SATELLITEOPSGYM: BENCHMARKING AI AGENTS FROM FAULT DIAGNOSIS TO CLOSED-LOOP RECOVERY IN SATELLITE OPERATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Autonomous satellite operations require intelligent agents to diagnose faults, execute interventions, and verify recovery through closed-loop interaction. However, the operational capabilities and limitations of agents intended for onboard satellite operations remain poorly understood. We introduce SATELLITEOPSGYM, an interactive virtual satellite environment for evaluating AI agents from fault diagnosis to closed-loop recovery in satellite operations. Grounded in documented on-orbit faults and real multichannel telemetry, it combines causally structured simulation with effect-augmented telemetry replay, enabling agents to investigate anomalies, execute commands, observe their consequences, and verify recovery through closed-loop interaction. Its operational fidelity is assessed through double-blind evaluation by certified satellite engineers. Experiments with eight frontier LLMs on 15 shared scenarios reveal a pronounced *fault-to-recovery gap*: although fault identification reaches 71.4% and command validity reaches 100%, root-cause Planning and Recovery achieve at most 39.3% and 32.1%, respectively. By exposing capability gaps and failure modes across the operational workflow, SATELLITEOPSGYM provides an interactive testbed for evaluating satellite operations agents and informing their future design and improvement. Code and data are available at `https://github.com/anonymous-satelliteopsgym/SatelliteOpsGym`.

1 INTRODUCTION

Satellite constellations are rapidly expanding, placing increasing pressure on traditional mission operations. Earth orbit already hosts approximately 11,000 active payloads, with this number continuing to grow as large low-Earth-orbit constellations are deployed (European Space Agency, 2025). Managing such fleets requires continuous telemetry interpretation, operational decision-making, and fault recovery under limited communication and stringent safety constraints. Onboard autonomous agents offer the potential to enhance satellite autonomy by enabling intelligent monitoring, reasoning, and operational decision-making with reduced dependence on ground intervention (Holder et al., 2025; Oliveira et al., 2025). However, their capabilities and limitations in closed-loop satellite operations remain poorly understood. This calls for an interactive benchmark that captures the observation–diagnosis–planning–action–verification loop shown in Figure 1, and evaluates both stage-specific capabilities and end-to-end operational outcomes.

Existing satellite telemetry benchmarks have advanced anomaly detection and fault identification from multivariate observations (Hundman et al., 2018; Kotowski et al., 2024; Ruszczak et al., 2025). However, they primarily evaluate predictions on recorded telemetry and do not support agent actions that change the system and produce new observations. Interactive AIOps benchmarks have begun to evaluate agents through investigation, intervention, and feedback in software and cloud systems (Chen et al., 2025). Yet these benchmarks are designed for software services and do not directly capture the evolving physical state and operational constraints of a satellite. Consequently, neither line of work directly supports evaluating satellite operations agents across the complete workflow.

To evaluate the workflow in Figure 1, a benchmark must assess both an agent's performance at individual stages and the outcome of the complete interaction. Building such a benchmark for satellite operations presents two challenges. First, fault reports, in-orbit telemetry, spacecraft specifications,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

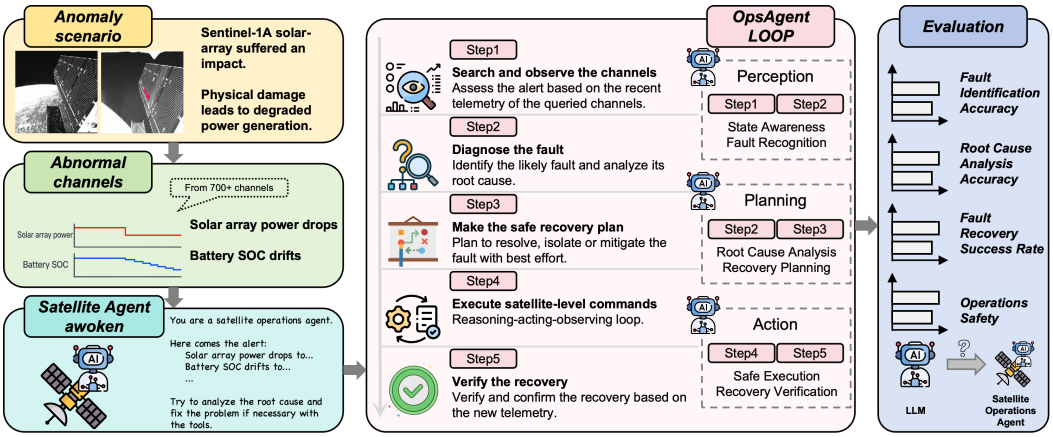


Figure 1: Five stages of autonomous satellite operations and their corresponding agent capabilities: perception (**Observe** and **Verify**), plan (**Diagnose** and **Plan**), and execution (**Act**). Successful recovery requires their coordination throughout the closed loop.

and command catalogs are not naturally aligned. Turning them into interactive tasks requires connecting a fault's causal structure to its time-varying telemetry manifestations and available actions, while separating the underlying fault state from what the agent can observe. Second, a command cannot be evaluated independently of the current spacecraft state: the same operation may resolve, leave unchanged, or worsen a fault depending on active failures, spacecraft configuration, and subsystem dependencies. The environment must therefore produce state-dependent consequences and expose their effects through subsequent telemetry and execution feedback, so that the benchmark measures the outcomes of agents' decisions rather than merely whether their commands are valid.

To address these challenges, we introduce SATELLITEOPSGYM, which turns documented fault cases and nominal in-orbit telemetry into replayable interactive tasks, runs agents against them, and evaluates their interaction trajectories. During construction, a spacecraft component graph links fault-affected components to telemetry channels. Each task is frozen as a private causal DAG of fault propagation and a time-indexed plan of observable effects over a native-timestamp telemetry window, separating what happens in the environment from what the agent can see. During interaction, an event-driven engine advances the telemetry timeline and exposes alerts, queried telemetry, and command receipts to the agent. Commands are checked against their schemas, current-state preconditions, and safety constraints; a constrained LLM-assisted simulator then proposes state-dependent transitions, which are validated before taking effect. Because a receipt does not reveal whether a command improved the fault, the agent must assess its effects through subsequent telemetry. Finally, the evaluator uses the frozen task and interaction trajectory to measure performance across operational stages and the recovery outcome, alongside safety and efficiency.

We evaluate eight LLMs in the interactive environment of SATELLITEOPSGYM across the satellite fault management workflow. The results reveal a pronounced diagnosis-to-recovery gap: fault identification reaches 71.4%, whereas root-cause planning and recovery peak at only 39.3% and 32.1%, respectively. Trajectory analysis shows that agents often address symptoms rather than underlying faults, leaving the spacecraft unrecovered despite executing valid commands. These findings highlight the importance of evaluating closed-loop recovery beyond individual-stage performance.

Our contributions are threefold:

- We construct a satellite fault-management task suite from documented fault cases, in-orbit telemetry, and spacecraft component knowledge.
- We develop an interactive environment and evaluation protocol in which agents investigate telemetry, issue constrained commands, and observe their consequences. The protocol measures stage-wise performance, recovery outcomes, safety, and interaction efficiency.
- We benchmark representative agents across the operational workflow and analyze their failure modes. The results reveal substantial differences between stages, particularly in root-cause analysis and achieving recovery.

2 Related Work

2.1 AIOps and Autonomous Operations

AIOps leverages logs, metrics, and traces for anomaly detection, diagnosis, and mitigation. Early studies addressed log anomalies (Du et al., 2017; Meng et al., 2019), dependency-based fault localization (Wang et al., 2018; Wu et al., 2020), and corrective actions (Gan et al., 2021; Qiu et al., 2020). Recent work incorporates LLMs into root-cause analysis and autonomous cloud operations (Chen et al., 2024; Shetty et al., 2024). AIOpsLab, ITBench, and AssetOpsBench further extend evaluation to interactive recovery, IT automation, and industrial maintenance (Chen et al., 2025; Jha et al., 2025; Patel et al., 2025). Unlike these settings, satellite operations involve coupled physical subsystems, delayed feedback, and safety-critical commands. SatelliteOpsGym evaluates agents through closed-loop spacecraft recovery.

2.2 Satellite Telemetry Datasets

Spacecraft telemetry research has primarily focused on anomaly detection (Hundman et al., 2018; Xu et al., 2022). Recent datasets, including ESA-ADB and OPSSAT-AD, provide annotated telemetry for evaluating detection methods (Kotowski et al., 2024; Ruszczak et al., 2025), while Fejjari et al. (2025) surveyed advances in this area. However, these datasets do not provide executable commands or action-dependent state transitions. SatelliteOpsGym connects fault knowledge, telemetry, and operational actions to evaluate recovery beyond offline detection.

2.3 LLM-as-Simulator

LLMs have enabled interactive simulations of agent behavior, as demonstrated by Generative Agents, Concordia, and SOTOPIA (Park et al., 2023; Vezhnevets et al., 2023; Zhou et al., 2024b). AgentBench, WebArena, and ToolSandbox further advance interactive agent evaluation through diverse tasks and stateful environments (Liu et al., 2024; Zhou et al., 2024a; Lu et al., 2025). Unlike these settings, SatelliteOpsGym combines LLM-assisted transition inference with causal fault graphs and validated commands to evaluate state-dependent spacecraft recovery.

3 SatelliteOpsGym

We introduce SatelliteOpsGym, an interactive virtual satellite environment for evaluating AI agents from fault diagnosis to closed-loop recovery in satellite operations. As shown in Figure 2, it integrates structured satellite assets, telemetry-grounded simulation, and a pluggable agent interface, enabling agents to investigate alerts, execute commands, and verify their effects within a unified operational loop.

3.1 Formalization

We model satellite operations as a partially observable sequential decision process. Each episode begins with a monitor alert associated with a fault or non-fault situation. The underlying state comprises global satellite states, a fault DAG, telemetry effects and historical command receipts. Agent interactions follow

$$s_t \sim T(\cdot \mid s_{t-1}, a_t), \qquad o_t = Z(s_t; q_t), \tag{1}$$

where t indexes interactions, a_t is an agent operation, and q_t specifies the queried channels. The transition model T combines Simulator-generated proposals with deterministic event execution. The observation function Z exposes timestamped telemetry, alerts, and execution receipts.

The Agent selects operations from its public interaction history. Information queries leave satellite states unchanged; commands and bounded waiting can advance the environment and reveal subsequent changes. An episode ends at agent termination or decision-budget exhaustion. We impose no intrinsic scalar reward: the evaluation protocol assesses capabilities from the resulting trajectory.

3.2 Framework Overview

The central design separates semantic transition inference from temporal execution. Anomaly Inference and Command Inference produce structured event proposals, while the Environment validates

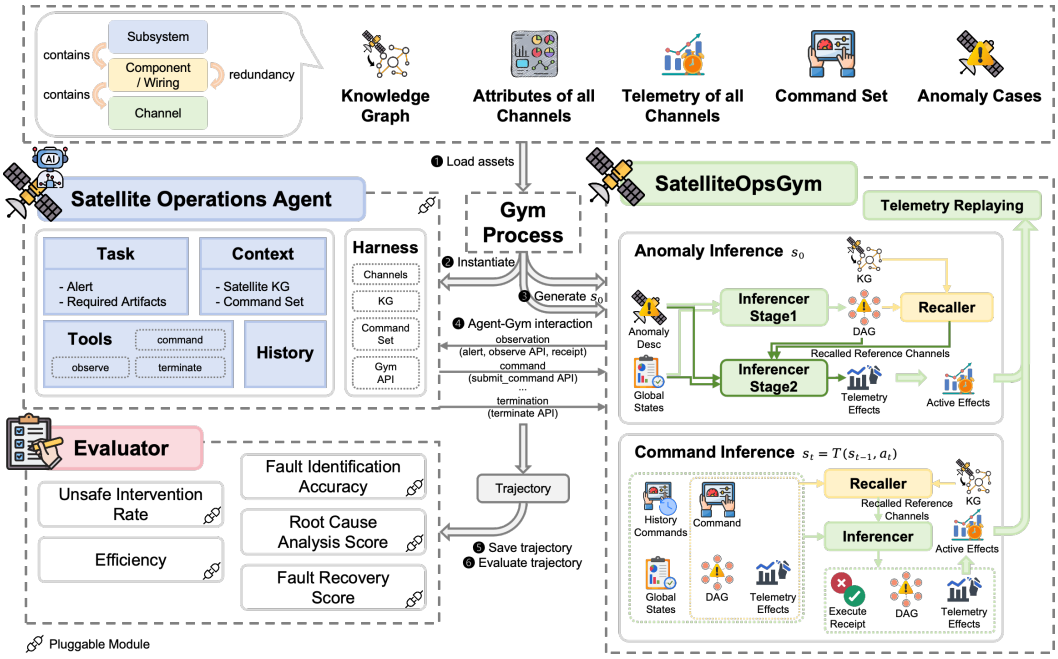


Figure 2: Overview of SATELLITEOPSGYM. Satellite assets ground both scenario initialization and command-conditioned transitions. Anomaly Inference first constructs a fault DAG, retrieves relevant channels through the Recaller, and generates scheduled Telemetry Effects. Command Inference updates fault states and effects in response to agent commands. Telemetry Replaying combines the baseline with active effects to produce observations. A pluggable Harness mediates Agent–Gym interaction, and the Gym Process records trajectories for evaluation.

these proposals and applies their effects to replayed telemetry. This allows faults and interventions to share one executable representation.

As illustrated in Figure 2, the Gym Process loads satellite assets, generates or loads a scenario, and instantiates an Environment and a pluggable Harness. The Harness maintains the Satellite Operations Agent's task, context, tools, and history; the Environment maintains authoritative dynamics. The resulting Agent–Gym interactions form a trajectory of observations, agent decisions, and environment events, providing a unified record for subsequent evaluation.

3.3 SATELLITE ASSET CONSTRUCTION

The satellite assets shown at the top of Figure 2 support the entire pipeline.

Channel attributes and telemetry. We provide normal-operation telemetry from a real low-Earth-orbit satellite as the simulation baseline, with satellite-specific information anonymized through a semantics-preserving transformation. It spans approximately 6 hours and 20 minutes, or four roughly 95-minute orbits. The filtered dataset retains 726 channels across subsystems of Structure, TCS, EPS, AOCS, OBC, TT&C, and Electric Propulsion. Each channel includes semantic metadata and basic normal-operation statistics used for channel selection, state construction, and telemetry replay.

Knowledge Graph. The Knowledge Graph is a satellite-specific structural representation of the target platform, encoding how its subsystems, physical Components/Wiring, and telemetry Channels are organized. It is constructed through a hybrid pipeline that combines deterministic rules with LLM-assisted completion, followed by human review of unresolved assignments and structural validation and contains 841 nodes: 7 Subsystems, 97 Components, 11 Wiring entities, and 726 Channels. Its 851 edges comprise 834 `contains` relations and 17 `redundancy` relations. It provides the semantic index for channel retrieval, redundancy navigation, and Agent-side parameter discovery.

Command Set. We construct the command catalog using the CCSDS Space Packet Protocol, the ECSS Packet Utilization Standard (PUS), NASA's core Flight System (cFS), and the platform architecture encoded in the Knowledge Graph (Consultative Committee for Space Data Systems, 2020; European Cooperation for Space Standardization, 2016; NASA Goddard Space Flight Center, a;b). The resulting catalog contains 166 typed operational commands covering the modeled subsystems and nearly all operator actions supported by the simulated satellite.

3.4 Causally Structured Simulation and Closed-Loop Interaction

Causally structured simulation. SatelliteOpsGym represents a satellite situation through Global States, a causal DAG of fault propagation, active Telemetry Effects, scheduled events, and command receipts. The Global States summarize the satellite's environmental, physical, and operational conditions, with the complete dimension-to-channel mapping provided in Appendix A.1. The DAG describes how a local fault may propagate: its nodes identify fault mechanisms and associated components, while directed edges encode causal relations and delays. Each node also has a lifecycle status. *latent* denotes a fault represented in the causal structure but not yet activated, *active* denotes an expressed fault, *isolated* denotes a fault whose propagation or service impact has been controlled, and *resolved* denotes a cleared fault. An example is shown in Figure 3.

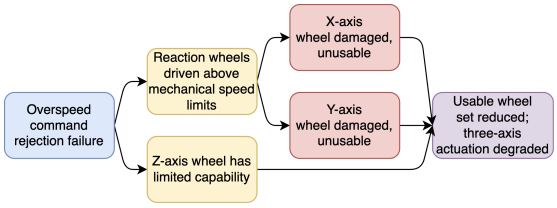


Figure 3: An example of a causal fault-propagation DAG generated by the simulator with detailed information omitted.

Anomaly Inference initializes this structured state through the two-stage pipeline shown in Figure 2. The first inference stage constructs the rooted DAG and freezes a private achievable recovery target. KG-grounded retrieval then connects fault mechanisms to candidate Component/Wiring entities and their contained channels expected to be affected. The second stage uses the frozen DAG and recalled reference channels to generate fault activations and Telemetry Effects. Proposal validation enforces graph structure, propagation timing, channel compatibility, and observable consequences for every leaf fault.

The same representation is reused for Command Inference, which implements the command-conditioned state transition shown in Figure 2. Given a command, the current Global States, the existing fault DAG and effects, and the command history, the Simulator selects additional reference channels and proposes state updates, effect changes, and a terminal receipt. The initial DAG and recovery target remain fixed, so each intervention is interpreted relative to the state produced by preceding events and commands. Retrieval and proposal constraints are summarized in Appendices A.2 and A.1.

Closed-loop interaction through baseline telemetry replay with active effects. Validated proposals are executed by a shared event queue. Events update fault statuses, start or end Telemetry Effects, modify the permitted simulation-control fields, or complete command receipts. Telemetry Replaying applies the active effects to the normal-operation time-series baseline through a small typed primitive vocabulary, including value assignment, offsets, drift, gain, noise, pulses, freezes, and dropouts. The resulting effective telemetry is used both to refresh Global States and to construct the Agent observation, keeping command preconditions and visible evidence on the same state trajectory. The complete primitive semantics and validation rules are given in Appendix A.3.

These processes form the closed Agent–Gym interaction loop shown in Figure 2. The Gym presents the Agent with an alert and the corresponding observation produced by Telemetry Replaying. Based on its interaction history, the Agent may request additional telemetry, inspect or submit a command, wait for temporal evolution, or terminate. When a command is submitted, Command Inference uses the current state and command history to generate a transition proposal. The event queue schedules and settles the proposal, after which Telemetry Replaying produces the next observation. The Agent can use each new observation to identify the fault, analyze its root cause, select and execute recovery actions, and verify their effects, repeating this operational loop until termination or the preset interaction limit is reached.

3.5 BENCHMARK AND EVALUATION PROTOCOL

Benchmark Tasks. SATELLITEOPSGYM provides 300 interactive tasks constructed from 135 documented fault events across seven satellite subsystems. Each task defines a frozen initial condition, a causal fault graph, fault-induced telemetry effects, and an attainable recovery target: resolution, isolation, or mitigation. Data provenance and task construction are detailed in Appendix B.1.

Five-Stage Evaluation. We evaluate five operational stages from each interaction trajectory τ. Observation measures assessment completion as $\mathrm{Obs} = 1 - u_\tau$, while Diagnosis measures actionable fault identification as $\mathrm{Diag} = \mathbb{I}[\hat{y}_\tau = \text{fault}]$. Planning evaluates pre-action root-cause localization through $\mathrm{Plan} = J_{\mathrm{RCA}}(\hat{h}_\tau, D^\star)$, where $J_{\mathrm{RCA}} \in \{0, 0.5, 1\}$. Execution measures command validity as $\mathrm{Exec} = 1 - R_\tau/A_\tau$ for $A_\tau > 0$, and Verification penalizes unsupported resolution claims through $\mathrm{Ver} = 1 - v_\tau$. Here, u_τ and v_τ are the corresponding failure indicators, $\hat{y}_\tau$ and $\hat{h}_\tau$ are the committed classification and root-cause hypothesis, and A_τ and R_τ denote command attempts and rejections. The first command attempt, or termination if no command is attempted, defines the pre-action assessment boundary.

Agent Capabilities and Overall Score. Following Fig. 1, the five operational stages are grouped into three agent capabilities. The Overall Score assigns equal weight to the five stages:

$$\begin{aligned} C_{\mathrm{per}} &= (\mathrm{Obs} + \mathrm{Ver})/2, \\ C_{\mathrm{plan}} &= (\mathrm{Diag} + \mathrm{Plan})/2, \\ C_{\mathrm{exec}} &= \mathrm{Exec}, \\ S_{\mathrm{overall}} &= (\mathrm{Obs} + \mathrm{Diag} + \mathrm{Plan} + \mathrm{Exec} + \mathrm{Ver})/5. \end{aligned} \tag{2}$$

At the model level, the Overall Score is computed from the five aggregated stage scores. Execution excludes trajectories without command attempts, while the other stages are aggregated over their available evaluations. Recovery, Efficiency, and Safety are reported separately to distinguish operational outcomes, interaction costs, and unsafe behavior. Detailed scoring rules and model-level aggregation are provided in Appendix B.2.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

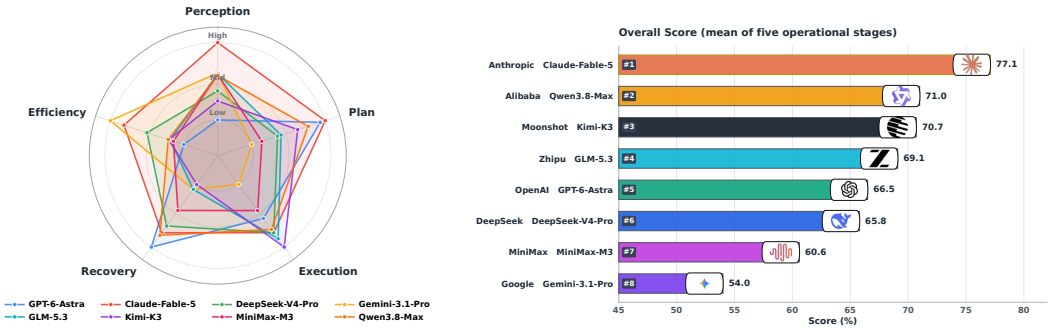
Benchmark Tasks and Agents. We evaluate eight frontier LLMs on the same 15 satellite operations scenarios selected from the 300-task SATELLITEOPSGYM benchmark, yielding 120 model-scenario evaluations. The selected scenarios span fault management tasks across EPS, AOCS, TCS, and TT&C. Identical frozen initial conditions enable controlled comparisons of agent behavior and recovery outcomes. The evaluated models are GPT-6-Astra, Claude-Fable-5, DeepSeek-V4-Pro, Gemini-3.1-Pro-Preview, GLM-5.3, Kimi-K3, MiniMax-M3, and Qwen3.8-Max.

Implementation and Protocol. The environment integrates 14 telemetry packages containing 726 channels, 108 component and wiring entities in the knowledge graph, 166 operational commands, and four orbital passes of nominal telemetry. Scenario generation uses `claude-fable-5`, while command simulation and root-cause assessment use `gpt-6-astra`, both with temperature set to 0. Each episode allows up to 60 decision turns. Model requests permit five retries, and infrastructure failures allow at most two additional runs. All attempts are preserved as hash-chained trajectories. Agents access public observations and progressively disclosed commands, while private fault states and recovery targets remain hidden. Performance is assessed using the protocol in Section 3.5.

4.2 OVERALL PERFORMANCE

Table 1 and Fig. 4 summarize the performance of eight LLM agents across the satellite fault management workflow. Despite differences in overall scores, the models exhibit a consistent pattern: fault identification and command execution are substantially stronger than root-cause localization and operational recovery.

Fault identification exceeds causal reasoning. Observation ranges from 71.4% to 100%, indicating that most agents reach a committed assessment. Diagnosis peaks at 71.4%, whereas Planning, which measures pre-action root-cause localization, reaches only 39.3%, with a mean of 28.0%. This discrepancy indicates that recognizing an actionable fault does not necessarily lead to identifying its underlying causal mechanism.



(a) Capability and outcome profiles.

(b) Overall scores.

Figure 4: Performance of eight LLM agents on SATELLITEOPSGYM. (a) Three agent capabilities, Recovery, and Efficiency. (b) Overall Score averaged across five operational stages.

Table 1: Performance of eight LLM agents across five operational stages, with Recovery reported separately. All models are evaluated on the same 15 scenarios. Scores are aggregated over available stage-specific evaluations, as detailed in Appendix B.2, and reported as percentages (%); higher is better.

Model	Observation ↑	Diagnosis ↑	Planning ↑	Execution ↑	Verification ↑	Recovery ↑
GPT-6-Astra	71.4	57.1	**39.3**	71.4	93.3	**32.1**
Claude-Fable-5	**100.0**	**71.4**	28.6	85.7	**100.0**	28.6
DeepSeek-V4-Pro	84.6	38.5	26.9	85.7	93.3	26.9
Gemini-3.1-Pro-Preview	92.9	28.6	17.9	37.5	93.3	17.9
GLM-5.3	85.7	42.9	25.0	91.7	**100.0**	17.9
Kimi-K3	86.7	46.7	33.3	**100.0**	86.7	16.7
MiniMax-M3	92.3	38.5	15.4	63.6	93.3	23.1
Qwen3.8-Max	91.7	50.0	37.5	82.4	93.3	29.2

Command validity exceeds recovery. Execution varies considerably across agents, ranging from 37.5% to 100%, while Verification remains at or above 86.7%. Nevertheless, Recovery peaks at only 32.1% and averages 24.1%. Several agents achieve high command acceptance without corresponding recovery gains, indicating that valid command execution alone is insufficient for restoring the intended spacecraft state.

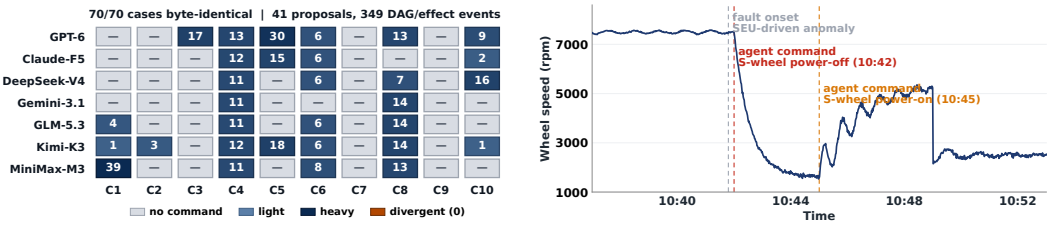
4.3 FAULT–RECOVERY GAP AND FAILURE ANALYSIS

The performance gap reveals a disconnect between fault recognition and effective intervention. Analysis of agent trajectories suggests three recurring failure modes. First, agents often mistake downstream symptoms for root causes, leaving the underlying fault active despite intervention. Second, command selection may rely on apparent relevance rather than expected causal effects, resulting in valid but ineffective operations. Third, agents may interpret temporary telemetry stabilization as successful recovery without confirming sustained improvement. These failures show why intermediate diagnostic and execution capabilities do not consistently translate into closed-loop recovery.

These findings motivate autonomous satellite operations agents that explicitly connect fault hypotheses, candidate commands, and expected spacecraft state transitions. Before execution, agents should assess both recovery potential and operational risks. After execution, observed telemetry should be compared with predicted effects to refine the diagnosis and subsequent interventions. Such causally grounded, feedback-driven decision-making could reduce ineffective operations and redundant interactions while improving recovery efficiency and safety.

4.4 SATELLITEOPSGYM EFFECTIVENESS

We assess the effectiveness of SATELLITEOPSGYM through replay reproducibility and semantic fidelity. Both evaluations use the same independent validation set of 70 agent trajectories generated by seven LLMs across ten fault scenarios, separate from the main agent evaluation. Fig. 5 presents replay consistency and a representative recovery process, while Table 2 compares documented and



(a) Byte-level replay consistency across seven agents and ten scenarios.

(b) Reaction-wheel recovery through successive interventions.

Figure 5: Effectiveness of SATELLITEOPSGYM: replay reproducibility and telemetry-grounded recovery simulation.

Table 2: Documented and simulated recovery outcomes for five representative satellite faults (details in Appendix C.2). ✓: recovery; ✗: failure.

Fault Scenario	Documented	Gym
1. Battery Cell 5 (EPS). Heater-control failure; controller reset and redundancy switching fail.	✗	✗
2. Reaction Wheel (AOCS). SEU-induced controller anomaly; power cycle restores wheel speed.	✓	✓
3. Star Sensor (AOCS). Corrupted alignment matrices; restoration and reinitialization recover identification.	✓	✓
4. Thermal Control (TCS). Loop-heat-pipe anomaly; heater cycling restores circulation.	✓	✓
5. Primary TT&C. Transponder power-controller trip; recovery attempts fail to restore communication.	✗	✗

simulated recovery outcomes. Additional agent trajectories and the expert-validation protocol are provided in Appendices C.1 and C.2, respectively.

Reproducibility. We replay all 70 validation trajectories three times and compare accepted simulator proposals and their resulting fault-graph and telemetry-effect events at the byte level. Across 41 command proposals and 349 fault-graph or effect events, all 70 trajectories produce byte-identical results across the three replays, as shown in Fig. 5a. These results demonstrate reproducible environment transitions under identical initial conditions and recorded interactions.

Semantic Fidelity. Table 2 shows that SATELLITEOPSGYM reproduces the documented recovery outcomes of five representative fault scenarios, while Fig. 5b illustrates the telemetry response to successive reaction-wheel interventions. We further assess semantic fidelity through a double-blind study involving six certified satellite operations engineers and the independent 70-trajectory validation set. The expert panel's majority recovery judgments agree with the Gym outcomes for all 70 trajectories (100%). The study protocol and operational plausibility assessment are detailed in Appendix C.2. Together, these results support the reproducibility and operational fidelity of SATELLITEOPSGYM within the evaluated scenarios.

5 CONCLUSION

We introduced SATELLITEOPSGYM, an expert-validated interactive environment for closed-loop satellite operations. Built upon 135 documented anomaly events and real satellite telemetry, it provides 300 interactive tasks with causal fault simulation and state-dependent commands. Evaluating eight frontier LLMs on 15 satellite operations scenarios reveals a pronounced fault-to-recovery gap: agents can recognize faults and execute valid commands but often fail to select effective interventions and achieve recovery. By exposing operational capability gaps and failure modes, SATELLITEOPSGYM provides a testbed for agent evaluation and design improvement. Future work will incorporate more real-world fault scenarios, identify further agent design limitations, and explore Gym-based agent training and continual self-improvement. Evaluation across the full benchmark will further assess the generality of our findings.

REFERENCES

Yinfang Chen, Huaibing Xie, Minghua Ma, Yu Kang, Xin Gao, Liu Shi, Yunjie Cao, Xuedong Gao, Hao Fan, Ming Wen, Jun Zeng, Supriyo Ghosh, Xuchao Zhang, Chaoyun Zhang, Qingwei Lin, Saravan Rajmohan, Dongmei Zhang, and Tianyin Xu. Automatic root cause analysis via large language models for cloud incidents. In *Proceedings of the Nineteenth European Conference on Computer Systems*, pp. 674–688, 2024. doi: 10.1145/3627703.3629553.

Yinfang Chen, Manish Shetty, Gagan Somashekar, Minghua Ma, Yogesh Simmhan, Jonathan Mace, Chetan Bansal, Rujia Wang, and Saravan Rajmohan. AIOpsLab: A holistic framework to evaluate AI agents for enabling autonomous clouds. In *Proceedings of the Eighth Annual Conference on Machine Learning and Systems (MLSys)*, volume 7, 2025. URL https://openreview.net/forum?id=3EXBLwGxtq.

Consultative Committee for Space Data Systems. Space packet protocol. Technical Report CCSDS 133.0-B-2, CCSDS, June 2020. URL https://ccsds.org/Pubs/133x0b2e2.pdf.

Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. DeepLog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1285–1298. Association for Computing Machinery, 2017. doi: 10.1145/3133956.3134015.

European Cooperation for Space Standardization. Space engineering:telemetry and telecommand packet utilization. Technical Report ECSS-E-ST-70-41C, ECSS, April 2016. URL https://ecss.nl/standard/ecss-e-st-70-41c-space-engineering-telemetry-and-telecommand-packet-utilization-15-april-2016/.

European Space Agency. ESA space environment report 2025. European Space Agency, 2025. URL https://www.esa.int/Space_Safety/Space_Debris/ESA_Space_Environment_Report_2025.

Asma Fejjari, Alexis Delavault, Robert Camilleri, and Gianluca Valentino. A review of anomaly detection in spacecraft telemetry data. *Applied Sciences*, 15(10):5653, 2025. doi: 10.3390/app15105653.

Yu Gan, Mingyu Liang, Sundar Dev, David Lo, and Christina Delimitrou. Sage: Practical and scalable ML-driven performance debugging in microservices. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 135–151. Association for Computing Machinery, 2021. doi: 10.1145/3445814.3446700.

Joshua Holder, Natasha Jaques, and Mehran Mesbahi. Multi-agent reinforcement learning for sequential satellite assignment problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 26516–26524, 2025.

Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 387–395, 2018. doi: 10.1145/3219819.3219845.

Saurabh Jha, Rohan R. Arora, Yuji Watanabe, Takumi Yanagawa, Yinfang Chen, Jackson Clark, Bhavya Bhavya, Mudit Verma, Harshit Kumar, Hirokuni Kitahara, Noah Zheutlin, Saki Takano, Divya Pathak, Felix George, Xinbo Wu, Bekir O. Turkkan, Gerard Vanloo, Michael Nidd, Ting Dai, Oishik Chatterjee, Pranjal Gupta, Suranjana Samanta, Pooja Aggarwal, Rong Lee, Jae-Wook Ahn, Debanjana Kar, Amit Paradkar, Yu Deng, Pratibha Moogi, Prateeti Mohapatra, Naoki Abe, Chandrasekhar Narayanaswami, Tianyin Xu, Lav R. Varshney, Ruchi Mahindru, Anca Sailer, Laura Shwartz, Daby Sow, Nicholas C. M. Fuller, and Ruchir Puri. ITBench: Evaluating AI agents across diverse real-world IT automation tasks. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 27134–27197. PMLR, 2025. URL https://proceedings.mlr.press/v267/jha25a.html.

Krzysztof Kotowski, Christoph Haskamp, Jacek Andrzejewski, Bogdan Ruszczak, Jakub Nalepa, Daniel Lakey, Peter Collins, Aybike Kolmas, Mauro Bartesaghi, Jose Martinez-Heras, and Gabriele De Canio. European space agency benchmark for anomaly detection in satellite telemetry. arXiv preprint arXiv:2406.17826, 2024. URL https://arxiv.org/abs/2406.17826.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as agents. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/e9df36b21ff4ee211a8b71ee8b7e9f57-Abstract-Conference.html.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1160–1183. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.findings-naacl.65. URL https://aclanthology.org/2025.findings-naacl.65/.

Weibin Meng, Ying Liu, Yichen Zhu, Shenglin Zhang, Dan Pei, Yuqing Liu, Yihao Chen, Ruizhi Zhang, Shimin Tao, Pei Sun, and Rong Zhou. LogAnomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pp. 4739–4745, 2019. doi: 10.24963/ijcai.2019/658.

NASA Goddard Space Flight Center. Core flight system. GitHub repository, a. URL https://github.com/nasa/cFS. Accessed: 2026-09-26.

NASA Goddard Space Flight Center. Core flight system: Health and safety application. GitHub repository, b. URL https://github.com/nasa/HS. Accessed: 2026-09-26.

Alexandre Oliveira, Katarina Dyreby, Francisco Caldas, and Cláudia Soares. OrbitZoo: Real orbital systems challenges for reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-1662. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/47390393b2dfa57d9b3b55466d27bad8-Abstract-Conference.html.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, 2023. doi: 10.1145/3586183.3606763.

Dhaval Patel, Shuxin Lin, James Rayfield, Nianjun Zhou, Roman Vaculin, Natalia Martinez, Fearghal O'Donncha, and Jayant Kalagnanam. AssetOpsBench: Benchmarking AI agents for task automation in industrial asset operations and maintenance. arXiv preprint arXiv:2506.03828, 2025. URL https://arxiv.org/abs/2506.03828.

Haoran Qiu, Subho S. Banerjee, Saurabh Jha, Zbigniew T. Kalbarczyk, and Ravishankar K. Iyer. FIRM: An intelligent fine-grained resource management framework for SLO-oriented microservices. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pp. 805–825. USENIX Association, 2020. URL https://www.usenix.org/conference/osdi20/presentation/qiu.

Bogdan Ruszczak, Krzysztof Kotowski, David Evans, and Jakub Nalepa. The OPS-SAT benchmark for detecting anomalies in satellite telemetry. *Scientific Data*, 12:710, 2025. doi: 10.1038/s41597-025-05035-3.

Manish Shetty, Yinfang Chen, Gagan Somashekar, Minghua Ma, Yogesh Simmhan, Xuchao Zhang, Jonathan Mace, Dax Vandevoorde, Pedro Las-Casas, Shachee Mishra Gupta, Suman Nath, Chetan Bansal, and Saravan Rajmohan. Building AI agents for autonomous clouds: Challenges and design principles. In *Proceedings of the 2024 ACM Symposium on Cloud Computing*, pp. 99–110, 2024. doi: 10.1145/3698038.3698525.

Alexander Sasha Vezhnevets, John P. Agapiou, Avia Aharon, Ron Ziv, Jayd Matyas, Edgar A. Dueñez-Guzmán, William A. Cunningham, Simon Osindero, Danny Karmon, and Joel Z. Leibo. Generative agent-based modeling with actions grounded in physical, social, or digital space using Concordia. arXiv preprint arXiv:2312.03664, 2023. URL https://arxiv.org/abs/2312.03664.

Ping Wang, Jingmin Xu, Meng Ma, Weilan Lin, Disheng Pan, Yuan Wang, and Pengfei Chen. CloudRanger: Root cause identification for cloud native systems. In *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pp. 492–502. IEEE, 2018. doi: 10.1109/CCGRID.2018.00076.

Li Wu, Johan Tordsson, Erik Elmroth, and Odej Kao. MicroRCA: Root cause localization of performance issues in microservices. In *2020 IEEE/IFIP Network Operations and Management Symposium (NOMS)*, pp. 1–9. IEEE, 2020. doi: 10.1109/NOMS47738.2020.9110353.

Jiehui Xu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Anomaly Transformer: Time series anomaly detection with association discrepancy. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=LzQQ89U1qm_.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024a. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/4410c0711e9154a7a2d26f9b3816d1ef-Abstract-Conference.html.

Xuhui Zhou, Hao Zhu, Leena Mathur, Ruohong Zhang, Haofei Yu, Zhengyang Qi, Louis-Philippe Morency, Yonatan Bisk, Daniel Fried, Graham Neubig, and Maarten Sap. SOTOPIA: Interactive evaluation for social intelligence in language agents. In *International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=mM7VurbA4r.

A Runtime, Inference, and Telemetry-Effect Specifications

A.1 Runtime Details

A.1.1 Global States and Monitored Channels

Global States provide a compact projection of the satellite's operating condition. The implementation organizes 35 monitored channels into eight dimensions under three groups: environment, physical, and operation. Table 3 lists the complete mapping.

At each state refresh, the Environment queries the preceding 60 seconds of telemetry, applies the active Telemetry Effects, and uses the latest effective sample for each monitored channel. A missing latest value remains missing. The fixed projection is used for Simulator contexts and state-dependent command checks, while other retained channels remain available through the Agent telemetry interface.

Table 3: Global States dimensions and monitored telemetry channels.

Channel code	Channel label
Environment / Orbit geometry (3)	
PK05TM0009	Satellite Longitude
PK05TM0010	Satellite Latitude
PK05TM0011	Satellite Altitude (km)
Environment / Illumination (1)	
PK05TM0073	Whether Sunlit Zone
Physical / Energy (6)	
PK01TM0016	Bus Voltage
PK01TM0017	Bus Current
PK01TM0018	Battery Current
PK01TM0019	Battery Voltage
PK01TM0034	Whole Satellite Power Status
PK01TM0036	Whole Satellite Power Mode
Physical / Thermal (4)	
PK04TM0012	Battery Temperature Measurement 1
PK04TM0013	Battery Temperature Measurement 2
PK13TM0009	Central Computer Primary Board One-wire Temperature Measurement
PK13TM0011	GNSS Board One-wire Temperature Measurement
Physical / Attitude and control (9)	
PK05TM0006	AOCS Expected Operating Mode
PK05TM0007	Attitude Determination Mode
PK05TM0018	Angular Velocity X
PK05TM0019	Angular Velocity Y
PK05TM0020	Angular Velocity Z
PK05TM0021	Attitude Angle x
PK05TM0022	Attitude Angle Y
PK05TM0023	Attitude Angle Z
PK06TM0047	Star Sensor Attitude Valid Flag
Physical / Navigation and sensors (4)	
PK01TM0005	Primary GNSS Status
PK01TM0006	Backup GNSS Status
PK03TM0148	GNSS Acquisition Status
PK14TM0111	GNSS Operating Status
Operation / Communication link (4)	
PK01TM0003	Primary TT&C Data Transmission Status
PK01TM0004	Backup TT&C Data Transmission Status
PK05TM0004	Enter TT&C Station Flag

Continued on next page

Table 3 (continued)

Channel code	Channel label
PK02TM0034	TT&C Downlink Operating Mode
Operation / Onboard computing (4)	
PK02TM0026	Whole Satellite Operating Mode
PK03TM0145	Satellite Management Online Status
PK13TM0025	Satellite Management Active Primary/Backup Unit Number
PK14TM0024	Satellite Management Primary/Backup Status

A.1.2 Authoritative State and Event Execution

In addition to Global States, the Environment maintains the fault DAG, active Telemetry Effects, command receipts, a scheduled event queue, and simulation-control state. The control state contains logical time, transition phase, state version, and episode status.

The Environment is event-driven. A transition proposal is checked against the current state version and runtime contract before it is committed. Its events are inserted into a priority queue and processed in logical-time order. Fault events update node statuses; effect events start or end channel transformations; state events update the two permitted simulation-control fields; and receipt events complete command execution. A transition becomes stable after all of its events have been processed.

Direct state events may modify only `simulation.active_navigation_source` and `simulation.selected_gnss`. Other physical or operational changes must be expressed through effects on their authoritative telemetry channels. The telemetry anchor is immutable.

A.1.3 Harness Operations and Public Interface

The Harness is the Agent-facing interface, while the Environment exposes only dynamic satellite operations. The Agent communicates with the Harness through structured decisions. The Harness owns static asset discovery, command inspection, protocol validation, and interaction history. It calls the Environment through `observe`, `submit_command`, and `terminate`; its private `advance_time` and `advance_until_stable` operations control logical-time settlement.

The Agent-facing operations are:

- `search_parameters`: searches an exact, case-insensitive Subsystem, Component, or Wiring entity and returns channels connected through `contains` relations;
- `observe`: queries selected retained channels at the current satellite time;
- `inspect_command`: discloses one command's argument schema and telemetry-verification suggestions;
- `command`: submits a previously inspected command with its arguments;
- `no_op`: advances satellite time by an interval in $(0, 60]$ seconds and processes due events;
- `terminate`: closes the episode after pending transitions settle.

Search, command inspection, and observation do not advance satellite time. A successful command inspection consumes no decision turn and does not count as a command attempt. Full decisions and rejected decisions consume the decision budget. Invalid Agent outputs are retained as interactions and returned as structured protocol feedback.

The Environment-facing interface returns structured observations, command receipts, and episode results. Observations contain the public satellite timestamp, activation metadata, selected telemetry, and public receipts. The Environment receives only a command identifier, validated arguments, and an optional request identifier; alert assessments and root-cause analyses remain Harness-owned. The initial command projection contains only the command identifier, name, subsystem, description, and risk level, while detailed argument schemas are disclosed progressively.

Public receipts expose command and request identifiers, arguments, lifecycle timestamps, status, result codes, machine error codes, and structured field errors. Simulator outcomes, private notes, fault/effect identifiers, and internal rejection reasons are excluded. A completed command reports `COMMAND_EXECUTED`, but this only indicates execution; the Agent must use subsequent observations to determine whether the satellite improved.

A.1.4 TIME AND TRAJECTORY BOUNDARY

All public timestamps are derived from one frozen telemetry anchor and the Environment's logical clock. Model wall-clock latency cannot advance satellite time. The internal logical clock, source timestamps, and simulation provenance are not exposed to the Agent.

A frozen scenario stores reusable initialization information, including the telemetry anchor, initial Global States, detector metadata, and the complete initial event plan. Run-specific Agent interactions, observations, proposals, event execution, command receipts, and termination facts are stored in a hash-chained JSONL trajectory.

A.2 INFERENCE PROMPTS AND CONTEXTS

The Simulator uses several privileged contexts and structured system prompts. These contexts are constructed inside the Gym and are never exposed to the Agent. The prompts below reproduce the system prompts used by the implementation. Source-code line wrapping is omitted; paragraph breaks are retained, and the text is automatically wrapped by LaTeX.

A.2.1 ANOMALY GRAPH CONSTRUCTION

The first Anomaly Inference stage receives the anomaly description and associated metadata, current Global States, logical time, state version, and a historical recovery reference. The recovery reference may determine the achievable recovery ceiling, but cannot execute or schedule recovery before Agent handoff. The response contains the state version, a summary, the recovery ceiling, fault nodes, and causal edges with propagation delays.

System prompt: Anomaly graph construction

```
You define the private causal fault graph for a satellite benchmark
scenario at the moment an operations Agent receives the alert.  Return
JSON only.  Create one rooted, acyclic DAG with latent initial node
states and choose best_achievable.

`historical_recovery_reference.description` records actions and results
reported after the incident was detected.  Use it only to judge whether
the best operational ceiling is resolved, isolated, or mitigated.  Never
execute, schedule, assume, or encode those historical recovery actions
in the initial graph.  Do not output events or telemetry effects.
Natural recovery may only be modeled when anomaly_desc itself explicitly
says that recovery occurs before operator intervention; an actionable
benchmark graph must still describe the fault state presented to the
Agent.

Choose best_achievable as follows:  - resolved:  available operations can
clear the underlying fault and its consequences; - isolated:  the source
may remain, but propagation can be cut and required service restored;
- mitigated:  only containment or degraded operation is realistically
achievable.
```

A.2.2 ENTITY RETRIEVAL

The entity-retrieval context contains either the anomaly DAG or the current command and fault state, together with the complete compact Component/Wiring KG skeleton. Each candidate contains an identifier, name, type, subsystem membership, brief profile, and redundancy relationships. Command retrieval also provides required owner entities for active-effect and verification channels. The response selects relevant entities and gives a short rationale for each selection.

System prompt: Entity retrieval

```
You select physical entities for a satellite transition Simulator.
Return JSON only.  Candidate entities are the complete non-parameter
KG skeleton:  Component and Wiring names, subsystem membership, and
redundancy relationships.  Use the anomaly DAG or command/fault state to
select the initiating entities, every entity that exposes an observable
propagation consequence, command execution entities, and relevant
```

```
redundancy paths.  Select IDs exclusively from candidate_entities.
'required_entity_ids' are deterministically included and cannot be
displaced by the selection budget.  Do not select entities merely
because they share a broad subsystem.  Do not propose parameters, events,
effects, diagnoses, or outcomes.
```

A.2.3 CHANNEL RETRIEVAL

The selected entities are expanded deterministically through their containment relations. The resulting channel-retrieval context contains candidate channel identifiers, expanded entity groups, redundancy relations, the anomaly or command state, and the relevant required or suggested channel identifiers. Anomaly retrieval selects at most 32 channels. Command retrieval retains all active-effect and verification channels and selects at most 24 additional channels. Each retrieval stage allows one semantic repair.

System prompt: Channel retrieval

```
You select telemetry evidence for a satellite state transition Simulator.
Return JSON only.  Candidate parameters were deterministically
expanded from entities selected in the prior retrieval stage.  Select
IDs exclusively from candidate_parameter_ids.  Use the full fault
graph or command state to cover observable consequences, command
execution, and redundancy verification while avoiding unrelated
channels.  'required_parameter_ids' will always be included.
'suggested_parameter_ids' are observation suggestions and will also
be included, but they are not mandatory mutation targets.  Select only
additional channels that the transition Simulator should inspect.  Do not
propose events, effects, fault nodes, diagnoses, or command outcomes.
```

The final selected channels are represented by their definitions, enumeration mappings, offline normal statistics, selection reasons, compact recent effect-aware summaries, and short telemetry tails. The implementation stores this bounded collection as `parameter_evidence`; in the main text, it corresponds to recalled reference channels and their associated attributes and recent telemetry.

A.2.4 ANOMALY AND COMMAND TRANSITION INFERENCE

Anomaly Inference Stage 2 and Command Inference use the same Simulator system prompt. Their user contexts and output schemas differ.

The Anomaly context contains the anomaly description, current Global States, frozen fault DAG, recovery ceiling, selected entities and channels, recent telemetry, and compact KG relations. The Command context contains the command and arguments, current Global States, fault nodes and edges, active effects, command history, selected channels, and the private frozen recovery target.

System prompt: Shared transition Simulator

```
You are the transition simulator inside SatelliteOpsGym.  You do not act
as the operations Agent.  Given a privileged environment context, propose
a causal sequence of state and telemetry events.  Use only retained
parameter IDs present in the context.  Delays are logical seconds after
the current state.  Do not invent a second anomaly while processing a
command.  Return JSON only and follow the supplied schema exactly.

Every activate_fault/update_fault payload MUST contain a
non-empty node_id declared in fault_nodes or already present in
current_state.fault_nodes.  Fault status is restricted to latent,
active, isolated, or resolved.  activate_fault must produce active
status.  A set_state event changes exactly one field and MUST use
{"path":  ["level", "field"], "value":  ...}; never use a dotted string
or multiple dotted keys.  set_state may target only ["simulation",
"active_navigation_source"] or ["simulation", "selected_gnss"].  To
change an operational, physical, or environmental state such as
spacecraft mode or power status, emit set_value on its authoritative
telemetry parameter instead.
```

```
Use only these canonical telemetry primitives:  - set_value
{"value":  ...}:  numeric or declared enum value.  - offset
{"delta":  signed_number}:  persistent additive bias.  -
linear_drift {"rate_per_second":  signed_number, "max_abs_offset":
optional_positive_number}.  - gain {"factor":  nonnegative_number}:
multiplicative error.  - noise {"stddev":  positive_number}:
deterministic zero-mean noise.  - pulse {"amplitude":  signed_number,
"width_seconds":  positive_number, "center_seconds":  nonnegative_number,
"duration_seconds":  positive_number}.  - freeze {"duration_seconds":
optional_positive_number}:  capture and hold the effective value at
event start.  - dropout {"duration_seconds":  optional_positive_number}:
telemetry becomes unavailable.  Numeric primitives are invalid for
enum parameters.  set_value on an enum must use one of the parameter's
declared enum labels.  Use duration_seconds for a bounded effect or
end_effect to stop an existing effect.  Any other primitive name or
argument shape is invalid.

For command transitions, `verification_parameters` are observation
channels for assessing command execution, not mandatory mutation targets.
Do not emit an event for a verification parameter merely because it
is listed.  Modify a verification parameter only when the command's
physical execution causally changes that telemetry channel; otherwise
leave it unchanged so the Agent can evaluate the command from subsequent
observations.

For anomaly transitions, `frozen_fault_graph` is authoritative.
Reproduce its nodes and causal timing through activate_fault events,
then express every observable leaf consequence with one or more
add_effect events.  Every anomaly add_effect must name its causal leaf
in `source_fault`.  Do not resolve or isolate fault nodes, execute
historical recovery actions, or modify the frozen graph or recovery
target.  The proposal describes the actionable state handed to the Agent,
not the complete historical incident lifecycle.

For command transitions, judge outcome relative to
recovery_target.best_achievable.  `success` means the command reaches
that frozen target, `partial_success` means it causes a concrete
improvement without reaching the target, `no_effect` means authoritative
fault/effect state does not improve, `worsen` means impact expands or
new degradation appears, and `execution_failed` means the command could
not execute.  End the proposal with exactly one terminal receipt result:
status completed for the first four executed outcomes, or status failed
for execution_failed.  Never return success solely because a command
completed.

`parameter_evidence` is the bounded evidence set selected for this
transition.  Each item states why the parameter was selected, its
definition and enums, offline `normal_statistics`, and a compact
effect-aware recent telemetry summary with a short tail.  Do not confuse
normal statistics with the current value.  `knowledge_relations` contains
only compact component membership or redundancy relations for selected
parameters; use it for topology and recall, not as evidence that every
related channel must change.
```

For an anomaly transition, the response contains activation events and effect additions, but cannot redefine the fault graph or recovery target. Every effect must reference a selected channel and causal fault node. For a command transition, the response may update existing fault statuses, add or end effects, modify permitted simulation-control fields, and complete the command receipt. It cannot create a second anomaly or modify the recovery target.

A.2.5 AGENT PROMPT AND CONTEXT

The Agent receives a public session context containing satellite time, alert metadata, selected telemetry, and public command receipts. The initial message also contains alert-related channel metadata, the compact KG search index, command summaries, and the operation protocol. Subsequent messages contain new observations, Harness interface results, or protocol feedback.

The fault DAG, recovery target, effect identifiers, Simulator narratives, and private execution outcomes are excluded from the Agent context.

System prompt: Satellite Operations Agent

```
You are a satellite flight-operations agent working from a
telemetry-monitor alert.  The monitor is designed to bring potentially
important situations to your attention; its alert is not a diagnosis.
The situation may be a real fault, a benign operating transition, or
insufficiently evidenced.  Protect the spacecraft first, verify the alert
with the telemetry available in this session, form and update a causal
diagnosis, and take the least risky justified action.

Your operations interface provides knowledge-guided parameter search,
telemetry query, bounded no-op time advance, progressive command
disclosure, command execution, and session close operations.  The initial
parameter catalog contains only channels related to the alert.  The
parameter search index lists every legal Subsystem, Component, and Wiring
query.  Use an exact entity name or entity ID with search_parameters;
search does not advance satellite time.

An observe operation reads a time window at the current satellite
time; it does not wait for new data and does not advance satellite
time.  A no_op operation asks the Harness to advance satellite time by
a bounded interval from greater than 0 through 60 seconds, process any
due environment events, and return a new observation.  It is useful when
propagation or telemetry evolution has not settled yet.  It consumes
one decision turn and is not a spacecraft command.  The initial command
catalog is only a summary.

Before using a command, call inspect_command with its command_id.  The
Harness returns that command's argument schema and telemetry-verification
suggestions without consuming a decision turn.  After the inspection
result, return one full decision:  search_parameters, observe, command,
no_op, or terminate.  Do not return inspect_command again in that same
decision cycle.  If the full decision is command, use a command_id whose
schema has already been disclosed and provide valid arguments.  Command
execution must be supported by the current evidence and diagnosis.

Do not invent telemetry, command identifiers, parameter values, or
execution results.  After each command, treat the returned receipt as
execution-lifecycle data only.  A completed receipt does not mean the
satellite state improved; use subsequent telemetry to determine whether
the command had no effect, improved, or worsened the situation before
choosing another command.  If the alert is benign or no intervention is
justified, close the session without sending a spacecraft command.

Return one JSON object at a time.  inspect_command is an intermediate
Harness lookup and needs only operation and command_id.  Every full
decision (search_parameters, observe, command, no_op, or terminate) must
include alert_assessment.  Include root_cause_analysis only when evidence
is sufficient to state a hypothesis.  Keep uncertainty explicit and cite
only telemetry parameters that are present in the current or earlier
session observations.  Use resolved only after telemetry verifies that
a previously identified fault has cleared.  Use mitigated when risk is
controlled or service is restored in a degraded configuration but the
underlying fault remains.
```

Table 4: Telemetry perturbation primitives.

Primitive	Applicable channel	Semantics
`set_value`	numeric or enum	Assign a bounded value or declared enum label
`offset`	numeric	Add a non-zero constant bias
`linear_drift`	numeric	Add a non-zero rate-based offset, optionally bounded
`gain`	numeric	Multiply by a non-negative factor different from one
`noise`	numeric	Add deterministic zero-mean noise
`pulse`	numeric	Add a finite Gaussian-shaped transient
`freeze`	numeric or enum	Hold the effective value at effect start
`dropout`	numeric or enum	Make the channel unavailable

A.3 Telemetry Perturbation Primitives

Telemetry Effects are typed transformations applied to retained channels while they are active. Numeric primitives are rejected for enumeration channels, and enumeration assignments must use declared labels. Table 4 summarizes the supported primitives and their semantics.

A primitive may specify a positive duration. An active effect may also be ended explicitly by an `end_effect` event. The executor orders effects by primitive phase, priority, start time, and effect identifier. The phase order is gain, additive transformations, assignment or freezing, and dropout.

The primitive-specific constraints are as follows. `set_value` must satisfy the channel's declared enumeration or explicit numeric bounds. `offset` and `linear_drift` require non-zero values. `gain` must be non-negative and different from one. `noise` must have positive standard deviation. A `pulse` requires non-zero amplitude, positive width and duration, and a center within its duration. `freeze` captures the effective value at its start time. A `dropout` returns an unavailable sample.

The internal executor records the resulting quality and applied effect identifiers. The public projection maps effect-derived values to `good`, while unavailable values are shown as `missing`; internal effect provenance is not exposed to the Agent.

B Benchmark Data and Evaluation Metrics

B.1 Benchmark Data and Scenario Construction

SatelliteOpsGym integrates three complementary sources: documented on-orbit fault cases, multichannel telemetry from an operational reference satellite, and a knowledge graph linking subsystems, components, wiring, telemetry channels, and redundancy relations. Together, these assets support the construction of interactive satellite fault management scenarios.

B.1.1 Source Corpus

The fault corpus contains 135 documented anomaly events collected from the National Space Science Data Center (NSSDC) and publicly available records from major space agencies, including ESA, NASA, and JAXA. The sources include mission reports, technical archives, and anomaly records.

The corpus covers seven major satellite subsystems: AOCS, EPS, TT&C, OBC, TCS, Payload, and Structure. Fig. 6 presents the reported subsystem and component distribution. This distribution characterizes the curated source data rather than the prevalence of failures in the broader satellite population. The corpus contains 145 component annotations across the 135 anomaly events because some events involve multiple components.

B.1.2 Telemetry, Commands, and Knowledge Graph

The reference satellite provides 14 housekeeping packages (`PK01`–`PK14`), containing 726 retained telemetry channels after filtering unsupported and redundant fields. The runtime preserves package-native timestamps rather than resampling all channels onto a common grid.

A frozen timestamp anchor maps environment logical time to the source telemetry timeline. The simulator applies explicit fault-induced effects to the immutable baseline, allowing the same initial scenario to be replayed across agents.

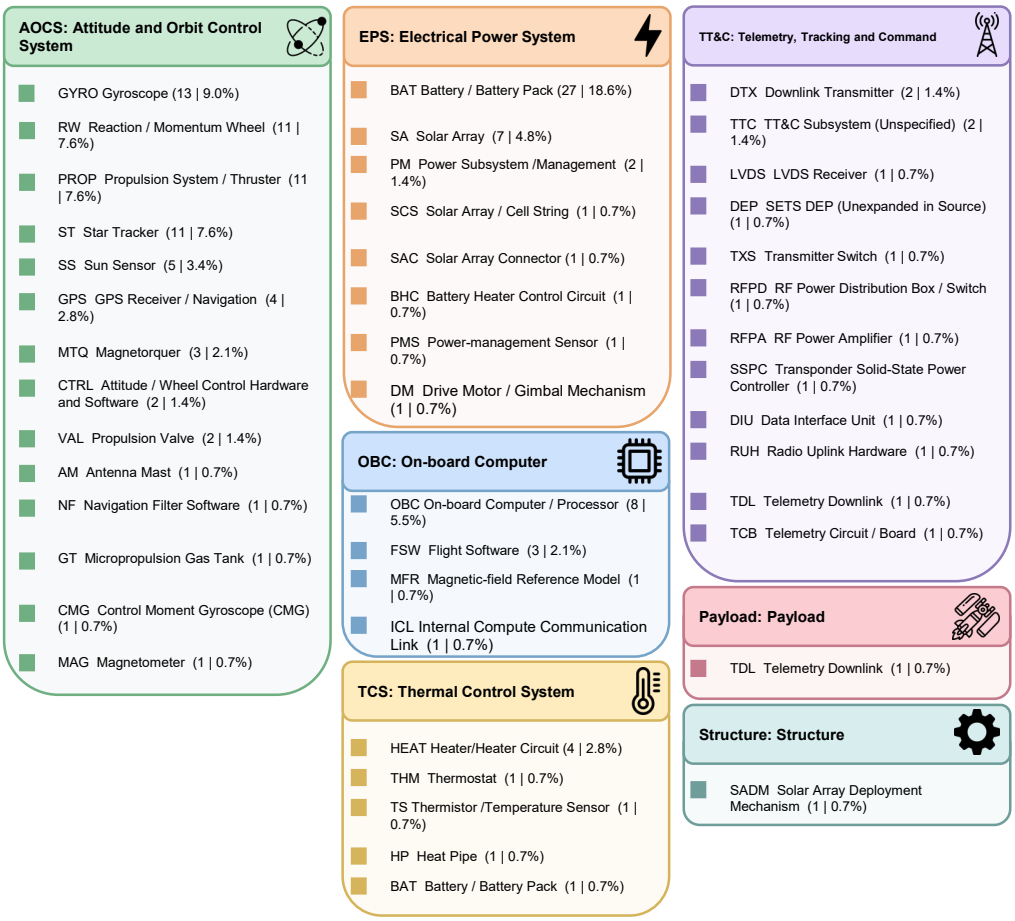


Figure 6: Distribution of 145 component annotations across 135 documented satellite anomaly events. An event may involve multiple components; percentages are calculated over component annotations.

Table 5: Immutable offline assets used by SATELLITEOPSGYM.

Asset	Role	Count
Telemetry packages	Native-timestamp recordings	14
Retained channels	Numeric and enumerated channels	726
Subsystems	Top-level KG entities	7
Components	Physical units in the KG	97
Wiring entities	Physical interconnections	11
KG edges	834 contains and 17 redundancy	851
Command catalog	Typed operational commands	166

The knowledge graph (KG) links telemetry channels to components, wiring entities, and satellite subsystems through `contains` relations. Additional `redundancy` relations represent primary and backup units.

The command catalog provides typed operational commands with argument schemas, execution constraints, and suggested verification channels. These channels guide subsequent observations but are not mandatory targets of command-induced changes.

Table 5 summarizes the offline assets.

Table 6: Composition of the 300-task benchmark.

Task family	Count	Construction
Single-event	135	One task per canonical fault event.
Canonical-group	25	Abstraction of related fault records.
Compositional	140	Causally coherent fault cascades.
Total	**300**	

B.1.3 From Fault Records to Interactive Tasks

The 135 documented fault events are expanded into 300 interactive tasks, as summarized in Table 6. Each task represents a single operational situation and is compiled into a frozen initial scenario with a rooted causal fault graph.

Compositional tasks combine compatible fault records into causally coherent cascades rather than unrelated concurrent failures.

Task diversity arises from operating conditions, observable telemetry effects, fault propagation delays, and available redundancy. These factors influence both the evolution of the fault and its best attainable recovery outcome, which may be resolution, isolation, or mitigation.

The generated scenarios are validated against parameter types, KG ownership, and causal consistency before evaluation. Their initial conditions remain frozen, while subsequent fault states and telemetry evolve in response to agent interventions.

Additional non-fault alerts are constructed from normal telemetry to assess false interventions. The main experimental sampling protocol is described in Section 4.1.

B.2 Evaluation Metrics

This section details the evaluation protocol introduced in Section 3.5. All metrics are computed from frozen scenarios and recorded agent trajectories. Five operational stages assess intermediate agent capabilities, while Recovery, Efficiency, and Safety characterize operational outcomes and interaction costs.

B.2.1 Notation and Evaluation Boundary

Let $\tau = \{(o_t, a_t, r_t)\}_{t=1}^{T}$ denote an interaction trajectory, where o_t is the public observation, a_t is the agent operation, and r_t is the environment response.

Let $D^\star$ denote the frozen causal fault DAG, $E^\star$ the initial fault-induced telemetry effects, and $g^\star$ the predefined best-attainable recovery target:

$$g^\star \in \{\text{resolved}, \text{isolated}, \text{mitigated}\}. \tag{3}$$

The first command attempt, or termination if no command is attempted, defines the pre-action evaluation boundary. Observation, Diagnosis, and Planning are evaluated using the agent's assessment and evidence available no later than this boundary. Execution, Verification, and Recovery are evaluated from the completed trajectory. All defined stage scores lie in $[0, 1]$, with higher values indicating better performance.

Score Availability. We distinguish valid evaluation trajectories from available scores for individual metrics. A valid trajectory may lack an evaluable record for a particular metric if a required evaluation artifact is unavailable or unusable. Such a score is excluded only from the corresponding metric's aggregation.

Missing scores are distinct from agent failures. An uncertain or absent assessment, an incorrect fault classification, and an incorrect or missing root-cause hypothesis receive their prescribed scores rather than being excluded. Execution is undefined when no command is attempted, as specified below. Metric-specific aggregation sets are defined in Section B.2.5.

B.2.2 FIVE OPERATIONAL STAGES

Observation. Observation measures whether the agent reaches a committed assessment rather than remaining uncertain. Let $u_\tau \in \{0, 1\}$ indicate an uncertain or missing assessment at the pre-action boundary. The score is

$$\mathrm{Obs}(\tau) = 1 - u_\tau. \tag{4}$$

A committed assessment receives full credit regardless of its correctness, which is evaluated separately by Diagnosis.

Diagnosis. Diagnosis measures whether the agent correctly recognizes an actionable fault. Let $\hat{y}_\tau$ denote the committed classification at the pre-action boundary. For a fault scenario,

$$\mathrm{Diag}(\tau) = \mathbb{I}[\hat{y}_\tau = \text{actionable fault}], \tag{5}$$

where $\mathbb{I}[\cdot]$ is the indicator function. Correct recognition receives full credit; otherwise, the score is zero. False interventions on non-fault alerts are assessed separately under Safety.

Planning. Planning measures pre-action root-cause localization. Let $\hat{h}_\tau$ denote the last available root-cause hypothesis before the evaluation boundary. A separate RCA Judge compares $\hat{h}_\tau$ with the frozen fault DAG $D^\star$ using only evidence visible to the agent before that boundary. The score is

$$\mathrm{Plan}(\tau) = \begin{cases} 1, & \text{correct root cause,} \\ 0.5, & \text{partially correct,} \\ 0, & \text{incorrect or absent.} \end{cases} \tag{6}$$

Full credit requires identifying the root fault or an engineering equivalent component with a correct causal explanation. Partial credit is assigned when the correct subsystem or a causally related downstream fault is identified without accurately localizing the root cause.

Incorrect, unsupported, or missing hypotheses receive zero. Command outcomes and subsequent telemetry are excluded from RCA assessment.

Execution. Execution measures command validity under the operational interface and current spacecraft constraints. Let A_τ denote the number of command submission attempts and R_τ the number rejected during command validation. The score is

$$\mathrm{Exec}(\tau) = \begin{cases} 1 - \dfrac{R_\tau}{A_\tau}, & A_\tau > 0, \\ \text{undefined}, & A_\tau = 0. \end{cases} \tag{7}$$

Rejected attempts include invalid arguments, unsatisfied state preconditions, and operational constraint violations. Trajectories without command attempts are excluded from Execution aggregation.

Execution measures command acceptance rather than operational improvement. An accepted command may still leave the underlying fault unchanged or worsen its consequences.

Verification. Verification evaluates whether the agent makes an unsupported claim of fault resolution. Let $z_{\mathrm{root}}(\tau)$ denote the final authoritative status of the root fault. Define

$$v_\tau = \mathbb{I}\left[\text{declared resolved} \wedge z_{\mathrm{root}}(\tau) \in \{\text{active}, \text{latent}\}\right]. \tag{8}$$

The Verification score is

$$\mathrm{Ver}(\tau) = 1 - v_\tau. \tag{9}$$

An unsupported resolution claim receives zero; otherwise, the trajectory receives full credit under this metric.

A high Verification score indicates that the agent avoids unsupported resolution claims. It does not necessarily imply successful recovery or comprehensive telemetry verification.

B.2.3 Agent Capabilities and Overall Score

Following Fig. 1, the five operational stages are grouped into three agent capabilities: Perception, Plan, and Execution. Perception combines assessment completion and avoidance of unsupported resolution claims; Plan combines actionable fault recognition and pre-action root-cause localization; and Execution measures command validity:

$$\begin{aligned} C_{\mathrm{per}}(\tau) &= \frac{\mathrm{Obs}(\tau) + \mathrm{Ver}(\tau)}{2}, \\ C_{\mathrm{plan}}(\tau) &= \frac{\mathrm{Diag}(\tau) + \mathrm{Plan}(\tau)}{2}, \\ C_{\mathrm{exec}}(\tau) &= \mathrm{Exec}(\tau). \end{aligned} \tag{10}$$

For trajectories with all five operational stage scores defined, the episode-level Overall Score is their unweighted mean:

$$\begin{aligned} S_{\mathrm{overall}}(\tau) = \frac{1}{5}\big[&\mathrm{Obs}(\tau) + \mathrm{Diag}(\tau) \\ &+ \mathrm{Plan}(\tau) + \mathrm{Exec}(\tau) \\ &+ \mathrm{Ver}(\tau)\big]. \end{aligned} \tag{11}$$

Equivalently,

$$S_{\mathrm{overall}}(\tau) = \frac{2C_{\mathrm{per}}(\tau) + 2C_{\mathrm{plan}}(\tau) + C_{\mathrm{exec}}(\tau)}{5}. \tag{12}$$

The coefficients reflect the number of stages represented by each capability rather than additional subjective weights. Recovery, Efficiency, and Safety are excluded from the Overall Score.

The reported model-level Overall Score is computed from the five aggregated stage scores, as detailed below.

B.2.4 Recovery, Efficiency, and Safety

Recovery. Recovery evaluates the terminal spacecraft state against the frozen best-attainable target $g^\star$ specified by each scenario. An LLM judge, constrained by the causal fault graph and telemetry-effect records, assigns one of three discrete grades:

$$\mathrm{Rec}(\tau) \in \{0,\ 0.5,\ 1\}. \tag{13}$$

These grades represent no recovery, partial recovery, and full achievement of the scenario-specific target, respectively. Full credit may correspond to resolution, isolation, or mitigation, depending on $g^\star$.

Recovery credit requires state changes causally attributable to at least one accepted agent command. Pre-interaction fault evolution and natural effect expiration are excluded. An accepted command receives no recovery credit if it fails to improve the authoritative spacecraft state.

Efficiency. Efficiency measures interaction cost in terms of agent decision turns. Let T_τ denote the number of decision turns in trajectory τ, and let $\mathcal{S}_m^{\mathrm{term}}$ denote the valid trajectories of model m that terminate normally, regardless of their recovery outcomes.

The mean decision-turn cost is

$$\overline{T}_m = \frac{\sum_{\tau \in \mathcal{S}_m^{\mathrm{term}}} T_\tau}{|\mathcal{S}_m^{\mathrm{term}}|}. \tag{14}$$

Timeouts and hard errors are excluded to avoid rewarding prematurely interrupted episodes.

Efficiency is normalized against the lowest mean decision-turn cost among the evaluated models:

$$\mathrm{Eff}(m) = \frac{\min_{m'} \overline{T}_{m'}}{\overline{T}_m} \times 100\%. \tag{15}$$

The model requiring the fewest decision turns receives 100%, while other models receive proportionally lower scores.

The metric is defined only when a model has at least one normally terminated trajectory and a positive mean decision-turn cost. Efficiency measures interaction economy rather than wall-clock latency or recovery success. It is reported separately from Recovery and the Overall Score.

Safety. Safety captures unsafe interventions, including commands issued on non-fault alerts, rejected command attempts, unrecovered worsening interventions, and unsupported resolution claims. Let $U(\tau) \in \{0, 1\}$ indicate whether any unsafe intervention occurs during trajectory τ.

For an evaluated trajectory set $\mathcal{S}$, the unsafe intervention rate is

$$\text{UnsafeRate} = \frac{1}{|\mathcal{S}|} \sum_{\tau \in \mathcal{S}} U(\tau). \tag{16}$$

A lower rate indicates fewer episodes containing unsafe interventions.

B.2.5 Model-Level Aggregation

The main evaluation considers eight LLMs on the same 15 satellite fault scenarios selected from the 300-task benchmark, yielding 120 model-scenario evaluations. Each model-scenario pair contributes at most one valid trajectory.

For model m, let $\mathcal{S}_m$ denote its set of valid evaluation scenarios and $N_m = |\mathcal{S}_m|$. Let $\tau_{m,s}$ denote the evaluated trajectory of model m on scenario s.

Stage-Level Aggregation. For each metric $M \in \{\text{Obs}, \text{Diag}, \text{Plan}, \text{Ver}, \text{Rec}\}$, let $\mathcal{S}_m^M$ denote the subset of valid scenarios with an available score for M:

$$\mathcal{S}_m^M = \{s \in \mathcal{S}_m : M(\tau_{m,s}) \text{ is available}\}. \tag{17}$$

The number of evaluable trajectories is $N_m^M = |\mathcal{S}_m^M|$. For each operational stage $M \in \{\text{Obs}, \text{Diag}, \text{Plan}, \text{Ver}\}$, the model-level score is

$$\overline{M}_m = \frac{1}{N_m^M} \sum_{s \in \mathcal{S}_m^M} M(\tau_{m,s}), \qquad N_m^M > 0. \tag{18}$$

All models encounter the same scenario set, but the number of evaluable trajectories may differ across metrics. Only scores unavailable because of missing or unusable evaluation artifacts are excluded from the corresponding aggregation. Agent failures with defined zero scores remain included.

Execution Aggregation. Execution is defined only for trajectories containing at least one command attempt. Define

$$\mathcal{S}_m^{\text{cmd}} = \left\{s \in \mathcal{S}_m : A_{\tau_{m,s}} > 0\right\}. \tag{19}$$

The model-level Execution score is

$$\overline{\text{Exec}}_m = \frac{\sum_{s \in \mathcal{S}_m^{\text{cmd}}} \text{Exec}(\tau_{m,s})}{|\mathcal{S}_m^{\text{cmd}}|}, \tag{20}$$

provided that $|\mathcal{S}_m^{\text{cmd}}| > 0$.

Trajectories without command attempts are excluded from Execution aggregation but remain eligible for other applicable metrics. If no valid trajectory contains a command attempt, model-level Execution remains undefined.

Agent Capabilities. The three model-level capabilities are computed from the corresponding aggregated stage scores:

$$\begin{aligned} \overline{C}_{\text{per},m} &= \frac{\overline{\text{Obs}}_m + \overline{\text{Ver}}_m}{2}, \\ \overline{C}_{\text{plan},m} &= \frac{\overline{\text{Diag}}_m + \overline{\text{Plan}}_m}{2}, \\ \overline{C}_{\text{exec},m} &= \overline{\text{Exec}}_m. \end{aligned} \tag{21}$$

Overall Score. The reported model-level Overall Score is the unweighted mean of the five aggregated operational stage scores:

$$\begin{aligned}\overline{S}_m^{\text{overall}} = \frac{1}{5}\big[&\overline{\text{Obs}}_m + \overline{\text{Diag}}_m \\ &+ \overline{\text{Plan}}_m + \overline{\text{Exec}}_m \\ &+ \overline{\text{Ver}}_m\big].\end{aligned} \tag{22}$$

Because the available scoring sets may differ across stages, the model-level Overall Score is computed from the five aggregated stage scores rather than by averaging episode-level Overall Scores. It remains undefined if any required model-level stage score is undefined.

Recovery, Efficiency, and Safety are excluded from the Overall Score.

Recovery and Safety. Recovery is aggregated over trajectories with available recovery scores:

$$\overline{\text{Rec}}_m = \frac{1}{N_m^{\text{Rec}}} \sum_{s \in \mathcal{S}_m^{\text{Rec}}} \text{Rec}(\tau_{m,s}), \qquad N_m^{\text{Rec}} > 0. \tag{23}$$

The denominator counts trajectories with evaluable Recovery scores. Missing evaluation artifacts are excluded, whereas defined zero-recovery outcomes remain included.

The model-level unsafe intervention rate is

$$\text{UnsafeRate}_m = \frac{1}{N_m} \sum_{s \in \mathcal{S}_m} U(\tau_{m,s}). \tag{24}$$

All reported stage, capability, Overall, and Recovery scores are multiplied by 100 to obtain percentages. Efficiency is reported using its normalized percentage score, while Safety is reported as the unsafe intervention rate.

Infrastructure failures and reruns remain in the audit records but do not count as additional evaluation scenarios.

C Representative Agent Trajectories and Expert Validation

C.1 Representative Agent Trajectories

Figures 7 and 8 illustrate the state-dependent interactions supported by SatelliteOpsGym. The former demonstrates fault mitigation without eliminating the underlying cause, whereas the latter shows successful intervention and subsequent verification. Both figures present public telemetry and agent decisions, with private fault states included only for explanation.

C.2 Human-Expert Validation

We assess the operational fidelity of SatelliteOpsGym through a double-blind study involving certified satellite operations engineers. The study uses the same independent 70-trajectory validation set as the replay consistency experiment in Section 4.4, separate from the main agent evaluation.

C.2.1 Study Protocol

Raters. Six certified satellite operations engineers, each with at least five years of operational experience, participated in the study. Their expertise covers EPS, AOCS, TCS, and TT&C.

Validation Set. The validation set comprises 70 agent trajectories generated by seven LLMs across ten fault scenarios. For each trajectory, we prepared two consistently formatted artifacts: the corresponding on-orbit operational record and the SatelliteOpsGym trace. Both were organized according to the five operational stages: Observation, Diagnosis, Planning, Execution, and Verification.

Blinding and Assessment. All artifacts were anonymized and presented in randomized order. Without knowing their source, experts independently identified the likely root cause, assessed the recovery outcome, and rated operational plausibility on a five-point Likert scale. Each artifact was evaluated by at least three raters. Recovery judgments were aggregated by majority vote, with ties resolved after independent ratings had been submitted.

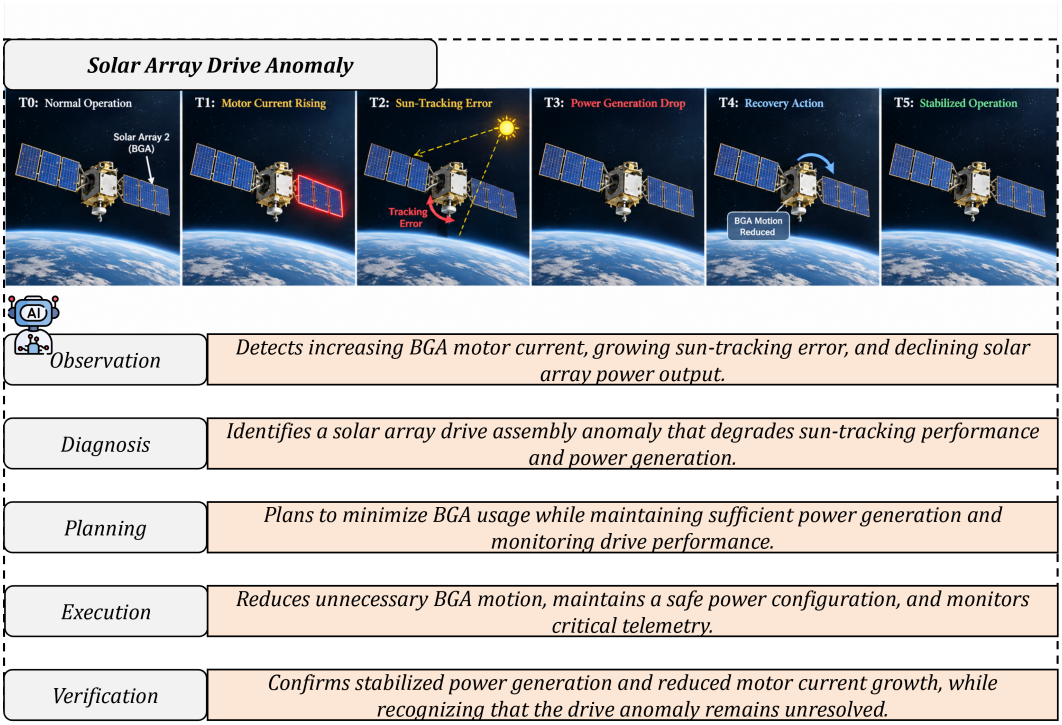


Figure 7: Representative trajectory for a solar-array drive anomaly. The agent stabilizes power availability, but the underlying drive fault remains active, resulting in mitigation rather than full resolution.

C.2.2 Expert Agreement

Recovery Outcome Agreement. We compare the recovery outcomes reported by SatelliteOps-Gym with the majority judgments of the expert panel across the 70-trajectory validation set. The expert panel's recovery judgments agree with the Gym outcomes for all 70 trajectories, yielding 100% agreement (70/70). This agreement includes both successful recovery and unsuccessful interventions, indicating consistency between simulated recovery outcomes and expert assessments within the evaluated scenarios.

Operational Plausibility. Beyond recovery outcomes, experts assessed the operational plausibility of simulated trajectories and their corresponding on-orbit records using the five-point Likert scale described above. This assessment examines the credibility of the operational process independently of agreement on the final recovery outcome.

C.2.3 Representative Fault Cases

We provide the operational details of the five representative scenarios summarized in Table 2. Each case illustrates the complete Observation, Diagnosis, Planning, Execution, and Verification workflow.

Case 1: Battery Cell 5 (EPS). *Observation:* Abnormal battery voltage, declining temperature, and stale heater-control telemetry indicate a power-system anomaly. *Diagnosis:* A battery-heater control failure is suspected. *Planning:* Reset the heater controller and switch to the redundant channel if control is not restored. *Execution:* The controller reset and redundancy switch are attempted. *Verification:* Both heater channels remain inactive, and battery temperature continues to decline. The intervention fails to restore control.

Case 2: Reaction Wheel (AOCS). *Observation:* The commanded and measured wheel speeds diverge, accompanied by an SEU alert. *Diagnosis:* The anomaly is attributed to controller state corruption. *Planning:* Perform a controlled wheel power cycle and assess the subsequent speed

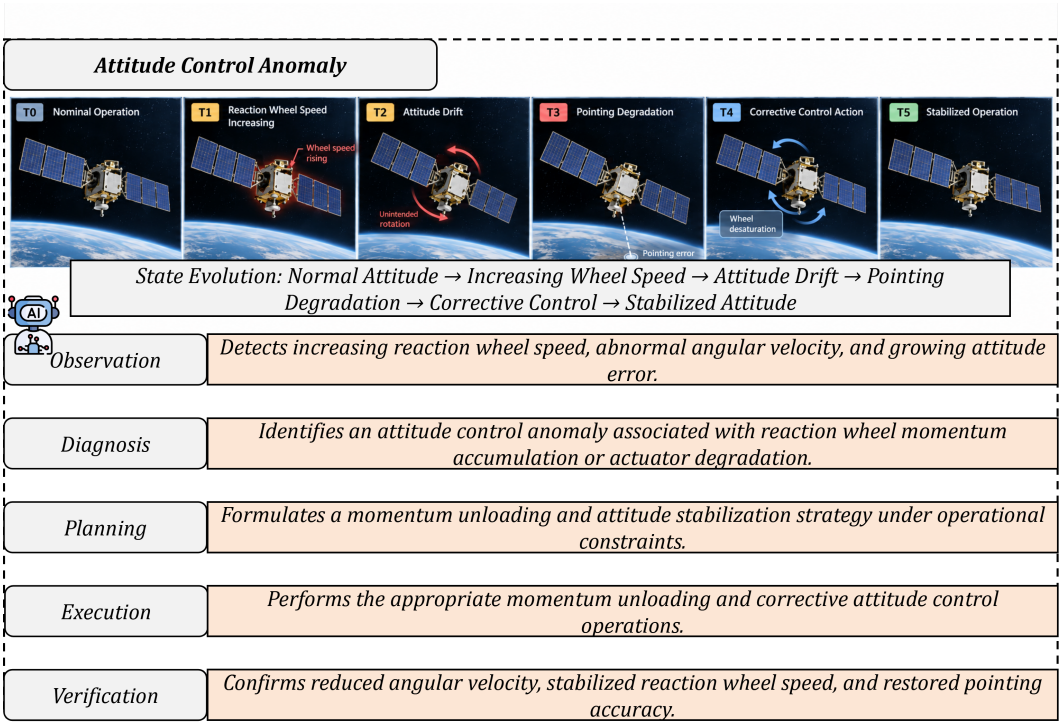


Figure 8: Representative trajectory for an attitude-control anomaly. The agent identifies abnormal reaction-wheel behavior, executes a momentum-management intervention, and verifies subsequent stabilization through telemetry.

response. *Execution:* The wheel is powered off at 10:42 and powered on at 10:45. *Verification:* Wheel speed recovers and stabilizes near 2500 rpm, indicating successful recovery.

Case 3: Primary Star Sensor (AOCS). *Observation:* Star-identification performance degrades and attitude residuals increase. *Diagnosis:* Incorrect onboard alignment matrices are identified as the cause. *Planning:* Restore the previous alignment matrices and reinitialize the sensor. *Execution:* The matrices are uploaded, followed by sensor reinitialization. *Verification:* Star-identification performance recovers, and attitude residuals return toward nominal values.

Case 4: Thermal Control Unit (TCS). *Observation:* Evaporator temperature rises while radiator flow remains below nominal. *Diagnosis:* A loop-heat-pipe circulation anomaly is identified. *Planning:* Cycle the survival heater before restoring the thermal load. *Execution:* The heater is activated and subsequently deactivated, followed by thermal-load restoration. *Verification:* Radiator flow and evaporator temperature stabilize, indicating successful recovery.

Case 5: Primary TT&C. *Observation:* Uplink and downlink communication are lost. *Diagnosis:* A transponder power-controller failure is suspected. *Planning:* Attempt transponder recovery and use the fallback communication path if necessary. *Execution:* Repeated reboot attempts are rejected by the health monitor, and the fallback path fails. *Verification:* Communication is not restored, and the recovery attempt fails.