Enhanced Multi-Robot System Architecture with Distributed Planning

Rafael Melo Santos[1], **Carlos Joel Tavares**[2], **Célia Ghedini Ralha**[1,2]

[1]Computer Science Institute – University of Bahia
Campus Ondina, 40.170-110 Salvador, Brazil

[2]Computer Science Department – Exact Sciences Institute – University of Brasília
Campus Darcy Ribeiro, 70.904-970 Brasília, Brazil

melo.r@ufba.br,carlosjoel.tavares@gmail.com, ghedini@unb.br

Abstract. *Achieving shared goals across real scenarios with multiple robots remains a challenge in autonomous multi-robot systems. This challenge is closely related to Multi-Agent Systems (MAS) and automated planning. The literature presents solutions for robotic agents in navigation and interaction, but planning recovery mechanisms is vital to ensure mission continuity and deserves more investigation. The Belief-Desire-Intention (BDI) agents present full potential for autonomy, adaptability, reactivity, and cooperation in MAS. This work presents a multi-robot system architecture with BDI robots enhanced with distributed planning. The validation applies heterogeneous robots to a hospital disinfection mission with a runtime replanning strategy by the Coordinator and local robots, achieving a 100% mission success rate.*

1. Introduction

In real-world domains such as medical assistance, search and rescue, and warehouse logistics, robots must collaborate, share information, adapt to unforeseen changes, and recover from plan failures. These features remain a central challenge in the Multi-Robot System (MRS) field of study [Klavins 2004, Aziz et al. 2021, Verma and Ranga 2021]. In these scenarios, relying exclusively on centralized decision-making may limit responsiveness, robustness, and scalability. Consequently, distributed planning capabilities have become increasingly important, allowing robots to make local decisions, coordinate with teammates, and react to environmental changes while maintaining mission objectives. However, achieving robust robot cooperation requires not only effective communication and task allocation strategies but also the integration of automated planning with plan-recovery mechanisms to support both centralized and distributed forms of coordination.

Traditional planning techniques often assume static environments and centralized control. This limits real-world utility for MRS deployments characterized by uncertainty, partial observability, and heterogeneous agents. Distributed planning approaches offer greater flexibility by enabling agents to reason about local goals while coordinating globally with other team members.

The distribution of Multi-Agent Systems (MAS) focuses on coordination, communication, and decentralized decision-making challenges, allowing Belief-Desire-Intention (BDI) rational agents to achieve robust cooperation [Wooldridge 2009]. BDI strengths are threefold: the underlying philosophy of practical reasoning in humans [Bratman 1987],

a software architecture implementable in real systems, such as the JaCaMo platform [Boissier et al. 2020] (integrating Jason language [Bordini et al. 2007], CArtAgO environment programming, and Moise), and a family of logics that support a formal theory of rational agency [Wooldridge 2020]. Multi-Agent Planning (MAP) integrates MAS with automated planning and adaptive architectures, exploring distributed coordination and dynamic plan recovery strategies (repair or replan) [Stolba et al. 2015, Torreño et al. 2017, Moreira and Ralha 2021, Moreira and Ralha 2022a, Moreira and Ralha 2022b].

This work presents an enhanced MRS architecture that extends the previous work of [da Silva et al. 2026]. The present architecture integrates BDI agents with the Jason language, the Robot Operating System (ROS2) [Macenski et al. 2022], and two Hierarchical Task Network (HTN) [Nau et al. 1999] planning levels. The global level uses a central *Coordinator*. The central contribution is the addition of a local level, allowing robots to autonomously detect and solve failures within their operational scope without requiring intervention from the *Coordinator*. To validate the architecture, we used an illustrative study with heterogeneous robots in an indoor hospital disinfection mission.

Figure 1 presents images of real-world robots that served as reference and motivation for the robot models used in our simulation, including the patrol robot, illustrated by the agile mobile robot called *Spot*® from Boston Dynamics,[1] and the *UVD* disinfection robot from UVD Robots, a Danish company part of Blue Ocean Robotics.[2] These robots were already used in hospital environments. *Spot* was used by MIT researchers at Brigham and Women's Hospital to remotely measure patients' vital signs, reducing the danger to healthcare workers during COVID-19 [Trafton 2020]. *UVD* is a mobile array of powerful short-wavelength ultraviolet-C (UVC) lights, with enough energy to shred the DNA or RNA of microorganisms, disinfecting hospitals and slowing down coronavirus infections [Ackerman 2020].

The illustrative study emphasizes autonomous task allocation, plan recovery, and real-time adaptation with distributed planning, testing the architecture under dynamic and uncertain conditions. By releasing the implementation and simulation on GitHub,[3] we aim to support reproducibility and promote further research. The rest of the article includes the architectural overview in Section 2, experiments with results in Section 3, and a conclusion with future work in Section 4.

2. Multi-Robot System Architecture

This work presents an enhanced MRS architecture with a global centralized replan and local distributed replanning. Our architectural design is rooted in a centralized MRS that coordinates robot missions with automated planning and a BDI framework to guide decision-making and the behavior of autonomous robots, as presented in [da Silva et al. 2026]. The design of an MRS architecture using BDI autonomous robots is challenging, requiring coordination of multiple agents without compromising the system's overall efficiency. The robots are autonomous and heterogeneous, using the Jason BDI language, and implemented using ROS2. The problem domain is defined using the IPyHOP planner syntax to recover execution [Bansod et al. 2022]; however, any

[1] https://bostondynamics.com/products/spot/
[2] https://uvd.blue-ocean-robotics.com/us
[3] https://github.com/CJTS/murosa-health

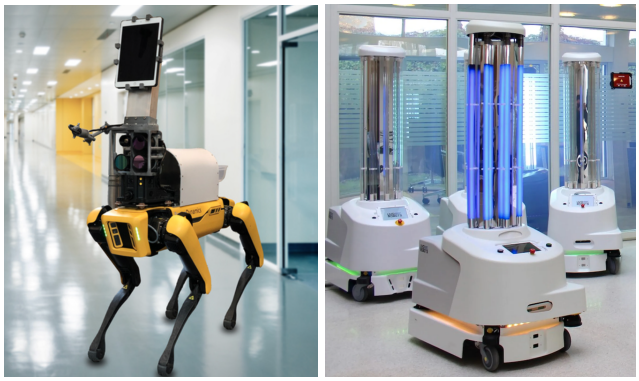


Figure 1. The *Spot* and *UVD* robots illustrate our simulated patrol and disinfection experiment. Credits: *Spot* image courtesy of the MIT researchers, edited by MIT News [Trafton 2020]. *UVD* image IEEE Spectrum's robotics editor [Ackerman 2020].

planner with a partially-order sequence of actions as output can be used instead.

The MRS architecture shown in Figure 2 comprises two levels:

- *Global Mission*: formed by the *Coordinator* agent responsible for managing mission execution and communication among robots and the *Global Planner*.
- *Local Mission*: including robots, each with a *Local Planners*, and the environment ROS node (*Env.*).

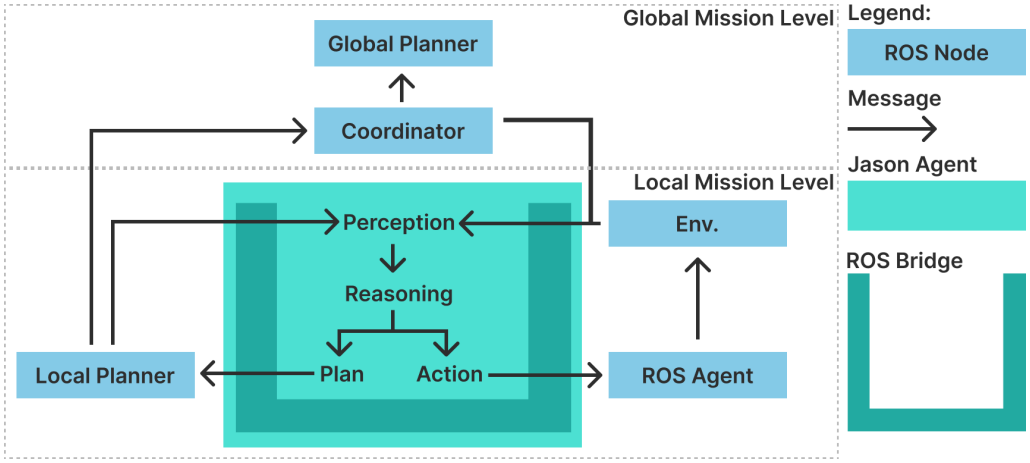


Figure 2. The high-level architecture.

The *Global Mission* level contains the *Coordinator* and a Global Planner, who generate global mission plans based on the overall environment. These plans are then distributed to the robots for execution. Communication between the *Global and Local Mission* is achieved through message exchanges, enabling continuous synchronization.

The robots continuously perform perception, reasoning, planning, and action activities. Environmental information is acquired through the perception process and incorporated into the agent's beliefs. Based on its current beliefs and mission objectives, the reasoning component deliberates on the most appropriate intentions and generates plans. The *Env.* node provides an abstraction of the context and integrates the simulated en-

vironment and the robots. In addition, the agent's *Local Planner* supports detailed plan generation, which is required for navigation and task execution.

The integration between Jason and ROS is achieved through a ROS Bridge [Cardoso et al. 2020], which enables bidirectional communication between symbolic reasoning components and robotic middleware services. This mechanism allows high-level cognitive reasoning to be combined with low-level robotic control, facilitating autonomous decision-making in dynamic and uncertain environments. By separating mission planning from local execution while maintaining continuous feedback of all components, the architecture promotes modularity and adaptability, making it suitable for complex multi-robot missions that require both strategic coordination and autonomous local behavior.

Figure 3 presents the execution process using Business Process Model and Notation (BPMN) 2.0 [OMG 2014].[4] The process begins with a trigger defined by the system integrator and domain. The *Coordinator* initializes the mission and invokes the planner to produce a valid global plan using the problem domain specified at design time. This plan is split into local missions and dispatched to the robots. During execution, each robot monitors its own environment and, upon detecting a failure, first attempts to recover autonomously using its planner (local replanning). If the failure is resolved locally, the robot resumes execution without the *Coordinator* intervention. Otherwise, the failure is escalated to the *Coordinator*, which monitors the overall mission state. If necessary, the *Coordinator* generates a new global plan and redistributes it to the affected robots. The robots execute the updated plan and notify the *Coordinator* upon mission completion or if there are problems in completing the mission.

Plan recovery is central to the architecture and relies on two complementary mechanisms. In *local replanning*, a robot attempts to recover from an execution failure using its planner node instance. If local recovery is not possible, the failure is sent to the *Coordinator*. In *global replanning*, the *Coordinator* generates a new global plan to handle failures that require coordination among multiple agents. The planner used for both global and local planning was IPyHOP. The architecture aligns with MRS design principles outlined in [Rizk et al. 2019] and [Torreño et al. 2017]:

- Task Decomposition & Allocation – domain experts define problem structures for HTN-based planning.
- Perception – sensors on robots and the *Coordinator* capture environment changes.
- Centralized Planning – one planner coordinates multiple executing agents at the mission level.
- Decentralized Recovery – individual agents maintain local planners to autonomously resolve failures without the *Coordinator* intervention.
- Computation Distribution – agents may operate on separate machines, following decentralized execution principles.
- Plan Synthesis – interleaved planning and coordination approach.
- Heuristic Search – uses improved depth-first search with pointer manipulations (efficiency).
- Privacy – robots access only their local plans, and the *Coordinator* retains global oversight.

[4]Using the bpmn.io online editor (available at `https://bpmn.io/`).

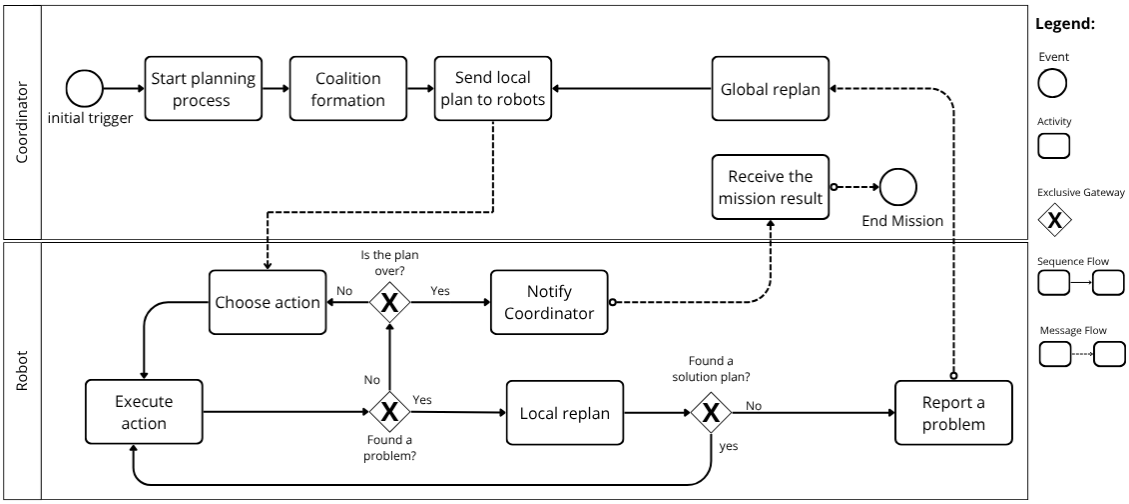


Figure 3. The replan execution process using BPMN 2.0.

This comprehensive architectural solution enables MRS to robustly manage dynamic and unpredictable environments through continuous monitoring, adaptive planning, and autonomous coordination.

3. Experiments

For the experiments, we used a medical domain inspired by the Robotic Mission Adaptation eXemplars (RoboMAX) [Askarpour et al. 2021]. The primary objective of the experiments is to evaluate the effectiveness of the distributed planning mechanism. To verify how agents can autonomously recover from failures before escalating to the *Coordinator*. This study explores an application involving a heterogeneous team of robots collaboratively executing a patrol and disinfection routine in a hospital environment. The aim is to assess the system's ability to handle additional real-world challenges.

3.1. Patrol and disinfection routine

A routine disinfection process is crucial for maintaining safe and hygienic environments, particularly in areas that demand contamination control, such as hospitals. In this context, patrol activities play a key role in preparing the environment before disinfection begins. By inspecting the area, patrols can detect obstacles or conditions that may decrease the effectiveness of the disinfection process, ensuring the environment is suitable before the procedure is executed.

Our patrol and disinfection routine is designed for a hospital environment and includes two types of robots working collaboratively. The first is an agile mobile patrol robot (*Spot*), responsible for inspecting the environment to ensure the room is ready for disinfection, for example, by detecting misplaced objects. The second is a disinfection robot (*UVD*) that can disinfect once the area has been confirmed safe. The *Spot* and *UVD* here denote the simulated robot models used in our experiments, inspired by the real-world robots shown in Figure 1. There is also a *Coordinator* agent capable of observing the environment, generating a plan based on current conditions, and assigning tasks to both the patrol and disinfection robots. When a failure exceeds the robots' capabilities,

the *Coordinator* issues a request to a simulated nurse agent to resolve it before the mission can proceed.

The mission is structured around two main stages:

- *Patrol phase*: ensures the environment is thoroughly prepared.
- *Disinfection phase*: room disinfection is executed.

Figure 4 presents the mission flow along with the agent's actions. As soon as the *Coordinator* triggers the mission start, the Spot robot initiates its patrol phase, which navigates to the target room (*a_navigate_to*) and patrols it (*a_patrol_room*) to verify conditions. If the room door is closed, the Spot robot first attempts to resolve the issue autonomously through local replanning, executing a door detection (*a_detect_macanet*), opening the door (*a_open_door*), and resuming navigation. If the failure exceeds the agent's local capabilities, such as a dirty room, the failure is escalated to the *Coordinator* for replanning. In this case, the *Coordinator* sends the nurse the clean room task (*a_clean_room*) and authorizes the Spot robot to resume patrolling (*a_authorize_patrol*) once the room is ready. Finally, the *UVD* robot initiates the disinfection phase. The robot receives authorization (*a_authorize_desinfect*), navigates to the room (*a_navigate_to*), and executes the final action, disinfection (*a_desinfect_room*).

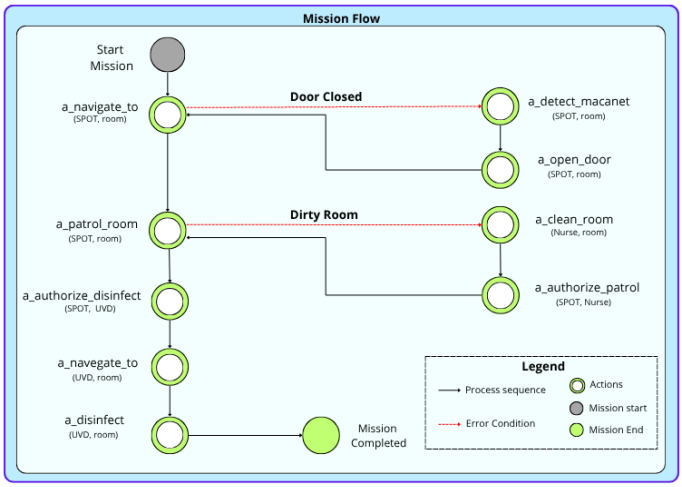


Figure 4. Mission flow.

Figure 5 presents a simplified layout of the hospital environment used in the experiments. It includes four rooms: three are assigned to the nurse for routine activities, while the fourth serves as a waiting area where the *Spot* and *UVD* robots remain on standby. Figure 6 presents the experimental workflow.

Experiment 1

The first experiment is implemented using the Jason language, ROS2, and the IpyHOP planner. A process creates a subprocess for each agent, including the *Coordinator*, the robots, and the nurse. Communication between agents is carried out through ROS2 topics and services. At the end of each execution, logs are automatically collected and processed by a script that organizes the data for analysis with the following setup:

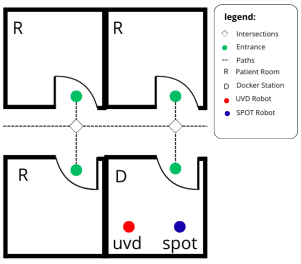


Figure 5. Hospital layout.

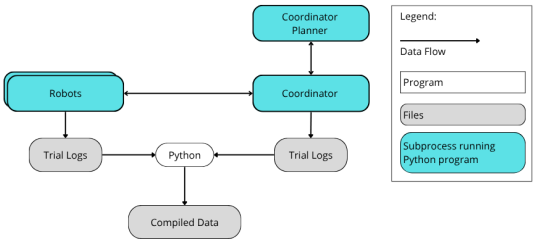


Figure 6. Experimental process.

- Objective – Evaluate how effectively the combination of local robot replanning and centralized coordinator replanning mitigates plan failures in a simulated dynamic hospital environment.
- Problem types – Two runtime problem categories were considered: (i) local problems, handled autonomously by the SPOT robot, specifically closed doors blocking navigation; and (ii) global problems, escalated to the *Coordinator*, specifically dirty rooms that cannot be patrolled or disinfected, triggering the *Coordinator* to replan.
- Illustrative study – Experimental scenarios with occurrence-failure probability and the replanning strategy used during task execution.
- Validation strategy –Each scenario was repeated 30 times to ensure statistical significance. Failures, including both closed-door and dirty-room occurrences, were introduced with probabilities of 0%, 25%, 50%, 75%, and 100%. To evaluate the impact of the replanning mechanisms, the results were compared against a system with only the coordinator-level replanning without local robot replanning. A system without any replanning, where any problem results in mission failure.
- Results evaluation metric – plan completion rate and total plan execution time were measured across all repetitions and scenarios.

We performed experiments using four different strategies. The first strategy did not allow replanning for unexpected events(*No Replan*). The second enabled only local replanning (*Local Replan Only*), allowing agents to recover from failures. The third enabled only coordinator-level replanning (*Coord Replan Only*), and the fourth combined both local and global replanning (*Coord+Local Replan*), allowing agents to attempt local recovery and contact the *Coordinator* only when local recovery was not possible.

Figure 7 shows the proportion of missions with problems during execution across problem occurrence rates of 0%, 25%, 50%, 75% and 100%. The frequency of failed missions increases with the problem occurrence rate across all strategies. This result demonstrates the system's sensitivity to environmental complexity, as higher problem rates naturally lead to more failure events being encountered during execution.

Figure 8 presents the success rates at two different stages of the mission. On the left, the plot shows the mission success rate up to the point where the Spot robot successfully navigates to the target room. This metric isolates the navigation phase, which is subject to closed-door failure. As shown, the strategies that include local replanning (*Coord+Local Replan* and *Local Replan Only*) achieve a 100% success rate across all problem occurrence rates, demonstrating that local replanning alone is sufficient to resolve door-closed failures without requiring the *Coordinator* intervention. In contrast, the

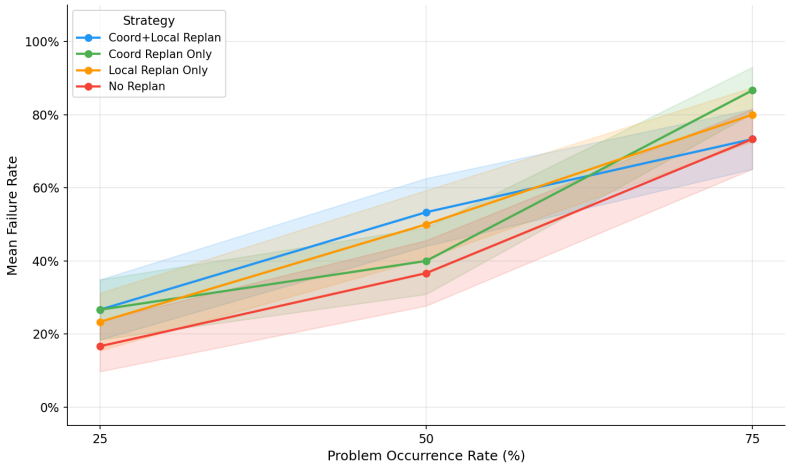


Figure 7. Proportion of mission failure rate by recovery strategy considering 25%, 50%, and 75% of failures during execution.

No Replan strategy degrades progressively as the problem rate increases. The *Coord Replan Only* strategy also maintains a 100% success rate, since the *Coordinator* can replan around closed-door failures when local recovery is unavailable. These results confirm that local replanning effectively absorbs navigation-level failures, preventing unnecessary escalation to the *Coordinator* and preserving overall mission progress up to this stage.

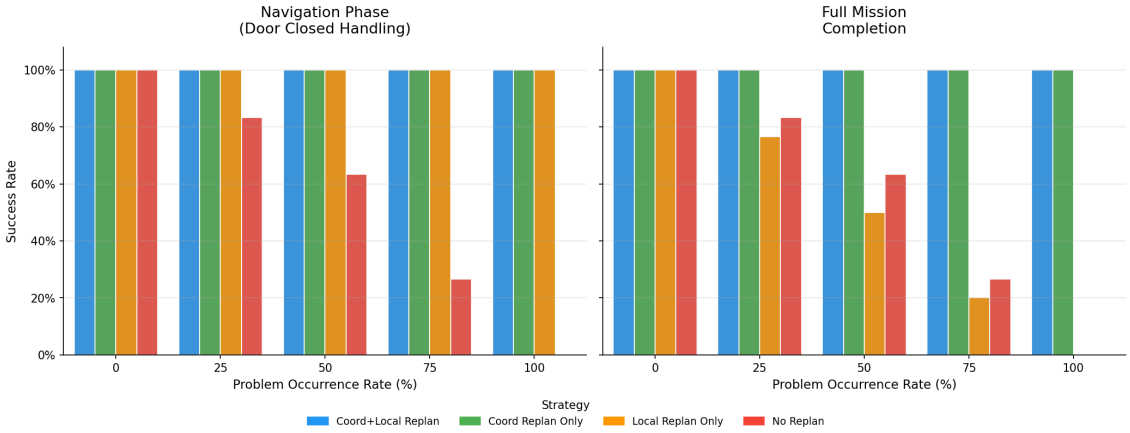


Figure 8. Mission success rate by recovery strategy and problem occurrence rate: navigation phase success and overall mission success

The plot on the right shows the mission success rate at different replanning strategies under the problem occurrence rates (0%, 25%, 50%, 75%, and 100%). When no failures are present (0%), all strategies achieve a 100% success rate. As the problem rate increases, the differences between the strategies become more evident. Both *Coord+Local Replan* and *Coord Replan Only* achieve a 100% success rate for all scenarios. Indicating that coordinator-level replanning is sufficient to ensure mission completion even when failures occur frequently. In contrast, *Local Replan Only* shows a significant decrease in performance as the problem rate increases. The local replanner can recover from closed-door failures on its own, but cannot handle dirty-room failures, which require coordinator-level replanning. Since the *Coordinator* replanning is disabled in this

strategy, failures in dirty rooms cannot be resolved, leading to mission failure. The No Replan strategy presents similar results, since neither failures behind closed doors nor in dirty rooms can be recovered.

Experiment 2

The second experiment focuses on the Unified Planning Framework (UPF) [Micheli et al. 2025] integrated with ROS2, called UPF4ROS2[5], using the patrol and disinfection routine. The UPF provides a unified interface for integrating multiple automated planning techniques and planners within a common software framework. UPF4ROS2 extends the UPF to ROS 2, enabling planning components to interact with robotic systems using ROS 2 communication mechanisms. This integration facilitates the use of symbolic planning alongside robotic perception, control, and execution, supporting the development of flexible and modular robotic applications. By decoupling planning algorithms from robot-specific implementations, UPF4ROS2 provides a practical interface for incorporating automated planning into ROS 2-based robotic architectures. However, UPF4ROS2 does not allow autonomous robots with replanning techniques.

As in the first experiment, the implementation creates a subprocess for each agent, and communication is also made with ROS2. Logs are automatically collected and processed by a script that organizes the data for analysis. Although the UPF can work with HTN planners, the UPF4ROS2 currently does not support this, so changes were made to define the problem and domain linearly. It uses the same setup as the first experiment. The main difference is the objective of comparing this work's solution with a state-of-the-art approach such as UPF4ROS2.

3.2. Comparative Analysis

The results demonstrate that both *Coord+Local Replan* and *Coord Replan Only* achieve 100% mission success across all problem rates. This confirms that coordinator-level replanning remains necessary for failure types that require information or resources outside the local agent's scope. However, although both strategies achieve the same success rate, they differ in execution time. *Coord+Local Replan* completes missions slightly faster than *Coord Replan Only*, especially at higher problem rates. At a 100% problem rate, the average execution time is 5.05 seconds for *Coord+Local Replan* and 5.37 seconds for *Coord Replan Only*. This difference occurs because the local replanner can resolve closed-door failures without requiring communication with the *Coordinator*, reducing the replanning overhead. Although this time reduction appears modest in the current scenario, we expect this gap to widen significantly in more complex missions with longer plans, more agents, and higher failures, where the cost of coordinator communication and global replanning becomes increasingly pronounced.

The *Local Replan Only* strategy demonstrates a limitation of local replanning alone in this architecture. Although it can effectively handle failures that are within an agent's capabilities, some problems cannot be resolved by a single agent and require

[5]`https://github.com/PlanSys2/UPF4ROS2`

coordination among multiple agents. Without coordinator-level replanning, these situations lead to mission failure, reducing the success rate. In summary, combining local and the *Coordinator* replanning is a promising approach for improving both failure recovery and mission completion. Local replanning allows agents to respond quickly to failures that can be resolved independently, reducing communication and coordination overhead. The coordinator-level replanning enables the system to handle more complex situations that require cooperation and coordination among multiple agents. The experimental results suggest that integrating these two mechanisms can improve the overall efficiency of multi-robot systems in dynamic environments.

Figure 9 presents the average mission runtime (in seconds) for different obstacle probabilities and replanning strategies. Strategies that use coordinator-level replanning (*Coord+Local* and *Coord Only*) show longer execution times as the problem rate increases, caused by the additional processing required for the *Coordinator* replanning. In contrast, Local Replan Only and No Replan show shorter execution times at higher problem rates because missions terminate early due to unresolved failures instead of being completed successfully. The reported runtime reflects the computational runtime of the planning and replanning process, not the physical duration of robot actions, which are not temporally modeled in the current simulation. Thus, the runtime comparisons should be interpreted as measures of algorithmic efficiency.

It is important to note that, although the UPF4ROS2 runtime is shorter than our architecture, their solution doesn't include BDI, so it has less autonomy than this work's solution. In [da Silva et al. 2026], the authors show that BDI can reduce the number of calls to planners, which, in more complex scenarios, means less overall time.

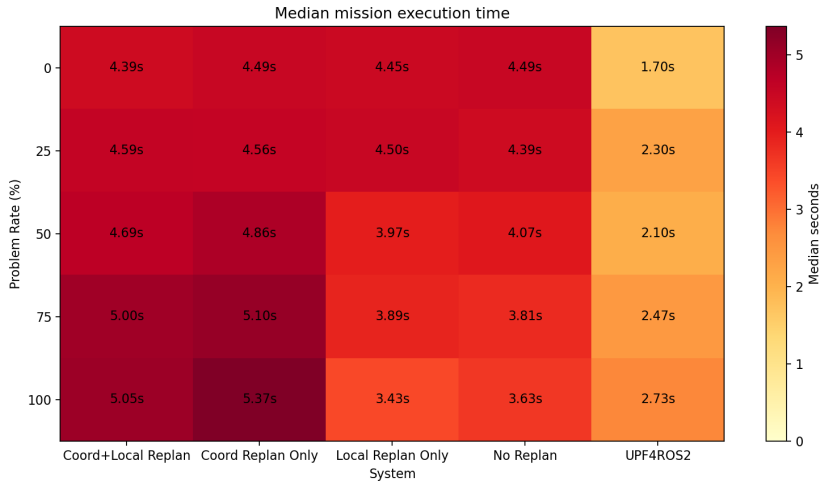


Figure 9. Average mission runtime by problem rate and recovery strategy.

4. Conclusion

The enhanced MRS architecture presented in this work integrates automated planning within a BDI-based autonomous multi-robot approach that explicitly separates local and global recovery. In contrast, the UPF4ROS2 unified planning interface for ROS2 provides centralized failure recovery. The focus of our contribution is the combination of

distributed autonomy and centralized coordination, providing a flexible mechanism for addressing both local failures and mission-level disruptions.

The experiments demonstrate that our architecture can effectively support mission execution in dynamic environments. The evaluation indicates that coordinator-level replanning is necessary to handle failures that require cooperation among multiple agents, while local replanning allows agents to recover from simple failures autonomously.

Desirable future research directions may focus on other domains to validate the architecture's scalability with an increasing number of robots, integrating with different planning algorithms, comparing UPF4ROS2 in terms of planning latency, recovery capability, and architectural decentralization, and augmenting metrics, such as resource efficiency, to enable more in-depth exploration of the contribution.

Acknowledgements

Some text editing used GPT-5.6 Luna, but all scientific claims, experimental results, tables, and analyses were generated and validated under the authors' responsibility.

References

Ackerman, E. (2020). Autonomous robots are helping kill coronavirus in hospitals – robots that can efficiently disinfect hospitals using uv light could slow coronavirus infections. IEEE Spectrum's robotics editor. March 11, 2020.

Askarpour, M., Tsigkanos, C., Menghi, C., Calinescu, R., Pelliccione, P., García, S., Caldas, R., von Oertzen, T. J., Wimmer, M., Berardinelli, L., Rossi, M., Bersani, M. M., and Rodrigues, G. S. (2021). RoboMAX: Robotic mission adaptation exemplars. In *Proc. of Int. Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, pages 245–251.

Aziz, H., Chan, H., Cseh, A., Li, B., Ramezani, F., and Wang, C. (2021). Multi-robot task allocation-complexity and approximation. In *Proc. of 20th Int. Conf. on Autonomous Agents and MultiAgent Systems (AAMAS)*, page 133–141.

Bansod, Y., Patra, S., Nau, D., and Roberts, M. (2022). Htn replanning from the middle. In *The International FLAIRS Conference Proceedings*, volume 35.

Boissier, O., Bordini, R. H., Hübner, J., and Ricci, A. (2020). *Multi-Agent Oriented Programming - Programming Multi-Agent Systems Using JaCaMo*. MIT Press. Intelligent Robotics and Autonomous Agents Series. ISBN: 9780262044578.

Bordini, R. H., Hübner, J. F., and Wooldridge, M. (2007). *Programming Multi-Agent Systems in AgentSpeak using Jason*. John Wiley & Sons. ISBN: 9780470029008.

Bratman, M. (1987). *Intention, Plans, and Practical Reason*. Cambridge, MA: Harvard University Press, Cambridge.

Cardoso, R. C., Ferrando, A., Dennis, L. A., and Fisher, M. (2020). An interface for programming verifiable autonomous agents in ROS. In *International Conference on Agreement Technologies (ICAT)*, pages 191–205.

da Silva, C. J. T., Santos, R. M., Cardoso, R. C., and Ralha, C. G. (2026). A Multi-Robot Architecture for Continuous Planning and Execution using BDI Agents. In

Proc. of 25nd Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS), page 422–430, Richland, SC. IFAAMAS.

Klavins, E. (2004). *Communication Complexity of Multi-robot Systems*, pages 275–291. Springer, Berlin, Heidelberg.

Macenski, S., Foote, T., Gerkey, B., Lalancette, C., and Woodall, W. (2022). Robot Operating System 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66).

Micheli, A., Bit-Monnot, A., Röger, G., Scala, E., Valentini, A., Framba, L., Rovetta, A., Trapasso, A., Bonassi, L., Gerevini, A. E., Iocchi, L., Ingrand, F., Köckemann, U., Patrizi, F., Saetti, A., Serina, I., and Stock, S. (2025). Unified Planning: Modeling, manipulating and solving AI planning problems in Python. *SoftwareX*, 29:102012.

Moreira, L. H. and Ralha, C. G. (2021). Evaluation of decision-making strategies for robots in intralogistics problems using multi-agent planning. In *Proc. of IEEE Congress on Evolutionary Computation (CEC)*, pages 1272–1279.

Moreira, L. H. and Ralha, C. G. (2022a). An efficient lightweight coordination model to multi-agent planning. *Knowledge and Information Systems*, 64:415–439.

Moreira, L. H. and Ralha, C. G. (2022b). Method for evaluating plan recovery strategies in dynamic multi-agent environments. *Journal of Experimental & Theoretical Artificial Intelligence*, pages 1–25.

Nau, D., Cao, Y., Lotem, A., and Munoz-Avila, H. (1999). Shop: simple hierarchical ordered planner. In *Proc. of 16th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, page 968–973, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

OMG (2014). Business process model and notation specification version 2.0.2 (BPMN™).

Rizk, Y., Awad, M., and Tunstel, E. W. (2019). Cooperative heterogeneous multi-robot systems: A survey. *ACM Comput. Surv.*, 52(2).

Stolba, M., Komenda, A., and Kovacs, D. L. (2015). Competition of Distributed and Multiagent Planners (CoDMAP). Workshop on Distributed and Multi-Agent Planning (DMAP), 25th Int. Conf. on Automated Planning and Scheduling (ICAPS). Available at `http://agents.fel.cvut.cz/codmap/`. Accessed on: 2020-02-05.

Torreño, A., Onaindia, E., Komenda, A., and Štolba, M. (2017). Cooperative multi-agent planning: A survey. *ACM Comput. Surv.*, 50(6).

Trafton, A. (2020). Robot takes contact-free measurements of patients' vital signs. MIT News Office. August 31, 2020.

Verma, J. K. and Ranga, V. (2021). Multi-robot coordination analysis, taxonomy, challenges and future scope. *Journal of Intelligent & Robotic Systems*, 102(1).

Wooldridge, M. (2009). *An introduction to multiagent systems*. John Wiley & Sons.

Wooldridge, M. (2020). *Reasoning about Rational Agents*. MIT Press. Intelligent Robotics and Autonomous Agents Series. ISBN: 9780262257237.