What Matters for Designing Reliable Systems: A New Taxonomy of Failure Cause-Condition-Effect Relationships Based on AUV Missions

Christopher Thierauf[1] Matthias Scheutz[2]

***Abstract*— Failures of autonomous robots are inevitable and the way to address them is not solely by design, but by way of methods in the robot's control architecture that are aware of the types of perturbations, their causes, conditions, and effects on the operation of the platform. In this paper, we assemble and analyze reports from approximately 450 deployments of the Sentry autonomous underwater vehicle (AUV) to categorize the different failure cases in an effort to develop a new failure taxonomy. The goal for the taxonomy is to provide robot architecture designers with a comprehensive framework that can serve as the basis for developing methods for monitoring and appropriately responding to the different types of faults that occur in actual deployments.**

I. Introduction

Autonomous robots are increasingly deployed in uncertain high-complexity environments where different types of failures are inevitable, ranging from mechanical issues like hardware wear and tear, to software problems like crashes or memory leaks, to sensor or processing problems like poor localization, to the impact of unforeseen environmental changes on the platform. While failures can also occur in tightly constrained domains like warehouse robots, or less constrained domains such as healthcare or home settings, the difference with autonomous field robots operating in remote environments such as under water or in space is that they usually cannot rely on an operator to intervene and help them recover from a fault. And while some recovery strategies are often implemented in these systems, they are often hard-coded responses to anticipated failure scenarios considered by their developers [1], leaving out ways to respond to unforeseen perturbations.

A recent taxonomy of perturbations distinguishes different types along six dimensions where each perturbation can either be (1) transient or permanent, (2) impactless or impactful, (3) predictable or unpredictable, (4) detectable or undetectable, (5) avoidable or unavoidable, and (6) mitigable or unmitigable [2]. Depending on the particular type of perturbation the response of an autonomous agent might take a different form. For example, if a perturbation is *transient*, *impactful*, *unpredictable*, *undetectable*, *unavoidable*, and *unmitigable* such as a transient communication failure between a surface vessel and an underwater robot, the simple strategy is to wait and may continue moving to get out of the zone that caused the interruption. If, on the other hand, the perturbation is predictable and avoidable, then the robot could make a plan to avoid it (such as a UAV drone flying through a thunderstorm). To make a robot as resilient to as many perturbations as possible, it needs to have extensive monitoring capabilities as well as explicit awareness of perturbation causes, conditions under which they occur, and the effects that they have on its operation. For this purpose, a *comprehensive taxonomy of perturbation causes, conditions, and effects* is a critical prerequisite for implementing detection and mitigation methods as well as recovery mechanisms in a robot control architecture.

Existing failure taxonomies, however, are too narrowly focused on hardware or software diagnostics, rather than considering what an autonomous agent can do about the failure. Some of these taxonomies borrow from the human interaction or safety literature. For example, the "Swiss Cheese model" of accident analysis [3] suggests accidents occur when unexpected events slip through unexpected layers, the "SHERPA model" of human error analysis [4] attempts to predict human error by decomposing it into tasks and actions. More relevant to robots, [5], [6], [1] each provide an analysis of robot platforms and consider how they may fail in the field. Yet, by not considering how to respond to faults, these taxonomies fail to include agents that have the capacity to assess, resolve, and communicate their mission obstacles. Additionally, causal failure analysis is often limited to considering a single chain of cause and effect, when, in reality, failure is often tightly connected to multiple states or other failure conditions that all need to be accounted for.

The goal of this paper is thus to propose a new taxonomy of robot mission failures which, unlike other taxonomies, is based on the analysis of reports from large autonomous robot deployments that describe faults that actually occurred, thus providing the basis for developing relevant methods for predicting and interpreting those failures. Together with methods for assessing failure impact, their integration into monitoring, reasoning and planning components in a robot architecture can then be utilized to avoid or mitigate the impact of perturbations. Specifically, we analyze a large dataset of real-world AUV failures to construct and validate a three-dimensional taxonomy that links low-level causes to symbolic-level failure reasoning, allowing robot failures to be considered from the perspective of task planning for debugging, mitigation, explanation, and adaptation—to our knowledge, this analysis of real-world deployments is based

[1]Christopher Thierauf is with the Deep Submergence Laboratory, Woods Hole Oceanographic Institution, Woods Hole, MA 02543, USA. Christopher.Thierauf@whoi.edu, work done while with the Human-Robot Interaction Laboratory, Tufts University.

[2]Matthias Scheutz is director of the Human-Robot Interaction Laboratory, Tufts University, Medford, MA 02155, USA. Matthias.Scheutz@tufts.edu

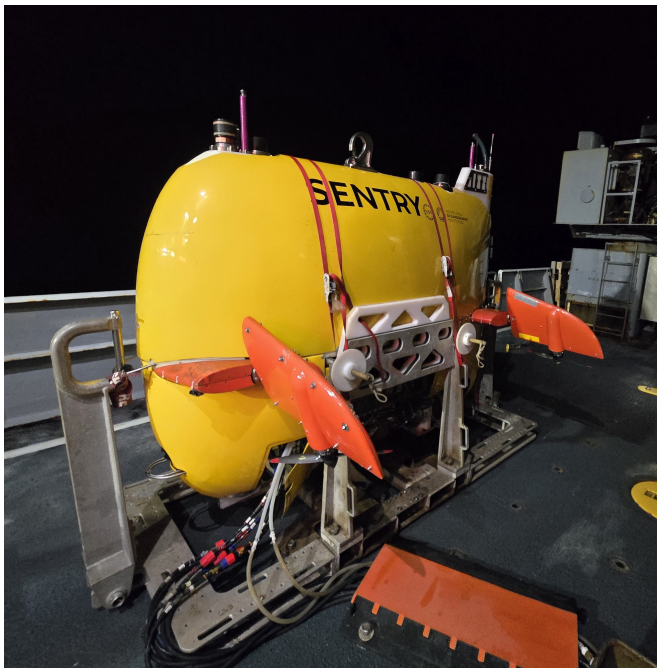

Fig. 1. The Sentry Autonomous Underwater Vehicle (AUV) between dives on the deck of the R/V Thomas G Thompson (photo: own work).

on the largest failure dataset available in the literature, and for AUVs, may be the only one of its kind. The proposed taxonomy then directly describes robot failures in ways that are human interpretable and usable for task planners to find mitigation plans (if they exist in the robot architecture and fault mitigation is possible).

II. Dataset and Methodology

We utilized the data contained in the "Sentry AUV Dive Report documents" to generate the dataset, which contains the records for over 700 AUV deployments (which typically range from 12 to 24 hour runtime). These documents are produced in the process of deploying the Sentry AUV as part of standard fieldwork operations within the National Deep Submergence Facility at WHOI[1]. For each deployment, operators produce a summary of critical dive information (e.g., launch coordinates, dive time, battery use, etc.) as well as sensors and system configurations. Most critically for this taxonomy, a "narrative" paragraph is written for each dive. This narrative includes a textual description of the dive goals, expected vs. produced behavior, the reason for the end of the dive, and any other information that may be operationally relevant to future users of the system or the data. This narrative is rich in information but challenging for an automated system to interpret.

For example, consider the following excerpt from a dive report for Sentry510:[2]

> Sentry510 is the first dive of AT42-05. Sentry510's primary objective was a stepped-altitude chemical survey of Cathedral Hill. Sentry flew the normal lawn-mowing pattern with 25m trackline spacing and 5m altitude steps starting from 5m up through 40m. Sentry's camera took photos for the first 5m survey level. The outboard methane sensor attached to the vehicle recovery hook stinger caused control issues at speeds over 0.6 m/s due to excessive, asymetric [sic] drag. Sentry was able to complete the desired altitude levels, but the tracklines wander considerably. Following surveys will restrict forward speed to 0.6 m/s or slower. Sentry left the bottom at 05:15 and was recovered on-time at 06:00.

The goal of our dataset construction was to avoid manual processing of these dive report documents and instead use computational means to extract relevant fault information. For this purpose, we developed a Python script which parses the dive report PDFs to extract the narrative (which contains a textual summary of mission events) and statistical information (which contains numeric statistics about the mission) for each deployment—this approach was feasible because dive reports generally follow the same template, and custom logic was implemented to catch the few edge cases. Then, a SQL database was programmatically constructed where each field in the statistics section of the dive report was inserted directly so that it could be later queried.

For purposes of taxonomization, the more useful data comes in the form of the narrative text. To ease the burden of manually analyzing this type of data, a Large Language Model (LLM) was used instead. Two were used: the GPT-4o model, via its online API [7], and Anthropic's Claude Opus 4.7 [8], zero shot, with default sampler settings. Results were human-reviewed where the models disagreed. They fully agreed on 264 of 387 multi-step categorizations (68%); the remaining 123 cases were hand-resolved by the authors. For each dive then, the LLM was prompted on the text several times with several summarization queries and other fact based questions:

- Did this mission succeed or fail in its goals? Return only SUCCESS or FAILURE.
- Summarize the mission goals into one sentence.
- Summarize the mission events into one sentence.

In the event of a failure, an additional set of questions were asked; for GPT-4o these are independent queries, for Claude these remain in zero-shot:

- Summarize the failure into one sentence.
- What was the cause of the failure? Return only HARDWARE, SOFTWARE, ENVIRONMENTAL, or HUMAN. Return HARDWARE if the failure is due to a

[1]The Sentry AUV is an underwater robot used for deep-sea scientific exploration, see `https://en.wikipedia.org/wiki/Sentry_(AUV)`.

[2]In Sentry's dive labeling nomenclature, "Sentry510" refers to the five hundred and tenth deployment of the Sentry AUV, which is available without embargo by contacting the National Deep Submergence Facility at the Woods Hole Oceanographic Institution, see `https://ndsf.whoi.edu/data/`.

hardware condition. Return SOFTWARE if the failure is due to a software condition. Return ENVIRONMENTAL if an element of the environment caused the failure. Return HUMAN if a human mistake led to the failure
- What was the temporal condition of the failure? Return BEFORE(X), AFTER(X), or DURING(X), where X is some state.
- Was the failure constant or intermittent? Return only CONSTANT or INTERMITTENT
- Was the failure correlative with a state, or caused by it? Return CORRELATION or CAUSATION
- What was the effect of the failure? Return only ABORTIVE, DEVIATIVE, or DEGRADATIVE. Return ABORTIVE if the mission aborted. Return DEVIATIVE if the behavior deviated from expected. Return DEGRADATIVE if the condition of the system degraded over time but no behavior changes were observed during operation.

Outcome disagreements for these queries were directional: every case GPT-4o flagged as "FAILURE" that Claude disagreed with was because Claude (correctly) did the more sophisticated inference that the deployment was successful despite the identification of a failure mode. To better explore these cases, Claude was then prompted:

- If the success contains some deviation from the original mission that was corrected, mark this as 'CONTAINS_DEVIATION'. Otherwise, mark it as 'FULLY_SUCCESSFUL'.

These cases actually highlight a core finding of this work (that failure can occur without it failing the mission outright), and in these cases we interpret Claude's choice of CONTAINS_DEVIATION as FAILURE, as intended by the taxonomy, so that DEGRADATIVE or DEVIATIVE may handle what Claude interpreted as non-failure events.

Each of these then became their own entry in the SQL database, used in future taxonomization and analysis. In the event that a dive was found to be a failure, the LLM was further prompted to categorize the failure into the various subcategories described in Section IV. Causal distributions are found to be broadly consistent under either pass; finer-grained conditional categorization and analysis is beyond the scope of this work.

Overall, Sentry deployment numbers 300 to 750 (inclusive) are represented in the dataset, ranging from the beginning of standardized mission reports to the most recently available as of 2024. This represents data from a total of approximately 5,400 deployed hours, and over 12,000 kilometers of distance traveled of the robot.

III. Existing Taxonomies

We consider three existing taxonomies that describe how robot deployments fail; Table I summarizes how each compares with the framework we propose in Section IV. In [5], a taxonomy is presented that first divides failure into being physical in origin, or human in origin. Should the failure be physical, it can then be divided further into specific subsystems (e.g., end effector, sensor, power). Should the failure be human in origin, this may stem from failure in design prior to deployment or it may stem from interaction during operation. An additional dimension of impact and repairability is added to the newer versions of this taxonomy. The repairability dimension considers if the robot can or cannot be repaired by a human operator in the field. The impact dimension considers if the fault is "terminal" (results in the end of the deployment) or "non-terminal" (deployment can continue).

For our purposes, this taxonomy is not suitable. We are interested in domains where we cannot assume field repairability: this may be because the robot is deployed to address domains that are inaccessible to their human operators (e.g., deep underwater or during space exploration) or because they have been deployed to domains where human experts to perform repair are not present (e.g., in a home). In these cases, the robot must still be capable of determining that it has or will fail, and should take steps to address this (at minimum failing safely without intervention).

In [1], the authors first consider stressors: open-ended disturbances to operation. They ask if these stressors are targeted (caused by antagonistic agents in the environment) or untargeted (e.g., standard wear and tear). They also ask if the stressor is modeled (fits within a known statistical distribution) or not. They then get closer to our needs by considering what an agent can do about the ongoing failure. They break this into three categories: Preoperative, intraoperative, and postoperative. The preoperative step are design-time strategies that attempt to mitigate failure (e.g., designing in redundancy). The intraoperative steps are to dynamically adjust behavior as needed to be robust against failures as they are encountered. Similarly, postoperative steps aim to adjust behavior in post after analyzing real vs. expected robot behavior.

Unfortunately for our applications, however, the preoperative category does not consider how an agent might understand that entering a condition will cause a failure. Instead, it focuses on how human designers or operators may incorrectly specify or design essential components in the software or hardware of the robot. Similarly, although the intra- and post-operative steps do consider modifications to the robots behavior, they do not do so in a way that is generic and applicable beyond hand-crafted problem-specific failure resolution strategies. Further, they assume a human operator is performing this learning and re-implementation on behalf of the robot. This is a key area we aim to address by providing the taxonomization in task planning framework, and so this approach is not suitable.

Finally, in [6], the authors present a system that focuses on human-robot interaction failure. They break the failure into being either technical or human-based in origin. If technical, the failure can be a problem of some hardware component, or it can be caused by software. If it is caused by software, this may be due to a design failure, communication failure, or processing failure. If human-based, this may be because of a social norm violation, a human error (like slipping,

TABLE I

COMPARISON OF EXISTING FAILURE TAXONOMIES WITH OUR PROPOSED APPROACH.

	Carlson & Murphy (2005)	**Prorok et al. (2020)**	**Honig & Oron-Gilad (2018)**	**Our proposal**
Core Focus	Field robot hardware/software failures	Resilience strategies in Multi-Robot Systems	User-centered HRI failure handling	Robot failure understanding
Levels/Dimensions	Physical vs Human failures	Pre-/Intra-/Post-operative + Stressor modeling	Technical vs Interaction failures (social, cognitive)	Symbolic cause, condition, and effect
Human Interaction	Human errors classified (slips, mistakes)	No	Central to taxonomy and model	Both as classification and output
Symbolic Reasoning	None	None	None	Explicit symbolic
Consider System Response	No	Yes	Yes	Yes
Failure Attributes	Severity, repairability, origin	Modeled/unmodeled, targeted/untargeted	Functional/social severity, symptoms, context	Severity, context, implication for task planning
Example Applications	Urban search/rescue, UGV ops	Multi-robot resilience in harsh environments	Home/social robots and user trust	Mission-level adaptation, debugging, interpretability in autonomous systems

forgetting, or making some other mistake), or general uncertainty in the environment. They then consider attributes of the failure itself: what is its severity (functionally or socially), relevance to other elements of the system, frequency with which it occurs, the conditions under which the failure occurred, and its observable indicators. The authors then go on to produce the RF-HIP model, which aims to describe how users of the robot may perceive these failures and then interpret or act on them.

This captures much of what we seek to taxonomize here. However, it focuses on perception, comprehension, and action of human partners or observers, rather than how the robot itself may perceive, comprehend, or act. In contrast, we seek a taxonomy that is tailored for autonomous, problem-solving agents: while the way in which this problem solving is perceived is certainly valuable, it is not the scope we seek here. Instead, the robot itself must be provided with this ability to interpret and re-task, and so this new taxonomy will differ by emphasizing machine actionable insight into conditions and patterns that indicate failure. Additionally, the severity will be determined not by human perception, but by the degree to which task completion is impacted.

IV. A NEW TAXONOMY OF ROBOTIC FAILURE

We propose a three-dimensional taxonomy that links cause, conditions, and effect of failure scenarios within a mission (see Figure 2). Each axis applies to a different element of a single event, allowing the event to be comprehensively and consistently described within this framework. This linking presents opportunity for avoiding failure, interpreting it, and communicating it.

A. Dimension 1: Cause

We build on [5] and [6] for describing how failures may be caused, describing the root causes of failure. This layer maps closely to diagnostic tools and fault reporting systems, and an implementation of this taxonomy would integrate such tools here. We can consider several sub-categories, e.g., failures that are internal to the embodied agent due to the agent's hardware or software:

a) Hardware failures: Mechanical or electrical failures of the system are the most common failure mode in the Sentry dataset, where the system experiences intense mechanical stress. This may be an input failure (for example, a sensor stops reporting values because it has flooded), an output failure (for example, an actuator stops responding to commands and cannot be moved by the software), or a overall system failure (for example, a power failure).

b) Software failures: Failures stemming from software implementations include behavioral failures, where the robot executes the expected code properly but the implementation produced a poor outcome such as a poorly tuned control policy, for example. Software communication failure (distinct from human communication failures) may also be a source of failure in that all components may be operating as expected, but not providing critical information to each other, preventing proper operation. Most commonly, this will be in the form of software bugs like crashes, memory overflow, or other errors leading to unexpected outcomes.

Other failures are external, and may be either due to the environment or the human operator (which we view as distinct from the environment):

c) Environmental disruptions: Elements of the environment can impede or modify the expected outcomes of behavior. For example, substantial underwater currents may force an AUV to arrive in a different location than a predetermined path would have computed. Weather conditions may force the operators to modify the planned deployment.

d) Human errors: When humans interact with autonomous systems, human errors can occur as defined in [5], for example, when human instruction are improperly interpreted, or human partners may incorrectly specify tasks, parameters, or constraints. Unlike [6], however, we do not consider the impact of the failure on a human partner, we only consider how the human may be a source of error for

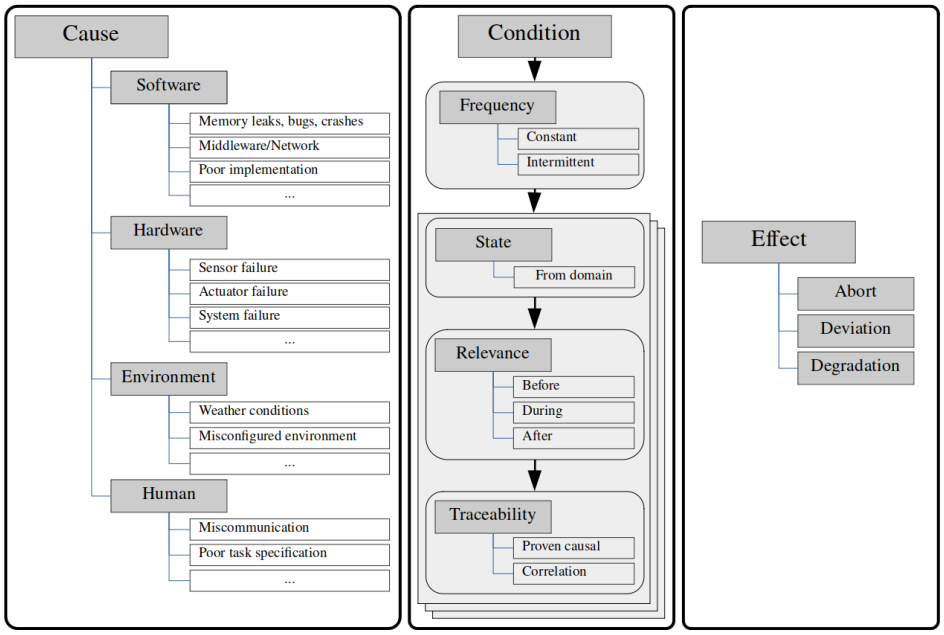

Fig. 2. The proposed taxonomy directly linking failure causes, conditions, and effects (see text for details).

the system operation.

B. Dimension 2: Condition

This dimension describes how the failure relates to some symbolic state. This dimension is unique in that it must be associated with some state in the agent's domain, we then consider the conditions under which this state is relevant to the failure. For this reason, a single failure mode may have multiple relevant conditions:

a) Frequency: First, we consider the nature of the failure itself. Does it occur constantly within a given time interval, or intermittently? This is a further refinement of the category of "permanent/transient" perturbations in [2] in that a failure may be constant within a given time interval ("transient perturbation") as in the case of a running thruster that kicks up sediment and constantly impedes visibility; or it may occur with some frequency within that transient time interval such as two SONAR-based sensors that interfere with each other on a specific clock cycle.

b) State: A failure state is then isolated with the aim of finding connections to other known states in the system and the environment, i.e., multiple states may be connected to the failure state. These states can be facts about the robot (e.g., "trap door is closed") or the environment (e.g., "robot is at location"). Ideally, these states will be encoded within the system's domain model so that the system can explicitly reason about them as in [9], but such representations are not guaranteed (e.g., for novel failure states).

c) Relevance: We aim to describe when this condition connects to the state being considered—before, during, or after it occurs—in order to be able to infer the causality. For example, we may see a relationship between a state and a failure where the state occurs immediately or substantially *before* the failure occurs as with a power surge either immediately (or, through repetition, only eventually) causing a motor controller failure. The state may be *during* the failure as in the case where two systems are running at the same time interfere with each other. Or, because we consider multiple conditions and states within one failure, we may note that this failure occurs after another one: for example, poor power regulation may cause a cascade of failures, in which multiple sensors and actuators sequentially fail.

d) Traceability: Traceability allows for representation of the fact that some conditions appear causal but actually require further analysis. For example, we may observe that a sensor consistently fails mid-deployment. This correlates with depth, but it is more reasonable to infer that it is because the sensor has been improperly configured to power off after some duration. This suggests different solutions and assessment strategies, in contrast to a flooded sensor, where we can reasonably state that the failure to produce data stems directly from the depth incurring a hardware failure.

C. Dimension 3: Effect

This dimension captures the effect of failures on the robot's task performance and observable behavior.

"Abortive effects" describe cases where the mission has irrecoverably failed due to unmitigable perturbations, which may or may not have been predictable, detectable, and avoidable [2]. This is typically because a critical system has failed, taking essential operation abilities with it. For example, in the dataset there are several examples of navigational sensors failing to provide reasonable data, leading to total positional failures that cannot be corrected from the surface. The result is the mission must be aborted.

"Deviative effects" are ones where a behavior did not produce the expected transition from one state to another. This may involve veering from paths, missing targets, or

repeating patterns. Such effects are often mitigable [2] but whether the system can recover will depend on an appropriate understanding of the failure. For example, if a key thruster fails, leading to increased drift when attempting to drive straight, the control policy can be modified to repair this deviation.

"Degradative effects" are ones where operation is not noticeably impacted yet, but that may have longer-term implications for system performance. These can generally be predicted, avoided, or repaired by learning more about the system. For example, a poorly tuned PID may cause excessive strain on the hardware, which can be mitigated by re-tuning. Poor line following may cause excessive battery drain, which is not a problem on its own but may impede the agent's ability to accomplish its task towards the end of the mission.

V. Analysis and Application to the Sentry Dataset

The Sentry dataset presents many examples of robots that encounter a different types of challenges and failure cases. For all mission deployments, a "failure" was considered any event that differed from the anticipated plan. These may have resulted in an abort, but may have been recovered through autonomous or operator intervention. Critically, "failure" here does not imply "mission failure", only a "failure event". In the vast majority of deployments, these failures were either recovered from during or corrected in post. Additionally, despite its use as an AUV that operates in remote environments, the Sentry dataset contains a surprising amount of human interaction cases with the Sentry as the operators plan, deploy, and monitor the platform, keeping the human interaction taxonomic elements relevant.

For each deployment flagged as differing from the anticipated plan, the LLM pipeline was prompted to consider each dimension of the taxonomy. The resulting distribution of failure causes and conditional properties is summarized in Table II. To categorize "Cause", the LLM assigned each event to one of four origins: software, hardware, environmental disruption, or human error. "Software errors" ranged from duplicate processes preventing critical sensors from informing the navigation system to string parsing issues. "Hardware errors" included failure of devices due to pressures at depth as well as routine mechanical degradation. "Environmental errors" included the vehicle contacting the seabed or weather conditions forcing modifications to the mission before or during deployment. "Human-rooted issues" ranged from mistakes during planning (e.g., the points specified for data collection did not correspond with the actual points of interest) to changing logistical needs (e.g., running out of time for the objective due to external planning factors).

The LLM was then tasked with associating these failures with a symbolic state and assessing the conditional elements of the failure. Among LLM-tagged failures, most were *intermittent* (see Table II), ranging from software processes that crashed and were automatically rebooted to gaps in navigation caused by out-of-range sensors. The remainder were *constant* and were generally associated with mission abort conditions, such as actuator failure or total loss of a sensor.

TABLE II

Outcomes of $n = 445$ reviewed Sentry deployments

Outcome	Count	Share
No issues before, during, or after	253	56.9%
Mission deviation but mission completed	168	37.8%
Mission failure (aborted mid-dive)	24	5.4%
Failure events (total)	**192**	**43.1%**
Mission successes (total)	**421**	**94.6%**

Traceability, relevance, and temporal conditions are challenging to assess from static records alone. Determining causal relationships generally requires real-time debugging access, which is not available within the dive report dataset. For this reason, improved autonomy of the platform and the ability to perform fault monitoring, fault detection, and recovery planning is required not only to respond to failures but also to interpret their underlying conditions and relationships.

Perhaps most notably, the catalog of failure types stems from hours deployed, not platform shortcomings. We draw a distinction between a "failure event" (any in-mission departure from a plan, which will include complete aborts as well as recoverable deviations) and "mission failure" in which the mission objectives could not be accomplished. Under this distinction, the human-reviewed pass found high overall mission reliability, with failure events being common but mission failures presenting as rare. Approximately 43% (192 of 445) of deployments contained at least one failure event, yet approximately 95% (421 of 445) still completed their primary science objectives, with on-board autonomy or operator intervention often being responsible for safe resolution. This contrast highlights that failure events will occur and that the mission can succeed regardless, providing empirical justification for an autonomy-focused architecture such as the one pursued here, and further motivates the need for agents which can independently interpret and address failure modes.

VI. Discussion

Our analysis of failure categories in Table II shows that no single cause type accounts for a majority of failure cases; software, hardware, environmental, and human origins are all materially represented. This points to a critical need for developing methods that monitor for all four: a system that only checks for hardware faults, for example, would miss a substantial share of failures. Moreover, in more than one in five cases faults are constant and can severely impact the mission performance without additional mechanisms in the control architecture to prevent them if they are preventable, for example, by predicting them in time to avoid them. Beyond the statistical information, the three dimensions of the taxonomy are designed to work alongside each other: By providing a clear interpretable separation of cause, condition, symbolic representations of failures that can be used for improving system autonomy, for associating

a failure with the conditions under which it may occur and impact future operation enables a system to use these dependencies directly in task planning to either predictively or reactively handle faults (e.g., see [10] for an architecture that enables such reasoning). Specifically, the connection "effect" with "cause" and "condition" provides the necessary information for processes to utilize fault conditions and their potentially detrimental effects for planning to monitor and categorize events that could cause failures. Moreover, the overall association of "cause" with "effect" provides the tools needed for explainability as to why a system might have decided to take certain actions in an effort to avoid a bad outcome.

Combined, the taxonomy provides the building blocks for agents to intelligently assess their environment and address failure on a task planning level, rather than simply reacting to them in retrospect. Failures can be avoided by automated planning systems by discovering the knowledge that entering a certain state triggers a new, undesirable one, for example, that fact that enabling a power channel consistently overwhelms the rest of the electrical system and so this power channel must not be enabled in future tasks. This paves the way to explainability that is informative for human operators: "I cannot complete this task because it requires the power channel to be on, but this prevents sensor use". This is particularly important for remote deployments where the system operation cannot be directly observed and human operators cannot directly intervene in the system operation (e.g., to correct faults). In such cases, human operators depend on system communications for understanding system state. And while explainability during deployment may not be desired, understanding deviations from the plan during post-mission analysis is valuable for repair and improvement in future missions. It also provides the system with a greater capability for failure avoidance and mitigation, allowing it to operate at a higher degree of reliability or, at minimum, a higher degree of safety. While mission success cannot be guaranteed, actions which are realized to place the agent in an unsafe condition can often be avoided by aborting the current deployment. Rather than treating failure as a terminal state that must be handled with human intervention, this framework considers failure from the perspective of an autonomous agent that has the ability to diagnose, trace, and mitigate failure as a part of an ongoing deliberative process. This is a shift from many of the prevailing approaches: Instead of custom-built responses for each imagined failure scenario, failure detection and recovery becomes part of a broader fault detection and recovery reasoning process that utilizes the system's existing domain model and planning capabilities. That way, systems are not limited to reacting to failures after they occur, but especially in cases of predictable avoidable perturbations they can take proactive steps that increase the chance of avoiding the fault altogether. Ultimately, this taxonomy supports the development of methods for real-time adaptive autonomy by providing a structure that can be used to interpret, respond to, and even anticipate failure events during autonomous operation in the field or during human interaction. And critically, despite being rooted in the presented AUV data, the framework itself is domain-general and thus equally applies to other types of robots and missions.

VII. Conclusion

The goal of this paper was to provide a new failure taxonomy that is rooted in the analysis of events that occurred in a large number of actual robot deployments, rather than just providing another conceptual analysis. We developed an automated method using LLMs for analyzing a large number of mission reports which contained information about different types of faults that occurred during Sentry AUV science diving missions. The results showed that no single category of "software", "hardware", "environmental", or "human" faults accounts for a majority of failure cases. Based on this analysis and different from other existing taxonomies, we then proposed a fault framework that takes an agent-centric perspective to explicitly render perturbations and faults in terms of their types of causes, characterizing conditions and properties (such as duration and mode of occurrence), and effects on task performance. That way the framework provides the information for robot architecture designers to develop methods for diagnosis and inference which are key for future autonomy strategies that handle faults intelligently, such as strategies that can mitigate faults by utilizing a task planner in the architecture. Moreover, the explicit cause–and–effect grounding supports communication strategies, like predictions of mission outcomes or decisions for behavior changes that can be relayed to human operators. Future work will develop implementations of prototypical monitoring and reasoning processes based on the framework that can be integrated into robotic architectures to improve their reliability and long-term autonomy.

References

[1] A. Prorok, M. Malencia, L. Carlone, G. S. Sukhatme, B. M. Sadler, and V. Kumar, "Beyond robustness: A taxonomy of approaches towards resilient multi-robot systems," *arXiv preprint arXiv:2109.12343*, 2021.

[2] J. Staley and M. Scheutz, "Evaluating task-general resilience mechanisms in a multi-robot team task," in *IFIP International Conference on Artificial Intelligence Applications and Innovations*. Springer, 2021, pp. 155–166.

[3] J. Reason, "Human error: models and management," *Bmj*, vol. 320, no. 7237, pp. 768–770, 2000.

[4] D. Embrey, "Sherpa: A systematic human error reduction and prediction approach," 1986.

[5] J. Carlson, R. R. Murphy, and A. Nelson, "Follow-up analysis of mobile robot failures," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 5. IEEE, 2004, pp. 4987–4994.

[6] S. Honig and T. Oron-Gilad, "Understanding and resolving failures in human-robot interaction: Literature review and model development," *Frontiers in psychology*, vol. 9, p. 861, 2018.

[7] OpenAI, "Gpt-4o system card," *arXiv preprint arXiv:2410.21276*, 2024.

[8] Anthropic, "Claude Opus 4.7," https://www.anthropic.com/news/claude-opus-4-7, Apr. 2026, large language model.

[9] C. Thierauf and M. Scheutz, "Self-debugging robots: Fault recovery through reasoning and planning," in *2024 Resilience Week (RWS)*. IEEE, 2024, pp. 1–10.

[10] C. Thierauf, T. Law, T. Frasca, and M. Scheutz, "Toward competent robot apprentices: Enabling proactive troubleshooting in collaborative robots," *Machines*, vol. 12, no. 1, p. 73, 2024.