AI-Enabled Monitoring of Space Robotics Telemetry

Adam Casey – Critical Systems Labs Inc.
Malav Naik – MDA Space
Simon Diemert - Critical Systems Labs Inc.
Nader Abu El Samid – MDA Space
Jeff Joyce – Critical Systems Labs Inc.
Emmanuel Lesser – MDA Space
Paul Grouchy – MDA Space

Proc. UK Safety -critical Systems Symposium in York, UK, Feb 10-12, 2026.

Copyright Notice:

AI-Enabled Monitoring of Space Robotics Telemetry

Adam Casey[1], Malav Naik[2], Simon Diemert[1], Nader Abu El Samid[2], Jeffrey Joyce[1], Emmanuel Lesser[2], Paul Grouchy[2]

[1] Critical Systems Labs Inc., Vancouver, Canada,
[2] MDA Space, Brampton, Canada

Abstract *This paper reports on the use of a structured argument to evaluate safety risk associated with the use of an AI/ML enabled system to monitor telemetry communicated to a ground station by a robotic arm being used in human spaceflight. In addition to addressing questions about the performance of the AI/ML enabled system (e.g., recall rate, false positives), the structured argument addresses human factors considerations such as the risk that operators might become complacent by trusting the AI-functionality to monitor operational safety. Rather than just using a structured argument to document the outcome of a safety assurance process, the evaluation approach is being driven by a structured argument concurrently with development of the system. Taking an approach based on a GSN variant, namely, Eliminative Argumentation, the use of defeaters (e.g., "unless the operator becomes de-sensitized to AI/ML generated reports of anomalies due to a high rate of false positives") is an effective means of guiding systematic evaluation of a new technology to better understand its impact on an operational process. A similar approach could be used in other technical domains such as medical devices, critical infrastructure and plant automation that are rapidly integrating AI/ML into safety-critical technology.*

1 Introduction

Artificial Intelligence (AI) and Machine Learning (ML) enable a wide range of system capabilities that are challenging, if not impossible, to implement using conventional software. A particularly important role for AI is in perceiving the operational environment of a system, where a learnt model analyses data from a wide range of sensors with the goal of detecting behaviours or objects of interest in the data. This perception task is applicable to a wide range of applications, including medical devices, geo-intelligence, autonomous vehicles, and robotics. The current generation of AI (ML) models are "trained" on large datasets that are representative of inputs the model is expected to encounter during operation. From this data the models "learn" to identify patterns that are associated with the specific objects or behaviours being classified. The trained models encode a statistical relationship between

the inputs and the output, allowing the model to infer what the output should be for a previously unseen input encountered during operations. However, the stochastic nature of AI models means that they can produce incorrect outputs (e.g., missed detections), with relatively high frequency compared to non-AI systems, especially when the inputs differ from the training data (Koop & Koopman, 2025).

Safety-critical systems are those where the consequences of incorrect or insufficient behaviour are significant, including damage to the environment or the loss of human lives (Engineers and Geoscientists of British Columbia, 2020). To mitigate risk, organizations developing or operating safety-critical systems invest significant resources to ensure that the system's behaviour is both reliable and predictable within the system's intended operational environment. For systems that do not depend on AI as part of their safety-related function, risk management processes and methods are described in guidance and technical standards[1]. However, these standards do not contain provisions or guidance for systems that incorporate AI as part of their safety-related functionality. In particular, there is limited guidance on how to establish confidence in systems where an AI-based function produces incorrect outputs with frequencies that are significantly higher than the historically acceptable rates of (random) failure used for safety-critical systems, typically in the range of 10^{-5} to 10^{-9} failures per hour of operation.

Some guidance[2] addresses the integration of AI into safety-critical systems. A pillar of the approaches proposed by this guidance is to develop an assurance case (AC) providing an argument that there is sufficient confidence in the correct behaviour of an AI-enabled system based on engineering evidence such as specifications, analyses, and verification results (ACWG, 2021).

MDA Space[3] is a provider and operator of technology for space-related applications that enable the exploration and scientific research of space, including robotic systems such as the Canadarm: a robotic system used to manoeuvre payloads about the International Space Station (Hudon-Castillo, Lamanque, & Thenault, 2024). As part of its next generation technologies, MDA Space is exploring the potential role of AI in its robotics platforms.

Since MDA Space's robotics platforms operate in an unforgiving environment (the vacuum of space) and might interact with high-value equipment, habitat modules for humans, or humans themselves, they are considered safety-critical. Like many safety-critical systems, MDA Space's robotics platforms include monitoring functions to detect and respond to anomalous behaviour (e.g., a sensor or actuator failure). These monitors use rule-based checking, implemented in conventional

[1] E.g., IEC 61508, ISO 26262, and NASA-STD-8719.13C (ISO, 2018; National Aeronautics and Space Administration, 2013; IEC61508 - Functional safety of electrical/electronic/programmable electronic safety-related systems, 2010)

[2] E.g., ISO/PAS 8800 – *Road vehicles – Safety and artificial Intelligence* (ISO 8800 - Road vehicles - Safety and artificial intelligence , 2024), ISO Technical Report 5469 - *Artificial intelligence - Functional safety and AI systems* (ISO TR 5469 - Artificial intelligence - Functional safety and AI systems, 2024), and the AMLAS framework (Hawkins, et al., 2021)

[3] https://mda.space/

software which can detect a wide range of failure conditions (e.g., out of range sensor values, unresponsive actuators, etc.). Conventional monitoring is, and will continue to be for the foreseeable future, necessary for safety-critical applications.

However, there are some patterns of anomalous behaviour that can be difficult to detect using rule-based monitoring. For instance, a small amount of additional noise in a sensor's outputs or reduced responsiveness of an actuator might indicate degraded system health and might precede a more profound system failure. These patterns can be subtle, perhaps not even detectable by human experts (engineers, system operators, etc.). There is an opportunity to employ the pattern recognition capabilities of AI models to detect anomalous system behaviours before they precipitate into high-consequence failures.

However, the incorporation of an AI-enabled monitoring function into a safety-critical robotics platform raises unique challenges. For instance, what happens if the AI-enabled monitoring produces a high number of spurious (false positive) detections? Or, what if system operators become dependent on the AI-enabled monitor to report anomalies and misses a genuine anomaly because the AI does not report it (i.e., a false negative)?

This paper reports on a case study that investigated methods for establishing trust in an AI-enabled monitoring function intended for use safety-critical space robotics platform. Drawing on the emerging guidance for AI-enabled safety-critical systems, such as ISO/PAS 8800 (ISO 8800 - Road vehicles - Safety and artificial intelligence , 2024) and AMLAS (Hawkins, et al., 2021), we created a safety assurance argument that positions the trained model's role as part of a wider monitoring function, addresses the impact of AI on system operators (i.e., human factors), and incorporates results from test-based verification activities, including metamorphic testing (Chen, Cheung, & Yiu, 1998) and combinatorial testing (Kuhn, Kacker, Lei, & Simos, 2020), the validate the AI model, using data generated from a digital twin (DT) of a robotic system that MDA Space is developing.

2 Foundations

The AC described in this paper uses the Eliminative Argumentation (EA) notation and method. EA is a variant of the Goal Structuring Notation (GSN) (ACWG, 2021) that encourages dialectic reasoning as a means of increasing confidence in argument (Goodenough, Weinstock, & Klein, 2015). This section briefly introduces the notion of an AC and EA as means of expressing and reasoning over ACs.

2.1 Assurance Cases

In the course of developing, operating, and maintaining a critical system it is common for many engineering artifacts to be produced. The relationship between these

artifacts has historically been established through engineering and operational processes and procedures. Confidence in the system's safe or reliable operation flowed from the rigour of the engineering processes employed that prescribed additional constraints (i.e., lower random failure rates, more demanding analysis activities to avoid systematic defects, etc.) on the system's design, development, or maintenance. The necessary activities to perform to achieve different levels of risk mitigation are documented in technical standards such as IEC 61508 (industrial control), ISO 26262 (automotive), EN 5012x (rail), and so on.

However, as systems have become increasingly complex and incorporate AI-enabled functions in their safety-related control pathways, engineering process is necessary, but not sufficient to achieve assurance objectives. Preparing an assurance argument (i.e., an assurance case) that expresses project or system specific rationale for why essential quality attributes are achieved has emerged as a necessary activity for AI-enabled safety-critical systems.

The GSN community standard defines an AC as "a reasoned and compelling **argument**, supported by a body of **evidence**, that a system, service or organisation will operate as intended for a defined application in a defined environment" (emphasis added) (Assurance Case Working Group). From this definition, and others such as (Koop & Koopman, 2025; ISO, 2018), we observe that an AC has two important elements: a structured argument and supporting evidence. The argument provides rationale for how a top-level claim (e.g., "my system is acceptably safe under conditions …") is supported by evidence produced by performing various activities, including those prescribed by engineering processes or operational procedures. Examples of evidence include test results, records of peer review, design analyses, specifications, and experimental data. Arguments can be expressed in many forms, including as part of a narrative "safety report". However, structured notations such as GSN, Claim-Argument-Evidence (CAE), or EA are increasingly favoured (Diemert, Shortt, & Weber, How do practitioners gain confidence in assurance cases?, 2025).

2.2 Reasoning with Eliminative Argumentation

Unless care is taken during their development, ACs are prone to "confirmation bias" where developers select evidence and express claims they believe to be true, rather than using the AC's argument to question the validity of claims or supporting evidence (Hadon-Cav, 2009). To address confirmation bias, Eliminative Argumentation (EA) encourages AC developers to express and reason about doubts they might have in the AC (Goodenough, Weinstock, & Klein, 2015). Once expressed, doubts (called "defeaters") can either be resolved through further argumentation or remain as residual sources of doubt in the argument. When deploying or operating the system, decision makers and interest holders must accept sources of residual doubt. Importantly, the existence of residual doubt does not necessarily imply a system's

design is flawed, incorrect, or unsafe, only that there is uncertainty about some aspect(s) of the AC.

Dialectic reasoning (i.e., using defeaters) has emerged as an important part of AC practice. In a recent survey 13 out of 19 practitioners reported using dialectic reasoning when preparing ACs (Diemert, Shortt, & Weber, How do practitioners gain confidence in assurance cases?, 2025). Several researchers have incorporated it into methods (Bloomfield & Rushby, 2023; Hawkins & Ryan Conmy, Identifying Run-Time Monitoring Requirements for Autonomous Systems Through the Analysis of Safety Arguments, 2023); and the current GSN community standard describes a "dialectic extension" that permits expression of counter-goals and counter-solutions (ACWG, 2021).

Like other structured argumentation methods, arguments in EA begin from a top-level claim, which is an assertion about some desirable property of the system. The top-level claim is then decomposed into sub-claims, which assert more specific properties of the system in support of the top-level claim, and these sub-claims are themselves decomposed further, until eventually they are either supported by evidence (facts that can be verified by examining another document) or terminated with a defeater (doubt), indicating that a residual risk exists. When visualized, the diagram is shown as a directed acyclic graph (DAG) with nodes containing statements about claims and evidence and edges showing dependencies.

Arguments in EA are expressed using seven main types of nodes, and three additional node types that can only be used to terminate a branch of an argument:

- **Claims** are assertions about system properties.
- **Evidence** nodes point to artifacts that support a claim, such as test results.
- **Defeaters** describe a doubt that can be raised in response to a claim, evidence, or inference. They can suggest flaws in the argument logic, unaddressed hazards, or doubts about the relevance or validity of cited documents.
- **Assumptions** are statements that are assumed to be true without providing evidence. Assumptions can be used to limit the scope of an argument, or to state facts that are believed to stand on their own.
- **Inference rules** describe the logic of a branch of an argument. These are often used where reasoning is inductive, rather than deductive, or where the logic of an argument is not immediately obvious to a reader.
- **Strategy** nodes describe how a claim will be supported by its subclaim. Strategies can be used in arguments that enumerate hazards, mitigations, or steps in a process.
- **Context** nodes provide information that helps the reader understand the argument or the system itself, but that is not necessary for the argument to be valid.

Evidence nodes and defeaters can be followed by a terminal node:

- The **Complete** ('OK') terminator indicates that a line of reasoning is complete.
- In contrast, the **Undeveloped** ('UND') terminator indicates that a line of reasoning is unfinished and requires elaboration to complete the argument.

- The **Residual** (‘Res’) terminator identifies that there is an unresolved doubt. These indicate a residual risk and contribute to the overall doubt in the top-level claim.

A small example argument expressed using EA can be seen in Figure 1 below. The argument begins with a top-level claim (C0001) about the weather in Vancouver, Canada. This claim is decomposed into two sub-claims (C0002 and C0009) by considering the current and predicted weather conditions (S0024). An inference rule (IR0006) describes how to combine these claims in support of the parent. Two defeaters (D0005 and D0013) are raised against the first claim, and counterevidence is presented. In the case of E0019, an additional defeater undermines the evidence which is left as residual (D0021). The second claim is supported by evidence directly (E0011). Finally, the inference rule is undercut by defeater (D0014) questioning the validity of inference, which is counter-argued with additional evidence.

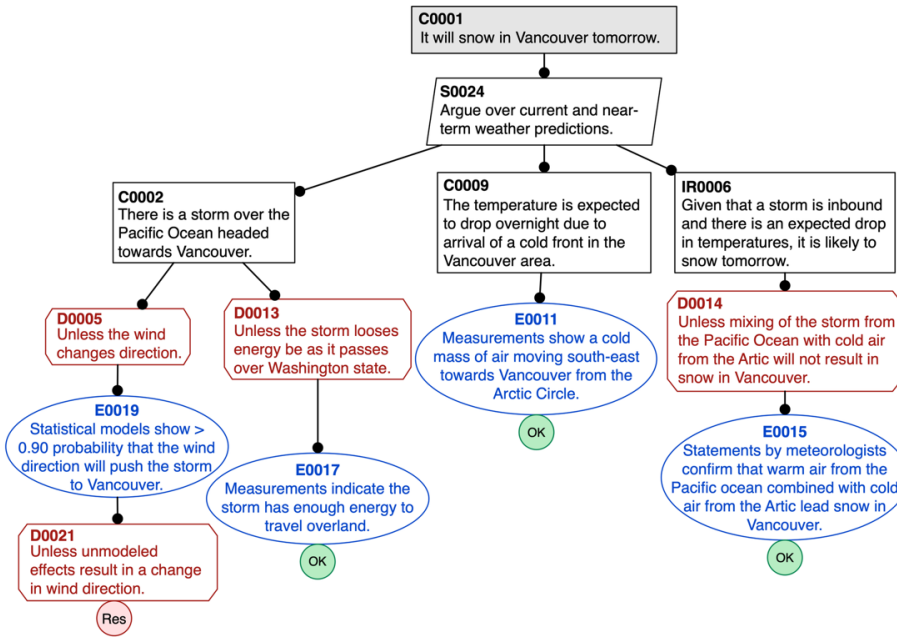

Figure 1 - Sample EA argument about the weather in Vancouver, Canada.

3 Space Robotics Case Study

The reference system consists of the following, as labelled in Figure 2 below:

1. The **Flight Segment** (comprised of a spacecraft and its cargo and/or occupants) collects data on its operations and transmits it to the Ground Segment (2-4).

2. Within the Ground Segment, the AI-based **Monitor** analyses the telemetry data, flagging suspected anomalies. There is also a complimentary monitoring system using conventional rule-based logic to flag telemetry data based on checks like “the temperature is too high” or “the angle is changing too quickly”.
3. The **Human Interface** presents both the raw telemetry data and the Monitor’s analysis to the Operator and provides means for the Operator to further analyse the data and take actions based on that analysis.
4. The **Operator** observes the information presented by the Human Interface and makes decisions on what data is truly anomalous and what action, if any, should be taken.

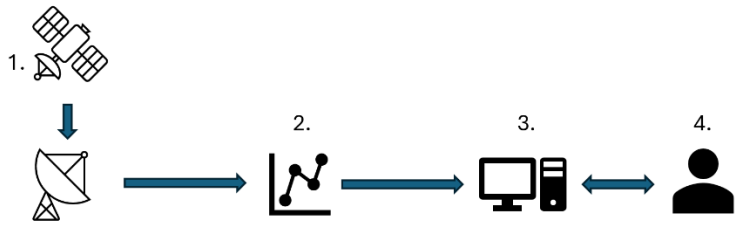

Figure 2 - Reference Monitoring System: 1. Flight Segment; 2. Monitoring Component; 3. Human Interface; 4. The Operator.

Telemetry data is analysed with some time lag and on a manoeuvre-by-manoeuvre basis, rather than as a continuous stream of real-time data. The Operator is not expected to intervene while a manoeuvre is ongoing, but rather to flag that an issue might exist so that future corrective action can be taken.

For this study, the Flight Segment was represented by a Digital Twin (DT), a realistic simulation of a space robotic manipulator, capable of simulating representative anomalies, such as loose connections, actuator wear, and sensor noise or drift. The Monitor was represented by several trained AI models (see 6.1 Materials). Data was not streamed live from the DT to the Monitor, but rather was recorded and analysed later. This does not affect the results. No stand-in was used for the human interface or the Operator, as these are not the focus of this study.

4 Assurance Argument

An AC was constructed for the monitoring system with a focus on identifying the evidence required to establish the safety of the system. While ACs are often constructed after the system itself is already complete, and use existing documentation as evidence, this AC was proactive and is used to guide development.

4.1 Argument Structure

The AC for the monitoring system argues for the following top-level claim:

> *Introducing an AI-based telemetry monitoring function to the Ground Segment will maintain or reduce safety risk[4] relative to the same system without the AI-based telemetry monitoring function.*

This claim was carefully chosen and has important implications. The AC is not trying to prove that the entire system is safe. Rather, it is trying to prove that a specific change (introducing the Monitor) does not increase risk. This is important as it limits the scope of the argument to the Monitor and its impact on the system and avoids having to analyse the safety of the pre-existing system. This top-level claim is decomposed into the following two branches as in Figure 3 below):

1. **Bounded Actions**: The only ways in which the Monitor can influence the system are to: 1) Inform the Operator that it has detected an anomaly, 2) inform the Operator that the Monitor is unable to function correctly, or 3) do nothing.
2. **Actions are Safe**: The net effect of the actions taken by the Monitor is to reduce the overall safety risk.

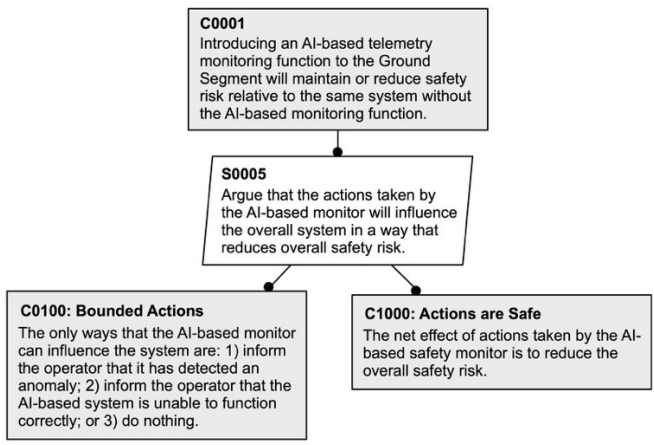

Figure 3 - Top-level argument decomposition

This structure bounds the scope of the argument. The first branch defines what actions can be taken by the Monitor, which limits the scope of further argumentation to only those actions. This branch is supported by design documentation showing the monitors input/output interface.

[4] In this AC, "safety risk" is defined as "*the combined likelihood and severity of an anomaly occurring on the spacecraft and going unmitigated, or of an Operator taking an action that is harmful*". Likewise, "safe" refers to an absence or a tolerable amount of safety risk.

The second branch contains most of the argument, and argues that those actions, when taken by the Monitor, are safe. More specifically, it assesses the risk associated with each possible action in context, and evaluates their net, collective risk:

1. **True Positives** - When the Monitor correctly reports that an anomaly exists (true positive), safety risk is maintained or decreased.
2. **False Positives** - When the Monitor spuriously reports a non-existent anomaly (false positive), safety risk is minimally increased.
3. **True Negative** - When the Monitor correctly does not report an anomaly (true negative), safety risk is unaffected.
4. **False Negative** - When the Monitor fails to report an anomaly (false negative), safety risk is minimally increased.
5. **Overall**, the collective impact of actions taken by the Monitor is to maintain or reduce risk.

Note that some risk is introduced when the Monitor makes an incorrect assessment, but that risk is argued to be minimal, and outweighed by the benefits when the Monitor is correct.

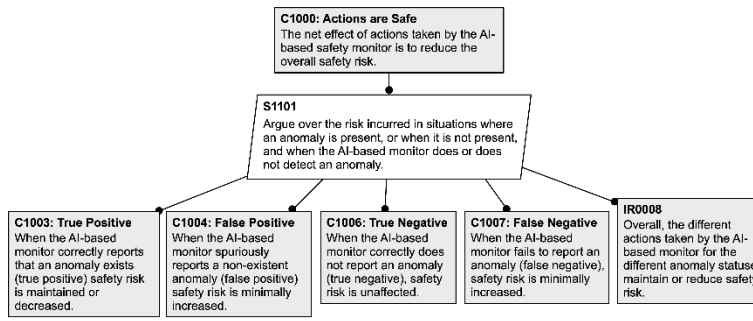

Figure 4 - Branch of the argument assessing the risk of the actions taken by the Monitor

Each of the first four branches addresses the risk for that action taken by the Monitor, in that context. It is important to remember that each branch addresses the risk *should the Monitor take that action*. There is no argumentation in these branches about *when* or *whether* that action will be taken, but rather about the *effect* of that action. Consequently, these four branches are focused on human factors mitigations for the cases when the Monitor makes an incorrect assessment.

The fifth branch addresses the net risk posed by these actions, which requires knowing how often each case will occur, relative to the others. This depends on both the occurrence rate of anomalies (which can be estimated from similar existing systems and updated after the system is deployed) and the performance of the Monitor. If the Monitor is more often incorrect than correct, it is likely that the risk will outweigh the benefits. This branch is supported by argumentation and evidence that the Monitor's performance is sufficient to ensure that overall risk is reduced.

The argument argues for sufficient model performance across three pillars: data, implementation, and validation. ML software is different from conventional software because it relies on large datasets for training, testing, and validation. The ML model will only ever be as good as the data used to create it. Thus, significant argumentation is made about the sufficiency of the data used during model development, including its coverage of operational scenarios, anomaly classes and manoeuvres; the suitability of the data sources; and the integrity of the data and labels. This branch of the argument is guided by ISO 8800 and AMLAS. The implementation branch is a process argument, asserting that model development followed best practices, and that a suitable model architecture was used. The validation branch argues that the testing methods used adequately demonstrate the performance and robustness of the model. This branch relies on evidence from combinatorial and metamorphic testing, among others (see Section 6 for details on AI model validation).

5 Human Factors Considerations

Because an AI-enabled system like the Monitor might have a higher failure rate than a conventionally-developed system, it is necessary to also address the risk introduced when the automation fails. As the Monitor interacts directly and exclusively with a human operator, the mitigations for this risk are related to human factors. We reviewed existing literature on AI-assisted monitoring systems examining sources of risk introduced by human-machine interaction ("human factors"), and strategies to mitigate these risks. This survey was focused on human factors for systems similar to the Monitor, i.e., systems where an AI-enabled component makes assessments, warnings, or recommendations to a human operator, who must then act on that information.

5.1 Sources of Risk

The human factors risks identified in the literature review for AI-augmented monitoring systems all involved "trust miscalibration": the situation where the trust that human operators place in automation does not match the actual reliability of the system (Karvonen, Heikkilä, & Wahlström, 2020; Parasuraman & Manzey, 2010). Trust miscalibration can be broken down into subcategories, each of which presents different sources of risk, and can require different mitigations:

- **Excessive Trust.** The operator trusts and relies on the AI system more than is justified (Karvonen, Heikkilä, & Wahlström, 2020; D. Xu, F. Liu, X. Ding, J. Ma, Y. Suo, Y. Peng, et al., 2025; Sanneman & Shah, 2022), deferring to the AI's analysis rather than trusting their own judgement. This can result in situations where an operator's AI-augmented performance is degraded relative to

their baseline, unassisted performance. Excessive trust can manifest in at least two ways (Parasuraman & Manzey, 2010): automation bias (commission errors) and complacency (omission errors). For automation bias, human operators tend to accept incorrect recommendations from the AI, even when presented with contradicting information. For complacency, human operators come to rely on automation and fail to act when the AI does not alert them to a problem, even when presented with information indicating a problem (Parasuraman & Manzey, 2010).

- **Insufficient Trust**. The operator distrusts the AI system more than is necessary, ignoring warnings from the Monitor and preventing the Operator from achieving the potential performance improvements promised by AI assistance.

Each of these is represented in the assurance case by defeater nodes. When the Monitor detects an anomaly when none is present, there is a risk that, due to automation bias, the human operator will take unnecessary action. Conversely, when the Monitor detects a true anomaly, there is a risk that the operator will distrust the automation and dismiss the detection, undermining the system.

5.2 Mitigation Strategies

Because the human factor risks presented by AI-augmented monitoring occur due to miscalibrated trust, these risks can be mitigated by helping operators to calibrate their trust in the system. These measures are not intended to specifically increase or decrease the operator's trust in the system, but rather to align that trust with the system's actual capability, to help the operator understand the system's strengths and weaknesses, and to allow the operator to make informed decisions using the system's outputs.

5.2.1 Operator Training

Trust calibration begins with operator training. It is crucial that operators understand the system's strengths, weaknesses, and limitations. Trust should be developed over time through incremental exposure, adaptive system feedback, and performance transparency (Karvonen, Heikkilä, & Wahlström, 2020). Operators should be guided through learning and system interaction, building confidence in system performance and identifying the boundaries of reliability. Since many human factor failures do not emerge until rare or high-pressure events occur in operations, simulation-based methods should be used to validate system behaviour, interface design, and operator-AI interactions under stress (Walsh & Beatty, 2002; M. Sethu, 2023). This allows for iterative refinement before live deployment and helps quantify human error risk in realistic conditions.

Training that includes intentional exposure to both automation bias and complacency failure modes can help operators build accurate mental models of system behaviour (Sauer, Chavaillaz, & Wastell, 2016). During training, operators should be exposed to known corner cases that cause the Monitor to fail, both for false positives and for false negatives. The types of faults to be injected into the simulation platform should align with known issues or inaccuracies identified during the validation phase of the AI monitoring system, or those observed in real-world deployment.

5.2.2 User Interface Design

The user interface (UI) should be designed to engage the operator and present information understandably and comprehensively. The UI should make it clear to the operator both why the Monitor has flagged a given segment of data as anomalous, and how confident it is in that label. Explainability (why) guides the Operator's attention to interesting areas of the data and helps the Operator decide whether an actual anomaly has occurred.

The UI should clearly differentiate between high-confidence and low-confidence outputs by providing contextual performance feedback, real-time confidence indicators, and mechanisms for users to understand system limitations (Karvonen, Heikkilä, & Wahlström, 2020). For example, probabilistic output confidence, visual uncertainty cues (such as a numeric confidence metric), and alert severity gradations can help users distinguish between high-certainty and low-certainty recommendations (Cummings, 2017). The Monitor's UI could display a numeric indication of its confidence in a detected anomaly, and should also alert the user to any conditions that might cause its performance to be degraded, such as data from manoeuvres for which the Monitor was not trained.

It is not recommended for the Monitor's human interface to rank possible anomalies.. A UI that displays several possible anomalies, ranked by confidence, might appear to be a useful extension of displaying confidence. However, human users interacting with systems that provide ranked options are more susceptible to automation bias than users of systems without ranked options (Cummings, 2017).

5.2.3 Operating Procedures

In systems where AI and human reasoning may diverge, designs should include methods to surface, communicate, and resolve observation, interpretation, and action conflicts (Arunthavanathan, Sajid, Khan, & Pistikopoulos, 2024). Conflict identification/notification, system reasoning traceability, and user intervention pathways are essential to prevent unsafe overrides or bias. Conflict could occur when an Operator dismisses a real anomaly detected by the Monitor. Strategies to mitigate this could include mandatory peer-review by a second Operator when an anomaly flagged by the Monitor is manually dismissed by the primary Operator, or

mandatory peer-review after a certain number of consecutive dismissals and collecting statistics on the frequency at which detections by the Monitor are dismissed.

Conflict can also occur between different automated anomaly detection systems, such as a conventional rule-based system and the AI-based Monitor. The UI should be designed such that warnings from both systems are merged gracefully. It should be clear which system has detected an anomaly, and neither system should be able to override or modify warnings from the other.

6 Methods for Validating AI Models

We assessed two validation methods for use with an AI-based monitor: combinatorial testing (CT) and metamorphic testing (MT) . This part of the project served two purposes:

1. **Engineering Demonstration.** The primary goal of this study was to demonstrate that these validation methods could be useful when applied to an AI-enabled system, like the telemetry monitor. This includes demonstrating that it is possible to apply these validation methods to the Monitor, and showing that these methods can detect previously unknown limitations.
2. **Scientific Experiment.** A secondary goal was to show experimentally that these techniques are effective at distinguishing between low-quality and high-quality ML models. This is driven by the hypothesis that "a useful validation method is one that is capable of differentiating high-quality from low-quality models." This is a higher bar than "Engineering Demonstration", as it demands some degree of statistical confidence in the results.

For each validation method, a single experiment was conducted, producing a single dataset for each validation method. This dataset was used for both the "engineering" and "scientific" aspects outlined above.

6.1 Materials

Although the processes used to apply CT and MT differ, they share some of the same materials:

- **Simulated Telemetry Data**. This data was generated using a digital twin (DT) and is a realistic representation of the telemetry data that the Monitor might encounter while in use. The DT allows varied scenarios to be simulated, including different manoeuvres, anomalies, and load conditions. The data consists of a time series of datapoints from several different inputs (e.g., sensors). The specific datasets for CT and MT differ, but both consist of the same type of data.

- **Monitoring Models**. The demonstration for both CT and MT requires several monitoring models of varying and known quality. For this experiment, quality variation was created using early stopping during training of the ML model, which causes the early-stop models to be less well-fit to the training data than the models trained to completion[5]. 50 such models were used, with 10 models at each of 5 different quality tiers. These numbers were chosen with the expectation that they would give a useful range and granularity of model quality, as well as enough variety at each quality tier to account for random variation in the generated models.

6.2 Combinatorial Testing

Combinatorial testing (CT) is a method of generating test cases for a software unit to achieve a desired level of coverage of that software's input space. This method was initially developed for conventional (non-ML) software, typically at the level of a single function taking a small number of input parameters (<10) with simple datatypes (e.g., "string", "integer", or "Boolean"). It works by generating test cases that cover combinations of the input parameters[6].

The premise of CT is that most failures of the system under test ("faults") are caused by interactions between a small number (<5) of input parameters. In one study which tested database management systems, over 90% of faults in each codebase were found by testing up to t=4. For medical devices tested, 99% of faults could be found by up to t= 3 (Kuhn, Kacker, Lei, & Simos, 2020).

ML-based systems present a unique challenge given the size of their input space and the fact that no individual input "parameter" is likely to correspond to a meaningful behaviour of the software. Consider an AI-enabled telemetry monitoring system for a robotic arm in space. The input to its ML component contains 600 time-series datapoints, each holding several dozen floating-point numbers. Clearly testing the system on random time-series data would not be useful. This problem can be sidestepped by redefining the input space based on the operational design domain (ODD) of the system (Diemert, Casey, & Robertson, Challenging Autonomy with Combinatorial Testing, 2023), rather than the direct inputs to the function. In this

[5] This method of varying model quality was chosen for its convenience, and for its ability to simulate a poorly-trained model. Alternatively, models could have been trained on a subset of the training data. However, this would have given an advantage to CT, which probes for gaps in the model's training.

[6] Consider a device that calculates drug dosages using the name of the drug (5 options), the patient's weight (<100 lb, 100-200 lb, >200 lb), and the patient's age (0-18 years, 18-65 years, >65 years). A combinatorial strength t is defined. For t=1, each input parameter must be tested at least once. For t=2, each *pair* of input parameters (e.g., 100 lb and Drug #1 or Drug #2 and 0-18 years) must be tested at least once. Given that there are only 3 input parameters, at t=3 every possible combination of drug, age and weight is tested.

example, the input space could potentially be defined by the type of anomaly present (including no anomaly), the operation being performed by robotic arm, the current temperature or any other relevant factor. Then CT can be used to generate test cases covering combinations of these parameters, ensuring that the test suite covers the breadth of situations in which the device will operate.

6.2.1 Validation Process

To begin applying CT, an ODD was defined for the Monitor, including the different types of anomalies the monitor is expected to detect (20 anomalies in 9 classes), manoeuvres that the Flight Segment could perform (2 manoeuvres), and different load conditions on the Flight Segment (5 load conditions in 2 classes). Using the ODD definition and the NIST ACTS tool (Kacker, 2013; User Guide for ACTS, 2018), test suites were generated for the combinatorial strengths *t=1* to *4*. These t values were used because any higher combinatorial strength would generate a number of test cases that would be infeasible to execute during this project. Each test suite consists of a list of test definitions, and each test definition includes specific values for each parameter in the ODD. For instance, a single test might be "Manoeuvre A, Unloaded, Small Interface Resistance, Large Noise on Input B". Each specified test case was simulated and data recorded. Then, each of the 50 monitoring models was run on the data from each test in each test suite, and the results were recorded for analysis.

6.2.2 Results

5816 test cases were generated in total (21 for t = 1, 441 for t = 2, 1384 for t = 3, and 3970 for t = 4). These tests were executed on each of the 50 models. From a scientific perspective, we found that as combinatorial strength increases, the test suite is increasingly capable of sorting the models by known quality, based on commonly used metrics like precision.

From an engineering perspective, combinatorial testing proved to be a useful tool. It identified specific combinations of parameters that caused even high-quality models to produce incorrect outputs. For example, it was found that certain combinations of load conditions caused models to exhibit an elevated false positive rate. Similarly, it was found that specific combinations of anomalies caused models to exhibit an elevated false negative rate. Many other combinations of parameters were found that consistently or sometimes caused the models to produce erroneous outputs. Knowledge of the combinations of parameters causing the model to fail could inform decisions about additional training data to use for future models, or important test cases to include in regular testing. Combinations of parameters which cause incorrect model outputs with high frequency could be included more often in the training data, and added to a regularly run test suite to prevent regressions.

6.3 Metamorphic Testing

Metamorphic testing (MT) is a method to test properties of a software function using "transformations", circumventing the software testing "oracle problem". For any given test case where the system receives specific inputs and expected to produce a specific output, that output must be defined by a "test oracle", which is generally a human creating the test cases. The involvement of a human limits the number of tests that can be created to the number that the human has time to create. Given that confidence in a system generally increases with the number and diversity of test cases, it is desirable to create test cases without relying on a human oracle.

MT avoids the oracle problem by using metamorphic relation (MRs) to generate many test cases from a smaller number of base test cases. An MR defines a transformation (change) to the model input parameters and the expected transformation to the model's output. These MRs are based on expert knowledge about the domain in which the system is working and reflect the analyst's intuition about how the system should behave[7].

6.3.1 Validation Process

Ten test cases were created to use as base cases in metamorphic test pairs. Some of these base cases are nominal, and some are anomalous. MRs were created, with a focus on robustness to noise and other imperfect input data. These MRs were expressed as natural language statements and later implemented in a script. For each combination of a single base case and an MR, the simulation data was transformed to produce a second, derived test case. The combination of a base case and a derived case is a "metamorphic test pair". The test pairs were executed, with each base and derived case run on each of the 50 models. Where the base case did not meet the conditions of the MR, the pair is discarded. Each result for each combination of an valid test pair and a model is checked against the conditions of the MR, and the results are collected for analysis.

6.3.2 Results

Metamorphic testing used 13 MRs. These MRs have the following structure (emphasis added to show the different parts of the MR):

[7] As an example, consider a medical device that classifies images of skin as either normal or malignant. Tumours have no inherent orientation, so reflections and rotations should not impact the classification model. One possible MR is therefore that "If a transformation consisting of any combination of reflection and rotation is applied to an image, the classification should not change."

When ("prerequisite") *the model correctly classifies the base case as nominal,* ***("transformation")*** *subsampling the data by 2, should* ***("expectation")*** *result in the model still classifying the derived case as nominal.*

Most MRs had the prerequisite that the base case be classified correctly, and some had the further requirement that the base case be nominal. The expectation of each MR was either that a given transformation would not affect the classification of a test case, or that it would cause a nominal test case to be classified as anomalous. Transformations that should not affect the classification of a test case include time-shifting the entire profile by a small amount, increasing the rate at which a single input changes by a small amount, shifting an input up or down by a small amount, subsampling, dropping up to 50% of datapoints, adding a small number of outliers, and adding a small amount of gaussian noise. Transformations that were expected to cause a case to be classified as anomalous include large time shifts, large increases in the rate of change of inputs, large shifts up or down, large amounts of gaussian noise, and large numbers of added outliers.

The results of MT were more mixed than for CT. Unlike for CT, in this experiment, MT did not prove to be effective at differentiating low-quality model from higher-quality ones. There was no apparent difference in the number of failed metamorphic test cases between known high-quality models and lower-quality models.

However, from an engineering perspective MT provided useful insight into the robustness of the models, as even the high-quality models failed some MRs, indicating that their behaviour differed from what was expected by the engineer who created the MRs. The high-quality models proved to be more sensitive to gaussian noise than expected, as a moderate amount of gaussian noise applied to a nominal simulation caused models to classify the case as anomalous. Similar results were found for time-shifting the data and shifting an input up or down slightly. This suggests that MT could be a useful tool to test a model's robustness to transformations like noise, dropped data, or other degradations. These failures could also indicate overfitting during model training. The model was also expected to detect a certain number of outliers as anomalous, but did not. This insight could be used to augment the training data, for instance by adding more anomalous cases with outliers.

7 Discussion

This paper presents elements of an assurance case for an AI-based telemetry monitoring function, the results of experiments conducted on validation methods for such a telemetry monitoring function, and an analysis of human factors for AI-based monitoring systems. Below we highlight key findings and limitations from our work in this area.

Assurance Cases. ACs are a useful tool for systems engineers. They provide a structured, logical connection from assurance work products ("evidence") like test results and engineering design documentation to assurance goals, like reliability or safety targets. Without an AC, it is difficult to determine whether the engineering

work which has been done sufficiently demonstrates the safety and reliability of the system. However, having an assurance case does not guarantee that the system itself is safe. Like any other, the AC presented in this paper is only valid so long as the evidence it is based on exists, is documented, and is correct. This includes implementing human factors mitigation strategies, such as those discussed in this paper, doing thorough testing, perhaps including CT and MT, and other evidence referenced in the AC.

Human Factors. The human factors strategies presented in this paper are a good basis for designing the human interface for operators monitoring telemetry, assisted by the AI-enabled monitor. Future work should implement prototypes using these techniques to validate their efficacy.

Validation Methods. The validation method demonstrations are most interesting from an engineering practitioner's perspective, but "scientific" results are also presented. From the engineer's perspective, both CT and MT demonstrated promise as techniques for validating AI-based monitoring systems.

CT was shown to be capable of finding combinations of input parameters which cause even high-quality monitoring models to produce erroneous outputs. This could indicate gaps in the training data, or overfitting during training. This information could be used to guide future conventional testing, as well as suggesting ways to augment the training data used to produce the model. For instance, if CT found that the combination of a particular anomaly with a specific orientation of the robotic manipulator caused the monitor to exhibit an increased false negative rate the anomaly, augmenting training data with cases including that combination could improve the model's robustness. However, some of the results found for CT might be attributable to limitations of the DT used to simulate the test cases and generate data. Future work on CT should ensure that the DT can simulate all test cases with sufficient fidelity.

Likewise, MT found model behaviour that conflicted with the engineer's expectations, as encoded in the MRs, suggesting that MT could be a useful tool for testing the robustness of AI-enabled telemetry monitoring systems. MT could be used to test the Monitor's response to different forms of data corruption, such as sensor noise, data dropout, reduced sampling rate, and others. When MRs are found to fail, this knowledge can be used to guide future model development, including adding training data or otherwise tuning the model.

From a scientific perspective, the results of this study were mixed. Combinatorial testing showed promise as a technique to differentiate between higher-quality and lower-quality models, but this result could not be tested for statistical significance and had sample size limitations. Further, the experiment varied model quality using early stopping, rather than varying the training data or the model architecture. If the various models used different training data, such as incomplete training data for the lower-quality models, CT would be expected to perform very well. This could be an interesting direction for future studies. Furthermore, the ODD used for CT was relatively small due to the simplicity of the DT used. Testing CT with an expanded ODD, defined for a more complex and physically realistic simulation might also yield interesting results. In our experiment, metamorphic testing did not

differentiate between higher-quality and lower-quality models. This does not mean that MT is not a useful tool, in fact our experiences using it as an engineering method suggest the opposite, but rather that the specific hypothesis tested was not proven to a sufficient degree of confidence, with the specific MRs used and implemented.

References

ACWG. (2021). *Goal Structuring Notation Community Standard Version 3.* SCSC.

Arunthavanathan, R., Sajid, Z., Khan, F., & Pistikopoulos, E. (2024). Artificial intelligence - Human intelligence conflict and its impact on process system safety. *Digital Chemical Engineering, 11*, 100151.

Assurance Case Working Group. (n.d.). *Goal Structuring Notation Community Standard (Version 3).* 2021: Safety Critical Systems Club.

Bloomfield, R., & Rushby, J. (2023). *Assessing Confidence with Assurance 2.0.* arXiv.

Chen, T., Cheung, S., & Yiu, S. (1998). *Metamorphic Testing: A New Approach for Generating Next Test Cases.* Hong Kong: Department of Computer Science, Hong Kong University of Science and Technology.

Cummings, M. L. (2017). Automation bias in intelligent time critical decision support systems. In *Decision making in aviation* (pp. 289-294). Routledge.

D. Xu, F. Liu, X. Ding, J. Ma, Y. Suo, Y. Peng, et al. (2025). Exploring ICU nurses' response to alarm management and strategies for alleviating alarm fatigue: a meta-synthesis and systematic review. *BMC nursing, 24*(1), 412.

Diemert, S., Casey, A., & Robertson, J. (2023). Challenging Autonomy with Combinatorial Testing. *IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW).* Dublin, Ireland.

Diemert, S., Shortt, C., & Weber, J. H. (2025). How do practitioners gain confidence in assurance cases? *Information and Software Technology, 185*, 107767.

Engineers and Geoscientists of British Columbia. (2020). *Professional Practice Guidelines on Software Development for Safety-Critical Systems.* Burnaby, BC.

Goodenough, J. B., Weinstock, C. B., & Klein, A. Z. (2015). *Eliminative Argumentation: A Basis for Arguing Confidence in System Properties.* Pittsburgh, United States: Carnegie Mellon University - Software Engineering Institute.

Hadon-Cav, C. (2009). *The Nimrod Review.* UK House of Commons.

Hawkins, R., & Ryan Conmy, P. (2023). Identifying Run-Time Monitoring Requirements for Autonomous Systems Through the Analysis of Safety Arguments. *Computer Safety, Reliability, and Security.* Toulouse, France.

Hawkins, R., Paterson, C., Picardi, C., Jia, Y., Calinescu, R., & Habli, I. (2021). *Guidance on the Assurance of Machine Learning in Autonomous Systems (AMLAS) v1.1.* Assuring Autonomy International Programme (AAIP), University of York.

Hobbs, C., Diemert, S., & Joyce, J. (2024). *Driving the development process from the safety case.* Safety-Critical Systems Club. .

Hudon-Castillo, Y., Lamanque, J. C., & Thenault, M. (2024). Canadarm, Canadarm2 and Canadarm3: The evolution of Canada's iconic robotic system and its impacts from space down to earth. *Proceedings of the 58th IAA History of Astronautics Symposium.* Milan, Italy.

(2010). *IEC61508 - Functional safety of electrical/electronic/programmable electronic safety-related systems.* International Electrotechnical Commission.

ISO. (2018). *ISO 26262 - Road Vehicles - Functional Safety.* International Organisation for Standardization.

(2024). *ISO 8800 - Road vehicles - Safety and artificial intelligence .* International Organisation for Standardisation.

(2024). *ISO TR 5469 - Artificial intelligence - Functional safety and AI systems.* International Organization for Standardization.

Kacker, R. N. (2013). ACTS: A Combinatorial Test Generation Tool. *Proceedings of Sixth IEEE International Conference on Software Testing, Verification and Validation ICST 2013.* Luxembourg.

Karvonen, H., Heikkilä, E., & Wahlström, M. (2020). Safety challenges of AI in autonomous systems design – solutions from human factors perspective emphasizing AI awareness. In *Engineering Psychology and Cognitive Ergonomics. Cognition and Design* (pp. 147-160). Springer International Publishing.

Koop, & Koopman, P. (2025). *Embodied AI Safety: Reimagining safety engineering for artificial intelligence in physical systems.* Pittsburgh, PA: Independently Published.

Kuhn, D., Kacker, R., Lei, Y., & Simos, D. (2020). Input Space Coverage Matters. *Computer (IEEE Computer), 53*(1), 37-44.

M. Sethu, B. K. (2023). Application of Artificial Intelligence in Detection and Mitigation of Human Factor Errors in Nuclear Power Plants: A Review. *Nuclear Technology, 209*(3), 276-294.

National Aeronautics and Space Administration. (2013). *NASA-STD-8719.13C - Software Safety Standard*. Washington, D.C.: NASA Office of Safety and Mission Assurance.

Parasuraman, R., & Manzey, D. H. (2010). Complacency and bias in human use of automation: An attentional integration. *Human factors, 52*(3), 381-410.

Sanneman, L., & Shah, J. A. (2022). The situation awareness framework for explainable AI (SAFE-AI) and human factors considerations for XAI systems. *International Journal of Human-Computer Interaction, 28*(18-20), 1772-1788.

Sauer, J., Chavaillaz, A., & Wastell, D. (2016). Experience of automation failures in training: effects on trust, automation bias, complacency and performance. *Ergonomics, 59*(6), 767-780.

User Guide for ACTS. (2018). NIST.

Walsh, T., & Beatty, P. C. (2002). Human factors error and patient monitoring. *Physiological measurement, 23*(3), R111.

Zhou, Z. Q., & Sun, L. (2019). Metamorphic Testing of Driverless Cars. *Communications of the ACM, 62*(3), 61-67.