Multi-Robot System Architecture Validation Using Disinfecting Robot Routine

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

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

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

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

Abstract. *Coordinating multiple robots to achieve shared goals across diverse scenarios remains a key challenge in Multi-Robot Systems (MRS). The literature proposes solutions where robots must collaborate, share information, and have planning recovery mechanisms to ensure mission continuity. The challenges involved are closely related to Multi-Agent Systems (MAS) integrated with Automated Planning (AP), often evaluated in limited and controlled scenarios. But to support claims of applicability, broader testing across diverse scenarios is essential. This work evaluates an MRS solution that integrates AP into an MAS, using a disinfecting robot routine as an illustrative study.*

1. Introduction

Coordinating multiple robots to achieve shared goals in dynamic and uncertain environments remains a central challenge in Multi-Robot Systems (MRS) [Aziz et al. 2021, Verma and Ranga 2021]. In domains such as search and rescue, warehouse logistics, or medical assistance, robots must collaborate, share information, adapt to unforeseen changes, and recover from partial plan failures [Klavins 2004]. Achieving robust cooperation demands not only effective communication and task allocation strategies but also integration with Automated Planning (AP) and plan recovery mechanisms to ensure mission continuity and efficiency.

These challenges closely parallel those found in Multi-Agent Systems (MAS), where coordination and communication are also key concerns to achieve robust cooperation [Wooldridge 2009], [Weiss 2016], [Salzman and Stern 2020]. Traditional planning techniques, however, often assume static environments and single-agent control, which limits their applicability in real-world MRS deployments. To address this, recent work has integrated Multi-Agent Planning (MAP) with flexible architectures capable of dynamic adaptation and plan repair [Komenda et al. 2016, Moreira and Ralha 2021, Moreira and Ralha 2022, da Silva and Ralha 2024].

The literature presents research involving architectural solutions to support MRS and AP integration, including centralized and decentralized coordination of robots using Robot Operating System (ROS) [Cashmore et al. 2015], multi-robot control for the RoboCup logistics league [González et al. 2020], PlanSys2: planning system framework for ROS2 [Martín et al. 2021], hierarchical architecture framework based on goal decomposition [Lesire et al. 2022]. One such architecture was recently introduced integrating MRS with MAP [da Silva 2024], combining agents with Hierarchical Task Network (HTN) planning for resilient mission execution with heterogeneous multi-robot [Erol et al. 1994, Erol et al. 1996]. While promising, that work evaluated the system in a relatively constrained context, leaving open the question of whether the architecture generalizes effectively across diverse operational domains.

The literature presents a gap in runtime planning recovery for autonomous MRS, with a need for broader testing across diverse scenarios. This work presents an illustrative study regarding this gap by validating the architecture through a different scenario using heterogeneous robots in an indoor disinfection mission. Figure 1 presents an example image of real robots used in our simulated study, including the patrol robot, illustrated by the agile mobile robot called *Spot*® from Boston Dynamics,[1] and the *UVD* disinfection robot from the UVD Robots, Part of Blue Ocean Robotics[2]. The illustrative study emphasizes autonomous task allocation, plan recovery, and real-time adaptation, thereby testing the architecture under dynamic and uncertain conditions not explored in the original evaluation. By releasing the implementation and simulation code openly[3], we aim to support reproducibility and promote further research in context-independent MRS planning.

Figure 1. The *Spot* and *UVD* robots used in the illustrative study.

The rest of the article includes, in Section 2, the architectural aspects of MAS with AP; in Section 3, the experiments together with the used illustrative study; and lastly, in Section 4 the conclusions and future work.

2. Architectural Overview

This work uses an MRS architecture that integrates AP into MAS. Adaptability is a crucial property in dynamic environments. However, MAS architecture design is challenging, as it includes individual agents without overhauling the entire system. Thus, our architectural design is rooted in an ensemble-based software architecture that coordinates missions of heterogeneous robots to autonomously form coalitions as presented in [Rodrigues et al. 2022].

The design enables hierarchical task decomposition, goal reasoning, and plan generation. It follows an interleaved planning and execution scheme to enhance adaptability

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

in dynamic environments. The architecture is divided by design time with the problem domain and runtime environment to provide the processes required for coordinating the execution of robots at runtime.

The architecture emphasizes the decomposition of complex tasks into simpler subtasks, the formation of coalitions of heterogeneous robots, and the use of MAP to convert tasks into executable actions [Rizk et al. 2019]. A human expert assists with initial task decomposition. The subsequent steps, task allocation, planning, and control, are autonomously managed by the robots.

The architecture adapted from [da Silva 2024] and shown in Figure 2 includes the following components:

- *System Integrator* – responsible for defining the problem domain in IPyHOP syntax [Bansod et al. 2022].
- *Coordinator* – handles mission requests, assigns roles to suitable robots, and manages execution and recovery processes.
- *Robots* – execute local plans through sensor-actuator loops and synchronize actions using a dedicated module.

The coordinator and robots are implemented using ROS2, an open-source framework designed to facilitate the development of robotic systems.[4] In this architecture, each robot is composed of modular ROS2 nodes, each one responsible for a specific function such as communication, actuation, or sensing. These nodes interact through message-passing interfaces, forming a distributed computational graph that represents the system's overall behavior.

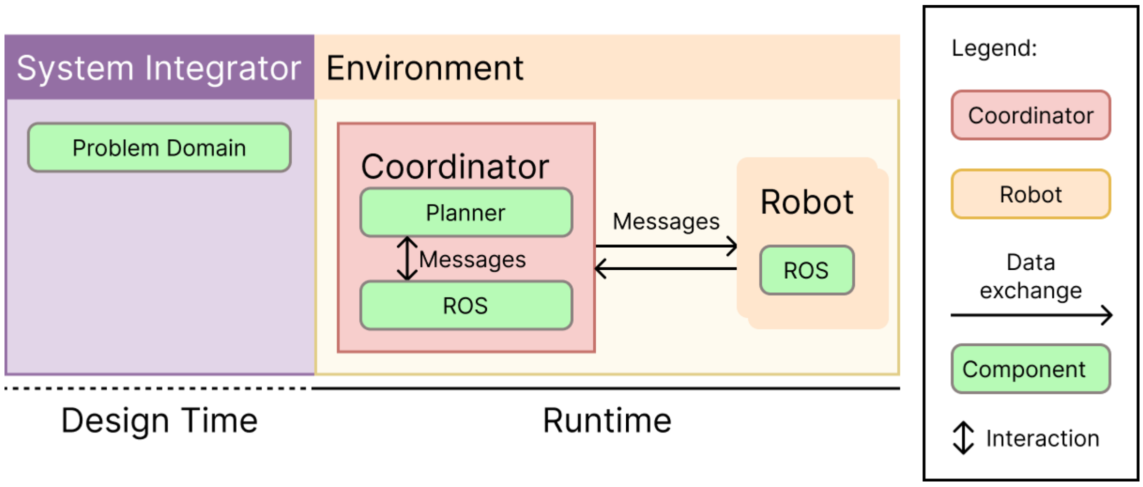

Figure 2. The high-level architecture.

The execution process, shown in Figure 3 using a Business Process Model and Notation (BPMN) 2.0, [OMG 2014][5] 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, the *Coordinator* monitors the environment for anomalies and, if necessary, creates a new plan to

[4]ROS2 documentation available at `https://docs.ros.org/en/rolling/index.html`
[5]Using bpmn.io online editor (`https://bpmn.io/`).

distribute to robots. The robots execute the plan and notify the *Coordinator* if there are problems in completing the mission.

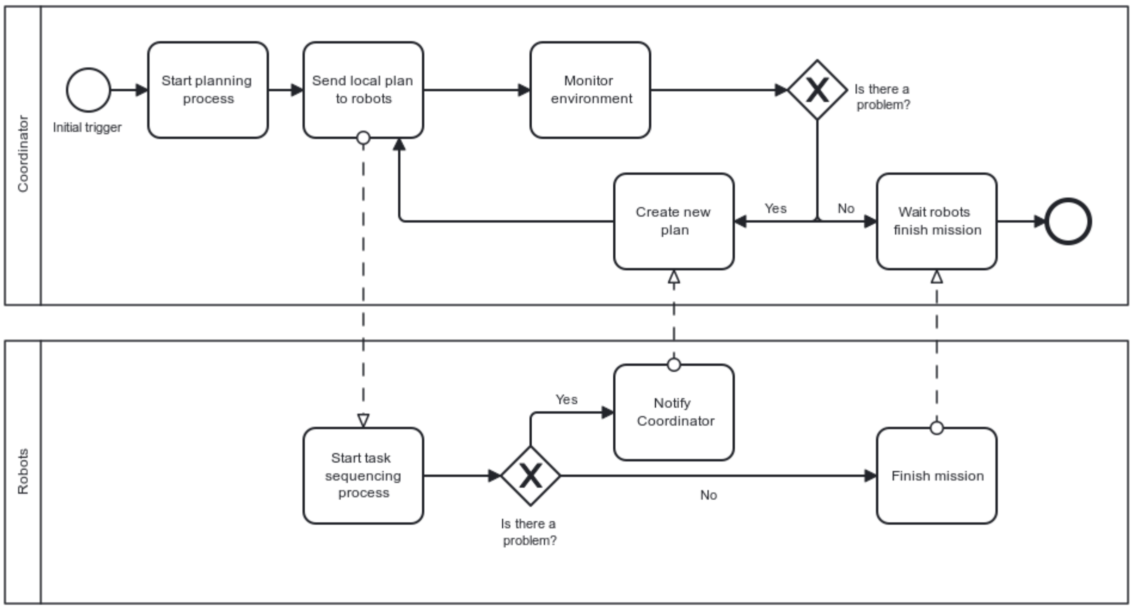

Figure 3. The solution's execution process using BPMN.

Plan recovery is central to the architecture and handles disruptions through *reactive planning*, triggered when a robot fails to execute an action due to issues like depleted battery or incorrect preconditions, and *proactive replanning*, involving continuous monitoring by the *Coordinator* to detect potential threats before they affect execution.

This architecture aligns with MRS design principles outlined in [Rizk et al. 2019] and [Torreño et al. 2017], featuring:

- Task Decomposition & Allocation – domain experts define problem structures for HTN-based planning.
- Perception – sensors on robots and the *Coordinator* capture changes in the environment.
- Centralized Planning – one planner coordinates multiple executing agents.
- 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 for efficiency.
- Privacy – robots access only their local plans; the *Coordinator* retains global oversight.

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 architecture, considering the plan recovery process [da Silva 2024]. Although previous studies have applied this framework to MRS in the domain of healthcare delivery, pick-up sample scenarios [da Silva and Ralha 2023], its versatility allows implementation in a variety of fields. To better assess its adaptability, this study explores an application involving heterogeneous robots performing a patrol and disinfection routine. The aim is to assess the system's ability to lead to 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 strict contamination control, such as hospitals, food industries, and laboratories. 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 hinder the effectiveness of the disinfection process. This preliminary step helps ensure that the environment is suitable for disinfection, increasing the success of the procedure.

Our patrol and disinfection routine is designed for a hospital environment and includes three 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 third is the *Coordinator*, a robot capable of observing the environment, generating a plan based on current conditions, and assigning tasks to both the patrol and disinfection robots.

The mission is structured around two main stages as presented in the mission flow of Figure 4:

- Patrol phase —- ensures the environment is properly prepared, which requires authorization from a nurse or technician.
- Disinfection phase – the room disinfection is executed.

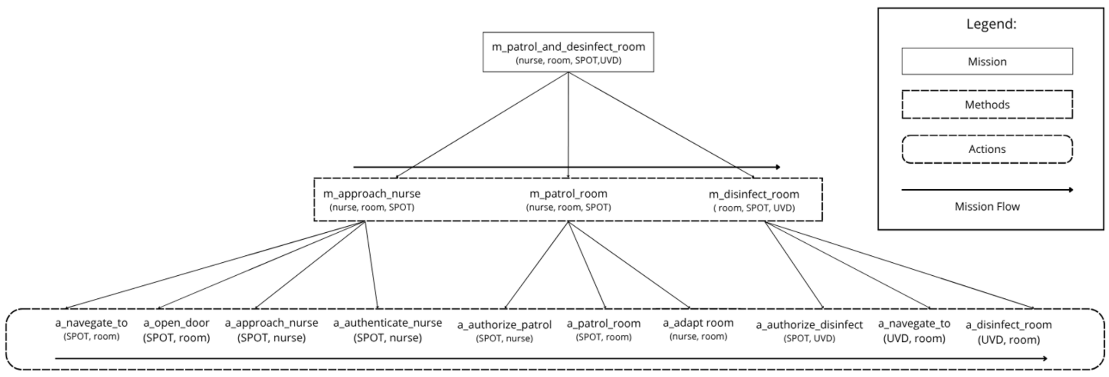

Figure 4. Mission flow.

Before the main stages, an authentication step is required for the nurse to authorize the robots to perform the mission. In the *m_approach_nurse* step, the *Spot* robot navigates to the nurse (*a_navigate_to*), opens the door (*a_open_door*), approaches the nurse (*a_approach_nurse*), and then performs the authentication procedure (*a_authenticate_nurse*).

Afterward, the *m_patrol_room* stage begins, in which the nurse authorizes the patrolling (*a_authorize_patrol*), the robot patrols the room (*a_patrol_room*), and the nurse adapts the room as needed (*a_adapt_room*). Finally, the *UVD* robot initiates the disinfection phase *m_desinfect_room*. The robot receives authorization (*a_authorize_desinfect*), navigates to the room (*a_navigate_to*), and executes the final action, disinfection (*a_desinfect_room*).

If the *Spot* robot detects a problem in the room, the *Coordinator* replans to maintain the effectiveness of the routine, alerting the human nurse or technician to adapt the room and resolve the problem.

The following PEAS (Performance, Environment, Actuators, Sensors) describes, respectively, the *Spot* and *UVD* robots in Tables 1 and 2. For the *Spot* robot, the performance measure is its ability to determine whether a room is currently unsuitable for disinfection and to report this information to the Coordinator. Its environment is a hospital ward that is partially observable, dynamic, sequential, and discrete and involves multiple agents. The actuators available to Spot include navigation capabilities for moving through corridors and rooms, an arm for interacting with doors, and communication tools to interface with the Coordinator and nurses. Its sensors include location and route perception, movement detection, and 360-degree monitoring sensors.

Table 1. PEAS description for the *Spot* robot.

PEAS	**Descriptions**
Performance	Identify whether the room is currently unsuitable for disinfection and report it to the *Coordinator*.
Environment	Hospital ward that is: partially observable, dynamic, sequential, discrete, and multiagent.
Actuators	Navigate through hospital corridors and rooms, interact with doors with a *Spot* Arm, and communicate with *Coordinator* and nurses.
Sensors	Position perception, the path to arrive and route, movement sensor, 360^o monitor sensors.

Table 2. PEAS description for the *UVD* robot.

PEAS	**Descriptions**
Performance	Disinfect assigned rooms using UV-C light or disinfectant spray with effectiveness.
Environment	Hospital ward that is: partially observable, dynamic, sequential, discrete, and multiagent.
Actuators	Move to target location, activate disinfection system (UV-C or spray), communicate with *Spot*.
Sensors	position perception, the path to arrive and route, movement sensors, and UV safety sensors.

For the *UVD* robot, the performance is defined by how effectively it can disinfect rooms using UV-C light or a disinfectant spray. The environment it operates in is the same as Spot's, a partially observable, dynamic, sequential, discrete, and multiagent hospital ward. Its actuators enable it to move to target rooms, activate its disinfection system, and communicate with the Spot robot. The sensors include those for position tracking, navigation, movement detection, and UV safety monitoring.

Listing 1 shows an example of a plan where no replan is needed. In this case, the *Spot* robot patrols the room and does not find anything unsafe. Consequently, the *UVD* robot proceeds with the disinfection process without interruptions. In Listing 2, replan is required after *Spot* robot detects an obstacle or harmful item in the room. Then, the nurse or healthcare professional must remove the obstacle before the *UVD* robot can safely continue and complete the disinfection task.

Listing 1. Plan execution.

```
a_navigate_to (SPOT, room)
a_open_door (SPOT, room)
a_approach_nurse (SPOT, nurse)
a_authenticate_nurse (SPOT, nurse)
a_authorize_patrol (SPOT, nurse)
a_patrol_room (SPOT, room)
a_authorize_disinfect (UVD, SPOT)
a_navigate_to(UVD, room)
a_disinfect_room (UVD, room)
–
–
–
–
–
```

Listing 2. Plan with replanning.

```
a_navigate_to (SPOT, room)
a_open_door (SPOT, room)
a_approach_nurse (SPOT, nurse)
a_authenticate_nurse (SPOT, nurse)
a_authorize_patrol (SPOT, nurse)
a_patrol_room (SPOT, room)
a_adapt_room (nurse, room)
a_approach_nurse (SPOT, nurse)
a_authenticate_nurse (SPOT, nurse)
a_authorize_patrol (SPOT, nurse)
a_patrol_room (SPOT, room)
a_authorize_disinfect (UVD, SPOT)
a_navigate_to(UVD, room)
a_disinfect_room (UVD, room)
```

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.

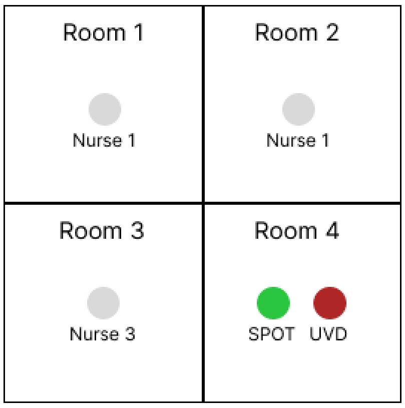

Figure 5. Hospital Layout

3.2. Experimental Setup

The experiments use ROS2 to implement the robot's behavior. A process creates a thread for each agent (*Coordinator* with planner and robots). The communication between agents is implemented using topics and services provided by the ROS2 framework. Figure 6 shows the execution process of the experiment.

The experiment execution took into account the following setup aspects:

- Objective – to evaluate how effectively the architecture mitigates issues when operating in a simulated dynamic environment.
- Problem types—The room is not suitable for disinfection due to misplaced objects or the presence of individuals who should not be in the room at that time.
- Illustrative study – four experimental scenarios related to the evaluated problem.
- Validation strategy – we conducted four scenarios, each repeated 30 times for statistical significance, to assess the *Coordinators*' ability to complete the plan. At the end of each execution, the plan results were analyzed. The analysis focused on the percentage of uncompleted missions related to the evaluated problem. The scenarios included an uncleaned room event that occurs with probabilities of 10%, 25%, 50%, and 75%.
- Results evaluation metric – plan completion and the time required to complete the plan were evaluated. We compared the experimental results with a scenario where the *Coordinator* was not allowed to replan.

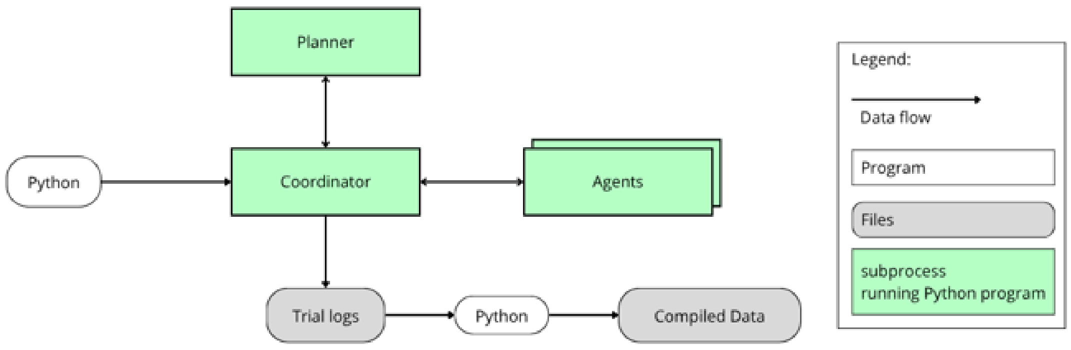

Figure 6. Experimental process.

3.3. Results

In this section, we analyze the experimental results to assess the effectiveness of the proposed architecture in dynamic environments. We focus on the plan recovery process illustrated with our patrol and disinfection routine.

3.3.1. Experiment Analysis

We performed experiments in two different scenarios: the first, when architecture cannot replan in response to unexpected events, and the second, replanning with unexpected events. Figure 7 shows the proportion of missions with problems during mission execution across different obstacle occurrence rates, shifting from 10%, 25%, 50%, and 75%. Note that the frequency of missions with problems increases proportionally with the obstacle occurrence rate. This result demonstrates the system's sensitivity to environmental complexity.

Figure 8 shows the rate of completed missions at different obstacle probabilities (10%, 25%, 50%, 75%). The results demonstrate that the presence of a replanning mechanism improves mission success, even as the problem rate increases. Although the success rates drop considerably without replanning, the system with replanning consistently maintains a 100% completion rate across all tested conditions.

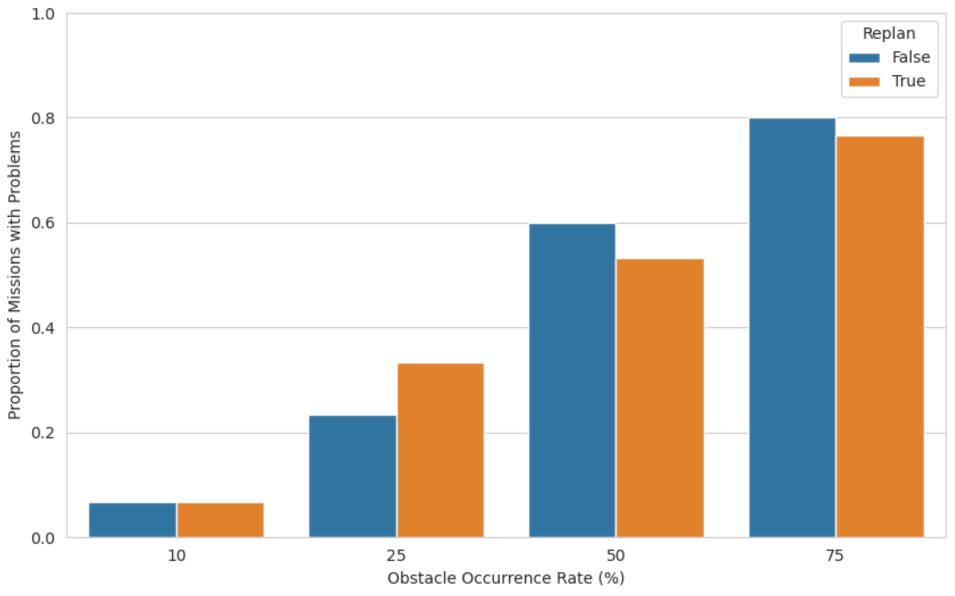

Figure 7. Mission execution with problems with different obstacle probability rates (10%, 25%, 50%, 75%), comparing with and without replanning.

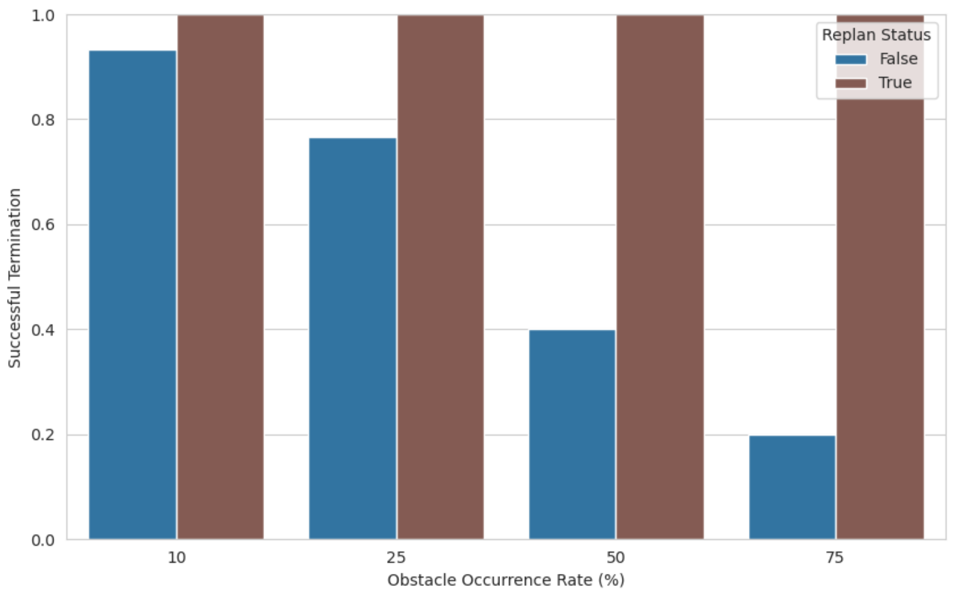

Figure 8. Rate of completed missions by increasing obstacle occurrence rate (10%, 25%, 50%, 75%) with replanning status (false and true).

Figure 9 presents the average mission runtime (in seconds) for the different obstacle probabilities with and without replanning. Although the runtimes slightly decrease without replanning, they increase progressively when replanning is enabled. This result is expected since the replanning mechanism introduces additional processing and coordination steps in response to detected problems.

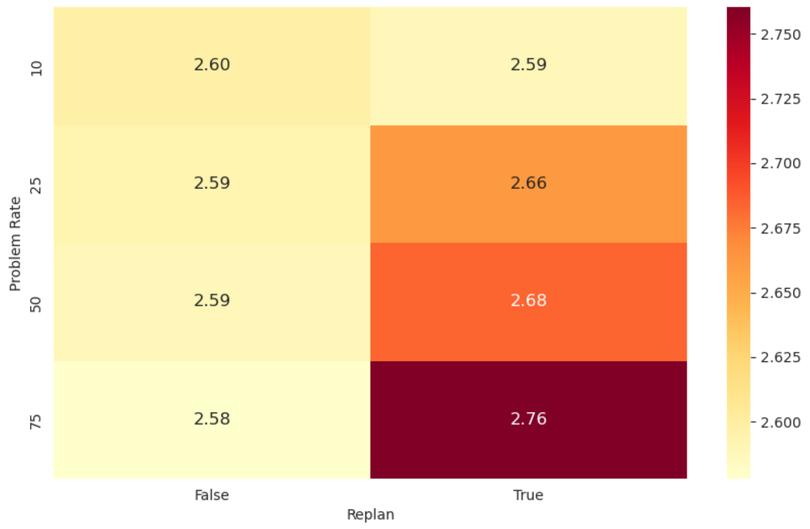

Figure 9. Average mission runtime by increasing obstacle probabilities (10%, 25%, 50%, 75%) with and without replanning.

3.3.2. Comparative Analysis

Despite the increase in execution time from 2.59 to 2.76 from 10% to 75% of obstacle occurrence, the advantages of the replanning capability become evident when analyzing Figures 8 and 9. The architecture replan ability enables the system to maintain high mission success rates, even under increasing obstacle probabilities, demonstrating that the added cost in execution time is justified by the system's adaptability in dynamic environments.

Although our experiments were carried out in a simulated environment, the proposed architecture dispenses an integrable capability into real-world systems thanks to its flexibility. In the light of the SPOT robot, it runs on an Ubuntu-based system and provides well-supported Python and C++ SDKs. Additionally, an official ROS2 wrapper enables integration with ROS-based systems, which aligns directly with our implementation, indicating possible deployment on physical robots (develop with SPOT).[6] The UVD robots are designed to be mobile and self-navigating in indoor spaces. They can be controlled remotely, enabling human operators or autonomous systems to initiate disinfection sequences and monitor progress. Thus, our solution can be handed over in real-world scenarios, including not only hospital settings but also industrial facilities, airports, shopping centers, educational institutions, and other high-traffic public spaces.

[6]`https://support.bostondynamics.com/s/article/About-ROS-with-Spot-67882`

In summary, the results indicate that the proposed architecture is efficient and flexible in dynamic environments. Its ability to adapt to unexpected changes ensures high mission success rates, making it a strong candidate for deployment in real-world multi-robot scenarios.

4. Conclusion

The experiments confirm the effectiveness and adaptability of the MAS architecture proposed earlier, which incorporates replanning to support multi-agent coordination in dynamic environments [da Silva 2024]. The patrol and disinfection routine example highlights the architecture's flexibility when a central *Coordinator* is responsible for planning, providing a reliable approach for managing heterogeneous robots within the evaluated scenario.

Future work may focus on enhancing the architecture's capabilities with false action results using sophisticated heuristic algorithms. In addition, performing experiments in other domains to validate the architecture's versatility, involving an increasing number of robots, integrating with more advanced planning algorithms to enhance communication protocols, finding a suitable benchmark for MAP, comparing the architecture with existing literature work, augmenting metrics like resource efficiency and scalability, to fully assess the potential of the contribution, will be desirable future research directions.

References

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.

Cashmore, M., Fox, M., Long, D., Magazzeni, D., Ridder, B., Carreraa, A., Palomeras, N., Hurtós, N., and Carrerasa, M. (2015). ROSPlan: Planning in the robot operating system. In *Proc. of 35th Int. Conf. on Automated Planning and Scheduling (ICAPS)*, page 333–341.

da Silva, C. J. T. (2024). *A Multi-robot System Architecture with Multi-agent Planning*. Master's thesis, Computer Science Department, University of Brasília, Brazil.

da Silva, C. J. T. and Ralha, C. G. (2023). Multi-robot system architecture focusing on plan recovery for dynamic environments. In *2023 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1668–1673.

da Silva, C. J. T. and Ralha, C. G. (2024). Multi-agent system architectural aspects for continuous replanning. In *Anais do XVIII Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações*, pages 39–50, Porto Alegre, RS, Brasil. SBC.

Erol, K., Hendler, J., and Nau, D. (1996). Complexity results for HTN planning. *Annals of Mathematics and Artificial Intelligence*, 18:69–93.

Erol, K., Hendler, J., and Nau, D. S. (1994). HTN planning: complexity and expressivity. In *Proceedings of the Twelfth AAAI National Conference on Artificial Intelligence*, AAAI'94, page 1123–1128. AAAI Press.

González, J. C., García-Olaya, A., and Fernández, F. (2020). Multi-layered multi-robot control architecture for the robocup logistics league. In *Proc. of IEEE Int. Conf. on Autonomous Robot Systems and Competitions*, pages 120–125.

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

Komenda, A., Stolba, M., and Kovacs, D. L. (2016). The international competition of distributed and multiagent planners (CoDMAP). *AI Magazine*, 37(3):109–115.

Lesire, C., Bailon-Ruiz, R., Barbier, M., and Grand, C. (2022). A hierarchical deliberative architecture framework based on goal decomposition. In *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 9865–9870.

Martín, F., Clavero, J. G., Matellán, V., and Rodríguez, F. J. (2021). PlanSys2: A planning system framework for ROS2. In *Proc. of IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, page 9742–9749.

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*, pages 1272–1279.

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

OMG (2014). Business process model and notation specification version 2.0.2 (BPMN™). `https://www.omg.org/spec/BPMN`. Accessed: 2025-08-03.

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

Rodrigues, G., Caldas, R., Araujo, G., de Moraes, V., Rodrigues, G., and Pelliccione, P. (2022). An architecture for mission coordination of heterogeneous robots. *Journal of Systems and Software*, 191(111363).

Salzman, O. and Stern, R. (2020). Research challenges and opportunities in multi-agent path finding and multi-agent pickup and delivery problems. In *Proc. of 19th Int. Conf. on Autonomous Agents and MultiAgent Systems (AAMAS)*, page 1711–1715.

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

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

Weiss, G. (2016). *Multiagent Systems*. The MIT Press, 2nd edition.

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