Reconciling Safety Measurement and Dynamic Assurance*

Ewen Denney and Ganesh Pai
KBR / NASA Ames Research Center, Moffett Field, CA 94035, USA
{ewen.denney, ganesh.pai}@nasa.gov

Abstract

We propose a new framework to facilitate dynamic assurance within a safety case approach by associating safety performance measurement with the core assurance artifacts of a safety case. The focus is mainly on the *safety architecture*, whose underlying risk assessment model gives the concrete link from safety measurement to operational risk. Using an aviation domain example of autonomous taxiing, we describe our approach to derive safety indicators and revise the risk assessment based on safety measurement. We then outline a notion of *consistency* between a collection of safety indicators and the safety case, as a formal basis for implementing the proposed framework in our tool, AdvoCATE.

1 Introduction

Software-based self-adaptation and machine learning (ML) technologies for enabling autonomy in complex systems—such as those in civil aviation—may induce new and unforeseen ways for operational safety performance to deviate from an approved baseline of acceptable risk. This phenomenon, known as *practical drift* [13], emerges from the inevitable variabilities in real-life operations to meet service expectations in an operating environment that is inherently dynamic. Conceptually, it can be understood as progressively imperceptible reductions in the safety margins built into a system in part due to initially benign operational tradeoffs between safety and performance. A system therefore appears to be operating safely but, in fact, is operating at a higher level of safety risk than what was originally considered acceptable, or approved for service. Left unchecked, practical drift may suddenly manifest as a serious *incident* or *accident*. Assessing the change in operational safety risk is thus key to identifying practical drift, its impact, and the mitigations needed.

1.1 Related Work

The conventional approach to operational safety assurance in aviation largely relies upon hazard tracking and safety performance monitoring and measurement, as part of a larger *safety management system* (SMS) [10]. The contemporary safety case approach to assurance has similarly employed safety monitoring and measurement: for example, our earlier work on *dynamic safety cases* [5] first suggested connecting safety monitoring to assurance argument modification actions. Subsequently, an approach to defining performance metrics and monitors by identifying the defeaters and counterarguments to a safety case has been developed in [12]. The concept has since also been applied to safety assurance of self-adaptive software [4], and to detect operational exposure to previously unknown hazardous conditions [18]. More recently, the use of *safety performance indicators* (SPIs)—a concept with a well-established history of use in aviation safety [13]—has been proposed for evaluating safety cases for autonomous vehicles [15]. These approaches all share a common motivation: using measurement based assessment to confirm at deployment, and maintain in operation, the validity of the assurance arguments of a safety case.

Although such an approach suggests which parts of an argument may have been invalidated, and thus require changing, the nature and extent of the change to operational safety risk levels is left implicit. Such analyses can also

*This work was performed under Contract No. 80ARC020D0010 with the National Aeronautics and Space Administration (NASA), with support from the System-wide Safety project, under the Airspace Operations and Safety Program of the NASA Aeronautics Research Mission Directorate.

meaningfully inform what modifications may be needed to the system and its safety case, especially when—due to practical drift—improved system performance is observed without detrimental safety effects, even though parts of the safety argument have become invalid. Current safety case approaches that use safety performance measurement to validate assurance arguments give limited guidance on how to facilitate what this paper considers as *dynamic* assurance (see Fig. 3): *continued, justified confidence that a system is operating at a safety risk level consistent with an approved risk baseline*.

There are other variations of the dynamic assurance concept [17], [21] that aim to optimize operational system performance, and thus opt for situation-specific runtime tradeoffs between safety and functional performance, instead of designing for the worst case. However, such tradeoffs may result in the initiating conditions for practical drift. Our proposed framework rather aims to identify and contain practical drift, whilst considering that a safety case for a system is always for a design that accounts for the worst credible safety effects. In [18], dynamic assurance refers to the automated aspects of so-called *continuous assurance*: a concept that, in effect, extends our prior work [5], by using monitors for different kinds of uncertainty that then trigger modifications to the system and its assurance case. The relationship of testing and operational metrics to safety assurance has been explored in [19], similar to our work in this paper (see Section 4), though there the focus is on providing confidence that a system meets its safety target, given evidence of mishap-free operation. In contrast, our focus here is on determining how safety risk has changed given similar measurement evidence.

1.2 Contributions and Paper Organization

To facilitate a framework for dynamic assurance within a safety case approach, the focus of this paper is on associating safety performance measurement with the *safety architecture* of a system, in addition to assurance arguments (Section 2). Using an aviation domain system (Section 3) as motivation, we present our approach to define safety metrics and indicators, through a concept of *safety measurement basis* (SMB), then revise the operational safety risk assessment based on safety measurement, and characterize the change to safety risk levels (Section 4). Additionally, we give illustrative numerical examples. Then (Section 5) we formalize a notion of *consistency* between the SMB for a system and the arguments of its safety case. We conclude (Section 6) by describing a preliminary implementation in AdvoCATE, and with a discussion of our future plans to further advance this work. The contributions above differentiate our work from prior related research.

2 Conceptual Background

2.1 Safety Case Metamodel

Our safety case concept [1] communicates confidence in safety through multiple viewpoints via a collection of core, interlinked *assurance artifacts*, namely: hazard, requirement, and evidence logs, a safety architecture, and an assurance rationale. Of those, the last two are particularly relevant for this paper. Assurance rationale captured as structured arguments expresses the reasoning why safety claims ought to be accepted on the basis of the evidence supplied. A safety architecture [6], [8] models the mitigations (and their interrelations) to the events characterizing the operational *risk scenarios* for a system, thereby offering a system-level viewpoint on how safety risk is reduced.

Fig. 1 shows a fragment of the metamodel associated with our safety case concept (as unshaded class nodes), for which we have a model-based implementation in our tool, AdvoCATE [7]. We use the *goal structuring notation* (GSN) [20] to represent structured arguments, and *bow tie diagrams* (BTDs) to represent views of a safety architecture. Those views capture a causal chain (e.g., see Fig. 2) of *threats* (initiating events) causing a *top event* (a hazard) that can lead to *consequence events* (undesired safety effects), along with the *barriers* (mitigations) necessary to reduce the safety risk posed. Each such event chain requires a combination of *hazardous activity*, *environmental condition*, and *system state* (together representing the operating context[1]), and can admit an arbitrary number of intermediate events between the initiating threat and terminating consequence events. Each barrier is itself a system comprising underlying *controls*; thus, it can have its own associated safety architecture, giving the overall model a layered structure that can mirror the system hierarchy.

A risk assessment model underlying a safety architecture gives the formal basis to: (i) characterize the extent of risk reduction, and (ii) link safety metrics and indicators to operational safety risk (see Section 4). In brief, this model

[1] Also known as an *operational design domain* (ODD) for systems integrating ML [14].

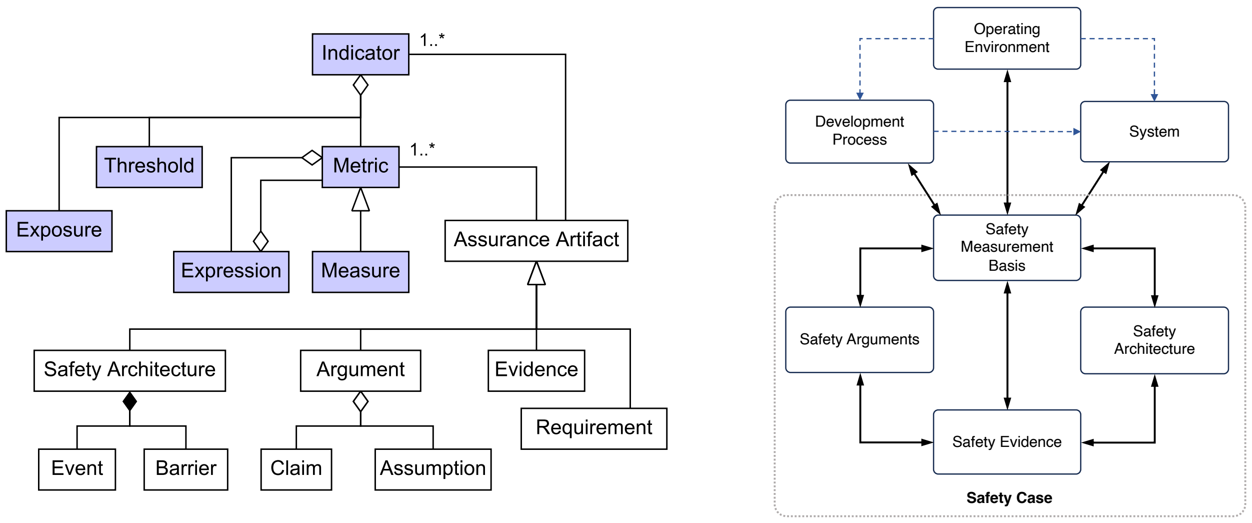


Figure 1: Fragment of AdvoCATE safety case metamodel (on the left) extended with measurement concepts that are part of a safety measurement basis (shown on the right), which is the interface between quantities in the system, its environment, its development process, and the assurance artifacts comprising a safety case (solid arrows denote consistency relations).

relates the risk of consequence events, i.e., their probability and severity, to that of the precursor events, and to the *integrity*[2] of the applicable barriers and their constituent controls. Depending on the stage of system development, we can interpret each of an event probability and barrier/control integrity both as a design target and verification goal. For the rest of this paper, we mainly consider the risk reduction contribution of barriers.

2.2 Safety Measurement

We extend the safety case metamodel in AdvoCATE with concepts for safety performance measurement (shown by the shaded class nodes in Fig. 1) as follows: we link the *indicators* to the core assurance artifacts—in particular, the event and barrier elements of a safety architecture, the claims and assumptions in arguments, to requirements, and to evidence artifacts. An *indicator* consists of a *metric* along with a *threshold*, representing the target that a metric should (or should not) reach, over a specified *exposure*, expressed either as a duration of continuous time or a specified number of occurrences of a discrete event. Indicators that have a bearing on safety can be called *safety indicators* (SIs) or *safety performance indicators* (SPIs). *Metrics* are computed values based on *measures*—directly observable parameters of the system, its environment, and its development process—and other metrics, which we represent using an expression language. Thus, they are arithmetic expressions over measured variables drawn from the most recent *mission*—which we term as a *data run*—or the missions conducted over the lifetime of the system. They can also refer to values referenced in assurance artifacts.

As shown in Fig. 1, a safety case can be seen as comprising a *dynamic* portion (indicators, metrics, and measures) and a static portion (safety arguments and safety architecture), with links associating the two. We refer to the set of interconnected indicators, metrics, and measures, along with their traceability links to the assurance artifacts of a safety case as a *safety measurement basis* (SMB). Roughly speaking, the connection between the dynamic and static portions is that the indicators represent the objectively quantifiable content of the arguments and the safety architecture which, in turn, give the justification for how those indicators collectively provide safety substantiation. Put another way, we want the SMB to be *consistent* with the static portions of the safety case, especially the arguments and the safety architecture (see Section 5).

[2]Integrity is the probability that a barrier or control is not breached, i.e., it delivers its intended function for reducing risk in the specified operating context and scenario [8].

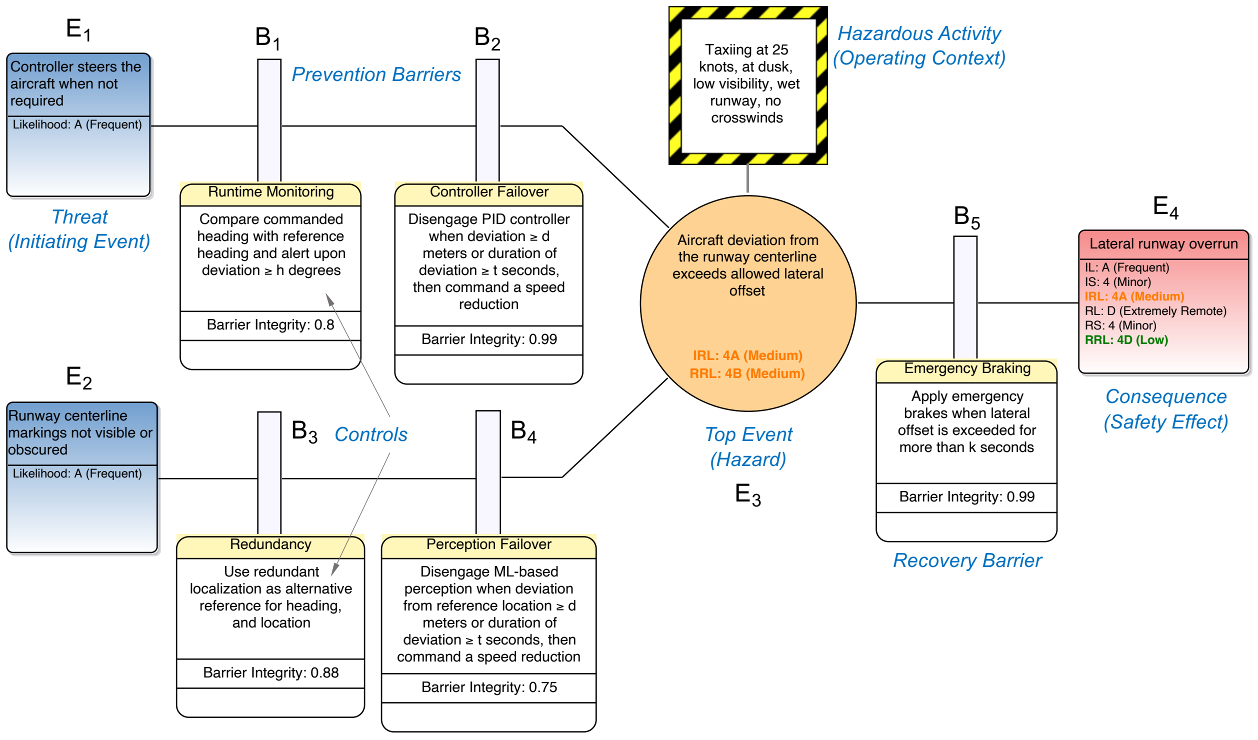

E_1
Controller steers the aircraft when not required
Likelihood: A (Frequent)
Threat (Initiating Event)
B_1
Prevention Barriers
B_2
Runtime Monitoring
Compare commanded heading with reference heading and alert upon deviation ≥ h degrees
Barrier Integrity: 0.8
Controller Failover
Disengage PID controller when deviation ≥ d meters or duration of deviation ≥ t seconds, then command a speed reduction
Barrier Integrity: 0.99
Taxiing at 25 knots, at dusk, low visibility, wet runway, no crosswinds
Hazardous Activity (Operating Context)
Aircraft deviation from the runway centerline exceeds allowed lateral offset
IRL: 4A (Medium)
RRL: 4B (Medium)
Top Event (Hazard)
E_3
B_5
Emergency Braking
Apply emergency brakes when lateral offset is exceeded for more than k seconds
Barrier Integrity: 0.99
Recovery Barrier
E_4
Lateral runway overrun
IL: A (Frequent)
IS: 4 (Minor)
IRL: 4A (Medium)
RL: D (Extremely Remote)
RS: 4 (Minor)
RRL: 4D (Low)
Consequence (Safety Effect)
E_2
Runway centerline markings not visible or obscured
Likelihood: A (Frequent)
B_3
Controls
B_4
Redundancy
Use redundant localization as alternative reference for heading, and location
Barrier Integrity: 0.88
Perception Failover
Disengage ML-based perception when deviation from reference location ≥ d meters or duration of deviation ≥ t seconds, then command a speed reduction
Barrier Integrity: 0.75

Figure 2: Annotated BTD fragment for an autonomous taxiing capability, showing how a lateral runway overrun is mitigated under specific initiating events leading to centerline tracking violation.

3 Motivating Example

We motivate this work using an aviation domain use case of autonomous aircraft taxiing [1]. This system uses a *runway centerline tracking* function comprising a classical controller coupled to a deep convolutional neural network that estimates aircraft position from optical sensor data. The functional objective is to maintain both the *cross-track error* (CTE) and the *heading error* (HE) within pre-defined bounds. CTE is the horizontal distance between the runway centerline and the aircraft body (or *roll*) axis; HE is the angle between the respective headings of the runway centerline and the roll axis. The safety objective is to avoid a lateral *runway overrun* (also known as a *runway excursion*), i.e., departing the sides of the runway.

Fig. 2 shows a BTD fragment for this example (annotated to show its graphical elements and their identifiers) as a view of its wider safety architecture (not shown), which composes [8] similar such BTDs, albeit for different operating contexts, threats, top events, and consequences. Here, the operating context involves a relatively low speed (25 kn), low visibility taxi operation on a wet runway, at dusk, under no crosswind conditions. The hazard to be controlled (E_3) is a violation of the allowed lateral offset from the runway centerline, failing which a lateral runway overrun (E_4) could occur. Two (out of many) initiating causes for this hazard have been shown: a controller malfunction that steers the aircraft away from the centerline when not required (E_1); and runway centerline markings that are not visible, or are obscured (E_2).

3.1 Baseline Safety

To characterize the safety risk level of an operating scenario, we use the risk assessment model associated with the safety architecture to establish a *baseline level* of operational safety risk for the identified safety effects.

For the scenario in Fig. 2, the *initial risk level* (IRL) of the consequence event E_4 is labeled $\mathsf{4A(Medium)}$. That is, E_4 has a *medium* level of unmitigated risk, and is assigned the *risk classification category* $\mathsf{4A}$. That refers to a region of the overall risk space that has been discretized using a classical 5×5 risk matrix of categories of consequence event probability, ranging from *Frequent* (A) to *Extremely Improbable* (E), and consequence event severity, ranging from *Minimal* (5) to *Catastrophic* (1). For a definition of those categories, see [10]. A similar interpretation applies to *residual risk level* (RRL) which, for E_4, is shown as $\mathsf{4D(Low)}$, representing the risk remaining after mitigation using the indicated barriers and the associated controls. Specifically, B_1: *Runtime Monitoring*, B_2: *Controller Failover*, B_3: *Redundancy*, and B_4: *Perception Failover*, serve as prevention barriers for exceeding the allowed CTE, while B_5: *Emergency Braking* is a recovery barrier invoked after the top event occurs.

For aeronautical applications, civil aviation regulations and the associated certification or approval processes generally establish what constitutes *acceptable* and *approved* baseline risk levels respectively. The two can be the same (though they need not be) and, typically, are given in terms of a so-called *target level of safety* (TLOS), which specifies the (maximum acceptable) probability of the undesired safety effect per unit of operational exposure, e.g., 10^{-6} lateral runway excursions per taxi operation. How TLOS is established and approved is out of scope for this paper[3]; as such, in Fig. 2, either of the values of the IRL, $\mathsf{4A(Medium)}$, or the RRL, $\mathsf{4D(Low)}$, may plausibly meet the TLOS, and therefore could be an approved baseline level of safety risk. For the purposes of this example, we assume that the RRL shown is the approved baseline that meets the TLOS. Once a system is deployed, note that the RRL for an event is, in fact, *dynamic*, i.e., as a sequence of values starting from the approved baseline, it represents how the risk of that event evolves over the system lifetime (also see Fig. 3).

3.2 Practical Drift

Some barriers or controls in the safety architecture of a system may be relaxed in operation to improve the performance of system services and/or to make local optimizations that address the operating context. In our running example, for instance, to increase runway throughput whilst operating on large runways in better environmental conditions (e.g., clear weather, and dry runway surface), the time an aircraft spends on a runway could be reduced. For that purpose, suppose that disengagement of the perception function or the controller is delayed (see Fig. 2), or that more permissive CTE bounds are admitted. In those cases, the system may enter certain states that would have been prohibited otherwise. In particular, such states represent violations of the barrier/control requirements that were stated as claims in the pre-deployment safety case.

However, when there is improved system performance without observed safety consequences or mishaps, those states are not perceived as violations that increase residual risk. This can lead to misplaced assurance in operational safety when the system as operated deviates from its safety case. Practical drift can then emerge when multiple safety mitigations may be progressively loosened, and continued, incident-free system operations under such changes obscure the increase in operational safety risk. It is important to emphasize that relaxing mitigations to improve performance represents an operational tradeoff rather than a deliberate attempt to subvert safety. An analogy, for example, is highway driving at the speed of traffic that exceeds the posted speed limits—a practice that is not always unsafe, but poses higher risk in general.

4 Framework

Dynamic assurance within a safety case approach gives a proactive means to assess and contain practical drift through continued assurance that the operational safety risk level for the system is aligned with its approved baseline (see Fig. 3). A framework that enables this must at least: (i) characterize how operational safety risk levels have changed; (ii) determine which mitigations, if any, may be legitimately relaxed without safety deteriorating; and (iii) identify the necessary modifications to both the system and its safety case, so that the two are mutually consistent during system operation. Next, we discuss how relating safety performance measurement to the safety architecture in a safety case, in addition to its arguments, gives the necessary elements and technical foundations for the first of the preceding three requirements—the main focus in this paper. The examples presented next are meant to be illustrative and not comprehensive.

4.1 Defining Safety Metrics and Indicators

A safety architecture and its associated risk assessment model [8] give a basis to allocate safety targets to the safety functions, and subsequently confirm them (analytically and empirically). TLOS is a system-level safety target always assigned to consequence event probability. Decomposing and allocating the TLOS across the elements of the safety

[3]Interested readers may refer to [3].

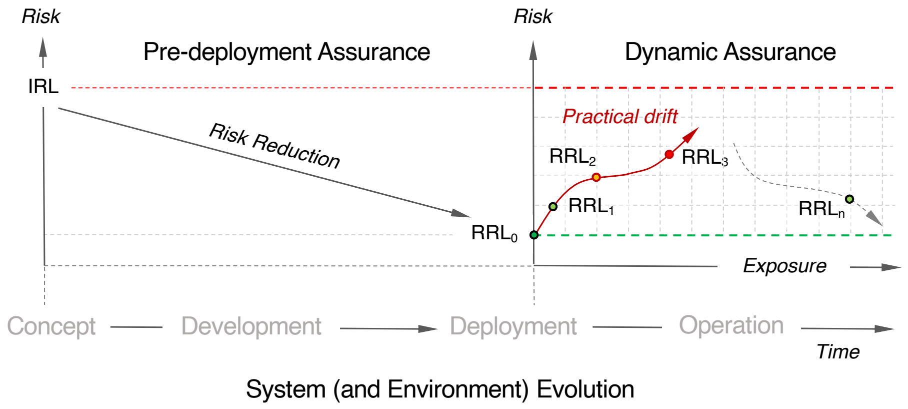


Figure 3: Pre-deployment assurance gives justified confidence in the reduction of an initial risk level (IRL) of the safety effects in a system concept, through system development, to a baseline residual risk level (RRL_0) that meets the TLOS at deployment. Dynamic assurance provides confidence in operation that the approved baseline is maintained, by identifying and managing practical drift.

architecture gives the safety integrity targets for barriers and controls, along with precursor event probabilities that we interpret as *scenario-specific safety targets*. Relating safety targets to safety performance measurement in general, and safety indicators (SIs) in particular, facilitates tracking and confirming that the mitigations are performing in operation as intended. One way to embed TLOS into an SI is by simply converting the corresponding probability value into an event frequency threshold applied to an appropriate safety metric used during development or in operation. In this section we focus on the operational safety metrics, addressing the metrics used during development in Section 4.2.

TLOS and the corresponding SIs can be *generic*, i.e., apply to all relevant operating contexts of a safety architecture, or *scenario-specific*, i.e., applicable to a particular operating context. For instance, let the TLOS for lateral runway overrun under all relevant operating conditions of the example system be 10^{-6} per taxi operation. We can then define a corresponding generic SI, $\mathcal{I}_{\texttt{LRE}}$: $\texttt{opLatRwyEx} \leq 1$ in 10^6 taxi operations, where `opLatRwyEx` is an operational safety metric[4] for the *number of lateral runway overrun events in operation*, whose threshold value is 1, measured over an exposure of 10^6 taxi operations. Another commonly used unit of exposure is *flight hours* [11], and the SI can be given equivalently as $\mathcal{I}_{\texttt{LRE}} : \texttt{opLatRwyEx} \leq 1$ in $10^6 \times t$ flight hours, where t is the average time in flight hours of a taxi operation.

Scenario-specific SI definition proceeds in the same way, but is applied to specific operating contexts after first decomposing and allocating the TLOS of a consequence event to its scenario-specific instance. For example, if 10% of all taxi operations occur under the operating context of Fig. 2, then we can modify the exposure of $\mathcal{I}_{\texttt{LRE}}$ to 10^5 taxi operations to get the scenario-specific SI for the consequence event E_4.

Similarly, we can define generic and scenario-specific SIs for the remaining safety architecture elements by converting the associated event probability and barrier integrity values as applicable. Moreover, recalling that a barrier can have its own safety architecture (Section 2.1), we can iteratively define SIs for the lower layers of a system hierarchy. Thus, in Fig. 2, we can define the scenario-specific SI for the barrier B_4: *Perception Failover* as $\mathcal{I}_{\texttt{PFO}}$: $\texttt{opPcpDisEngF} \leq y$ in n taxi operations. In Section 4.4, we illustrate one approach to instantiate y and n.

Here, `opPcpDisEngF` is a metric related to the integrity of B_4 (itself a metric) that counts the *number of failed disengagements of ML-based perception in operation*; its threshold value is y, to be measured over an exposure of n taxi operations conducted in the stated operating context for the specified scenario. This metric relies upon a precise definition of a *failed disengagement* (not given here), which may itself be given in terms of other metrics, e.g., those associated with its *functional deviations* (i.e., violation of the requirements for the barrier, its constituent controls, or their verification), and its *failure modes* (of the physical systems to which the barrier function is allocated). Additional operational safety metrics related to barrier integrity include `opTxLowVisW`, counting the *number of taxi operations conducted at dusk under low visibility, no crosswind, and wet runway conditions* (i.e., the operating context of Fig. 2), from which we may infer the *number of successful disengagements of ML-based perception* as the metric $\texttt{opPcpDisEngS} = \texttt{opTxLowVisW} - \texttt{opPcpDisEngF}$.

[4]Henceforth, identifiers with the prefix ‘`dev`’ refer to metrics used during system development, and the prefix ‘`op`’ indicates an operational safety metric.

4.2 Updating and Revising the Operational Risk Assessment

A pre-deployment safety case represents what (we believe) a system design achieves at deployment, and will continue to achieve in operation. Some of the metrics and SIs applicable during system development constitute measurement evidence verifying safety performance, e.g., during pre-deployment system testing or flight testing. Thus, by associating those metrics and SIs with the safety architecture, we get the *prior* values of event probability and barrier integrity. For the scenario and operating context of Fig. 2, some of the metrics used during system development for the barrier B_4 are: `devTxLowVisW`: the *number of tests for* $\mathsf{B}_4 = t$ (say); `devPcpDisEngS`: the *number of successful disengagements of ML-based perception* $= s$; and `devPcpDisEngF`: the *number of failed disengagements of ML-based perception* $= (t - s) = f$.

If the test campaign during system development is designed as a Bernoulli process [16] then we can model the sequence of test results as a binomial distribution, $\texttt{Binom}\,(\chi : \eta, \theta)$, whose parameters are χ: the number of successes, η: the number of independent trials, and θ: the probability of success in each trial. Hence, we can assign the values of the metrics `devPcpDisEngS` and `devTxLowVisW`, respectively, to the first two parameters as $\chi := s$, and $\eta := t$. Let $\theta := p$, the unknown (fixed) probability that each test produces a successful disengagement. We can model p as the conjugate prior beta distribution, $\pi(p) \sim \texttt{Beta}\,(\alpha, \beta)$. The *hyperparameters* (i.e., the parameters of the prior distribution) represent our prior knowledge of the number of successful and failed tests during development. Hence we assign to them the values of the metrics `devPcpDisEngS` and `devPcpDisEngF`, respectively, as $\alpha := s$, and $\beta := f$. The beta distribution mean, $\mu_p = {}^{s}\!/_{t}$, gives a point estimate of the prior barrier integrity, and its variance, $\sigma_p^2 = {}^{sf}\!/_{t^2(t+1)}$, gives the uncertainty in that estimate.

In operation, safety performance measurement yields a sequence of observations of the state of the safety system. We can transform this data into a *likelihood function*, i.e., a joint probability of the observations given as a function of the parameters of a model of the underlying data generation process. In our example, a binomial probability density function (PDF) is a reasonable *initial* model (i) assuming that the pre-deployment safety case provides the argument and evidence that testing is representative of actual operations (as would likely be necessary), and (ii) since a binomial distribution models the sequence of test results. Thus, supposing that over n taxi operations conducted in the operating context of Fig. 2, there were x failures to disengage ML-based perception on demand. We now have the operational safety metrics `opTxLowVisW` $= n$, and `opPcpDisEngF` $= x$, so that `opPcpDisEngS` $= (n - x) = y$, and the likelihood function is $\mathcal{L}\,(p|n, y) = \binom{n}{y}\, p^y (1 - p)^x$. As before, p is the unknown probability of a successful disengagement of ML-based perception on a random demand, representing a surrogate measure of barrier integrity.

Bayesian inference gives the formal procedure to update the priors into *posterior* values of barrier integrity (and event probability), which represent what the operational system *currently* achieves. Thus, for our running example, the posterior integrity for B_4 is given by (the proportional form of) Bayes' theorem as $\pi(p|y) \propto \mathcal{L}\,(p|n, y) \times \pi(p)$. Since the beta prior and the binomial likelihood are a conjugate pair, the posterior has a closed form solution, $\pi(p|y) \sim \texttt{Beta}\,(s + y, (t - s) + x)$. The distribution mean, $\mu_{p|y} = {}^{(s+y)}\!/_{(t+n)}$, is the updated point estimate of barrier integrity. To get a revised assessment of the operational safety risk level for the system, we propagate the posterior barrier integrity through the risk assessment model underlying the safety architecture.

4.3 Characterizing the Change to Safety Risk

We use *risk ratio* (RR), a metric of relative risk, to quantify the change in operational safety risk. In operation, the RR for a consequence event is the ratio of its current estimated probability of occurrence and the approved baseline. More generally, we will (re)compute the RR for any event of interest in the safety architecture, typically after the operational risk assessment has been revised (as in Section 4.2) as the ratio of its updated (i.e., prior or posterior, as appropriate) probability to its (scenario-specific) safety target. Denoting the RR for event E_i by $\texttt{RR}(\mathsf{E}_i)$, $\texttt{RR}(\mathsf{E}_i) > 1$ indicates an increase in the safety risk of E_i. Similarly, $\texttt{RR}(\mathsf{E}_i) < 1$ indicates a decrease, while $\texttt{RR}(\mathsf{E}_i) = 1$ indicates no change. By itself, RR reflects how effective the safety architecture is in reducing the risk of the identified safety effects.[5] By considering the trend of RR over time, we can construct a powerful SI of practical drift, e.g., by fitting a linear trend line to a temporally ordered sequence of RR values computed over some pre-determined exposure, the sign and magnitude of the slope indicate, respectively, the direction and the rate of the change in safety risk.

[5] RR has also been used as a development safety metric, e.g., in designing aircraft collision avoidance systems [9].

4.4 Numerical Examples

We now give some numerical examples to concretize the preceding discussion.

Example 1 (Prior Barrier Integrity)**.** During the development of our running example system and its pre-deployment safety case, assume we have a total of $\texttt{devTxLowVisW} = 32$ flight tests in which there are $\texttt{devPcpDisEngF} = 8$ failing tests for the *Perception Failover* barrier. Thus a prior distribution for its integrity is $\pi(p) \sim \texttt{Beta}(24, 8)$, whose mean is $\mu_p = 0.75$, and variance is $\sigma_p^2 = 0.0057$. The mean gives a point prior value of barrier integrity which we show in the corresponding node in the BTD of Fig. 2.

Example 2 (Scenario-specific Barrier Safety Indicator)**.** Recall that a scenario-specific SI for the *Perception Failover* barrier is $\mathcal{I}_{\texttt{PFO}}$: $\texttt{opPcpDisEngF} \leq y$ in n taxi operations (Section 4.1). As before, $\texttt{opPcpDisEngF}$ measures the number of failed disengagements of ML-based perception in operation. To determine a suitable exposure n and threshold y, consider that a conservative range of values for p that would provide the same, or better, risk reduction performance as its prior mean is the closed interval $[\mu_p + \sigma_p, 1] = [0.8254, 1]$. In other words, observing 8 or more successful disengagements or, equivalently, 2 or fewer failed disengagements on demand of ML-based perception over at least 10 taxi operations conducted in the specified operating context would validate the safety performance of the barrier. Thus, here, $n = 10$ and $y = 2$.

Example 3 (Likelihood of Data and Posterior Integrity)**.** After system deployment, suppose that to improve runway utilization, the control in B_4 (see Fig. 2) is relaxed such that ML-based perception is disengaged after a larger distance (or duration) of position deviation than what was specified in the safety architecture. The metric that records the number of failed disengagements in operation, $\texttt{opPcpDisEngF}$ (Section 4.2), includes violations of the barrier requirement as initially specified, which itself includes violations of the barrier requirement after operational modification. That is, the operational safety metric *should not be modified* even though the barrier function has been operationally changed. Supposing $\texttt{opPcpDisEngF} = 4$ violations have been observed over $\texttt{opTxLowVisW} = 10$ taxi operations. Given this data, the likelihood function is $\mathcal{L}(p|6) = \binom{10}{4} p^6 (1-p)^4$, and the posterior distribution is $\pi(p|6) \sim \texttt{Beta}(30, 12)$, whose mean is $\mu_{p|6} = 0.7143$ and variance is $\sigma_{p|6}^2 = 0.0047$.

Example 4 (Operational Safety Risk Update)**.** We assume prior data is available (from characterizing the ODD [14] for the autonomous taxiing function) on how often runway markings are obscured during taxiing due to runway surface and weather conditions. Hence we can give a prior distribution, say $\pi(\mathsf{E}_2) \sim \texttt{Beta}(10, 190)$, whose mean is the prior point estimate $\texttt{Pr}(\mathsf{E}_2) = 0.05$. Similarly, let $\texttt{Pr}(\mathsf{E}_1) = 0.05$. Given these priors and the barrier integrity values as in Fig. 2, the prior probability of the consequence event is $\texttt{Pr}(\mathsf{E}_4) = 1.5998 \times 10^{-5}$ corresponding to an RRL of $\mathsf{4D(Low)}$. We recall from Example 3 that 4 barrier violations were observed in 10 taxi operations. Hence E_2 must have occurred on $z = 4$ occasions for B_4 to have been invoked and have failed on demand. Thus, we may reasonably model this event as a Bernoulli process with a binomial PDF as the likelihood function for the observed data. Thus, the posterior distribution over E_2 is $\pi(\mathsf{E}_2|z) \sim \texttt{Beta}(14, 196)$ so that $\mu_{\mathsf{E}_2|z} = 0.0667$ is the point posterior for $\texttt{Pr}(\mathsf{E}_2)$. Propagating both the posteriors for E_2 and B_4 through the risk assessment model of the safety architecture [8], we get the updated prior $\texttt{Pr}(\mathsf{E}_4) = 2.386 \times 10^{-5}$ for the consequence event. The corresponding RRL remains unchanged suggesting that the operational modification to B_4 may be acceptably safe.

Example 5 (Safety Risk Level Change and Practical Drift)**.** The risk ratio for the consequence event E_4 given the change to barrier B_4 (as in Example 3) is $\texttt{RR}(\mathsf{E}_4) = {}^{2.386}/_{1.5998} \approx 1.49$. Thus, despite an unchanged risk level (see Example 4), the RR indicates increasing safety risk. Now, further suppose that to improve runway utilization, a greater deviation in CTE from the stated bounds is operationally admitted (see the top event E_3 in Fig. 2). Consequently barrier B_5 needs to be relaxed to be invoked after a longer duration than specified (see Fig. 2). Suppose that the posterior integrity computed from operational safety metrics (omitted here due to space constraints) is $\texttt{Pr}(\mathsf{B}_5) \approx 0.96$. In this case, the revised prior for E_4 is $\texttt{Pr}(\mathsf{E}_4) \approx 2.4 \times 10^{-4}$, the revised RRL is $\mathsf{4C(Medium)}$, and $\texttt{RR}(\mathsf{E}_4) \approx 10$. The updated RRL now violates the TLOS even if no safety effects may have been observed. Moreover, the modifications to the barriers B_4 and B_5 are at least *an order of magnitude more likely* to result in a lateral runway overrun, indicating an appreciable increase in safety risk relative to the approved baseline, and suggests practical drift.

5 Towards Formal Foundations

As mentioned earlier (Section 2.2), we want to formalize a notion of *consistency* between the static portion of the safety case (i.e., its assurance artifacts, see Fig. 1) and the collection of indicators that constitute the SMB. The safety metrics and indicators represent the objectively quantifiable content referenced in the arguments and the safety architecture, which in turn provide the justification for how the metrics and indicators collectively provide safety substantiation. Although operational safety measurement entails updating and revising the risk assessment (Section 4), changing the SMB may not be necessary. However, in situations where replacing, modifying, or adding metrics and indicators is required—e.g., to reflect new observable phenomena in the environment—the SMB will change and so would the associated assurance artifacts to retain consistency. Note that currently we are not considering changes that would entail modification of the safety architecture (e.g., replacing a barrier). Hence we exclude that from our notion of consistency for now and focus on consistency with arguments.

We can achieve this consistency if the argument structure reflects the risk reduction rationale implicit in the safety architecture. That is, the form of the argument structure proceeds from all terminating consequence events in the safety architecture, working recursively backwards (i.e., leftwards) to all initiating (leftmost) threat events. Thus, each level of the argument has the following form: all consequence events are acceptably mitigated (i.e., the residual risk level meets the allocated TLOS), *which is supported by the argument that*: all their identified precursor events (causes) are acceptably mitigated, *which is supported by the argument that*: (a) all applicable barriers are operational and effective, and (b) all causes have the stated probability of occurrence. In a GSN representation of this argument, the leaves are *solution nodes* [20] that have the following *evidence assertion*: the initiating threat has the stated (assumed) probability.

Thus, the overall argument states that if the barriers are effective and operational, and the events have the assumed probabilities, then the consequences have acceptable risk levels. Indicators map into the corresponding claims of barrier effectiveness and event probabilities, serving to monitor that those values are within the required limits.

Now we briefly outline how to place this consistency on a more rigorous basis. Let $\mathbf{Arg}$ and $\mathbf{SMB}$ represent the sets of well-formed arguments and SMBs, respectively, and define mappings $F : \mathbf{Arg} \to \mathbf{SMB}$ and $G : \mathbf{SMB} \to \mathbf{Arg}$, such that F extracts the associated indicators from an argument, and G embeds an SMB into a skeleton argument of the form outlined above. Then we require that $F; G \leq I$ and $G; F = I$ (where I is the identity mapping), where arguments are ordered by *refinement*. The first inequality ensures that the argument contains the necessary rationale for the SMB, with the refinement allowing that the argument can contain additional reasoning; the second ensures that all quantifiable components of the argument are represented in the SMB.

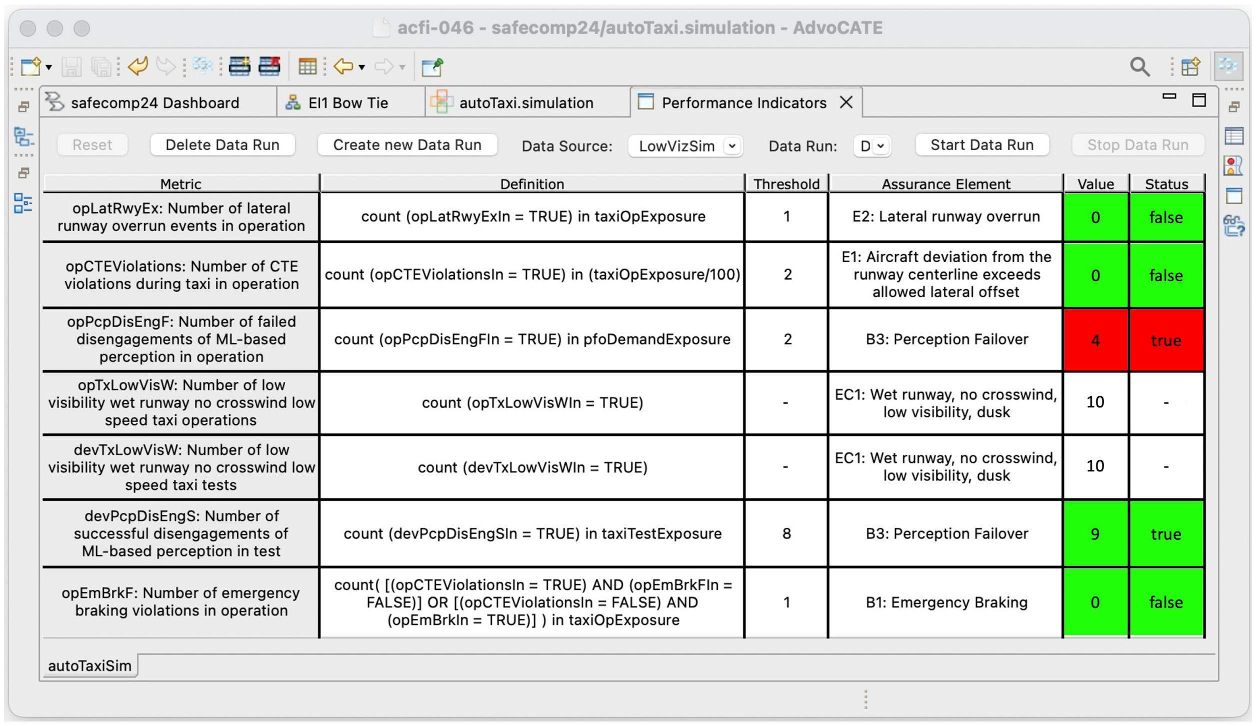
6 Concluding Remarks

We have a preliminary implementation of the SMB in AdvoCATE that currently supports the following functionality: real-time import of data (i.e., measures) from multiple data sources (simulations or feeds from external sensors); computation of derived metrics and indicators over multiple data runs; and tracing to assurance artifacts (events and barriers in the safety architecture, and goals and assumptions in the safety arguments). We display indicators and the associated assurance artifacts in a dynamically updated table (Fig. 4 shows an example) that highlights when the conditions on the indicator thresholds have been met (in green) or have not been met (in red). A dashboard (not shown) allows selection between the various metrics of the SMB with charts displaying real-time updates of their values as well as other dynamically updated risk status, such as hazards ordered by risk level, and barriers ordered by integrity.

The goal of managing practical drift has mainly informed our choice of safety metrics and indicators. We plan to leverage the *Goal Question Metrics* (GQM) approach [2] to define additional metrics suitable for other dynamic assurance goals, e.g., improving functional performance whilst maintaining safety.

A binomial likelihood may be only initially appropriate for certain kinds of measurement data. Indeed, as more data is gathered, the preconditions for using a binomial PDF need to be reconfirmed. As such, it may be necessary to use other PDFs for the likelihood of the data, along with numerical methods for Bayesian inference. Our choice of beta priors is motivated, in part, by computational convenience, its flexibility to approximate a variety of distributions, and the domain-specific interpretation of the distribution parameters in different safety metrics. Although we represent the uncertainty in barrier integrity and event probability by specifying their distributions in the theoretical framework, our prototype implementation currently represents and propagates their point values (i.e., the distribution means) for both the pre-deployment risk assessment, and the revisions of the operational risk assessment. We plan to refine this approach by also propagating the uncertainties through the risk assessment model so as to quantify the corresponding uncertainty in the residual risk of the safety effects of interest. By so doing, we aim to ground the quantification of assurance in safety measurement.

Since TLOS is typically assigned to rare events, legitimate concerns can arise about the credibility of using quan-



Data Source: LowVizSim Data Run: D

Metric	Definition	Threshold	Assurance Element	Value	Status
opLatRwyEx: Number of lateral runway overrun events in operation	count (opLatRwyExIn = TRUE) in taxiOpExposure	1	E2: Lateral runway overrun	0	false
opCTEViolations: Number of CTE violations during taxi in operation	count (opCTEViolationsIn = TRUE) in (taxiOpExposure/100)	2	E1: Aircraft deviation from the runway centerline exceeds allowed lateral offset	0	false
opPcpDisEngF: Number of failed disengagements of ML-based perception in operation	count (opPcpDisEngFIn = TRUE) in pfoDemandExposure	2	B3: Perception Failover	4	true
opTxLowVisW: Number of low visibility wet runway no crosswind low speed taxi operations	count (opTxLowVisWIn = TRUE)	-	EC1: Wet runway, no crosswind, low visibility, dusk	10	-
devTxLowVisW: Number of low visibility wet runway no crosswind low speed taxi tests	count (devTxLowVisWIn = TRUE)	-	EC1: Wet runway, no crosswind, low visibility, dusk	10	-
devPcpDisEngS: Number of successful disengagements of ML-based perception in test	count (devPcpDisEngSIn = TRUE) in taxiTestExposure	8	B3: Perception Failover	9	true
opEmBrkF: Number of emergency braking violations in operation	count([(opCTEViolationsIn = TRUE) AND (opEmBrkFIn = FALSE)] OR [(opCTEViolationsIn = FALSE) AND (opEmBrkIn = TRUE)]) in taxiOpExposure	1	B1: Emergency Braking	0	false

Figure 4: AdvoCATE screenshot: Table of safety indicators for the example system in Section 3.

titative methods as in this paper. Though we have yet to explore how *conservative Bayesian inference* [19] could be used in our approach, relative risk metrics such as risk ratio (RR) are a step towards circumventing those concerns.

Practical drift is distinct from *operating environment drift* in that the former results from changes within the system boundary, whereas the latter occurs outside that boundary. We reflect the assumptions about the operating environment in the pre-deployment safety case, for example, as the prior probabilities (conditional on the operating context) associated with the threat events. We can reflect environment drift via the posterior distributions of the corresponding event probabilities updated by operational safety metrics associated with the respective events (see Section 4.2, and Example 4). We additionally distinguish *runtime risk assessment* [1], from the update and revision of operational risk as described in this paper: the former occurs during the shorter time span of a mission (e.g., during a taxi operation), whereas the latter occurs over longer time intervals, between missions, and through the lifecycle of the system (e.g., over multiple taxiing operations, possibly involving an aircraft fleet).

The numerical examples (Section 4.4) have described a scenario-specific application of our approach, where the event probabilities and barrier integrities are *conditional* on the operating context. For a system-level characterization of how operational risk changes, we must consider the *marginal* probabilities and integrities in the overall safety architecture that composes different risk scenarios. However, we have not considered it in this paper, and it is one avenue for future work.

To further develop our proposed dynamic assurance framework we aim to explore how by thresholding, ranking, and comparing RR under changes made to individual mitigations or their combinations, we may infer: (i) which mitigations may be optimized for system performance whilst maintaining safety (possibly necessitating a change to the safety architecture itself); and, in turn, (ii) which system and safety case changes may be necessary. Some changes may be automated while will induce *tasks* requiring manual attention [5]. Additionally, we aim to define a tool-supported methodology on top of the main components of the framework. This will involve defining and formalizing the methods and procedures to decompose and allocate safety targets, derive safety indicators, and close the safety assurance loop, i.e., maintain consistency of the arguments with the SMB) through targeted changes to the system and its safety case.

Observations of system operations constitute one specific form of evidence that we can use to reason about system safety. We seek to systematize this through a notion of *evidence requirement* that will also cover *static* data. We are also extending the metrics expression language to express trends, although work remains to integrate it into our methodology and to relate it to the concept of *safety objective*. A need to update the SMB, e.g., modify indicators and possibly their thresholds, accompanies operational safety measurement. We aim to better understand the principles that underlie those modifications and, subsequently, implement the corresponding tool features. However, practically deploying this framework will necessitate harmonizing with existing safety management system (SMS) [10] infrastructure, whilst carefully considering the roles of different stakeholders in safety performance monitoring, measurement, and assurance.

References

[1] Asaadi, E., Denney, E., Menzies, J., Pai, G., Petroff, D.: Dynamic Assurance Cases: A Pathway to Trusted Autonomy. IEEE Computer **53**(12), 35–46 (2020). https://doi.org/10.1109/MC.2020.3022030

[2] Basili, V., Caldiera, G., Rombach, D.: Goal Question Metric Paradigm, pp. 528–532. Encyclopedia of Software Engineering, John Wiley & Sons, Inc., 2nd edn. (1994)

[3] Busch, A.C.: Methodology for Establishing a Target Level of Safety. Technical Report DOT/FAA/CT-TN85/36, US DOT, FAA Technical Center (Aug 1985)

[4] Calinescu, R., Weyns, D., Gerasimou, S., Iftikhar, M.U., Habli, I., Kelly, T.: Engineering Trustworthy Self-Adaptive Software with Dynamic Assurance Cases. IEEE Transactions on Software Engineering **44**(11), 1039–1069 (Nov 2018)

[5] Denney, E., Habli, I., Pai, G.: Dynamic Safety Cases for Through-life Safety Assurance. In: 37th Intl. Conference on Software Engineering - Vol. 2, pp. 587–590. (May 2015). https://doi.org/10.1109/ICSE.2015.199

[6] Denney, E., Johnson, M., Pai, G.: Towards a Rigorous Basis for Specific Operations Risk Assessment of UAS. In: 37th IEEE/AIAA Digital Avionics Systems Conference (2018). https://doi.org/10.1109/DASC.2018.8569475

[7] Denney, E., Pai, G.: Tool Support for Assurance Case Development. Journal of Automated Software Engineering **25**(3), pp. 435–499 (Sep 2018). https://doi.org/10.1007/s10515-017-0230-5

[8] Denney, E., Pai, G., Whiteside, I.: The Role of Safety Architectures in Aviation Safety Cases. Reliability Engineering and System Safety **191** (2019). https://doi.org/10.1016/j.ress.2019.106502

[9] Edwards, M., Mackay, J.: Determining Required Surveillance Performance for Unmanned Aircraft Sense and Avoid. In: 17th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference. AIAA 2017-4385 (Jun 2017)

[10] FAA Air Traffic Organization: Safety Management System Manual (Dec 2022)

[11] US Dept. of Transportation, FAA: Safety Risk Management Policy. Order 8040.4C (2023)

[12] Hawkins, R., Conmy, P. R.: Identifying Run-time Monitoring Requirements for Autonomous Systems through the Analysis of Safety Arguments. In: 42nd Intl. Conference on Computer Safety, Reliability, and Security (SAFECOMP), LNCS 14181 (Sep 2023)

[13] International Civil Aviation Organization (ICAO): Safety Management Manual (Doc 9859), 4 edn. (2018)

[14] Kaakai, F., Adibhatla, S., Pai, G., Escorihuela, E.: Data-centric Operational Design Domain Characterization for Machine Learning-based Aeronautical Products. In: 42nd Intl. Conference on Computer Safety, Reliability, and Security (SAFECOMP), LNCS 14181 (Sep 2023)

[15] Koopman, P.: How Safe is Safe Enough? Measuring and Predicting Autonomous Vehicle Safety. 1st edn. (2022)

[16] Ladkin, P.: Evaluating Software Execution as a Bernoulli Process. Safety-Critical Systems eJournal **1**(2) (July 2022)

[17] Reich, J., Trapp, M.: SINADRA: Towards a Framework for Assurable Situation-aware Dynamic Risk Assessment of Autonomous Vehicles. In: 16th European Dependable Computing Conference (EDCC). pp. 47–50 (Sep 2020). https://doi.org/10.1109/EDCC51268.2020.00017

[18] Schleiss, P., Carella, F., Kurzidem, I.: Towards continuous safety assurance for autonomous systems. In: 6th Intl. Conference on System Reliability and Safety (ICSRS 2022). pp. 457–462 (2022). https://doi.org/10.1109/ICSRS56243.2022.10067323

[19] Strigini, L.: Trustworthy Quantitative Arguments for the Safety of AVs: Challenges and some Modest Proposals. In: 1st IFIP Workshop on Intelligent Vehicle Dependability and Security (IVDS) (Jan 2021)

[20] The Assurance Case Working Group (ACWG): Goal Structuring Notation Community Standard Version 3. SCSC-141C (May 2021) https://scsc.uk/r141C:1

[21] Trapp, M., Weiss, G.: Towards Dynamic Safety Management for Autonomous Systems. In: 27th Safety-Critical Systems Symposium (SSS). pp. 193–204. (Feb 2019)