Reflex-Plan: A Safety Monitoring Architecture for Thinking Fast and Slow

Momina Rizwan[1] Christoph Reichenbach[1] and Volker Krueger[1]

***Abstract*— Ensuring functional safety is crucial for the deployment of autonomous systems in real-life dynamic environments, as they must operate reliably and safely among humans. However, existing safety systems are designed with a closed-world assumption and can over-constrain the system by shutting down the robot at every safety violation, limiting the robot's ability to complete its tasks. To address this problem, we present a novel operational safety approach supported by our software architecture Reflex-Plan, where a safety monitor proactively influences high-level planning to enable safe and adaptive recovery behaviors thus preventing unnecessary stops. Unlike traditional safety monitors that primarily react to violations through predefined stop mechanisms, our software architecture follows a two-step process: the fast-thinking safety monitor provides immediate reflexive responses, while the slow-thinking high-level planner processes the safety monitor's feedback to plan recovery strategies. This allows the robot to respond quickly to safety-critical situations while maintaining adaptability for long-term autonomy. We validate the effectiveness of Reflex-Plan through real-world robot experiments in a mock hospital environment. Our experimental results confirm that keeping immediate safety responses within the safety monitor ensures fast reactivity while delegating recovery strategies to the reasoning layer enables efficient adaptation, reducing failures and ensuring more stable operation without reliance on external intervention.**

I. Introduction

Functional safety is "the part of the overall safety that depends on a system or equipment operating correctly in response to its inputs, including safe management of potential faults within the system" [1]. Functional safety is key to realizing the potential of autonomous robots, from AI-driven assistants to self-driving cars, when human lives are at stake, as these robots increasingly work alongside humans in unpredictable environments [2], [3]. However, overly conservative safety approaches, such as emergency stops or strict operational constraints compromise autonomy and efficiency, especially in dynamic environments [4].

To standardize safety in robotics and automation, ISO norms [5], [6], [7], [8] establish global safety benchmarks that manufacturers and operators follow. ISO norms establish strict safety requirements including Human-robot interaction limits, Operational Safety Zones and Fail-Safe Mechanisms. While these measures ensure safe operation, they limit autonomy in unpredictable environments. For example, autonomous robots might be forced to stop if they can't clearly detect obstacles or rule out human presence, limiting autonomy in dynamic settings.

[1]Department of Computer Science, Faculty of Engineering (LTH), Lund University, SE 221 00 Lund, Sweden. E-mail: <firstname>.<lastname>@cs.lth.se.

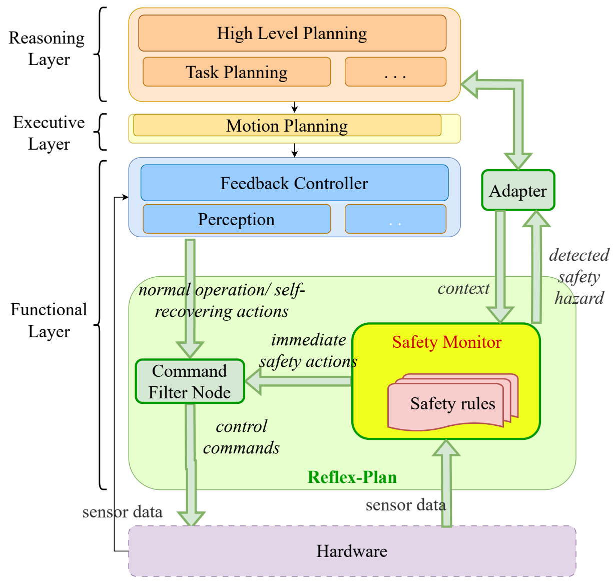

Fig. 1: The proposed Reflex-Plan architecture, highlighted in green, is integrated within the standard three-layer software framework for autonomous systems, illustrating the interplay between fast and slow thinking. Fast thinking—embodied by the safety monitor and command filter—enables rapid responses to hazards, while slow thinking—reflected in the planner—supports deliberative decision-making. The diagram highlights the bidirectional communication channels that connect these components.

Consider a medicine delivery robot in a hospital, autonomously transporting medicines from the nurse's desk to the patient wards. The robot is equipped with sensors and a safety monitor that relies on predefined safety rules and assumptions about what constitutes a hazard. As the robot moves through a corridor, it encounters a wheelchair access ramp leading to a patient room. The incline is unexpected, slightly steeper than what is considered safe in the safety rules. The safety monitor detects this unmodeled anomaly and enforces a full stop. This full stop delays essential medicine delivery by hours due to required human intervention and blocks the ramp, which also serves as an emergency exit, thereby creating a safety risk. Such catastrophic outcomes highlight the critical need for an operational safety framework that allows the robot to recover autonomously in these situations, finding an alternate but safe route to the patient without introducing new hazards.

State-of-the-art runtime safety monitors [9], [10] for robots often defaults to a generic fail-safe action, such as emergency stops, because they cannot dynamically formulate new safety reactions. In essence, current rule-based monitors only act as independent safety overrides, rather than as guiding components that updates the planner to find a solution. This contrasts with what we would like to see: an operational safety approach that allows the system to continue to function (perhaps in a limited capacity) despite the hazard. While some state-of-the-art systems [11] offer alternatives to full stops, e.g. in the form of a switch to a low-speed "manual mode", these reactions are usually tied to pre-programmed mode switches rather than a re-planning of its task. High-level planners and safety monitors often run in parallel with minimal interaction, which means that safety overrides halt the mission instead of guiding it to a safe success.

To address these limitations, we present a runtime safety monitoring architecture, *Reflex-Plan*, that supports our new operational safety approach, as illustrated by the green blocks in Fig. 1. The figure shows how our approach integrates "fast thinking" safety monitoring at the functional layer, with "slow-thinking" context-aware high-level planning in the reasoning layer. An Adapter mediates between these layers. This connected safety–planning framework enables robots to apply high-level reasoning to open-world uncertainty without compromising on fast reactions.

Key contributions of this paper are as follows:

1) An operational safety approach that goes beyond traditional reactive shut-down mechanisms by enabling communication between runtime safety monitoring and high-level planning. This ensures that safety monitors provide real-time situational awareness, while high-level planners supply the conceptual context and operational goals.
2) A software architecture, Reflex-Plan, that facilitates real-time communication between fast reacting safety monitoring and slow-thinking high-level planning, ensuring both reflex-like rapid safety enforcement and context-aware recovery strategies without unnecessary task interruptions.
3) Experimental validation on a real robot, demonstrating the effectiveness of our approach. We evaluate the architecture's performance based on task completion rate and response time, while trigger strength provides insight into how the system dynamically adjusts safety responses to varying hazard levels..

II. Related Work

A safety monitor (also known as a safety manager [12], or a safety bag [13]) is a critical runtime component that observes sensor data, detects potential hazards, and enforces safety constraints to prevent unsafe system behaviors. Some safety monitors enable robots to function as self-aware systems [14], dynamically modifying their behavior in response to environmental or internal system changes. Such safety monitors are integrated into the control logic [15], enforcing safety at the execution level. However, we adopt a different approach, treating the safety monitor as an independent component, ensuring real-time safety compliance without being affected by faults or delays in the control system. This follows Baudin et al.'s requirement that "safety systems have to be independent, i.e., designed from an independent specification composed of a set of safety properties [. . .] [enforcing] safety properties independently from any faults of the functional system." [4]

Safety monitors can be classified into two main types: rule-based, and model-based. Rule-based monitors enforce predefined safety constraints using if-then logic, providing deterministic and interpretable safety enforcement [11], [16], [17]. These approaches are computationally lightweight and reactive, making them well-suited for real-time safety enforcement. However, they are limited in adaptability and cannot handle unforeseen hazards.

Model-based monitors [18], [19] rely on formal system models to predict and verify safe behaviors before execution. These methods can systematically rule out hazards under a closed-world assumption, ensuring that safety properties are mathematically enforced. One of those model-based approaches by Weber et. al. [20] focuses on runtime diagnosis of known failure, and graceful degradation, enabling systems to continue operation with degraded capabilities. However, model-based methods are inherently not designed to function in open-world settings, where unforeseen hazards require adaptive recovery strategies beyond predefined models.

Several works attempt to move beyond rigid safety enforcement by integrating adaptive recovery strategies. Honda et al. [21] employ reinforcement learning to bridge fast and slow reactions, allowing autonomous systems to modify behavior based on real-time sensor feedback. Trapp and Weiss [22] propose dynamic safety management, where different safety configurations are applied depending on contextual awareness. Bonfanti et al. [23] integrate adaptive control techniques to dynamically adjust robot actions, minimizing task disruption. Masson et al. [9] introduce active safety monitors, which define safety and permissiveness properties in a state-space framework, ensuring goal reachability despite safety interventions. While these methods enable more flexible safety enforcement, they primarily focus on motion-level adaptation rather than task-level recovery planning. Our approach extends beyond motion planners, influencing higher-level decision-making to enable task recovery rather than just enforcing constraints.

Most rule-based, and model-based safety monitors enforce constraints at the execution level, ensuring immediate safety compliance but often leading to unnecessary task failures due to strict rules that lack recovery mechanisms. Our approach differs by maintaining an independent safety monitor, ensuring real-time enforcement without computational slowdowns. We separate fast reflexive responses from slow recovery planning, but extend beyond motion-level corrections to enable task-level adaptation. Unlike model-based approaches that assume closed-world conditions, our framework dynamically recovers from unstructured hazards in open-world settings. Additionally, we go beyond context-aware safety rule switching by actively integrating safety constraints into the task planner, enabling high-level recovery strategies rather than mere execution adjustments.

ce of

III. Overview

In this Section, we further elaborate on our approach compared to the classical rule-based safety monitoring approaches. We then introduce our software architecture Reflex-Plan to describe how different components are organized and communicate to ensure both immediate safety compliance and adaptive recovery.

A. *Operational Safety Behaviors*

A typical safety monitoring system consists of a set of safety rules, as shown in Fig. 2 (left), guiding the monitor's behavior by specifying conditions that must be met to ensure safe operation. A safety rule might specify an unsafe state, e.g., when a robot's speed has exceeded a certain maximum speed. When a safety rule is triggered (represented as red block in Fig. 2), the system adopts a predefined *safety behavior* to bring the system to a fail-safe state.

For built-in safety monitors in industrial robots such as the KUKA LBR iiwa or UR5 that rely on PLC-based or FPGA-based safety systems [24], [25], [26] but also for software safety monitors such as proposed by Adam et al. [11], these safety behaviors are either (1) *Terminal* such as stopping, or permanent reduction of some safety-relevant behavior, such as movement speed [11] or (2) *Temporary* such as entering fail-safe states, or resetting the system to a known safe condition. The runtime behavior of these state-of-the-art safety monitors is shown in Fig. 2 (left). Once a safety monitor transitions into a fail-safe state, it typically remains in that state until manually reset or overridden, preventing autonomous recovery.

Inspired by human cognition, we propose two complementary safety behaviors: "fast thinking" *Adaptive behavior* for immediate, low-level safety responses, which can adjust the parameters of the safety monitor's safety behaviors to reduce the impact of the safety action on the robot's task and "slow thinking" *Self-recovering behavior* which requires high-level decision making and can enact complex safe recovery plans with the help of the robot control logic.

1) Adaptive behavior

Fig. 2 (centre) illustrates how the *Adaptive behavior* is activated by the safety monitor after a safety rule is triggered (shown as red). An adaptive behavior is when the robot temporarily modifies its parameters to manage a detected change or hazard. *Adaptive behavior* is a broader concept than the temporary behaviors seen in existing safety monitors, as it allows for graded responses instead of binary enforcement mechanisms.

A key factor in determining the robot's *Adaptive behavior* is trigger strength, which represents the severity of the detected hazard. Rather than applying a fixed response, the safety monitor scales adaptive actions based on trigger strength, allowing finer control over safety interventions. For example, a low trigger strength may result in gradual speed reduction when approaching a crowded area, while a higher trigger strength—such as a rapidly approaching obstacle—may force an immediate halt.

Adaptive behavior is activated for less complex hazards, where the safety monitor can recover without interacting with high-level planning. Examples include temporarily reducing speed when approaching a crowded area, making a small detour around a human, or adjusting the robot platform's speed and torque based on direct sensor input, thereby avoiding an unnecessary full stop.

Since *Adaptive behavior* must execute in real-time to ensure immediate safety compliance, they are predefined and computationally lightweight. These responses are short-lived and automatically revert when the hazard disappears, preventing unnecessary delays while maintaining safety.

2) Self-recovering behavior

Fig. 2 (center and right) illustrates self-recovering behavior of the system. When a safety rule is triggered, the safety monitor immediately executes an *Adaptive behavior* to mitigate the detected hazard and ensure real-time safety compliance as shown in fig. 2 (center). This adaptive response—such as slowing down due to excessive tilt—remains in effect as long as the hazard persists. Crucially, the safety monitor does not depend on the high-level planner; it continues enforcing safety constraints independently while simultaneously sending structured feedback to inform the planner of the situation.

Once the planner receives this feedback, it can initiate a *Self-recovering behavior* by sending commands through the robot control logic, as illustrated in fig. 2 (right). This recovery process involves deliberative planning [27], where the control logic re-evaluates the constraints, updates the world model, and determine alternative approaches to resume the task safely. Unlike *Adaptive behavior*, *Self-recovering behavior* incorporates reasoning over time, considering mission objectives, environmental changes, and broader system constraints to ensure long-term operational safety.

By maintaining clear role separation of the two behaviors, our approach ensures that the safety monitor reacts instantly to hazards while allowing the high-level planner to independently adapt task execution without delaying critical safety enforcement.

B. *Reflex-Plan*

To implement the operational safety approach described in Section III-A, we formalize the safety monitoring architecture, Reflex-Plan. Fig. 1 illustrates our design, which builds upon the standard three-layered architecture for decisional autonomy [28].

In traditional runtime safety monitoring, fail-safe mechanisms—such as stopping or resetting the system—are enforced at the functional layer, which operates closest to the hardware with direct access to sensors and actuators. Our proposed safety monitoring architecture, shown in Fig. 1, extends the functional layer to also communicate with the reasoning layer while maintaining its independence in enforcing immediate safety. This extension enables real-time safety compliance through *Adaptive behavior*,

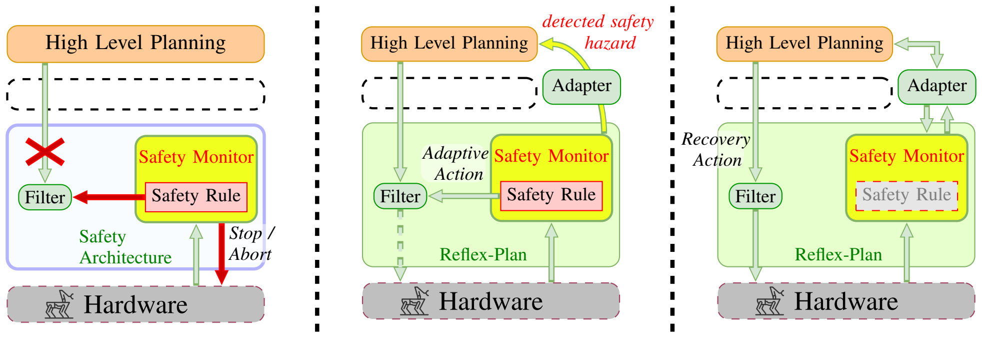

Fig. 2: A comparison between the runtime behavior of state-of-the-art Safety Monitoring approaches (left) and our proposed approach in safety mode (centre), and our approach reverting back to operational mode (right).

which acts like a human reflex, while simultaneously providing structured feedback to the reasoning layer to facilitate long-term *Self-recovering behavior*. It is important to note here that the safety monitor does not wait for the reasoning layer to approve actions; instead, it continuously enforces safety constraints while allowing the planner to independently evaluate recovery options. This dual-layer approach ensures that non-trivial recovery actions remain separate from immediate responses, balancing fast reactivity with strategic recovery planning."

Our architecture consists of the following core modules:

1) **Safety Monitor** (shown as a yellow box in Fig. 1) serves as the core runtime safety enforcement module. It enforces safety rules, executes *Adaptive behavior*, and provide feedback for the high-level planner to plan for *Self-recovering behavior* upon detecting a safety hazard.
2) **Command Filter Node** intercepts and validates control commands, ensuring that only commands compliant with active safety rules are executed.
3) **Adapter** which provides a bidirectional interface to facilitate communication between the safety monitor and the high-level planner.

To avoid increasing the inherent complexity of the safety monitor, we delegate deliberative recovery actions to high-level planning at the reasoning layer. This delegation does not require modifications to the high-level planner itself; instead, an Adapter (shown in Fig. 1) establishes structured communication channels, defining the "language" between the safety monitor and the planner. The adapter translates safety signals into new observations or constraints that the planner can interpret within its existing framework, enabling it to seek alternative solutions to achieve its current goal. Meanwhile, the Command Filter Node processes recovery actions like any other operational command, ensuring that the safety monitor can intervene if necessary.

To further understand the role of Adapter, consider a robotic arm with a gripper attempting to reach an object behind an obstacle. Upon detecting the obstacle, the safety monitor immediately halts movement and possibly withdraws the arm. The adapter then translates this safety event into a format the control logic can understand, allowing it to update its world model with the new constraint. The planner may then attempt to replan its approach by adjusting the arm pose or removing the obstacle entirely [27]. However, the safety monitor remains independent and retains override authority, ensuring that any new movement violating safety constraints is blocked, regardless of the planner's decisions.

Design Decisions

The separation of roles in our safety monitoring architecture is essential to ensure fast reaction times while enabling context-aware recovery planning. Since *Self-recovering behavior* require a higher level of decision-making, their execution is delegated to the reasoning layer, where the planner assesses mission impact, evaluates constraints, and formulates strategic recovery strategies. This separation ensures that safety enforcement remains fast and reactive, while long-term adaptations are made with a broader understanding of the system's goals and constraints.

To enable this coordination, the adapter needs to define a "language" in which low-level safety constraints are communicated to the planner and vice versa.. Based on the observable sensor readings, the safety monitor can provide the following types of information: the nature of the hazard (e.g., a blocked path or an unexpected obstacle), the affected system components (e.g., an overheated motor causing functional degradation), and recommended constraints or adaptations (e.g., marking restricted areas in navigation planning or limiting torque in specific joints to prevent mechanical stress). Based on this information, the reasoning layer takes corrective actions, which may include: updating the world model (e.g., recognizing a new obstacle in the environment), modifying task execution (e.g., switching grasping to an alternative arm if one is compromised), and replanning motion constraints (e.g., dynamically adjusting safe operating areas based on newly detected hazards). These corrective strategies vary depending on the reasoning layer implementation and system requirements, allowing flexibility in how adaptation is handled across different robotic architectures.

While the primary role of the safety monitor is to enforce immediate safety actions and provide structured feedback to the planner, minimal but essential context from the planner can also refine safety rule interpretation. Since safety rules are often context-dependent [22] rigid enforcement without context may lead to overly conservative behaviors that unnecessarily disrupt execution.

For example, detecting light pressure on a gripper's force sensor could indicate a safety violation in certain cases, such as handover tasks, where unintended contact with a human could be dangerous. However, in other tasks—such as board cleaning or insertion tasks—this force may be necessary and intentional. Without additional context, the safety monitor might incorrectly interpret these expected forces as hazards, leading to unnecessary stops or constraints that degrade system performance. To prevent such misinterpretations, planner reports task-specific features to the safety monitor via the adapter, treating them as additional input signals ("context" arrow shown in Figure 1). This allows the safety monitor to refine its enforcement criteria while still ensuring real-time safety compliance.

By structuring communication in this way, the safety monitor remains independent and reactive, while the planner receives meaningful safety information to refine long-term task execution. This ensures adaptability without compromising strict safety enforcement, making the approach modular and applicable across different robotic systems.

IV. Experimental Evaluation

To validate our proposed operational safety approach and its supporting software architecture Reflex-Plan, we conduct experiments on a mobile robot navigating through a mimicked unstructured hospital environment. Fig. 3 depicts the layout of the hospital task, including the planned navigation goal and potential unknown safety hazards encountered along the way, including a wheel chair ramp that is too steep for safe navigation. The real-world execution of these experiments takes place in a lab environment designed to simulate the user-case conditions (see Fig. 4) with a wooden ramp mimicking the unsafe wheel chair ramp. Our evaluation focuses on assessing the impact of integrating safety monitoring with high-level planning, ensuring both fast-reacting adaptive behaviors and slow-thinking context-aware recovery behaviors. We assess our framework using the following key metrics: trigger strength, which evaluates how well the adaptive behavior responds to varying hazard intensities, task completion, which measures whether Reflex-Plan can successfully resume or modify the original plan rather than defaulting to failure, and response time, which assesses whether the architecture enables fast execution of adaptive actions, ensuring immediate safety compliance.

A. Experimental Setup

1) Robot Platform and the software setup

We conducted our experiments on our *Heron* robot platform (shown in Fig. 4), which integrates a MIR 200 mobile platform and a UR5e manipulator. The MIR is equipped with two laser scanners, wheel encoders and an IMU sensor to detect and respond to safety hazards while maneuvering in diverse environments. Mounted on the MIR, the UR5 provides six degrees of freedom and has a six-axis force/torque sensor at the wrist to detect collisions. A real-sense camera is also connected to the wrist of the UR5.

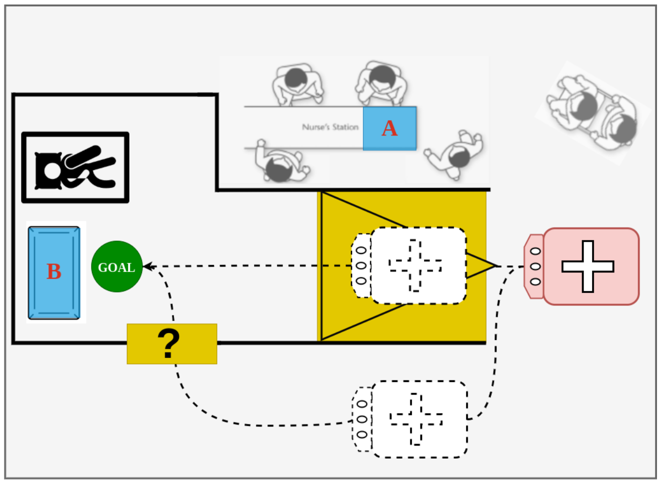

Fig. 3: A medicine delivery robot navigating in a hospital with dynamic, uncertain environment that require quick, safe decision-making. Possible paths to deliver medicines are shown using dashed lines. Unknown safety hazards are marked in yellow.

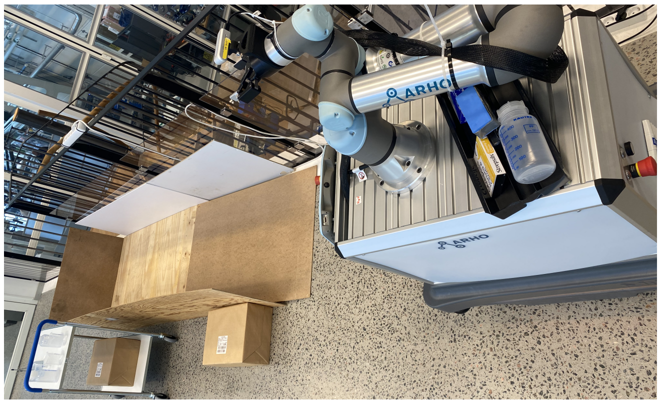
Fig. 4: Our mobile robot Heron delivering medicines to specified locations in our hospital corridor mock-up with a wooden ramp as the unknown wheel chair ramp.

We implement Reflex-Plan in ROS, using a modular design in which each component operates as an independent ROS node. The Safety Monitor is generated using our DSL ROSSMARie [29] for defining and enforcing safety constraints. The architecture integrates with our skill-based robot control platform SkiROS2 [30], but the modularity of Reflex-Plan ensures that it can be adapted to any other control frameworks as well.

To facilitate structured interactions between safety monitoring and control logic, the Adapter and Command Filter (see Fig. 1) are implemented as ROS nodes, enabling message-based communication over ROS topics. The Command Filter validates and regulates control commands based on active safety constraints before execution, while the Adapter translates low-level safety events into planner-relevant constraints and vice versa. These ROS-based communication channels ensure integration of fast-reacting adaptive behaviors and long-term self-recovering behaviors, providing a scalable solution for runtime safety monitoring in autonomous robotic systems.

The risk assessment of the robot indicates that while the MIR is well equipped with bumpers and safety features to maintain a certain distance from obstacles, it is less effective at adapting to uneven terrain, which thus introduce a risk of damaging the robot and the medicines it is carrying.

2) Environment Setup

The setup simulates a healthcare environment with a patient's room and a nurse's room (see the SLAM map in Fig. 5). Table A simulates the nurse's station, where the robot picks up new medications. Table B simulates the patient's room, to and from which the robot delivers and and retrieves medicine boxes. As obstacles, we introduce a ramp in the corridor, bumps on the path, and variations in table heights (Fig. 4) to test the robot's response.

The experiment consists of two scenarios:

1) **Scenario 1:** The robot starts at Location A and navigates to Location B to collect empty bottles. Here, we have two hazard points: 1 – the robot detects an upward-sloping ramp; 2 – the robot is carrying the box and detects a downward-sloping ramp;
2) **Scenario 2:** The robot collects medications from Location A and delivers them to Location B. Again we have two hazard points: 3 – the robot is carrying the box and encounters an upward-sloping ramp; 4 – the robot attempts to place the medicines on Table B but experiences arm jerks due to the height difference from the planned placement.

For each run, we assume that the robot operates in an environment that is unknown, and it relies solely on the runtime safety monitor to detect hazards, including ramps, obstacles, etc.

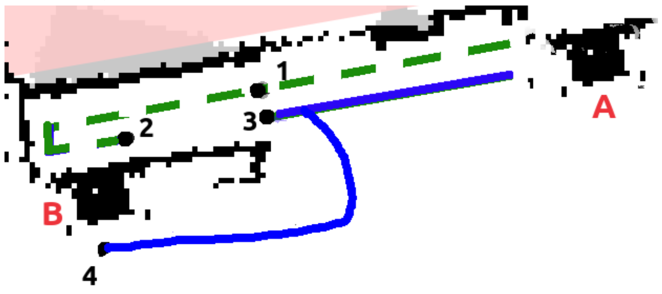

Fig. 5: SLAM map of the experimental setup generated by the MIR's LiDARs. The green dashed line route shows the robot's path for *Scenario 1*. The blue route represents the robot's path for *Scenario 2*. Hazard points are labeled as follows: 1 – robot detects an upward-sloping ramp; 2 – robot is carrying the box and detects a downward-sloping ramp; 3 – robot is carrying the box and encounters an upward-sloping ramp; 4 – robot attempts to place the medicines on Table B but experiences arm jerks due to a height difference from the planned placement.

B. Safety Monitor

We utilized ROSSMARie [29] to generate the safety monitor based on the design outlined in Section III-A. Following the risk assessment in Section IV-A.1, we manually crafted safety rules to specify the normal ranges of sensor inputs, to detect any violation in unstructured environments where not all hazards can be anticipated. Listing 1 presents our safety specifications in the ROSSMARie DSL [29], a dialect of DeROS [11]. The safety behaviors are detailed in Lines 24–38, showcasing the system's response strategies. Fig. 4.

```
1  # Actions
2  action stop;
3  ...
4  # Input
5  input orientation = topic /fmInformation/imu.orientation
6  ...
7  # Thresholds
8  const max_force = -15 N/sec
9  ...
10 # Events
11 entity imu_sensorsystem {
12    tilt_no_tok :
13       orientation.pitch() < max_tilt_neg or
14       orientation.pitch() > max_tilt_pos for 4.0 sec;
15 }
16 entity drive_system {
17    moving : linear_speed > non_stationary_speed;
18 }
19 entity force_monitor {
20    oscillation_not_safe :
21      force.z > max_force for 3.0 sec;
22 }
23
24 # Behaviors
25 if imu_sensorsystem.tilt_not_ok and
26     drive_system.moving and
27     object_on_top  then {
28        stop;
29        goBack; stop;
30        notify_planner("Obstacle_detected_at ", (currect_x,
     current_y,current_z))
31 };
32 if imu_sensorsystem.tilt_not_ok and
33     drive_system.moving and
34     not object_on_top  then {
35       slowDown;
36 };
37 if force_monitor.oscillation_not_safe then {
38       decrease_stiffness;
39       retract;
40       notify_planner("High_force_detected")
41 };
```

Listing 1: An example ROSSMARie specification to describe safety rules and recovery actions for the MIR robot navigating in an unstructured environment, as in e.g.

1) Scenario 1

When the robot encounters a ramp (Hazard Point 1 in Fig. 5), the integrated IMU sensor detects this event (defined in Lines 11–14) and triggers the *Adaptive behavior* in Lines 32–35. Instead of resorting to a stop or a complete shutdown, the adaptive behavior decreases the robot's speed to increase torque and stability while climbing the ramp, allowing the task to continue safely. At Hazard Point 2, the robot encounters the ramp after the robot has picked up the medicine box which triggers the *Self-recovering behavior* in Lines 25–30. The safety monitor first activates *Adaptive behavior*, stopping and tracing its footsteps back a certain number of steps. It then communicates the updated situation to the robot's high-level planner. In our scenario, the high-level planner is unable to find an alternative plan (as the robot is confined within a closed area, cf. Fig. 5), forcing the robot trigger a fail-safe state and calling for help.

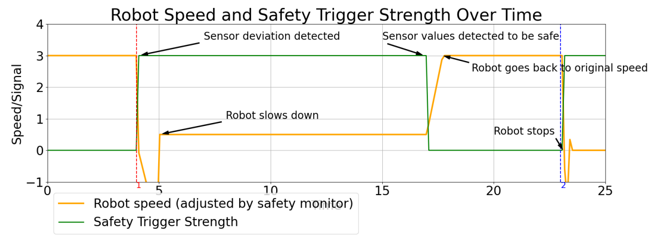

Fig. 6: Trigger strength diagram of the safety monitor over time (in seconds) for Scenario 1.

Fig. 6 shows the trigger strength [18] of the `tilt_not_ok` safety rule (green) and the robot's speed (orange) during Scenario 1, for one representative run. We show the hazard points as vertical dashed lines, and the threshold settings as activation and deactivation points of the green line. When sensor values exceed the thresholds, the safety monitor applies the *Adaptive behavior*, causing the robot to slow down, as shown by the corresponding changes in the robot's speed (orange line). At approximately 17 seconds, the sensor detects that the values have returned to a safe range, prompting the safety monitor to disengage and transfer control back to normal operations.

2) Scenario 2

The *Self-recovering behavior* in Lines 25–30 is triggered again at Hazard Point 3. Here, the control logic finds an alternative plan to reach **Table B**. At Hazard Point 4, the robot encounters an unexpected event when approaching the table from a lower position (not via the ramp). The actual table height is different from the assumed table height, causing the robot arm to jerk during the approach. This triggers safety behavior defined in Lines 37–40: first, the *Adaptive behavior* decreases the controller stiffness and retracts the arm to a safer height. Then it informs the control logic through the Adapter that the applied force is dangerously high. Subsequently, the robot control logic executes a *Self-recovering behavior* that switches to a compliant controller, allowing the arm to adapt to the new height and safely continue the task without interruption, showcasing our fast and slow thinking operational safety approach's ability to handle unexpected scenarios adaptively and maintain operational flow.

C. Response Time

Response time is defined as the duration between when a trigger is detected by the sensors and when the safety monitor or the high-level planner issues a command to respond [11]. We measured response time for both the safety monitor and for the high-level planner to execute *Self-recovering behavior* in response to unexpected changes, repeating each scenario 10 times. Table I reports the averages of our measurements: the *Adaptive behavior*s executed by the safety monitor respond in roughly $\frac{1}{10}$th of the time needed by the high-level planner, validating their role as a fast reflex system. Immediate safety compliance requires fast reactions in high-risk scenarios, while planner-driven recovery actions take longer due to communication latency and execution complexity (Table I).

D. Discussion

The robot successfully completed its tasks despite unexpected environmental changes, demonstrating the effectiveness of integrating fast thinking *Adaptive behavior* and slow-thinking *Self-recovering behavior*. However, at Hazard point 2 in Scenario 1, recovery was not possible autonomously, requiring human intervention to remove the payload (medicine box), enabling the robot to cross the ramp without damaging medicine bottles.

Importantly, safety was never compromised. The robot made informed decisions, such as avoiding unnecessary stops near the emergency exit in Scenario 2, enabling more efficient navigation. The safety monitor's sensitivity and thresholds were calibrated to balance risk management and task performance, preventing excessive interruptions while enforcing strict safety boundaries. Unlike traditional binary approaches—either continuing operation or shutting down—our method ensures controlled recovery within safe operational limits, enhancing both safety and efficiency.

V. Conclusions

In this paper, we introduced an operational safety framework supported by our software architecture, Reflex-Plan, that integrates a fast-thinking safety monitor with a slow-thinking high-level planner to achieve adaptive recovery actions with minimal task interruptions. Our experiments demonstrate that Reflex-Plan enables robots to dynamically adjust their original plan in response to detected hazards by executing *Adaptive behavior* first and then *Self-recovering behavior*. Safety is highly contextual, and traditional measures like emergency stops can sometimes introduce new risks, such as blocking emergency exits. Our framework addresses this challenge by enabling informed, context-aware actions that maintain safety without unnecessarily disrupting task execution.

Our architecture is broadly applicable across a wide range of autonomous robotic systems following the standard three-layered architecture. It is especially beneficial for robots operating in dynamic and unpredictable environments, where relying exclusively on system shutdowns is neither efficient nor practical. Instead of rigid, predefined fail-safe mechanisms, our framework enables nuanced, controlled safety responses that balance immediate hazard mitigation with long-term operational continuity. Future work will involve expanding validation across diverse robotic platforms and refining adaptive recovery strategies in increasingly complex and dynamic settings.

We believe that this work marks the beginning of a shift away from conservative safety approaches, such as emergency shutdowns, toward fully utilizing the capabilities of autonomous robots. By integrating cognitive reflexes with planning mechanisms, our architecture Reflex-Plan enables adaptive safety behaviors and balance safety with continuous recovery operations in dynamic environments.

Acknowledgements

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded

Hazard Points	Violated Rule in Listing 1	Safety Monitor			High-level Planning			Task Completion
		Actions	Response Time (sec)		Actions	Response Time (msec)		
			Mean	Std. D		Mean	Std. D	
1	Line 29	Slow down	0.88	0.039	–	–	–	✓
2	Line 22	Stop Go back	0.34 0.50	0.002 0.006	No alternative plan found	5.05	2.00	✗
3	Line 22	Stop Go back	0.40 0.68	0.008 0.002	Take a longer route	6.43	1.76	✓
4	Line 34	Retract Switch controller	0.49 0.41	0.002 0.325	–	–	–	✓

TABLE I: Experimental results of the two scenarios (repeated 10 times) are shown in Fig. 5. The hazard points 1, 2, 3, and 4 are the critical point (highlighted in Fig. 5) the robot encounters unexpected changes in the environment.

by Knut and Alice Wallenberg Foundation.

References

[1] *IEC 61508: Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems*, International Electrotechnical Commission Std., 2010, edition 2.0.

[2] A. Aniculaesei, D. Arnsberger, F. Howar, and A. Rausch, "Towards the verification of safety-critical autonomous systems in dynamic environments," *arXiv preprint arXiv:1612.04977*, 2016.

[3] M. A. Javed, F. U. Muram, H. Hansson, S. Punnekkat, and H. Thane, "Towards dynamic safety assurance for industry 4.0," *Journal of Systems Architecture*, vol. 114, p. 101914, 2021.

[4] É. Baudin, J.-P. Blanquart, J. Guiochet, and D. Powell, "Independent safety systems for autonomy: state of the art and future directions," Ph.D. dissertation, LAAS-CNRS, 2007.

[5] I. O. for Standardization, *Robots and robotic devices — Safety requirements for industrial robots*, ISO 10218-1:2011 ed. Vernier, Geneva, Switzerland: International Organization for Standardization, 2011. [Online]. Available: https://www.iso.org/standard/51330.html

[6] ——, *Robots and robotic devices — Collaborative robots*, ISO/TS 15066:2016 ed. Vernier, Geneva, Switzerland: International Organization for Standardization, 2016. [Online]. Available: https://www.iso.org/standard/62996.html

[7] I. O. for Standardization (ISO), "Earth-moving machinery and mining autonomous and semi-autonomous machine system safety," International Organization for Standardization," Standard, july 2019.

[8] I. O. for Standardization, *Robots and robotic devices — Safety requirements for personal care robots*, ISO 13482:2014 ed. Vernier, Geneva, Switzerland: International Organization for Standardization, 2014. [Online]. Available: https://www.iso.org/standard/53820.html

[9] L. Masson, J. Guiochet, H. Waeselynck, K. Cabrera, S. Cassel, and M. Törngren, "Tuning permissiveness of active safety monitors for autonomous systems," in *NASA Formal Methods Symposium*. Springer, 2018, pp. 333–348.

[10] P. Mallozzi, E. Castellano, P. Pelliccione, G. Schneider, and K. Tei, "A runtime monitoring framework to enforce invariants on reinforcement learning agents exploring complex environments," in *2019 IEEE/ACM 2nd International Workshop on Robotics Software Engineering (RoSE)*. IEEE, 2019, pp. 5–12.

[11] S. Adam, M. Larsen, K. Jensen, and U. P. Schultz, "Rule-based dynamic safety monitoring for mobile robots," *Journal of Software Engineering for Robotics*, vol. 7, no. 1, pp. 121–141, 2016.

[12] C. Pace and D. Seward, "A safety integrated architecture for an autonomous safety excavator," in *International Symposium on Automation and Robotics in Construction*, vol. 2, 2000, p. 2.

[13] P. Klein, "The safety-bag expert system in the electronic railway interlocking system elektra," in *Operational Expert System Applications in Europe*. Elsevier, 1991, pp. 1–15.

[14] J. Schlatow, M. Moostl, R. Ernst, M. Nolte, I. Jatzkowski, M. Maurer, C. Herber, and A. Herkersdorf, "Self-awareness in autonomous automotive systems," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, 2017, pp. 1050–1055.

[15] Y. Jiang, H. Liu, H. Song, H. Kong, R. Wang, Y. Guan, and L. Sha, "Safety-assured model-driven design of the multifunction vehicle bus controller," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 10, pp. 3320–3333, 2018.

[16] A. Ferrando, R. C. Cardoso, M. Fisher, D. Ancona, L. Franceschini, and V. Mascardi, "Rosmonitoring: a runtime verification framework for ros," in *Towards Autonomous Robotic Systems: 21st Annual Conference, TAROS 2020, Nottingham, UK, September 16, 2020, Proceedings 21*. Springer, 2020, pp. 387–399.

[17] C. Paul, L. Benjamin, S. Walter, and M. Brini, "Validation of safety necessities for a safety-bag component in experimental autonomous vehicles," in *2018 14th European Dependable Computing Conference (EDCC)*. IEEE, 2018, pp. 33–40.

[18] N. B. Haupt and P. Liggesmeyer, "A runtime safety monitoring approach for adaptable autonomous systems," in *Computer Safety, Reliability, and Security: SAFECOMP 2019 Workshops, ASSURE, DECSoS, SASSUR, STRIVE, and WAISE, Turku, Finland, September 10, 2019, Proceedings 38*. Springer, 2019, pp. 166–177.

[19] M. Stadler, M. Vierhauser, and J. Cleland-Huang, "Towards flexible runtime monitoring support for ros-based applications," in *Proceedings of the 4th International Workshop on Robotics Software Engineering*, 2022, pp. 43–46.

[20] J. Weber and F. Wotawa, "Combining runtime diagnosis and ai-planning in a mobile autonomous robot to achieve a graceful degradation after software failures," in *International Conference on Agents and Artificial Intelligence*, vol. 2. SCITEPRESS, 2010, pp. 127–134.

[21] K. Honda, R. Yonetani, M. Nishimura, and T. Kozuno, "When to replan? an adaptive replanning strategy for autonomous navigation using deep reinforcement learning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 6650–6656.

[22] M. Trapp and G. Weiss, "Towards dynamic safety management for autonomous systems," in *27th Safety-Critical Systems Symposium (SSS)*, 2019, pp. 193–204.

[23] S. Bonfanti, E. Riccobene, and P. Scandurra, "A runtime safety enforcement approach by monitoring and adaptation," in *European Conference on Software Architecture*. Springer, 2021, pp. 20–36.

[24] E. Kyrkjebø, M. J. Laastad, and Ø. Stavdahl, "Feasibility of the ur5 industrial robot for robotic rehabilitation of the upper limbs after stroke," in *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2018, pp. 1–6.

[25] E. Galan-Uribe, L. Morales-Velazquez, and R. A. Osornio-Rios, "Fpga-based methodology for detecting positional accuracy degradation in industrial robots," *Applied Sciences*, vol. 13, no. 14, p. 8493, 2023.

[26] B. P. Jeppesen, N. Roy, L. Moro, and F. Baronti, "An fpga-based controller for collaborative robotics," in *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*, 2017, pp. 1067–1072.

[27] F. Ahmad, M. Mayr, S. Suresh-Fazeela, and V. Kreuger, "Adaptable recovery behaviors in robotics: A behavior trees and motion generators (btmg) approach for failure management," *arXiv preprint arXiv:2404.06129*, 2024.

[28] J. Guiochet, M. Machin, and H. Waeselynck, "Safety-critical advanced robots: A survey," *Robotics and Autonomous Systems*, vol. 94, pp. 43–52, 2017.

[29] M. Rizwan, C. Reichenbach, and V. Krueger, "Rossmarie: A domain-specific language to express dynamic safety rules and recovery strategies for autonomous robots," in *Second Workshop on Quality and Reliability Assessment of Robotic Software Architectures and Components*, 2023.

[30] M. Mayr, F. Rovida, and V. Krueger, "Skiros2: A skill-based robot control platform for ros," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 6273–6280.