IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received August 26, 2020, accepted September 22, 2020, date of publication September 25, 2020, date of current version October 7, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3026946

Fault-Resilient Collective Ternary-Hierarchical Behavior to Smart Factories

VIVIAN CREMER KALEMPA[1], LUIS PIARDI[2], MARCELO LIMEIRA[3], AND ANDRÉ SCHNEIDER DE OLIVEIRA[3], (Member, IEEE)

[1]Department of Information Systems, Universidade do Estado de Santa Catarina (UDESC), São Bento do Sul 89283-081, Brazil
[2]Research Center in Digitalization and Intelligent Robotics (CeDRI), Instituto Politécnico de Bragança (IPB), 5300-253 Bragança, Portugal
[3]Graduate Program in Electrical and Computer Engineering, Universidade Tecnológica Federal do Paraná (UTFPR), Curitiba 80230-901, Brazil

Corresponding author: Vivian Cremer Kalempa (vivian.kalempa@ udesc.br)

This work was supported in part by the National Council for Scientific and Technological Development of Brazil (CNPq) and in part by the Coordination for the Improvement of Higher Level People (CAPES).

ABSTRACT Smart factories are introducing new technologies to improve production and expand flexibility, which denotes the integration of intelligent, autonomous, and interconnected agents. The conceptual transition to dynamic multiple agents generates some dilemmas, mainly regarding the occurrence of unexpected situations. This paper aims to discuss the collective behavior of multi-agent systems in smart factories for achieving fault resilience. The proposed approach is based on three hierarchical plans: imposition, negotiation, and consensus. Fault restoration is achieved through the collective behavior that manages the ternary decisions made in these plans. The approach can help the smart factories that employ autonomous multi-agents improve their production, reliability, and robustness to failure. The proposed method was evaluated using a virtual warehouse logistics but employing real scenarios. Experiments were performed through logistic tasks to prove the collective behavior implemented in the approach for fault resilience. Quantitative analysis of the experiments shows the efficiency of the approach under various situations.

INDEX TERMS Smart factories, multi-agent, resilience, consensus.

I. INTRODUCTION

I ndustry 4.0 uses intelligent behaviors for manufacturing and expands conventional motions that are purely based on ground lines, such as automated guided vehicles (AGVs). Autonomy is a crucial requirement of smart factories to enable them to interact with their environment for localization and mapping, learn their environment, and avoid previously known and unknown obstacles. However, flexibility and robustness cannot be achieved through the application of AGVs in the ground lines alone. Industrial processes are a set of distributed actions (parallel, serial, cascade, and coupled) whose global objective cannot be achieved by their agents' independent actions.

For smart factories, collective behaviors are key to ensuring faster, more efficient, and customer-centric manufacturing. These can help industrial agents interact with other agents to not only execute tasks but also make decisions, express experiences and expectations, and make global decisions taking into consideration the collective polities as a consensus. Intelligence is achieved when agents have higher perception, can retain information, and can adapt to unspecified situations. Interaction with other agents occurs only when all agents recognize that they are part of a shared environment and perform collaborative tasks. The uniqueness of distributed agents introduces redundancy of processes and ability to recover from unexpected situations through collective decisions.

Agents in Industry 4.0 are collaborative mobile robots designed with advanced perception systems, which are capable of executing tasks side-by-side with humans and other robots and interacting and communicating actively. One of the main tasks of a multi-robot system (MRS) is task allocation, which is commonly known as the multi-robot task allocation (MRTA) problem. In this case, for a group of robots to perform a particular task at the system level effectively, the designer must address which robot should do what task and when [1]. It is also necessary to have a contingency mechanism in case the chosen robot suffers failures, as using multiple agents to perform a task makes the system redundant and consequently, fault-tolerant [2]. Thus, in multi-agent management, collective behavior is used for executing individual sub-tasks and group tasks to make the system resilient to execution failures.

The associate editor coordinating the review of this manuscript and approving it for publication was Weiguo Xia.

This paper proposes a fault-resilient collective ternary-hierarchical behavior for managing and adapting multi-agent systems and achieving resilience to failures. The proposed collective method is organized as a three-level decision: imposition, negotiation, and consensus. Thus, resilience is achieved by considering choices at different hierarchical levels: global process, agents' groups, and individual agents. The recovery from failures is the consequence of managing collective actions through ternary choices. This approach aims to improve the production efficiency of smart factories.

The remainder of this paper is organized as follows. Section 2 presents the related works. Section 3 presents the problem statement and assumptions. Section 4 presents the fault-resilient collective ternary-hierarchical behavior. Section 5 explains the warehouse logistics. Section 6 discusses the experimentation and evaluation of the approach. Section 7 presents the accuracy and precision of the approach. Finally, Section 8 presents the conclusions and future work.

II. RELATED WORK

Resilient and fault-tolerant approaches for multi-robot systems have been explored in several studies. However, from the studies presented in this section and based on the experiments conducted in the present paper, it can be observed that the development of resilient or fault-tolerant multi-robot systems entails certain costs. In addition, the three phases of fault tolerance (i.e., detection, diagnosis, and recovery from failures) need to be addressed [3].

Beckman and Aldrich (2007) [4] presented a programming framework with fault tolerance support for collaborative robotics. The application was implemented among robots and executed through consecutive remote procedure call (RPC). The framework provides a primitive for the programmer to annotate critical code blocks that can be problematic in a failure event. The application programmer defines some compensatory actions within these blocks for cases when faults occur during execution (e.g., compensatory actions to avoid deadlocks). This ensures that the application can progress despite failures. However, the critical part of handling failures, which are the compensatory actions, remains the programmer's responsibility.

Yang *et al.* (2011) [5] proposed a distributed fault-tolerant flocking algorithm that could tolerate crash of robots, including initial crashes and crashes during flocking. Tarapore *et al.* (2017) [6] presented a generic fault-detection system for robot swarms. In this system, robots can observe and classify one another's behavior through an immune system-inspired algorithm, which distinguishes between normal and abnormal behaviors online. The goal was to provide long-term autonomy for multi-robot systems.

Guo *et al.* (2018) [7] presented a framework called ALLIANCE-ROS to develop cooperative multi-robot fault-tolerant systems. Koutsoubelias and Lalis (2018) [8] presented a selective replay technique to tolerate failures of centralized applications that control multiple mobile robots to perform a task in a coordinated way. The approach uses a combination of verification, passive replication, and logging and can resume application execution consistently, considering system/environment dynamics. Li *et al.*(2018) [9] presented a new approach for the creation of patrolling policies with multiple robots, fault-tolerant and self-adaptive based on the Hoplites for MRTA framework. Panerati *et al.* (2019) [10] analyzed a set of techniques to assess, control, and enforce connectivity in fallible robots. A controller for connectivity maintenance in the presence of faults was presented, and parameter and performance optimizations were discussed.

Fault-based approaches remain a challenging issue owing to the inherent dynamics of multi-robot systems. Versatility and dynamism of multi-robot systems intensify the challenges, making fault-handling more complicated without complete degeneration of the system. This paper presents a solution to the MRTA problem on fault resilience. Redundancy of robots to address a fault is not always available; however, despite this situation, the system can remain stable to continue fulfilling its tasks.

III. PROBLEM STATEMENT AND ASSUMPTIONS

According to Blanke *et al.* (1997) [11], fault-tolerant systems can be understood as *"systems that may degrade performance when a fault occurs, but a fault will not develop into failure at the system level if this could be prevented through proper action in the programmable parts of a control loop"*. A system employing this kind of behavior can be described as one that has the following properties:

- prevent any simple fault from developing into failure at the system level;
- use information redundancy to detect faults;
- reconfigure programmable system components to accommodate faults;
- accept degraded performance owing to a fault but maintain plant availability.

Hence, fault tolerance indicates the way a system treats the faults that have occurred. When a failure occurs, its impact should not be noticed owing to existing redundancy mechanisms.

However, in several systems, faults result in instantaneous performance degeneration but are then suppressed by recovery mechanisms. Hollnagel *et al.* (2006) [12] defined resilience as *"the ability of a system to keep or recover quickly to a stable state, allowing to continue operations during and after a major fault or in the presence of continuous significant faults"*. Thus, fault resilience is the capability of a system to recover and continue operations despite system failures.

Multi-agent systems (MAS) are highly dependent on the flexibility of the environment and their agents' interactions. The dynamic behavior of MAS is not compatible with traditional recovery methods as it may provide obsolete information or indicate inconsistent task execution, leading to wrong decisions and undesirable actions [8].

The present work addresses the collective behavior of MAS employed in smart factories to achieve

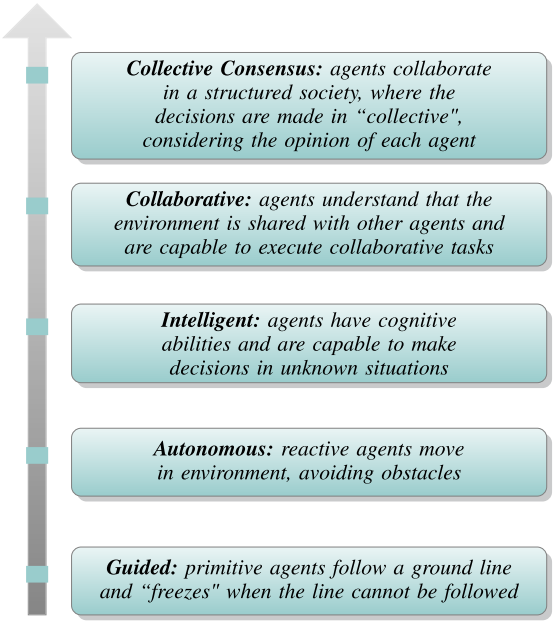


FIGURE 1. Cognition level of agents.

fault-resilience capability. The agents are freed from the primitive tracking of ground lines, allowing autonomous and intelligent behaviors and interacting actively with other agents in collective actions. One of its abilities is to recover from system failures through broad and collective decisions. Agents can have different behaviors in smart factories due to their cognition level, as shown in Figure 1.

This paper focuses on collective behavior for achieving fault resilience in multi-agent systems. The approach does not aim to discuss the detection and classification of faults but to present support so that failures cannot interrupt an industrial process.

IV. FAULT-RESILIENT COLLECTIVE TERNARY-HIERARCHICAL BEHAVIOR

The proposed ternary-hierarchical behavior is a method for achieving fault resilience in multi-agent systems through a collective decision for the fault recovery approach. The method is hierarchical because the decision making is performed in three hierarchical plans: imposition (task scheduler), negotiation (task groups), and consensus (individual agents), as shown in Figure 2.

The fault recovery approach is composed of the three hierarchical plans or decisions. The first-level plan introduces requirements for the decision making of the other plans.

A. IMPOSITION

The first decision level is the *imposition*, where the scheduler delimits the recovery behavior of one or more robots' failure events in terms of process characteristics. The scheduler defines which process, with a given priority, should release robots to recover from fault. It is not determined precisely the process, but which process type should give robots.

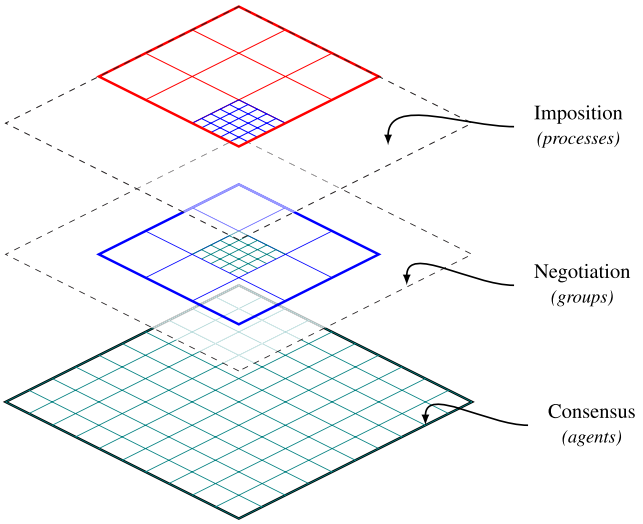


FIGURE 2. Ternary-Hierarchical Behavior.

This decision plan provides instructions to lower levels on which need to adapt to follow the order. In this case, the recovery does not affect the execution of other processes.

The instruction given to lower levels is accompanied by the director's process specification, which manages the decision in the second level. The selection is based on the highest priority or, in the case of equal priorities, the oldest.

B. NEGOTIATION

The second level of the approach is *negotiation*, where the scheduler's order is received and the guidelines imposed by the scheduler are followed in the processes of the referred plan. This level is composed of all processes with the priority determined by the scheduler and also by the redundant processes (i.e., having more agents than the minimum number required by the priority). The group of processes in the plan determined by the scheduler must negotiate the robot's assignment to recover from failure. The negotiation is based on its impact on the plan. Thus, processes must inform the process director on its status based on the following:

- *redundancy degrees*: which is the quantity of additional robots in the process. The process has a minimum number of robots to serve it, according to priority. If more than one process has extra robots, the analysis will start from the process with a lower priority;
- *laxity*: this information is based on Least Laxity First (LLF) Scheduling algorithm [13]. The process that has less laxity in an instant will have a higher priority than others at that instant. Thus, for processes that have equal priorities, the process with the highest laxity interval will be considered the least important at that instant.

The director will mediate the negotiation between candidate processes. First, evaluate redundancy of robots. If positive, the process gives way and the rest continues normally. The second hypothesis is if any process is too far from its deadline, i.e., it has a high laxity. In this case, the process

with the most significant laxity interval is the one that will have to release robots.

There is a prerogative in this plan. The process that releases robots does not lose priority, even if it does not have the minimum required number. This is the benefit of negotiation, where the process gains temporary bypass requirements and the robot recovers when it finishes the process with the highest priority, while keeping its original priority.

However, if the process that has a failed robot is the one with the lowest priority or the one with the highest laxity, that process will suffer depreciation and be preempted. This mechanism aims to preserve the priority of more essential processes and disallow priority inversion owing to the dependency between parallel processes.

C. CONSENSUS

The third level of the approach is *consensus*. After the staff determines which of the processes must release robots to those in need, the team decides which workers (i.e., agents) are to be deallocated. This decision is consensual and collective between agents of the process and is made through voting.

In the voting process, it is necessary to define the process manager, i.e., the robot with the highest battery power in the current activity. Then, an analysis is performed to determine eligibility, where the robots that have no faults and have more battery are listed. The number of eligible robots is the number of robots needed to meet a failure plus one.

In addition to battery analysis, robots that incur the highest cost to complete the current process are also prioritized in the eligibility list. The remaining robots inform the manager of their choices based on the impact of their processes. That is, each robot votes for the one that brings the least work. It is a purely individual decision. After the assignment of a robot, the recomposition in the process will be fundamental for the vote. In this sense, each robot will vote for the one that is the farthest away. If the recomposition requires the nearest robot to assume the task of the other robot, it will not be affected. Finally, the process manager counts the votes and chooses the robot with the highest cost to complete the task in the event of a tie.

D. HYPOTHETICAL CASE

A hypothetical ternary-hierarchical behavior is shown in Figure 3, where a failure occurs during the execution of ψ processes in a smart factory. In this case, two definitions were established.

Definition 1: Staff $\{S\}$ is the number of agents working together to achieve the same process.

Definition 2: Process constraint $\{P\}$ is the minimum number of agents working together to achieve the same process.

Once a failure occurs, the scheduler determines whether a recovery can take place within the process group. This is *possibility 1*. Here, the recovery does not affect the execution of other processes and there is no interference from the scheduler.

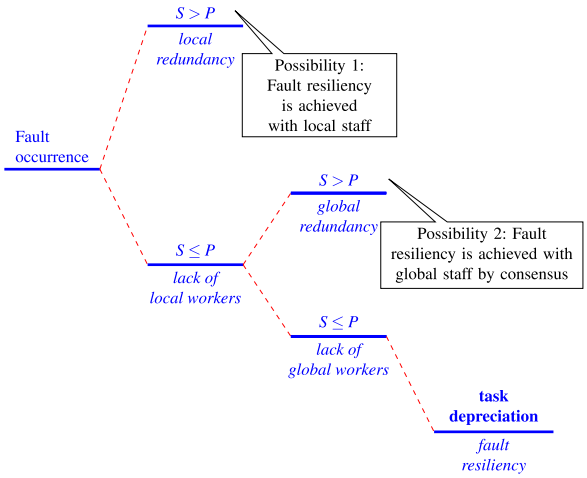


FIGURE 3. Hypothetical case of Ternary-Hierarchical behavior.

Overall recomposition occurs when the fault cannot be recovered by the local staff, degenerating the minimum requirements of priority. The other processes are analyzed to identify redundancies. In case a redundancy exists, agents can be reassigned for fault recovery in another process. The scheduler warns the groups that one process needs to assign agents to another process. The staff must decide among themselves which agents have to be available for others. Within the team, workers consensually determine which workers have to be dismissed (*possibility 2*).

If there is no redundancy in the other processes, the scheduler starts the negotiation process in the same way, forcing the lowest priority task or the task with highest laxity interval to supply agents for the process. The process that will release robots will be without the minimum number of agents needed to meet its priority. However, it does not suffer preemption until its agent(s) has recovered. If the process that losses agent(s) is not attended to in any of these situations, that process will have the lowest priority among the running processes or have the highest laxity interval and suffer depreciation and preemption. Thus, the priority of other processes for execution is preserved without inversion of priorities between processes.

V. WAREHOUSE LOGISTICS

This work was developed based on a real warehouse located in Brazil, which required a complete automated process. All departments, such as maintenance, robot loading, product handling, warehousing, and sorting, were replicated in the model. Forklifts performed different operations in the small-scale warehouse, such as loading and unloading of trucks, traveling to loading stations or maintenance areas, and storing and unloading of goods from shelves. Figure 4 illustrates the warehouse plan, in which the processes are divided into seven parts: incoming cargo, outgoing cargo, checking, staging, warehouse, charging station, and maintenance.

The incoming cargo is the department in which the truck is parked for the forklift to unload the cargo. These goods or packages are then sent to the checking department.

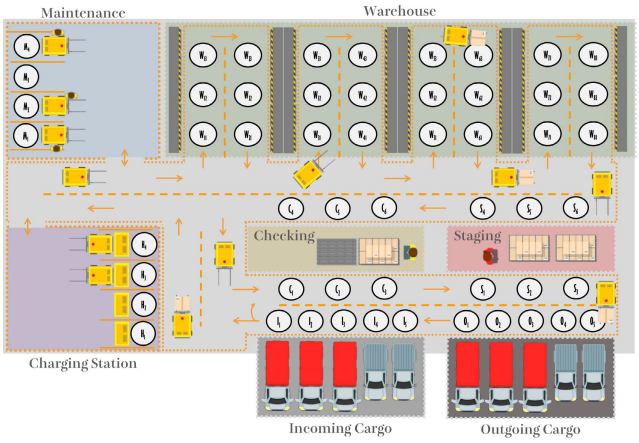


FIGURE 4. Organization of warehouse logistics.

FIGURE 5. Description of processes and tasks in warehouse logistics.

Then, the integrity of each package is checked and classified to determine the location of the package in the warehouse. The warehouse contains four aisles, one for each type of product. The first aisle is for auto supplies, the second is for medicines, the third is for food, and the fourth is for miscellaneous items. There are two closets in each aisle and each closet has three shelves.

Outgoing cargo is the department in which the truck is loaded with goods to be delivered. First, the goods are removed from the warehouse and delivered to the Staging sector. Then, they are packaged and transported to the outgoing cargo section. For goods received and transported, if there are many packages to be processed, more than one forklift can be selected to meet the requirements.

The forklift support is mainly concentrated in two areas: maintenance and charging station. If the forklift needs to be refilled, it will be directed to the charging station. If the forklift needs maintenance, it will be directed to the maintenance department.

The warehouse logistics process is divided into various states, divided into the sector's super stages. The transition state indicates the direction of the forklift to avoid possible collisions. Each state (marked with a white circle in Figure 4) indicates a position that can only be occupied by a forklift.

Warehouse logistics is a dynamic process, i.e., goods enter and leave at the same time, and it performs these tasks simultaneously. Urgent orders must be executed immediately within the time required by consumers or warehouses, such as those that involve refrigerated goods. Management is dynamic as the number of operators performing tasks changes constantly and tasks can be interrupted by temporary lack of operators. As forklifts can be damaged or can run out of batteries, random circumstances can change the way logistics are handled.

A. CHARACTERIZATION OF THE TASKS

Collection, storage, distribution, delivery, and inventory management are the responsibilities of warehouse logistics, as illustrated in Figure 4. The two processes, namely, *incoming cargo* and *outgoing cargo*, and their corresponding tasks make up the logistics, as detailed in Figure 5.

The receiving, checking, sorting, and storage are the *incoming cargo* processes. The *receiving task* involves managing receipts. When a receiving event occurs, a collection process is initiated to unload the goods from the trucks parked in the incoming cargo sector. The forklift must process the goods from the incoming cargo and take it to the checking sector for inspection and record it in the inventory, designated as the *checking task*. After inspection, goods must be classified and stored in the *sorting task*. Finally, the *storage task* stores the goods in the warehouse.

Picking, staging, and delivery are the *outgoing cargo* processes. The *picking task* starts when there is a customer order to be delivered, which requires removing the goods from the warehouse and placing them in the staging department. The *staging task* prepares or packs the goods to be delivered. Finally, the shipping of goods takes place in the outgoing cargo department, which is the *delivery task*.

The entire logistics operation is considered by the scheduler as a high-level abstraction (HAL), the decomposition of the *incoming cargo* and *outgoing cargo* processes. These processes are a series of standard tasks (as shown in the Figure 5). The *incoming cargo process* is the sequence of *receiving* $\rightarrow$ *checking* $\rightarrow$ *sorting* $\rightarrow$ *storage*. The *outgoing cargo process* is the sequence of *picking* $\rightarrow$ *staging* $\rightarrow$ *delivery*.

For each standard task in the HAL process, a coalition is formed, i.e., different groups of robots can complete these processes. When performing tasks, the robot may need to be replaced owing to lack of battery power or errors. When other robots are closer to the next subtask to be performed, the coalition of robots can be re-assigned.

B. TASK SCHEDULING

The approach scheduler maintains a processing queue, classified as an *incoming cargo* or *outgoing cargo*. Each process

has a priority that determines its importance, and the priority is distributed equally according to its tasks. The scheduler has four types of priority.

Priority 1 (Minor): this is for non-essential processes (those that do not inhibit the staff's functionality or the main purpose of the warehouse). These processes are performed when possible or by idle agents.

Priority 2 (Normal): this is for processes that are not critical or important, have isolated effects, and may have alternative solutions (variable staff). They have no special requirements, except that they must be performed when created. At least one agent must be provided.

Priority 3 (Major): this is for processes that are not critical; however, they have significant impacts on the warehouse staff. For example, an entry or exit of a truck with many boxes. This type of process requires at least two autonomous robots.

Priority 4 (Critical): this is for processes that must start immediately and be completed as quickly as possible. Processes given with this priority usually involve with refrigerated goods, emergency deliveries, or limited truck parking. These processes require at least three agents.

Up to three parallel processes can be executed by the scheduler to redirect resources to new higher priority processes. This restriction is also designed to ensure optimal flow of mobile agents in the warehouse and avoid queues in crowded environments. The allocation of robots is based on priority constraints. If the number of robots required for a given task is less than the expected number, the remaining robots will be reassigned to the task with the highest priority or to the oldest task, if they all have the same priority.

In addition, the scheduler has a process preemption mechanism. The preemption mechanism is used to prioritize the most important processes, so that the lowest priority active processes can be stopped. In this case, the lowest priority process is stopped and will re-enter the process queue. When it is possible to resume the halted process, it will continue from where it was paused.

VI. EXPERIMENTAL EVALUATION

The proposed approach was evaluated in a small warehouse called ARENA, which simulates a real industrial plant located in Brazil. ARENA was developed in a cyber-physical structure that reproduces a real industrial plant, in accordance with the tiny-scale, as shown in Figure 6. Agents perceive this virtual environment as a real plant and memorize the environment through their perception sources (or perform mapping). The aim was to use the ARENA combined with MRS and mixed reality to demonstrate an autonomous warehouse system. More detailed information about the real ARENA can be obtained in [14].

The proposed evaluation plan was developed through two experiments. The first experiment had a higher priority process and two others with equal priorities and number of robots. The robot with the highest priority process failed. The decision between the two processes with equal priorities and number of robots were made based on each process degree of laxity, giving the robot the farthest to be completed. The second experiment was a situation with three processes that had different priorities. The two robots for the process with intermediate priority failed, leaving no robots. In this scenario, as the other two processes had one surplus robot, each of them released one robot for the process that lost robots.

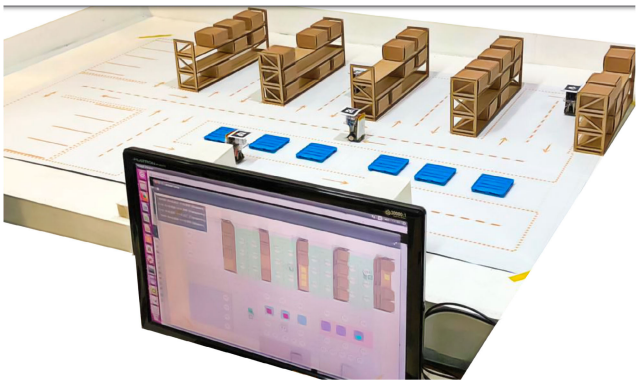


FIGURE 6. ARENA: real scenario for experimentation with mixed reality.

A. EXPERIMENT 1

The first experiment was an example of fault resilience, where two tasks that had similar priorities had to be analyzed to decide which of them would release a robot for the highest priority task. In this experiment, it was assumed that there were three boxes in aisle 1 of the warehouse, two boxes in aisle 2, and three boxes in aisle 4 from previous processes 1, 2, and 3. In this scenario, there were one *incoming cargo* type process and two *outgoing cargo* type processes, as shown in Table 1.

TABLE 1. Description of Experiment 1.

#	Type of Process	Priority	Number of Boxes	Initial State	Warehouse Aisle
Process-4	*Incoming Cargo*	4	2	I1	3
Process-5	*Outgoing Cargo*	3	3	O1	4
Process-6	*Outgoing Cargo*	3	3	O5	1

Initially, as shown in Figure 7 (a), two robots were selected for process 4 (robots 0 and 1), three robots for process 5 (robots 3, 4, and 7), and three robots for process 6 (robots 2, 5, and 6). After removing the boxes from the warehouse, the robots from processes 5 and 6 delivered the boxes to the staging sector, as shown in Figure 7 (b). After this process, the robots headed to the exit of the staging sector to take the boxes to the outgoing cargo sector, as shown in Figure 7 (c). In Figure 7 (c), it can be seen that robots 0 and 1 approach the incoming cargo sector to remove the boxes and transport them to the checking sector. However, robot 1 had some errors and stopped its execution. At that moment, the scheduler signaled that one robot was needed for priority process 4 and assigned process 4 as the manager as it had the highest priority, as shown in Figure 8. Both processes 5 and 6 had the

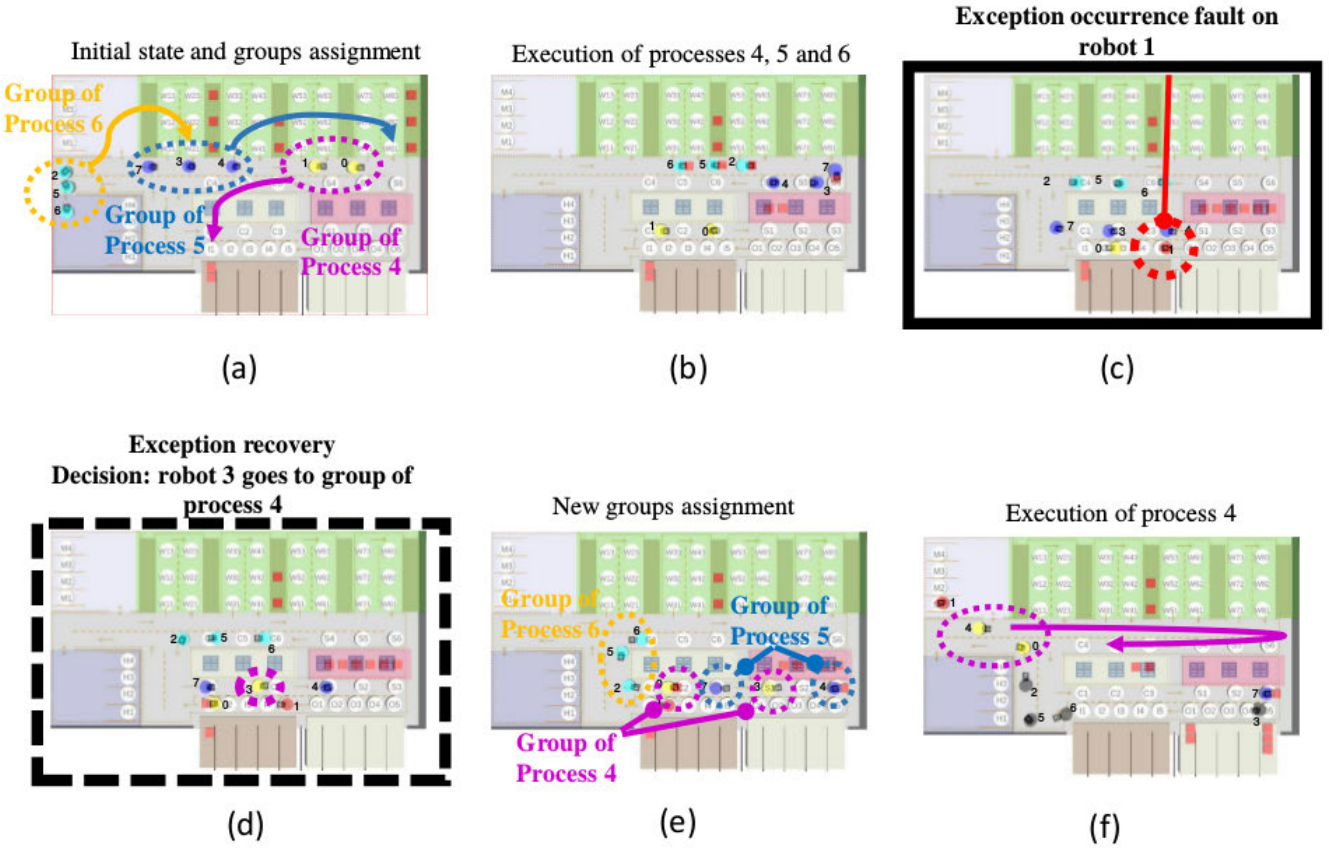
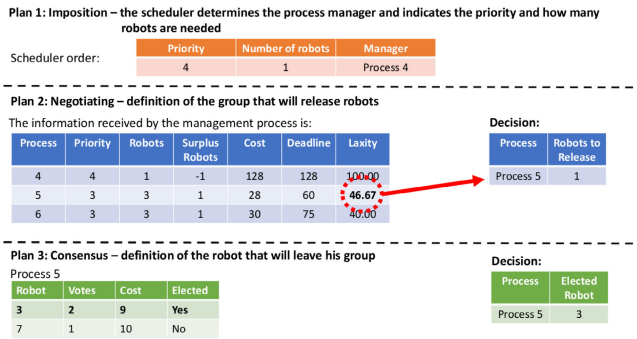


FIGURE 7. Execution of experiment 1.



Plan 1: Imposition – the scheduler determines the process manager and indicates the priority and how many robots are needed

Scheduler order:

Priority	Number of robots	Manager
4	1	Process 4

Plan 2: Negotiating – definition of the group that will release robots

The information received by the management process is:

Process	Priority	Robots	Surplus Robots	Cost	Deadline	Laxity
4	4	1	-1	128	128	100.00
5	3	3	1	28	60	46.67
6	3	3	1	30	75	40.00

Decision:

Process	Robots to Release
Process 5	1

Plan 3: Consensus – definition of the robot that will leave his group

Process 5

Robot	Votes	Cost	Elected
3	2	9	Yes
7	1	10	No

Decision:

Process	Elected Robot
Process 5	3

FIGURE 8. Ternary-Hierarchical decisions of the experiment 1.

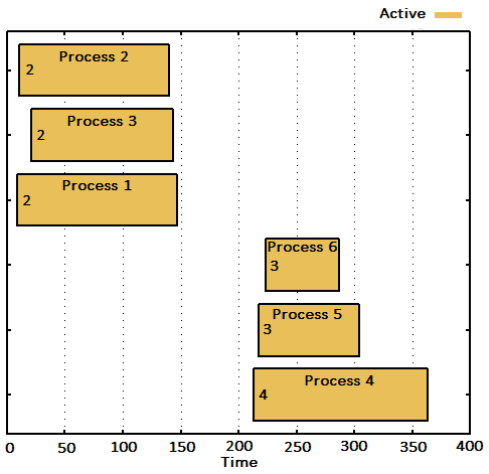


FIGURE 9. Gantt chart of experiment 1.

same number of robots, the same number of surplus robots, and the same priorities. Thus, process 4 needed to decide between the two and chose the one with the highest degree of laxity, which in this case was process 5. Process 6 had a higher estimated deadline as its task had a greater extension of states in the warehouse, which started from aisle 1, whereas process 5 started only from aisle 4.

After choosing process 5 to release a robot to process 4, the process of deciding which robot to release started. In this case, robot 7 was selected as the election manager and received the vote of the other robots in the staff. The eligible robots are the ones with the highest cost to complete the current stage of the process, which were robots 3 and 7. During voting, each robot will vote among the eligible robots, which are those farthest from each other. Each robot will vote for the farthest robot, hence, robot 3 was the one chosen. In Figure 7 (d), robot 3 appeared in yellow, indicating that it was part of process 4.

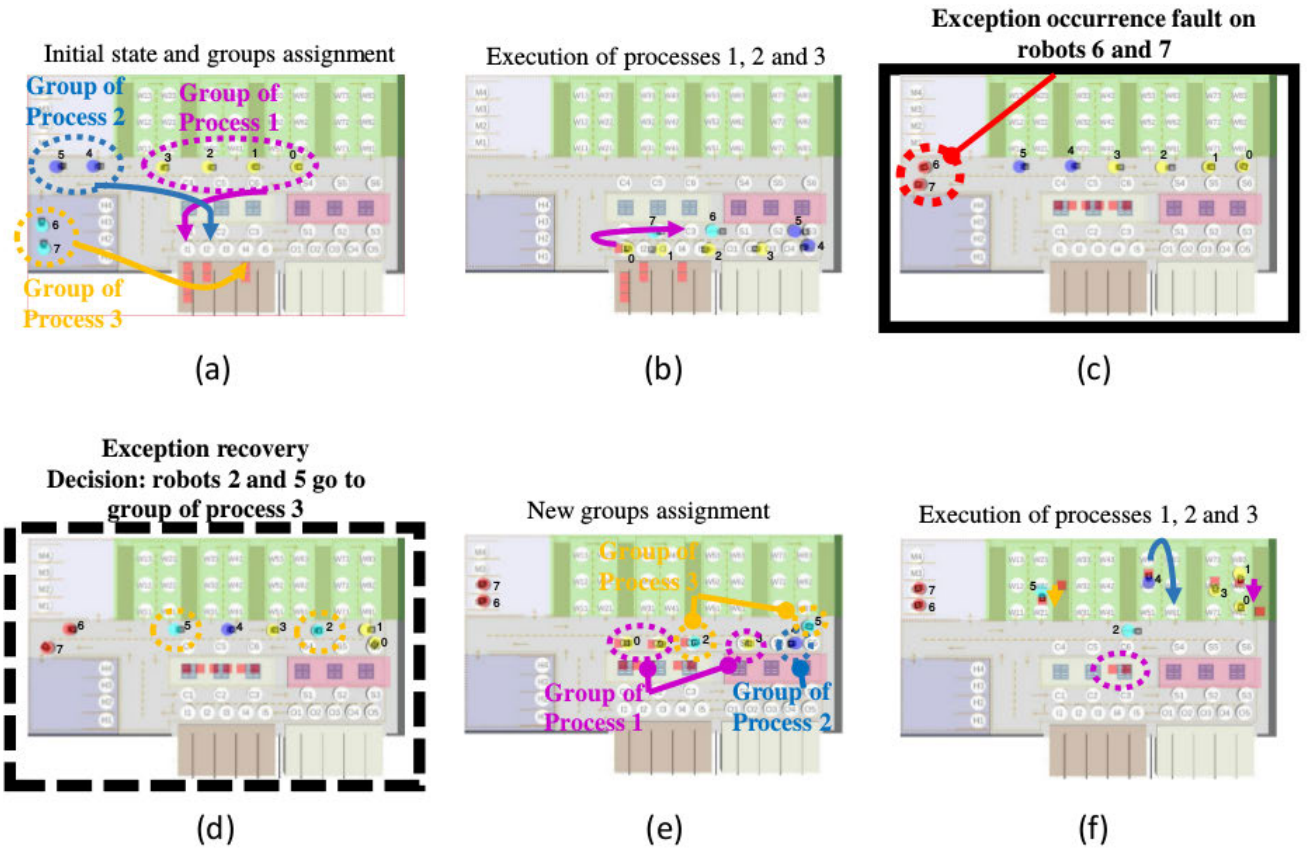


FIGURE 10. Execution of experiment 2.

Figure 7 (e) shows the scenario where robot 4 of process 5 starts transporting its box to the outgoing cargo sector. After the delivery of the boxes in the checking sector, robots 0 and 3 in process 4 were deallocated, and the process chose the robots closest to state C4 to continue the last stage of the process, which was the transportation of the boxes from the checking sector to the warehouse. The robots chosen by process 4 for this last stage were robots 0 and 4, as shown in Figure 7 (f). As shown in Figure 7 (f), it would also be possible for robot 7 to transport the last box in process 5 to the outgoing cargo sector.

The Gantt chart for this experiment is presented in Figure 9. The processes of interest in this analysis are processes 4, 5, and 6. Process 4, despite having the highest priority, was the one that had the longest execution time as it was an *Incoming Cargo* type process, which had more scenarios to be completed. Process 6 started after process 5; however, as there was no need to give robots because it was more advanced in its execution about process 5, it ended first.

B. EXPERIMENT 2

The second experiment is another example of fault resilience with three *incoming cargo* processes, as described in Table 2. Two robots from process 3 failed and each of the other processes yielded one robot.

TABLE 2. Description of Experiment 2.

#	Type of Process	Priority	Number of Boxes	Initial State	Warehouse Aisle
Process-1	*Incoming Cargo*	4	4	I1	4
Process-2	*Incoming Cargo*	2	2	I2	3
Process-3	*Incoming Cargo*	3	2	I3	1

The first scenario of experiment 2 is shown in Figure 10 (a), where four robots were allocated for process 1 (robots 0-3), two for process 2 (robots 4 and 5), and two robots for process 3 (robots 6 and 7). Figure 10 (b) shows when the robots from process 1 started to transport the boxes to the checking sector. Figure 10 (c) shows all the boxes in the checking sector waiting to be transported to their respective warehouse aisles. However, at that moment, robots 6 and 7 experienced some errors and could not proceed with process 3. The scheduler then signaled that priority process 3 requires two robots and assigned process 1 as the manager as it had the highest priority, as shown in Figure 11.

In Figure 11, it is shown that processes 1 and 2 each have a surplus robot. Thus, the process manager signaled to these two processes to give up one robot each. As process 1 had four robots and process 2 had two robots, the process of choosing among the robots in each process started. The number of

TABLE 3. Experiments carried out to provide accurate information on the approach.

	Process 1				Process 2				Process 3				Results				
Experiment	Fault	Start tick	End tick	Duration	Fault	Start tick	End tick	Duration	Fault	Start tick	End tick	Duration	Total cost	Average cost	Increase %	Total duration	Increase %
1	**No**	3	125	122	**No**	6	126	120	**No**	10	129	119	361	120,33	-	126	-
2	Yes	2	142	140	No	7	132	125	No	11	168	157	422	140,67	16,90	166	31,75
3	No	2	129	127	Yes	6	164	158	No	10	139	129	414	138,00	14,68	162	28,57
4	No	2	130	128	No	6	131	125	Yes	11	163	152	405	135,00	12,19	161	27,78
5	Yes	2	136	134	Yes	6	155	149	No	10	154	144	427	142,33	18,28	153	21,43
6	No	2	124	122	Yes	6	158	152	Yes	10	156	146	420	140,00	16,34	156	23,81
7	Yes	2	131	129	No	6	155	149	Yes	10	155	145	423	141,00	17,17	153	21,43
8	Yes	2	133	131	Yes	8	157	149	Yes	13	167	154	434	144,67	20,22	165	30,95

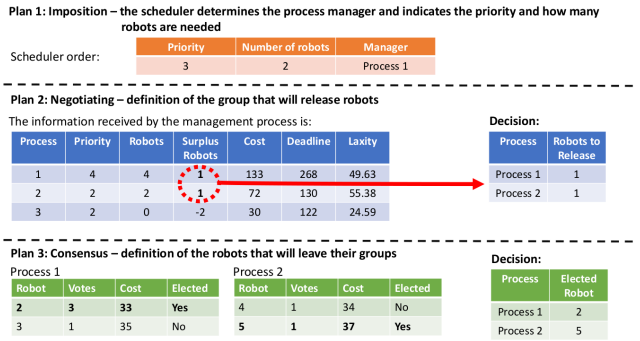


FIGURE 11. Ternary-Hierarchical decisions of the experiment 2.

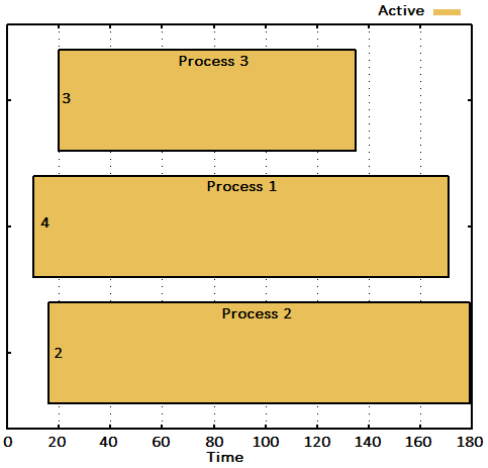


FIGURE 12. Gantt chart of experiment 2.

TABLE 4. Description of experiments.

#	Type of Process	Priority	Number of Boxes	Initial State	Warehouse Aisle
Process-1	*Incoming Cargo*	3	2	I1	3
Process-2	*Incoming Cargo*	3	3	I2	2
Process-3	*Incoming Cargo*	3	3	I3	1

eligible robots will be the number of required robots plus one. Thus, the manager of the robots decides which one is voted for the most. From process 1, the eligible robots were robots 2 and 3 as they had the highest costs for completing process 1 and the managing robot was robot 3. Each robot chose between robot 2 and 3 according to the distance from them, i.e., voting for the robot that is farthest away. Robot 2 was selected to be assigned to process 3 as it was the farthest from robots 0, 1, and 3.

For process 2, the choice of which robot to be assigned was made between robots 4 and 5. As each robot received a vote, the choice was up to the managing robot, which in this case was robot 5. Robot 5 was chosen as the one that had the highest cost to complete the task. Figure 11 shows that the robot with the highest cost was robot 5; hence, it was elected to leave the staff and go to process 3.

Robots 2 and 5 are represented using light blue in Figure 10 (d), indicating that they were assigned to process 3. Figure 10 (e) shows the scene where the robots in process 1 started shipping their boxes to the warehouse. Figure 10 (f) shows the boxes being delivered to their respective aisles and shows that there are still two boxes to be transported, referring to processes 1 and 2.

The Gantt chart for this experiment is presented in Figure 12. The Gantt chart shows that the process that finished its tasks first was process 3 as the robots assigned to it when its robots failed were already in a more advanced position. Process 1 was the second to finish, considering that it was the process with the highest number of boxes. Finally, process 2, with the lowest priority, was the last to complete its tasks, even if it had only two boxes, as it was left with only one robot to complete its tasks.

VII. ACCURACY AND PRECISION

The accuracy of the proposed approach to fault resilience through a ternary-hierarchical decision was evaluated through eight different experiments, which aimed to quantify the results. In this case, three *incoming cargo* processes were performed, as described in Table 4. Each process was introduced into the system at different times.

The first experiment, presented in Table 3, was considered the base experiment, where no failure occurs in the three processes. The other experiments presented all possible

combinations of failures between the processes, with the last experiment showing a failure. These failures correspond to the failure of one robot in each process. The purpose of these experiments was to verify the increase in the scheduler's number of cycles in executing each process, called *tick*. Table 3 also shows the results obtained from the eight experiments and their analysis. In the *results* column, the *total cost* column is the sum of the *duration* column for each process. The *average cost* column is the average of the *total cost* column for the three processes and the *total duration* column is the total duration in ticks of each experiment.

Experiments 2 to 8 had an average total cost of 420 ticks, which was 16.54% higher compared with the total cost of experiment 1. Experiment 4 had the lowest total cost in the presence of failures. The only failure occurs in process 3, which did not require negotiation between the other processes to fill the gap. The experiment with the highest total cost was experiment 8 as all its processes failed. Consequently, these are also experiments with the lowest and highest average costs, respectively.

The average total duration of experiments 2-8 was 159 ticks, which was 26.53% higher compared with experiment 1. Experiments 5 and 7 were the ones with the shortest total duration, and experiment 2 was the one with the highest total duration.

In experiment 5, one robot in process 1 and one robot in process 2 failed. Thus, one robot from process 3 was assigned to process 1 to keep the minimum. Even with the failure of one robot, process 2 still had the required minimum and did not require replacement. The process ended quickly because process 2 had finished loading two boxes earlier than process 1, making one robot available for process 3.

In experiment 7, one robot in process 1 and one robot in process 3 failed. For the failure in process 1, the robot was replaced by one robot from process 3. When one robot in process 3 failed, the number of robots became less than the required minimum number, causing process 2 to yield one robot. However, process 2 maintained the minimum requirement; thus, it did not require additional robots. As in experiment 5, it supplied process 3 with one robot to transport the last box faster.

In experiment 2, one robot in process 1 failed, forcing process 3 to give up one robot, so that process 1 could keep the minimum of two robots required for priority 3 processes. However, this had caused a considerable delay in the experiment as there was no other process that could make the robots available, such as in experiments 5 and 7.

Lastly, even with the failures, all processes were attended to, demonstrating the resilience of the proposed approach to failures.

VIII. CONCLUSION

This work proposes the fault-resilient collective ternary-hierarchical behavior approach for multi-agent systems for achieving resilience to failures. The approach is based on three hierarchical plans. Fault recovery is a collective behavior obtained through the decisions made in these plans, where resilience is achieved through the decisions taken at global, procedural, and individual levels. This collective behavior helps smart factories improve the reliability and robustness of their solutions from failure.

Quantitative analysis of the approach through various experiments demonstrated the efficiency of the method in recovering from failures. In this analysis, eight experiments, each with three processes, were conducted. Results show an average total duration of 159 ticks for the experiments that failed, which was 26.53% higher than that of the base experiment. In addition, the average total cost of these experiments was 420 ticks, which was 16.54% higher than that of the base experiment. The results show that all processes were attended when failures occurred. Although there was an increase in the total duration and total cost of the experiments after recovery from faults, the resilience to failures of the multi-robot system was obtained.

As future work, we intend to expand the level of consensus and carry out the experiments in an environment with more robots.

REFERENCES

[1] B. P. Gerkey and M. J. Matariá, "A formal analysis and taxonomy of task allocation in multi-robot systems," *Int. J. Robot. Res.*, vol. 23, no. 9, pp. 939–954, Sep. 2004.
[2] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated multi-robot exploration," *IEEE Trans. Robot.*, vol. 21, no. 3, pp. 376–386, Jun. 2005.
[3] D. Crestani, K. Godary-Dejean, and L. Lapierre, "Enhancing fault tolerance of autonomous mobile robots," *Robot. Auto. Syst.*, vol. 68, pp. 140–155, Jun. 2015.
[4] N. Beckman and J. Aldrich, "A programming model for failure-prone, collaborative robots," in *Proc. Int. Workshop Softw. Develop. Integr. Robot.*, 2007, pp. 1–5.
[5] Y. Yang, S. Souissi, X. Défago, and M. Takizawa, "Fault-tolerant flocking for a group of autonomous mobile robots," *J. Syst. Softw.*, vol. 84, no. 1, pp. 29–36, Jan. 2011.
[6] D. Tarapore, A. L. Christensen, and J. Timmis, "Generic, scalable and decentralized fault detection for robot swarms," *PLoS ONE*, vol. 12, no. 8, pp. 1–29, Aug. 2017.
[7] Z. Guo, W. Yang, M. Li, X. Yi, Z. Cai, and Y. Wang, "ALLIANCE-ROS: A software framework on ROS for fault-tolerant and cooperative mobile robots," *Chin. J. Electron.*, vol. 27, no. 3, pp. 467–475, May 2018.
[8] M. Koutsoubelias and S. Lalis, "Fault- tolerance support for mobile robotic applications," in *Proc. IEEE 13th Int. Symp. Ind. Embedded Syst. (SIES)*, Jun. 2018, pp. 1–10.
[9] N. Li, M. Li, Y. Wang, D. Huang, and W. Yi, "Fault-tolerant and self-adaptive market-based coordination using hoplites framework for multi-robot patrolling tasks," in *Proc. IEEE Int. Conf. Real-time Comput. Robot. (RCAR)*, Aug. 2018, pp. 514–519.
[10] J. Panerati, M. Minelli, C. Ghedini, L. Meyer, M. Kaufmann, L. Sabattini, and G. Beltrame, "Robust connectivity maintenance for fallible robots," *Auto. Robots*, vol. 43, no. 3, pp. 769–787, Mar. 2019.
[11] M. Blanke, R. Izadi-Zamanabadi, S. A. Bøgh, and C. P. Lunau, "Fault-tolerant control systems — A holistic view," *Control Eng. Pract.*, vol. 5, no. 5, pp. 693–702, May 1997.
[12] E. Hollnagel, D. Woods, and N. Leveson, *Resilience Engineering: Concepts Precepts*. Farnham, U.K.: Ashgate, 2006.
[13] S.-H. Oh and S.-M. Yang, "A modified least-laxity-first scheduling algorithm for real-time tasks," in *Proc. 5th Int. Conf. Real-Time Comput. Syst. Appl.*, 1998, pp. 31–36.
[14] L. Piardi, V. C. Kalempa, M. Limeira, A. S. de Oliveira, and P. Leitáo, "ARENA—Augmented reality to enhanced experimentation in smart warehouses," *Sensors*, vol. 19, no. 19, p. 4308, Oct. 2019.

VIVIAN CREMER KALEMPA received the degree in computer science from Santa Catarina State University (UDESC) and the master's degree in computer science from the Federal University of Santa Catarina (UFSC). She is currently pursuing the Ph.D. degree with the Federal University of Technology-Parana (UTFPR). She is also a Professor with UDESC. Her research interests include intelligent production systems, collaborative robots, and autonomous mobile robot.

LUIS PIARDI received the degree in electronic engineering from the Federal University of Technology-Parana (UTFPR), Brazil, and the master's degree in industrial engineering from the Instituto Politécnico de Bragança, Portugal. He is currently pursuing the Ph.D. degree with the Graduate Program in Electrical and Computer Engineering, UTFPR. He is also a Researcher with the Research Centre in Digitalization and Intelligent Robotics (CeDRI). His research interests include intelligent production systems, machine learning, collaborative robots, and autonomous mobile robot.

MARCELO LIMEIRA received the degree in electrical engineering from Positivo University (UP) and the master's degree from the Federal University of Technology-Parana (UTFPR), where he is currently pursuing the Ph.D. degree. His research interests include industrial automation, mobile robotics, and artificial intelligence.

ANDRÉ SCHNEIDER DE OLIVEIRA (Member, IEEE) received the M.Sc. degree in mechanical engineering, concentrate in force control of rigid manipulators from the Federal University of Santa Catarina (UFSC), in 2007, and the Ph.D. degree in engineering of automation and systems, in 2011. His thesis focused on differential kinematics through dual-quaternions for vehicle-manipulator systems. He is currently an Adjunct Professor with the Federal University of Technology-Parana (UTFPR) and a member with the Advanced Laboratory of Robotics and Embedded Systems (LASER) and the Laboratory of Automation and Advanced Control Systems (LASCA). His research interests include robotics, mechatronics and automation, with particular focus on navigation and localization of mobile robots, autonomous and intelligent systems, perception and environment identification, cognition, deliberative decisions, and human-interaction and navigation control.