SpaceTry: An LLM-Assisted Simulation Testbed for Space Mission Autonomy Engineering

Ricardo Caldas, Keila Lima, Roberto Natella, and Patrizio Pelliccione
Gran Sasso Science Institute (GSSI), L'Aquila, Italy
{firstname.lastname}@gssi.it

***Abstract*—Autonomy is essential for planetary exploration because space robots must operate under uncertainty, limited supervision, and strong resource constraints while continuing to achieve mission goals. Because lab and field campaigns are costly and hard to iterate, in-house (in-vitro) testing can help screen scenarios before embodied validation. Existing robotics testbeds do not jointly provide this workflow for the space domain. We present SpaceTry, an open-source simulation-based testbed for scenario-based evaluation of autonomous space exploration missions. SpaceTry combines LLM-assisted scenario authoring from natural-language mission descriptions, configurable scenario artifacts for mission logic and monitoring, and a Space ROS/Gazebo runtime substrate for executing planetary-autonomy experiments. The framework does not prescribe any specific adaptation mechanism; instead, it exposes monitored violations as runtime events and leaves the response to the autonomy solution under evaluation. We describe the architecture, introduce a reference Mars outpost scenario extending the Space ROS NASA's Curiosity Rover Demo, and position SpaceTry as a testbed for comparing self-adaptive autonomy approaches before later lab or field testing.**

***Index Terms*—space autonomy, self-adaptive systems, space robotics, behavior trees, runtime monitoring, testbed, Space ROS**

I. Introduction

Autonomy is a prerequisite for planetary exploration. Space robots must achieve mission goals under uncertainty, limited supervision, and strong resource constraints [1]–[4]. NASA's Curiosity rover offers a compelling example: unexpectedly severe wheel damage led to adaptations in rover driving behavior [5], illustrating a broader point in resilient planetary autonomy [6]: robots must reason and react when mission assumptions break down.

If autonomy is the scientific problem, its evaluation must be methodologically rigorous. Scenario-based testing is a natural foundation because it treats the scenario as the evaluation unit: a structured combination of goals, uncertainties, constraints, and observable outcomes [7], [8]. For space robotics, field and lab campaigns are costly, low-throughput, and hard to reproduce, so in-house (in-vitro) testing [9] helps screen missions before embodied validation. This requires testbeds that combine representative simulation, explicit safety and mission constraints, and runtime evidence that autonomy preserves them. Existing robotics testbeds cover only parts of this space. RoboMAX [10] catalogs mission-adaptation descriptions but has no executable simulator. SUAVE [11] and Aloft [12] provide ROS-based environments for underwater vehicles and drones, but do not target planetary robotics or integrate constraint monitoring as a first-class evaluation mechanism. More broadly, robotics software engineering has long argued for systematic mission specification and verification [13], yet a unified testbed for scenario-based evaluation of space autonomy remains absent.

This gap is growing as LLMs change how scenarios and robotic behaviors are authored. For scenario-based testing, they can translate natural-language mission intent into diverse, executable, and semantically controlled scenarios, which is useful when real space-mission data are scarce and costly [14]. Recent work shows LLM support for scenario generation from grounded requirements [15] and interactive simulation-scenario editing [16]. Yet these advances do not provide a methodology for evaluating space autonomy. We need a testbed where LLMs assist scenario engineering, while autonomy is assessed through executable scenarios, runtime monitors, and explicit evaluation artifacts.

We present **SpaceTry**, an open-source simulation testbed for scenario-based evaluation of autonomous space exploration missions. Built on Space ROS [17] and Gazebo, SpaceTry lets teams describe a mission in natural language and derive artifacts to configure, execute, and assess a planetary-autonomy scenario, including behavior-tree mission logic [18], structured safety specifications [19], uncertainty injection [20], and runtime monitoring [21], [22]. Crucially, SpaceTry does not prescribe the adaptation mechanism. When a monitored property is violated, the testbed exposes that event and leaves the response to the autonomy solution. This makes SpaceTry a neutral platform for comparing autonomy solutions before lab or field testing.

The contributions are: (1) a scenario-based architecture for in-house evaluation of self-adaptive autonomy in space exploration missions; (2) an LLM-assisted workflow for translating natural-language mission descriptions into executable and monitorable planetary-autonomy artifacts; and (3) an open-source reference implementation with a Mars outpost case study integrating mission execution, runtime monitoring, and evaluation reporting.

II. Related Work

The work most closely related to SpaceTry spans space-robotics frameworks, autonomy-evaluation environments, and scenario-generation methods. Space ROS targets flight-oriented ROS 2 infrastructure [23], and ROS simulation

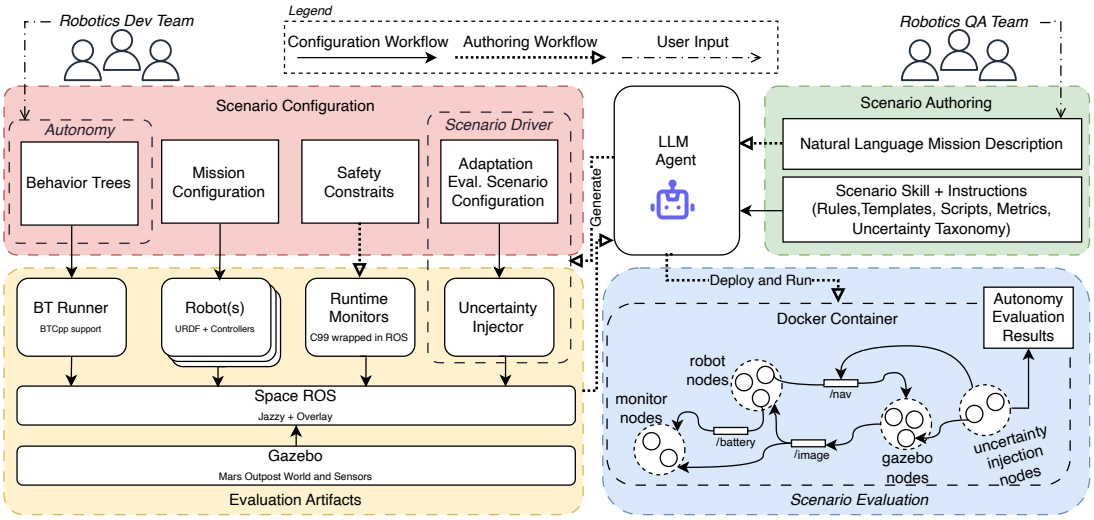

Fig. 1: SpaceTry architecture for autonomy specification and evaluation.

supports on-orbit servicing [24]. Planetary-autonomy evaluation uses analogue missions, field tests, and operational-scenario simulators [25]–[27]. These works define non-LLM baselines: expert-authored scenarios, fixed libraries, parameter sweeps, and scripted fault injection. Scenario-based testing adds model-, search-, and data-driven generation [7], [8]. In SpaceTry, LLMs complement these alternatives by supporting natural-language mission intent, scarce domain data, and links among goals, uncertainties, constraints, and executable ROS artifacts. Prior work supports LLMs for grounded robot plans and requirements-driven scenarios [15], [16], [28]. SpaceTry combines these strands under Space ROS/Gazebo with runtime-monitor evidence.

III. SpaceTry Architecture

Fig. 1 presents SpaceTry as a reusable architecture for simulation-based, scenario-based autonomy evaluation in planetary robotics. The design is grounded in scenario-based testing, where the central challenge is to preserve a clear bridge between scenario design and executable evaluation under controlled and reproducible conditions. In this perspective, a scenario is treated as a first-class artifact that links mission goals, environmental assumptions, uncertainty sources, runtime constraints, and observable outcomes to an executable evaluation setup [7], [8]. This abstraction decouples evaluation from any specific planner, controller, or adaptation mechanism, enabling comparative experiments across autonomy solutions before later embodied testing.

Scenario Authoring Layer. SpaceTry's automated authoring workflow uses LLM-based agents extended with domain-specific instructions and reusable Skills to transform a natural-language mission description into evaluation artifacts. Prior work suggests that LLMs can support grounded test-scenario generation and interactive scenario editing when guided by domain knowledge [15], [16]. In SpaceTry, the scenario prompt template captures mission intent, uncertainty sources, and expected outcomes. The agent then generates a Scenario Driver that specifies how the test will be configured, executed, and observed, guided by Skill resources such as templates, rules, taxonomies, and models. This layer does not implement adaptation; it constructs traceable scenarios for autonomy solutions supplied separately.

Scenario Configuration Layer. The second layer binds the generated Scenario Driver to a concrete experiment. In SpaceTry, this includes mission waypoints and objects, behavior-tree specifications, launch parameters, safety constraints, and scenario configuration files. Here, high-level uncertainty descriptions are translated into executable scenario instances by binding mission configuration, runtime constraints, uncertainty assumptions, and observability hooks. The same scenario intent can therefore yield multiple runs with different fault placements or manifestations, enabling exploration of parameter sweeps. The architecture becomes reusable across families of missions without further structural changes.

Evaluation Artifacts Layer. SpaceTry instantiates this architecture in a Mars outpost scenario built on the Space ROS Curiosity Demo [17]. The Gazebo world preserves fixed Martian physics properties, such as gravity and directional sunlight, while exposing configurable objects including the outpost safe-return station, science targets, and a meteorite hazard. On the ROS 2 side, SpaceTry provides a `BT Runner` for mission-level actions, a custom battery and perception package, runtime monitors synthesized through the FRET+Copilot pipeline [19], [21], and an uncertainty injector driven by a taxonomy adapted from [29]. Concrete injections can include hazardous terrain configurations, accelerated battery drain, and sensor noise.

Scenario Evaluation Layer. The configured Scenario Driver is executed over a Space ROS and Gazebo substrate where the rover, simulated world, and autonomy software interact through ROS 2 topics, services, and parameters. Behavior trees encode mission-level control, while perception and resource-awareness components derive predicates such as obstacle

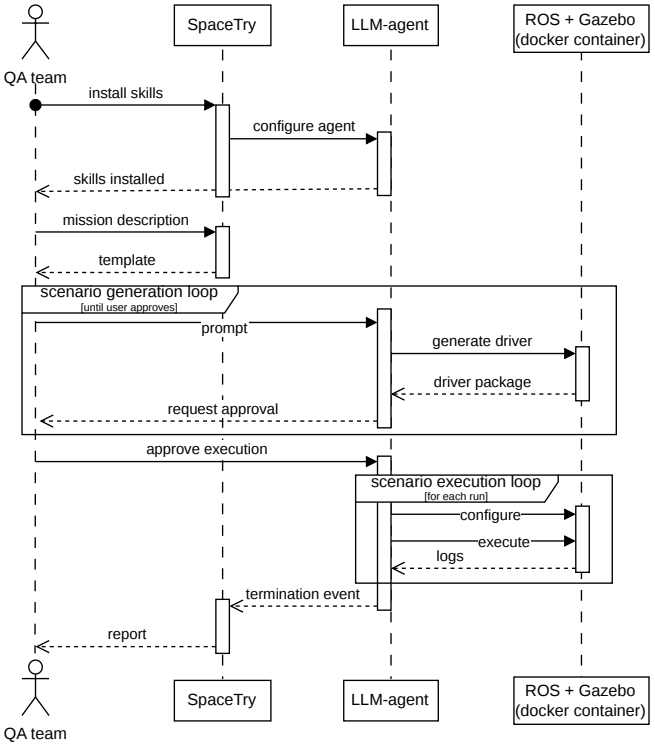

Fig. 2: Sequence Diagram for SpaceTry Authoring Workflow.

direction, battery state of charge, and proximity to safe locations [7], [30]. Formal monitors assess safety properties at runtime and publish verdicts as ROS 2 events for metrics collection and reporting. By externalizing violations instead of embedding recovery policies, SpaceTry remains neutral with respect to the adaptation strategy under evaluation.

IV. SpaceTry in Practice

SpaceTry [31] is intended to be used jointly by robotics developers and quality assurance (QA) teams. Developers supply the autonomy artifacts under test, namely the behavior tree, monitors, and supporting perception or resource-awareness components. QA teams use the same project to author scenario prompts, generate scenario drivers, execute runs, and analyze autonomy-focused outcomes rather than low-level control traces. This separation keeps the evaluated autonomy fixed while allowing the test scenario, uncertainty parameters, and reporting criteria to vary in a controlled way.

Autonomy Specification in SpaceTry. Robotics developers integrate an autonomy solution into SpaceTry by providing the artifacts under test, such as a behavior tree, supporting ROS 2 nodes, and optional runtime monitors. In the reference scenario, `base_bt.xml` serves as the baseline autonomy: it encodes waypoint-driven exploration with reactive obstacle avoidance using lidar data. During evaluation, SpaceTry keeps these autonomy artifacts fixed so that only the scenario configuration and uncertainty parameters vary across runs.

Autonomy Evaluation in SpaceTry. Fig. 2 summarizes how the reference scenario is enacted in practice. From the scenario prompt template, the Skill generates a ROS 2 scenario-driver

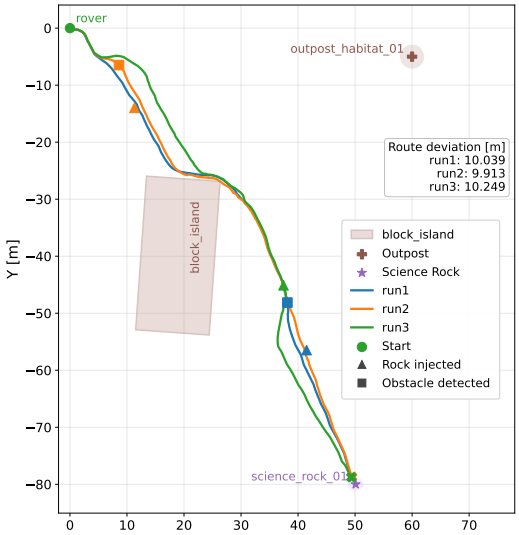

Fig. 3: Evaluation results for the SpaceTry reference scenario.

package, configures launch parameters from YAML, validates syntax and workspace compilation, executes the baseline autonomy in the containerized testbed, and reports the locations of the generated logs and evaluation artifacts. During evaluation, the agent acts adversarially with respect to the tested autonomy by adjusting scenario parameters and re-running the simulation. Excerpts of the generated scenario parameters and the evaluation report are depicted in the Appendix.

We ran the reference scenario prompt 3 times to evaluate `base_bt.xml` and demonstrate the Authoring Workflow. The targeted uncertainty locations in the user prompt were changes in the rover's environment and sensing. Fig. 3 shows the resulting rover trajectory per run, alongside the amount of deviation of the route per trajectory during execution.

In all three full runs, the injected obstacle was encountered, and the sensing degradation was active, yet the rover still reached the goal. The clearest difference across runs was post-avoidance recovery: the recovery duration was 3.044s in run1, 11.9s in run2, and was not measured by the driver in run3. Obstacle detection latency was 8.0s in run1, 8.6s in run2, and 1.9s in run3. Route deviation was 10.039m in run1, 9.913m in run2, and 10.249m in run3. As seen in the run3 path, the rover hits the simulated rock (▲), this indicates that in run3, the rover avoidance autonomy strategy failed.

V. Conclusion and Future Work

We presented SpaceTry, an open-source simulation-based testbed for scenario-based evaluation of autonomy in space exploration missions. SpaceTry combines LLM-assisted scenario authoring with configurable evaluation artifacts and Space ROS/Gazebo execution, while remaining neutral with respect to the autonomy strategy under study. We demonstrated the testbed on a Mars outpost scenario derived from NASA's Curiosity Demo. Future work will expand the testbed to additional missions (e.g., Lunar), uncertainty models, evaluation metrics, and stronger links from in-house evaluation to later embodied testing.

ACKNOWLEDGMENTS

This work has been (partially) funded by (i) Space It Up! and received funding from the Agenzia Spaziale Italiana (ASI) and the Ministero dell'Università e della Ricerca (MUR) – Contratto n. 2024-5-E.0 - CUP n. I53D24000060005 and (ii) the European HORIZON-KDT-JU-2023-2-RIA research project MATISSE "Model-based engineering of Digital Twins for early verification and validation of Industrial Systems" (grant 101140216-2, KDT232RIA 00017).

LIMITATIONS

SpaceTry is currently demonstrated on a single reference planetary scenario and a limited set of autonomy artifacts, so the evidence is not yet sufficient for broad claims across mission classes or adaptation strategies. It supports in-house (in-vitro) rather than field testing, so it is intended to prioritize candidate scenarios for later lab or field campaigns rather than replace them. The LLM-assisted workflow is evaluated here as a research prototype rather than as a mature engineering tool, and additional validation is needed across more missions, uncertainty models, repeated executions, and different LLM agents, models, and reasoning settings.

APPENDIX

This appendix provides excerpts of the generated scenario-driver configuration and evaluation report used to document the reference SpaceTry run.

Scenario Driver Configuration.

```yaml
runtime:
  timeout_s: 1200.0
  start_guard_s: 20.0
  min_progress_ratio_before_injection: 0.55
  max_progress_ratio_for_injection: 0.92
  max_distance_to_injection_pose_m: 14.0
  min_distance_to_goal_after_injection_m: 18.0
  min_distance_from_baseline_hazard_m: 18.0
  progress_window_s: 6.0
  min_progress_distance_m_over_window: 2.0
  minimum_post_encounter_observation_window_s: 30.0

injected_obstacle:
  model_name: rock_5
  model_relative_path: rock_5/model.sdf
  entity_name_prefix: injected_route_rock
  x: 36.0
  y: -57.5
  z: -14.5
  roll: -1.1
  pitch: -0.1
  yaw: 3.07
  encounter_distance_m: 11.0
  attribution_distance_m: 12.0
  clear_distance_m: 14.0
  collision_distance_m: 3.0

degradation:
  publish_hz: 5.0
  max_duration_s: 22.0
  pre_encounter_noise_duration_s: 6.0
  side_flip_period_s: 0.4

metrics:
  obstacle_threshold_m: 9.0
  reaction_angular_threshold_radps: 0.45
  slow_linear_threshold_mps: 0.70
  resumed_linear_threshold_mps: 0.80
  resumed_progress_window_s: 6.0
```

Listing 1: Run 3 scenario driver parametrization (excerpt).

Scenario Evaluation Report.

```
# Scenario Report: spacetry_scenario_navigation_obstacle_degraded_perception

- Run label: `full_run_20260411T100300`
- Termination reason: `goal_reached`
- Outcome assessment: `DEGRADED`
- Baseline outcome assessment: `DEGRADED`
- Injected outcome assessment: `FAIL`

## Scenario Summary

- Primary evaluation target: runtime obstacle blocking on the route to
↪  `science_rock_01`.
- Secondary injected uncertainty: degraded obstacle interpretation on the
↪  autonomy-facing obstacle topics.
- Baseline autonomy intentionally left untouched:
↪  `src/spacetry_bt/trees/base_bt.xml`, `spacetry_bringup`,
↪  `spacetry_perception`, `spacetry_battery`, and `spacetry_monitors` remained
↪  unchanged and were only observed.

## Key Metrics

- Adaptation speed: `None` ms
- Obstacle detection latency: `1900.0` ms via `/obstacle/left`
- Raw scan detection latency: `None` ms
- Recovery rate: `None` ms
- Route deviation: `10.249` m
- Post-injection route deviation: `7.817` m
- False obstacle rate: `33` / `108` (`30.56`%)
- Evaluation window after encounter: `95.2` s

## Safety And Goal

- MR_009 preserved: `True`
- MR_011 preserved: `False`
- Collision with injected obstacle avoided: `False`
- science_rock_01 reached: `True`
- Mission deadline met: `True`

## Attribution

- Observed control rationale: `unknown`
- Reaction scope: `indeterminate`
- Reaction attribution status: `False`
- Injected uncertainty source: `[]`
- Baseline uncertainty exercised: `True`

## Observability Notes

- Collision status is inferred conservatively from rover-to-fault distance
↪  because the baseline stack does not expose a direct contact signal on a ROS
↪  topic.
- The primary obstacle detection metric prefers the autonomy-facing obstacle
↪  topics, with raw `/scan` timing reported separately.
```

Listing 2: Generated report for run 3 (excerpt).

REFERENCES

[1] G. Ishigami, K. Nagatani, and K. Yoshida, "Path planning for planetary exploration rovers and its evaluation based on wheel slip dynamics," in *2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 2361–2366. [Online]. Available: https://doi.org/10.1109/ROBOT.2007.363672

[2] R. Domínguez *et al.*, "Cooperative robotic exploration of a planetary skylight surface and lava cave," *Science Robotics*, vol. 10, no. 105, p. eadj9699, 2025. [Online]. Available: https://doi.org/10.1126/scirobotics.adj9699

[3] T. S. Vaquero *et al.*, "Eels: Autonomous snake-like robot with task and motion planning capabilities for ice world exploration," *Science Robotics*, vol. 9, no. 88, p. eadh8332, 2024. [Online]. Available: https://doi.org/10.1126/scirobotics.adh8332

[4] G. Falcone and Z. R. Putnam, "Autonomous decision-making for aerobraking via parallel randomized deep reinforcement learning," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 59, no. 3, pp. 3055–3070, 2023. [Online]. Available: https://doi.org/10.1109/TAES.2022.3221697

[5] A. Rankin, N. Patel, E. Graser, J.-K. F. Wang, and K. Rink, "Assessing Mars Curiosity rover wheel damage," in *Proc. IEEE Aerospace Conference*, 2022.

[6] M. Rostamnia, G. Filippone, R. Caldas, and P. Pelliccione, "Towards adaptable and uncertainty-aware behavior trees," in *2025 IEEE/ACM 7th International Workshop on Robotics Software Engineering (RoSE)*. IEEE, 2025, pp. 9–16.

[7] R. Queiroz *et al.*, "A driver-vehicle model for ads scenario-based testing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 8, pp. 8641–8654, 2024. [Online]. Available: https://doi.org/10.1109/TITS.2024.3373531

[8] W. Ding *et al.*, "A survey on safety-critical driving scenario generation: A methodological perspective," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 7, pp. 6971–6988, 2023. [Online]. Available: https://doi.org/10.1109/TITS.2023.3259322

[9] A. Bertolino *et al.*, "A survey of field-based testing techniques," *ACM Computing Surveys (CSUR)*, vol. 54, no. 5, pp. 1–39, 2021.

[10] M. Askarpour *et al.*, "RoboMAX: Robotic mission adaptation exemplars," in *Proc. IEEE/ACM International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, 2021, pp. 245–251.

[11] G. R. Silva *et al.*, "SUAVE: An exemplar for self-adaptive underwater vehicles," in *Proc. IEEE/ACM International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, 2023, pp. 181–187.

[12] C. Imrie *et al.*, "Aloft: Self-adaptive drone controller testbed," in *Proc. ACM/IEEE International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, 2024.

[13] S. García, D. Strüber, D. Brugali, T. Berger, and P. Pelliccione, "Robotics software engineering: A perspective from the service robotics domain," in *Proc. ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, 2020, pp. 593–604.

[14] Y. Zhao *et al.*, "A survey on the application of large language models in scenario-based testing of automated driving systems," *IEEE Transactions on Intelligent Transportation Systems*, 2026.

[15] C. Arora, T. Herda, and V. Homm, "Generating test scenarios from nl requirements using retrieval-augmented llms: An industrial study," in *2024 IEEE 32nd International Requirements Engineering Conference (RE)*. IEEE, 2024, pp. 240–251. [Online]. Available: https://doi.org/10.1109/RE59067.2024.00031

[16] S. Li, T. Azfar, and R. Ke, "Chatsumo: Large language model for automating traffic scenario generation in simulation of urban mobility," *IEEE Transactions on Intelligent Vehicles*, vol. 10, no. 11, pp. 4962–4973, 2025. [Online]. Available: https://doi.org/10.1109/TIV.2024.3508471

[17] Space ROS, "Open-source framework for space robotics," https://github.com/space-ros, 2024.

[18] M. Colledanchise and P. Ögren, *Behavior Trees in Robotics and AI: An Introduction*. CRC Press, 2018.

[19] D. Giannakopoulou, T. Pressburger, A. Mavridou, and J. Schumann, "Automated formalization of structured natural language requirements," *Information and Software Technology*, vol. 137, p. 106590, 2021.

[20] E. Khalastchi and M. Kalech, "On Fault Detection and Diagnosis in Robotic Systems," *ACM Comput. Surv.*, vol. 51, no. 1, Jan. 2018. [Online]. Available: https://doi.org/10.1145/3146389

[21] I. Perez, A. Mavridou, T. Pressburger, A. Will, and P. J. Martin, "Monitoring ROS2: From requirements to autonomous robots," in *Proc. International Workshop on Formal Methods for Autonomous Systems (FMAS)*, ser. EPTCS, vol. 371, 2022, pp. 208–216.

[22] R. Caldas *et al.*, "Runtime verification and field-based testing for ROS-based robotic systems," *IEEE Transactions on Software Engineering*, vol. 50, no. 10, pp. 2544–2567, 2024.

[23] A. B. Probe *et al.*, "Space ros: An open-source framework for space robotics and flight software," in *AIAA SciTech 2023 Forum*, 2023, nASA NTRS 20220017761. [Online]. Available: https://ntrs.nasa.gov/citations/20220017761

[24] J. L. Ramón, J. Pomares, and L. Felicetti, "Task space control for on-orbit space robotics using a new ros-based framework," *Simulation Modelling Practice and Theory*, vol. 127, p. 102790, 2023. [Online]. Available: https://doi.org/10.1016/j.simpat.2023.102790

[25] R. Francis *et al.*, "Utility and applications of rover science autonomy capabilities: Outcomes from a high-fidelity analogue mission simulation," *Planetary and Space Science*, vol. 170, pp. 52–60, 2019. [Online]. Available: https://doi.org/10.1016/j.pss.2019.03.007

[26] E. Z. N. Dobrea *et al.*, "Rover science autonomy in planetary exploration: Field analog tests," *The Planetary Science Journal*, vol. 6, no. 2, p. 51, 2025. [Online]. Available: https://doi.org/10.3847/PSJ/adaa78

[27] J. Rimani, N. Viola, and S. Lizy-Destrez, "Simulating operational concepts for autonomous robotic space exploration systems: A framework for early design validation," *Aerospace*, vol. 10, no. 5, p. 408, 2023. [Online]. Available: https://doi.org/10.3390/aerospace10050408

[28] I. Singh *et al.*, "Progprompt: Generating situated robot task plans using large language models," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 11 523–11 530. [Online]. Available: https://doi.org/10.1109/ICRA48891.2023.10161317

[29] S. Mahdavi-Hezavehi, P. Avgeriou, and D. Weyns, "A classification framework of uncertainty in architecture-based self-adaptive systems with multiple quality requirements," in *Managing trade-offs in adaptable software architectures*. Elsevier, 2017, pp. 45–77.

[30] S. Merino-Fidalgo, C. Sánchez-Girón, E. Zalama, J. Gómez-García-Bermejo, and J. Duque-Domingo, "Behavior tree generation and adaptation for a social robot control with llms," *Robotics and Autonomous Systems*, vol. 194, p. 105165, 2025. [Online]. Available: https://doi.org/10.1016/j.robot.2025.105165

[31] R. Caldas and K. Lima, "Spacetry testbed for self-adaptive space exploration missions," Apr. 2026. [Online]. Available: https://doi.org/10.5281/zenodo.19921736