4 h of data acquisition. The expected a given as follows:

$$s(t) = h(t) + \eta, \quad \eta \sim N(0, \delta_\eta^2) \tag{5}$$

where $h(t)$ is the (temperature or humidity) sensor output and $\eta \sim N(0, \delta_\eta^2)$ is the noise with zero mean and low variance associated with each sensor.

Among the seven sensors, one is distinct because it is not subjected to fault injection. Meanwhile, the remaining six samples were processed, injecting precision degradation fault into the sensors' output. As described in Jan et al. (2021), this type of sensor fault occurs because of sparse connections in the circuit, noise resulting from high-frequency operations, and physical damage to the sensor. When a fault manifests itself, a precision degradation is injected according to the model described in Eq. (6). The noise associated with a precision fault has zero mean and high variance. Faults can manifest individually in temperature or humidity or concurrently affect both measurements.

$$s(t) = h(t) + \eta + \nu, \nu \sim N(0, \delta_\nu^2) : \delta_\nu^2 \gg \delta_\eta^2 \tag{6}$$

For the experiments, the dataset was divided into two parts. The first part contained five samples with precision faults injected, as shown in Fig. 8. These five samples were used to train the fault detection models embedded in the respective agents responsible for the racks according to ID, e.g., $agent_1$ responsible for $rack_1$ was trained with the dataset with ID 1.

The second part of the dataset (shown in Fig. 9) contained two samples used in different experiments. The sample on the left shows the data obtained from a healthy sensor to represent a fault-free experimental scenario. The sample on the right contains data obtained from the sensors with precision degradation, representing an experimental scenario with fault manifestation.

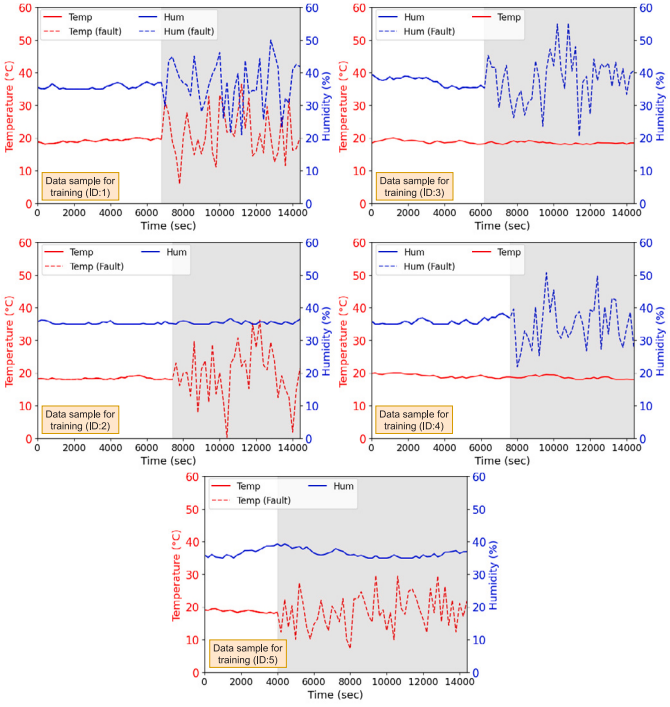

Fig. 8. Data used for training the detection models.

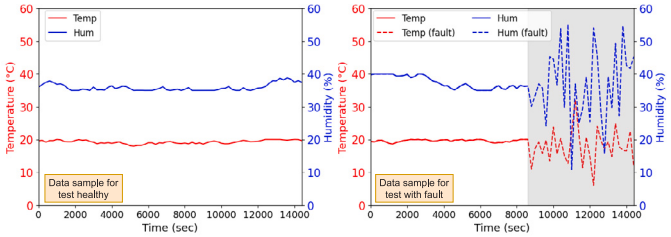

Fig. 9. Data used for the fault-free and faulty experiments.

5.3. Experimental procedure

To compare the fault detection capabilities of traditional methodologies (LFT and MCFT) with the collaborative fault detection of the proposed CF*, a strategy involving a binary classifier based on a SVM algorithm was employed. To achieve this, it is necessary to train the fault detection models (offline process) and integrate them into agents to perform online fault classification. It is worth noting that the objective of the experiments was not to optimize the classifier model for fault detection (for this, it is possible to consult Yoo, 2020; Yang et al., 2023; Chen et al., 2011), but to compare different methodologies. Consequently, the compared methodologies did not incorporate feature extraction techniques or principal component analysis as a means to optimize the classification models.

The Python library scikit-learn was employed to implement the SVM within the agents to classify the faults in the rack sensors. The input consisted of a temporal window encompassing sixty temperature and humidity observation each. The output domain was binary, with "0" representing healthy data and "1" representing faulty data. All SVM classifier models were trained using the parameters listed in Table 2.

Fig. 10 summarizes the classifier training procedure and illustrates the process through which each model was derived. A uniform training procedure was employed for both the LFT and CF*, given that the classifier models were locally integrated within each agent. In contrast, the MCFT consolidates all training data samples to develop a model with enhanced generalization capacity and minimized overfitting. This is expected to yield a superior performance, which is typical of a centralized model trained on a larger dataset.

The experimental procedure for comparing the different FT methodologies is shown in Fig. 11. Two experiments were conducted for each methodology based on the test data from the created dataset (Fig. 9). For fault detection using the LFT methodology, each model was embedded in the agents that received temperature and humidity data from the physical part, that is, the racks. In the MCFT methodology, one agent is responsible for detecting faults in the five racks, and must internally manage the data received from each component and the fault detection sequence.

For CF* experiment, each agent was responsible for a physical rack component and performed self-fault detection and collaboration when needed, according to the collaboration parameters τ, σ, and ρ. These parameters were uniformly implemented across all agents for fault detection in CF*. τ was determined via the confidence level of the model's inferences for each observation in the test data. Collaboration

Table 2
Parameters used in the SVM model for fault detection.

Parameter	Value
Kernel	rbf
Regularization parameter (C)	1.0
Gamma	Scale
Decision function shape	ovr
Probability	True
Tolerance for stopping criterion	0.001

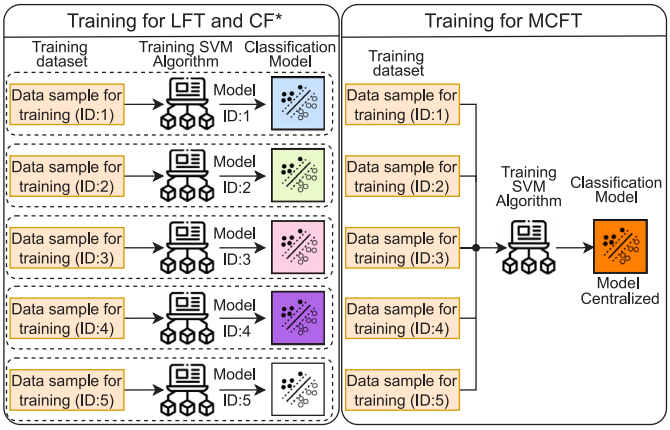

Fig. 10. Procedure used for training the SVM models.

was initiated if an inference classified an observation with a confidence level lower than 85% (a threshold established based on experimental practice) for both healthy and faulty data. The component selection for collaboration σ adopted in this case study encompasses all five agent racks of the system since they are compatible. Finally, the decision makes ρ parameters is based on the majority of agent votes.

5.4. Results and discussion

In the fault-free experiment, the data sample for testing the sensor condition (Fig. 9) represents a scenario in which the agents are not expected to detect any faults. However, owing to the inherent limitations of the SVM models, some observations were incorrectly classified as faulty. Fig. 12 shows the performance of the MCFT, LFT, and CF* methodologies to elucidate on the misidentified faults. With a detection model trained on a more comprehensive dataset, the MCFT methodology outperformed the LFT methodology for all agents. However, in the LFT methodology, each agent demonstrated unique fault detection characteristics. Through the collaboration facilitated by the CF* methodology, the agents improved the performance of the LFT methodology, reducing the number of incorrect classifications.

Fig. 13 complements the comparison by presenting the observations exposed through a confusion matrix for each methodology, showing the accuracy improvement for the fault-free experiment when considering the proposed CF* methodology for collaborative detection implemented in each agent compared with the LFT methodology, representing an improvement of 13% on average.

Table 3
Accuracy for the fault-free data test.

	–	Agent 1		Agent 2		Agent 3		Agent 4		Agent 5	
	MCFT	LFT	CF*	LFT	CF*	LFT	CF*	LFT	CF*	LFT	CF*
Precision	1	1	1	1	1	1	1	1	1	1	1
Recall	0,86	0,69	0,92	0,74	0,76	0,50	0,79	0,60	0,70	0,59	0,69
F1-score	0,93	0,82	0,96	0,85	0,86	0,67	0,88	0,75	0,82	0,74	0,81

Table 3 showcases the precision, recall, and F1-score metrics for the fault-free experiment. Considering the focus on healthy data exclusively, the models' precision metric attains a score of unity, as no faulty data is present. Consequently, the emphasis shifts to evaluating the recall and F1-score metrics. The MCFT methodology exhibits satisfactory fault detection results, with an F1-score of 93%, outperforming all LFT agents, with some agents displaying a low F1-score value, notably $agent_3$, $agent_4$, and $agent_5$ with values of 67%, 75%, and 74% respectively. However, when adopting the agent's collaboration for fault detection, the recall metrics have a discernible enhancement and, consequently, the F1-score across all agents. Although $agent_2$, the best agent without collaboration, does not maintain its lead in the CF* methodology, but it plays a pivotal role in collaboration, significantly bolstering the performance of other agents. Notably, $agent_1$ excels under CF*, surpassing the baseline set by the MCFT by 3%. Comparing the MCFT and CF* methodologies, it is observed that, on average, the F1-score is higher for MCFT (93% vs. 86%). However, CF* demonstrates competitiveness due to its distributed approach, which enhances the scalability and avoids the risk of a single failure point.

The second experiment, which considered test data with precision degradation faults in the sensors, also demonstrated improved performance in agents that utilized the collaboration protocol in the proposed CF* fault detection methodology. Fig. 14 shows the performance of the MCFT, LFT, and CF* methodologies.

Fig. 15 shows the observations delineated through a confusion matrix for the methodology comparison, providing an overview of the performance for the precision fault experiment. The improvement introduced by the collaborative fault detection implemented in the agents was considerable. However, for $agent_4$, the collaboration resulted in a minor induction of error (inserting two falsely healthy and five falsely faulty observations).

Table 4 provides a detailed breakdown of the precision, recall, and F1-score metrics derived from the fault-free experiments. Upon further comparison between the LFT and CF* methodologies, an enhancement the precision and recall was observed for the latter, resulting in an improvement of 6.7% for the F1-score on average. Notably, $agent_3$ exhibited a F1-score improvement of 1% under the proposed CF* methodology as compared to the MCFT, while $agent_1$ exhibited a similar F1-score.

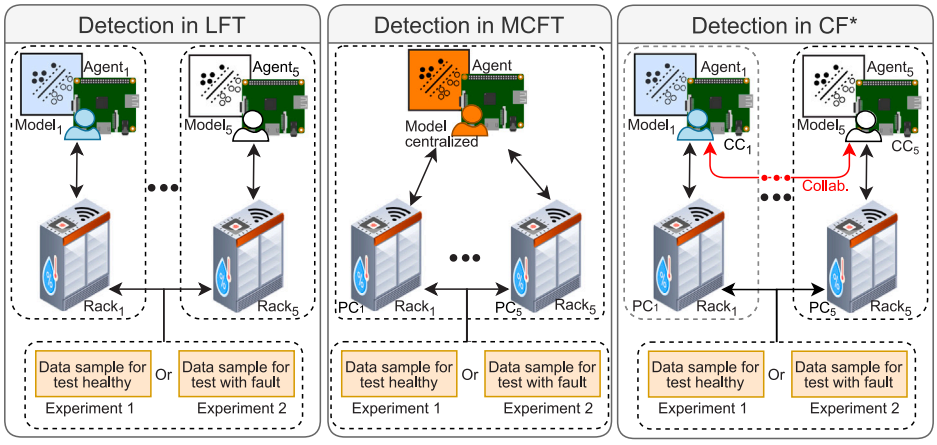

Fig. 11. Experimental procedure used in this study.

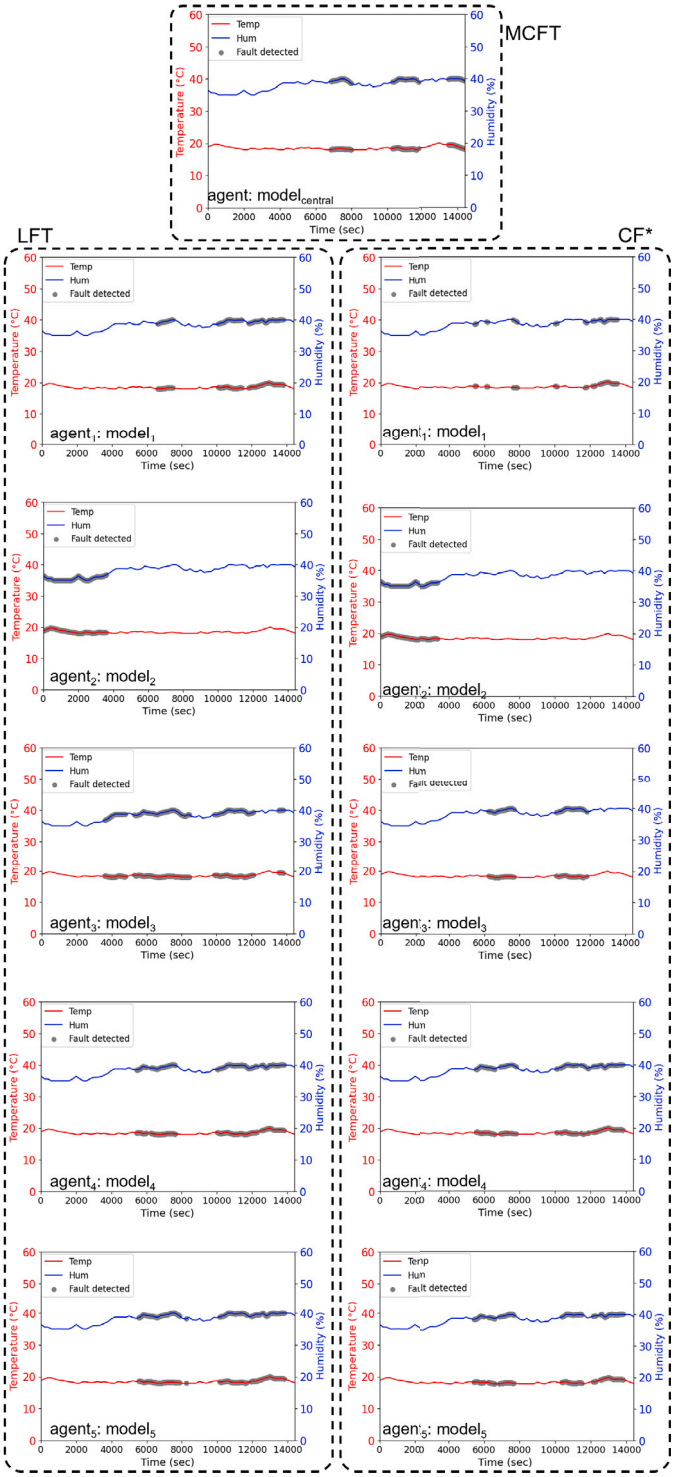

Fig. 12. Fault detection classification comparison between the MCFT, LFT, and CF* methodologies for the fault-free experiment.

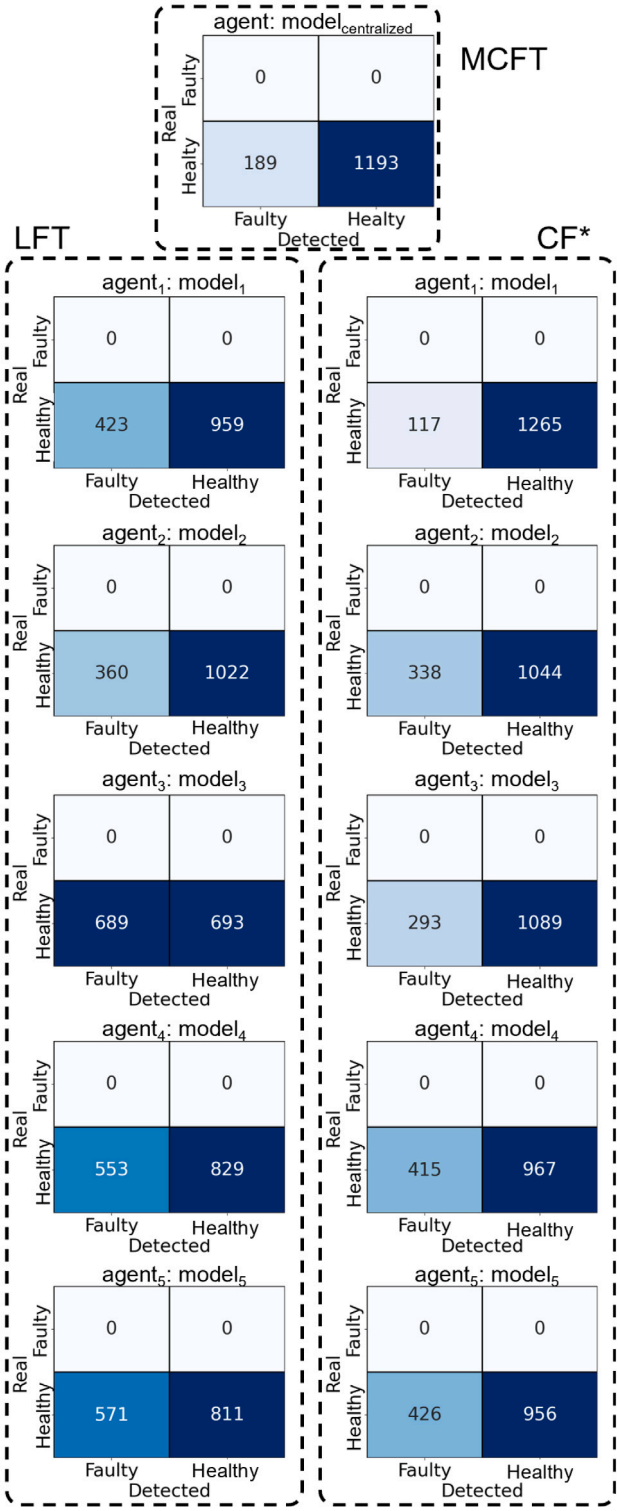

Fig. 13. Confusion matrix for the fault-free experiment.

6. Conclusion and future works

In this paper, a collaborative fault tolerance methodology for cyber–physical systems is introduced. The proposed approach, named CF*, parameterized the criteria for when, who, and how a component should collaborate with others compatible components to improve its fault detection, diagnosis, and recovery capabilities. Each component is self-fault tolerant when possible and collaborates when needed, resulting in collective intelligence for fault tolerance. To this end, a specification is presented with a meta-model for collaborative fault tolerance based on MAS concepts. Additionally, Petri net formalism was used to model the dynamic behavior of the agents, emphasizing the collaborative fault detection stage. For agent collaboration under the proposed CF*, in addition to the ability to adopt the protocols established in FIPA-ACL, a collaborative fault detection protocol is also proposed.

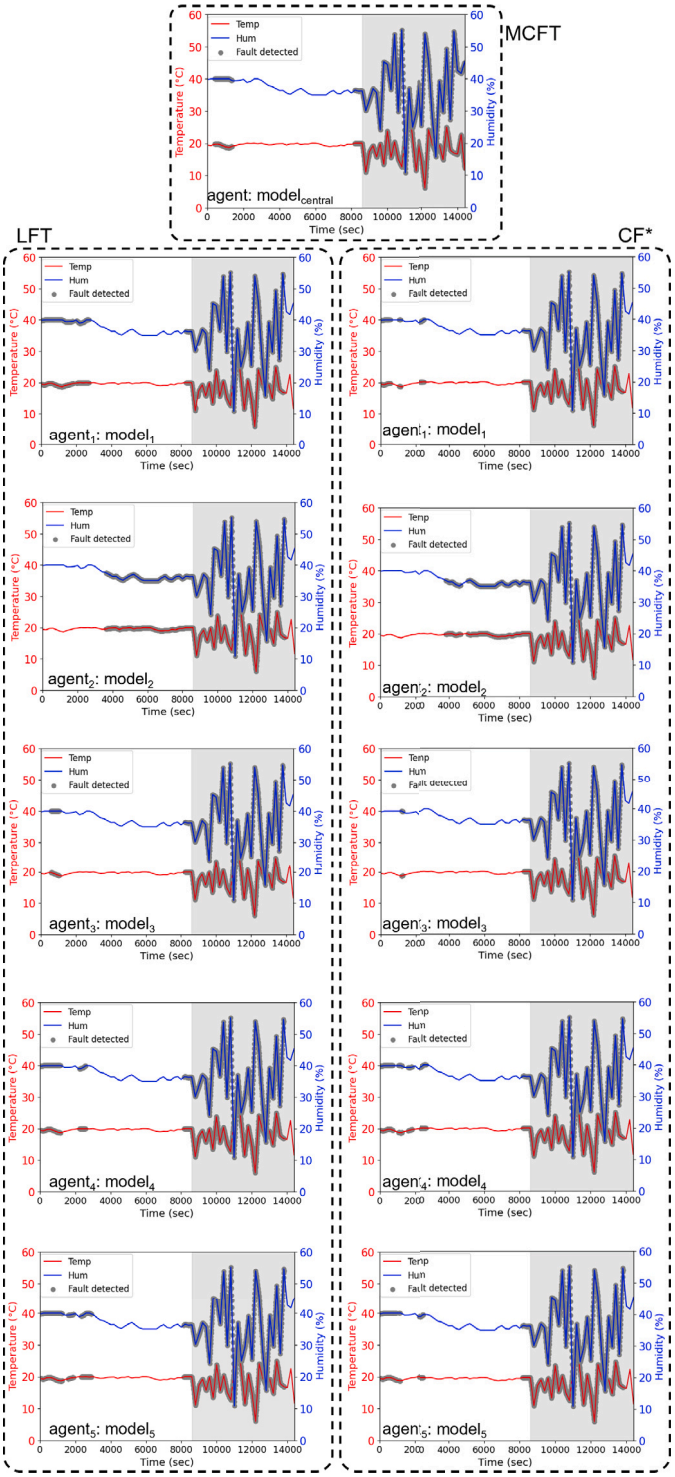

Fig. 14. Fault detection classification comparison between the MCFT, LFT, and CF* methodologies for the precision degradation fault experiment.

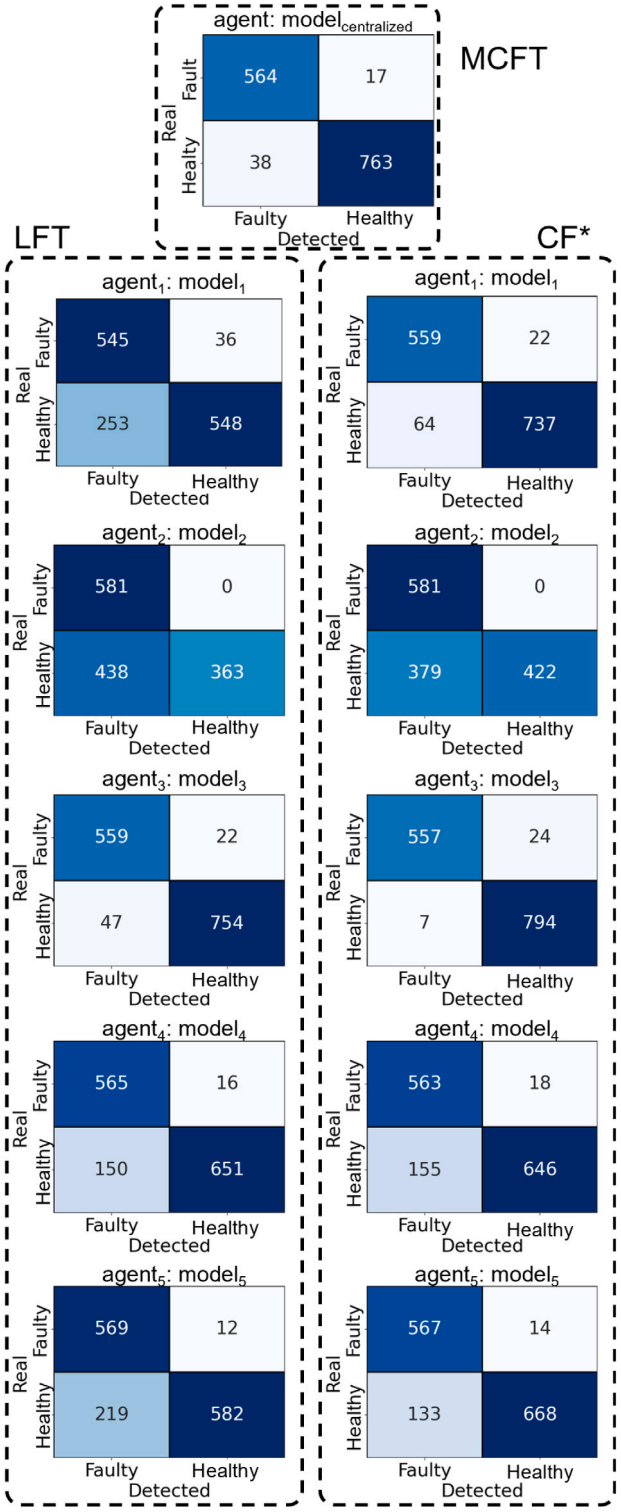

Fig. 15. Confusion matrix for precision fault experiment.

As a case study, five racks on a mock-up warehouse were adopted to conduct two fault detection experiments on the temperature and humidity sensors installed in each rack. The experiments focused on the detection stage and compared the precision, recall, and F1-score of traditional fault tolerance methodologies, namely, LFT and MCFT, with those of the proposed CF* approach. The results confirmed the benefits of using collaboration, with a significant improvement when using the proposed CF* compared to the LFT; the latter even surpassed the MCFT.

Overall, the work presented in this paper opens new research avenues in the field of collaborative fault tolerance for CPS and provides a foundation for future development. Future work will focus on the diagnostic stage, implementing heterogeneous models to identify the nature of faults, exploring recent data-driven methods, as ToMFIR to mitigate incipient faults, enhancing the fault tolerance robustness in CPS as well as integration with the diagnosis and recovery stages.

Table 4
Accuracy for the precision degradation fault data test.

	–	Agent 1		Agent 2		Agent 3		Agent 4		Agent 5	
	MCFT	LFT	CF*	LFT	CF*	LFT	CF*	LFT	CF*	LFT	CF*
Precision	0,98	0,94	0,97	1	1	0,97	0,97	0,98	0,97	0,98	0,98
Recall	0,95	0,68	0,92	0,45	0,53	0,94	0,99	0,81	0,81	0,73	0,83
F1-score	0,97	0,79	0,94	0,62	0,69	0,96	0,98	0,89	0,88	0,83	0,90

Further research is needed to enhance the capabilities of the proposed CF* methodology, mainly for the fault prognosis, recovery stage and learning capacity, to online improve the detection, diagnostic and recovery models and the collaboration parameters.

CRediT authorship contribution statement

Luis Piardi: Writing – original draft, Visualization, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **André Schneider de Oliveira:** Writing – review & editing, Validation, Supervision, Resources, Conceptualization. **Pedro Costa:** Writing – review & editing, Validation, Supervision, Resources, Conceptualization. **Paulo Leitão:** Writing – review & editing, Writing – original draft, Validation, Supervision, Resources, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by Fundação para a Ciência e a Tecnologia (FCT) through Portugal National funds FCT/MCTES (PIDDAC): CeDRI, UIDB/05757/2020 (DOI: 10.54499/UIDB/05757/2020) and UIDP/05757/2020 (DOI: 10.54499/UIDP/05757/2020) and SusTEC, LA/P/0007/2020 (DOI: 10.54499/LA/P/0007/2020). The author Luis Piardi thank Fundação para a Ciência e a Tecnologia (FCT) for the PhD Grant UI/BD/151286/2021 (DOI: 10.54499/UI/BD/151286/2021).

Data availability

No data was used for the research described in the article.

References

Avizienis, A., Laprie, J.C., Randell, B., Landwehr, C., 2004. Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. Dependable Secur. Comput. 1 (1), 11–33.

Bayar, N., Darmoul, S., Hajri-Gabouj, S., Pierreval, H., 2015. Fault detection, diagnosis and recovery using artificial immune systems: A review. Eng. Appl. Artif. Intell. 46, 43–57.

Boi-Ukeme, J., Ruiz-Martin, C., Wainer, G., 2020. Real-time fault detection and diagnosis of CPS faults in DEVS. In: 2020 IEEE 6th International Conference on Dependability in Sensor, Cloud and Big Data Systems and Application (DependSys). IEEE, pp. 57–64.

Caiazzo, B., Di Nardo, M., Murino, T., Petrillo, A., Piccirillo, G., Santini, S., 2022. Towards zero defect manufacturing paradigm: A review of the state-of-the-art methods and open challenges. Comput. Ind. 134.

Chaterji, S., Naghizadeh, P., Alam, M.A., Bagchi, S., Chiang, M., Corman, D., Henz, B., Jana, S., Li, N., Mou, S., et al., 2019. Resilient cyberphysical systems and their application drivers: A technology roadmap. arXiv preprint arXiv:2001.00090.

Chen, K.-Y., Chen, L.-S., Chen, M.-C., Lee, C.-L., 2011. Using SVM based method for equipment fault detection in a thermal power plant. Comput. Ind. 62 (1), 42–50.

Dowdeswell, B., Sinha, R., MacDonell, S.G., 2020. Finding faults: A scoping study of fault diagnostics for industrial cyber-physical systems. J. Syst. Softw. 168, 110638.

Fidelis, S.A., Castro, M., Siqueira, F., 2022. Distributed learning using consensus on edge ai. In: 2022 XII Brazilian Symposium on Computing Systems Engineering. SBESC, IEEE, pp. 1–8.

Grethler, M., Marinov, M.B., Klumpp, V., 2021. Embedded machine learning for machine condition monitoring. In: International Conference on Future Access Enablers of Ubiquitous and Intelligent Infrastructures. Springer, pp. 217–228.

Hastbacka, D., Halme, J., Barna, L., Hoikka, H., Pettinen, H., Larranaga, M., Bjorkbom, M., Mesia, H., Jaatinen, A., Elo, M., 2021. Dynamic edge and cloud service integration for industrial IoT and production monitoring applications of industrial cyber–physical systems. IEEE Trans. Ind. Inform. 18, 498–508.

He, Y., Xiang, H., Zhou, H., Chen, J., 2023. In-situ fault detection for the spindle motor of CNC machines via multi-stage residual fusion convolution neural networks. Comput. Ind. 145, 103810.

Jan, S.U., Lee, Y.D., Koo, I.S., 2021. A distributed sensor-fault detection and diagnosis framework using machine learning. Inform. Sci. 547, 777–796.

Lee, J., Bagheri, B., Kao, H.-A., 2015. A cyber–physical systems architecture for industry 4.0-based manufacturing systems. Manuf. Lett. 3, 18–23.

Lee, J., Kundu, P., 2022. Integrated cyber–physical systems and industrial metaverse for remote manufacturing. Manuf. Lett. 34, 12–15.

Li, Z., Wang, Y., Wang, K., 2019. A deep learning driven method for fault classification and degradation assessment in mechanical equipment. Comput. Ind. 104, 1–10.

Moussa, N., Hamidi-Alaoui, Z., El Alaoui, A.E.B., 2019. Cftm: A centralized fault tolerant mechanism for wireless sensor networks. In: 2019 5th International Conference on Optimization and Applications. ICOA, IEEE, pp. 1–6.

Piardi, L., Costa, P., Oliveira, A., Leitão, P., 2022. Collaborative fault detection and diagnosis architecture for industrial cyber-physical systems. In: IEEE Int. Conf. on Industrial Technology. ICIT, pp. 1–6.

Piardi, L., Leitão, P., de Oliveira, A.S., 2020. Fault-tolerance in cyber–physical systems: Literature review and challenges. In: IEEE 18th Int. Conf. on Industrial Informatics. INDIN, pp. 29–34.

Pivoto, D.G., de Almeida, L.F., da Rosa Righi, R., Rodrigues, J.J., Lugli, A.B., Alberti, A.M., 2021. Cyber-physical systems architectures for industrial internet of things applications in industry 4.0: A literature review. J. Manuf. Syst. 58, 176–192.

Poslad, S., 2007. Specifying protocols for multi-agent systems interaction. ACM Trans. Auton. Adapt. Syst. (TAAS) 2 (4), 15–es.

Reijnen, F., Leliveld, E.-B., van de Mortel-Fronczak, J., van Dinther, J., Rooda, J., Fokkink, W., 2021. Synthesized fault-tolerant supervisory controllers, with an application to a rotating bridge. Comput. Ind. 130, 103473.

Ribeiro, L., Barata, J., 2011. Re-thinking diagnosis for future automation systems: An analysis of current diagnostic practices and their applicability in emerging it based production paradigms. Comput. Ind. 62 (7), 639–659.

Safari, S., Ansari, M., Khdr, H., Gohari-Nazari, P., Yari-Karin, S., Yeganeh-Khaksar, A., Hessabi, S., Ejlali, A., Henkel, J., 2022. A survey of fault-tolerance techniques for embedded systems from the perspective of power, energy, and thermal issues. IEEE Access 12229–12251.

Wu, Y., Su, Y., Shi, P., 2024a. Data-driven tomfir-based incipient fault detection and estimation for high-speed rail vehicle suspension systems. IEEE Trans. Ind. Inform..

Wu, Y., Su, Y., Wang, Y.-L., Shi, P., 2024b. Ts fuzzy data-driven tomfir with application to incipient fault detection and isolation for high-speed rail vehicle suspension systems. IEEE Trans. Intell. Transp. Syst..

Yang, Z., Huang, Y., Nazeer, F., Zi, Y., Valentino, G., Li, C., Long, J., Huang, H., 2023. A novel fault detection method for rotating machinery based on self-supervised contrastive representations. Comput. Ind. 147, 103878.

Yoo, Y., 2020. Data-driven fault detection process using correlation based clustering. Comput. Ind. 122, 103279.

Zhang, K., Shi, Y., Karnouskos, S., Sauter, T., Fang, H., Colombo, A.W., 2022. Advancements in industrial cyber–physical systems: An overview and perspectives. IEEE Trans. Ind. Inform..