Table of Contents

Appendix E AIC Systems Approach Key Concepts Extended Definitions

The primary purpose of the AIC systems approach is to forge an Ideal System out of a chaotic, harder-to-predict, and more complex phenomenon involving autonomous, non-autonomous, living, and non-living systems into an easier-to-predict and more comprehensible system. The whole makes up the architect's epistemic complexity field where all that needs to be known exists. However, to achieve complete coverage of all that needs to be known, the architect requires a systematic and iterative approach, along with a comprehensive ontology, to reduce epistemic uncertainty. Architectural epistemic uncertainty can stem from gaps in knowing what needs to be observed and understanding the complexity of observed field dynamics (past, present, and future). Thus, a solution for this (as well as the other sources of epistemic uncertainty) is a comprehensive categorical system that intends to enable the architect to think laterally and vertically.

To achieve this, we require an ontology that maps perceived information into reasoned, categorical knowledge. Hence, the AIC systems approach is a systematic knowledge acquisition activity based on general universal systems theory known as AIC Systems Theory. In simple terms, AIC stands for Appreciation, Influence and Control. The general premise here is that all systems in the universe, in any scenario, form three categories of relationships: AIC. As described by its founder, Dr Smith (see literal review), the current system theory describes AIC as a form of purpose and actions. We adapt the theory to make it more practical for systems engineering and formulate the following concepts, some of which are inspired by AIC philosophy. In contrast, others are derived from our understanding of general systems and interpretations of prior art.

E.1 Environment, Architect and Machine Uncertainty

We encountered uncertainty in several aspects of our research. During our experiments, we needed to convey how we demonstrated the discovery of Black Swans (rare but impactful events, such as a moon appearing to be a red traffic light for Tesla's ML component) and mitigated them in a pictorial format. We needed to establish the conceptual link between mitigating Black Swans and ensuring ourselves and a regulator (for example, CAA or clients) that emergent properties are preserved during unforeseen scenarios (the expected properties of ML models). Another instance of confronting the concept of "uncertainty" occurred during our attempt to understand the nature of open, complicated problem domains. We needed to distinguish between the concepts of "complexity" and "complicatedness". We also encountered uncertainty when we tried to make sense of predictive thinking and the role of lateral thinking in solving hard-to-predict interactions.

Taking a first-principles approach, accounting for all possible interactions among components in a problem environment, helps predict Black Swan scenarios. This approach led us to confront specific interactions that we could not understand, such as the interaction between trees' oscillatory motion and traffic over a bridge.

Because of those experiences, we were forced to study uncertainty as a topic, and later we realised that our approach can be described objectively as about solving the uncertainty reduction problem. We realised that we are trying to reduce the architect's (via a first principles approach to problem analysis) and the machine's uncertainty (via pictorial datasets) about the problem complexity. We realised that the real challenge we face in the observed problem is the predictability of the probability distribution of events. In other words, how can an architect (and by extension a regulator or a customer) be confident (certain) in their predictions of the likelihood of an event to occur in the real world and what events may occur immediately or at a later stage of an AS deployment.

Unlike traditional systems, in which the architect can have assured, proven certainty on some environmental and system behaviours, where the architect can afford to ignore some events as rare. In our problem, the architect can not afford to ignore rare events since some of them may have a bizarre impact on AS behaviour, like a half-visible person walking with their bike is not a person (see the famous self-driving Uber car accident [3]). How could the uber design team made up of talented and experienced individuals could have thought of such a scenario?

Although the driver eventually pleaded guilty, we believe that the systems engineering process is partially to blame for such a tragic accident. NSTB found that Uber fostered "inadequate safety culture" [4] and that is a holistic problem that can be easily rectified by a more suitable holistic paradigm that prompts the design team to take a different approach to safety of autonomous systems. One that takes Black Swan scenarios more seriously. This led us to understand that to predict Black Swan scenarios, we needed to understand uncertainty in general and then how our approach manages and manipulates it.

We further support our motivation to understand complexity and complicatedness through SACE observation of the nature between AS and its environment. The expert authors of SACE (Safe Assurance of Autonomous Systems in Complex Environments), guidance suggest the following observation about the relationships between autonomous systems and an open operational environment:

> "The different possible scenarios and environmental states that relate to the decision can then be identified by considering the real-world state and the belief state of the AS at the point at which the decision is made, along with each of the possible options. The real-world state represents the actual state of the operating environment, whilst the

belief state represents the understanding that the AS has of the state of the operating environment"

This prompts the question:

How do we account for all possible operational environment states?

Now we don't only have to worry about the AS states and their failure modes, but also the states of all other systems around the AS and their failure mode. But then, how can the surrounding space between, for example, train tracks and a boundary fence fail in such a way that we need to account for potential responses from a security drone tasked with performing security patrols over train tracks?

Do you notice the level of uncertainty we are dealing with? What do you reckon, whose uncertainty is at stake here?

Hence, we learned that one of the differences between the role of a traditional systems architect and AS architect, the latter is required to design the whole complexity field of the problem domain, not only the system of interest. We can summarise the observation as a general expression:

The open real-world states influence the AS open-ended belief state about them, and vice versa. Failures in the real-world systems to achieve their intended purposes, impact failures of AS failure states.

We can see this dependency in the Uber accident. There was tow environmental failures that led to the unfortunate accident:

1. Uber attempt to reduce costs by dropping the requirement to have 2 pilots in their car.
2. The driver's social problems that led them to be distracted.
3. The driver is distracted.
4. Additionally, the victim shares some level of responsibility; they did not use the pedestrian crossing to cross the street safely or miscalculated the speed of the approaching car. The footage shows the victim walking casually, which could be due to a number of reasons.
    a. Maybe they ignored the upcoming traffic.
    b. Maybe they have failed to take into account that the driver was not paying attention.
    c. Maybe it was a habit where the victim does not walk fast in front of any car (some people are too relaxed).

We are not trying to speculate here, but we are trying to make the point and support the SACE guidance. The AS architect cannot ignore the behaviours of other systems and how they may fail to achieve their purposes as well.

So what?

Note in our description of the general problem we stated, "Failures in the real-world systems to achieve their intended purposes". This is a key concept and a hint for design teams; they need to make assumptions about operational environment complexes' purposes and how they behave to achieve such purposes, not only worry about AS purpose. In other words, the design approach needs to encourage practitioners to be more "sympathetic" toward other systems' desire to achieve their primary purposes. The architect can easily predict a system's behaviour in some scenario by knowing the primary purpose the system is trying to achieve. AIC (Appreciation, Influence and Control) systems theory incorporates the concept of "purpose" to be the key motivator of systems dynamics in any complexity. Hence, we realised the natural synergy between AIC theory and what we want to achieve.

E.1.1 Failed appreciation vs failed control uncertainty

We ask another question about the victim and the driver?

> **Would you describe the driver's and victim's failures as failed control actions? or failed appreciation actions?**

The car was driving reliably, and the control was performing as expected. We cannot entirely blame the perception system for its inability to detect the half-visible person walking with their bike at night, because its control action is directly dependent on the level of appreciation involved in its training dataset. Suppose the training datasets (and training dataset architect) were **ignorant (lacking knowledge, i.e. a higher epistemic uncertainty about some information relative to some previous a priori perceived lower epistemic uncertainty)**[1] of such scenarios. For example, the architect initially assumes that adversarial

[1] Note how epistemic uncertainty played a key factor in the problem. Think of ignorance as the relative increase in epistemic uncertainty due to an increase in imbalanced cognitive perception of information received. In other words, over-fitting or specialisation. Sometimes, ineffective information-gathering processes lead to an imbalance or imperfection in the information received, where some aspects (features, classes, dimensions, or components of a problem) become more informed than others, which can lead to a situation where an increase in information causes ignorance and epistemic blind spots. To reduce this possibility, we devised an n×n matrix. We mandated the consideration of all interactions within such matrices, which would then reduce the likelihood of favouring some aspects of a problem over others.

drone attacks target one side of a train track zone, so he designs a security drone that specialises in one side of a train track zone to reduce the costs of training. However, over time, new information comes in. It adds to their knowledge system, where the adversarial drones that focused on one side were nothing but decoys for other adversarial drones to attack from the other side of the train track zone. This additional information did not reduce the architect's epistemic uncertainty, but instead increased their uncertainty about their problem-solving process and information processing. The adversarial drone played the architect. This is what epistemic ignorance is: an increase in information leading to the rise in epistemic uncertainty. Such ignorance is reducible but the improvement of information gathering and processing quality.

In that case, we cannot say with utmost confidence that it was a control action failure. It was instead an appreciation failure on the part of the architect, whose understanding of the problem domain failed to consider such a scenario. From the victim's perspective, they were in complete control of themselves and reliably crossing the road; no control action had failed, but rather a lack of appreciation for the environment. Both the architect, the machine and the victim's knowledge base can be described as having experience "epistemic ignorance" which is a general flaw or intellectual fault of knowledge systems. However, the architect, and by extension their training datasets, and by extension the reliability of the trained model, can be understood to suffer from an inherent deficiency that leads to "epistemic ignorance". We understand this deficiency as a general assumption associated with any finite knowledge system, in particular the architect, by proposing the following general assumption and explanation:

The epistemic perception of finite knowledge systems, which underpins mental models, is inherently deficient. However, this deficiency is relatively reducible by gathering a uniformly distributed amount of information across various classes of required information. It can never be 0, because the number of classes needed to achieve 0 epistemic ignorance can be infinite, and the amount of information for each class also needs to be boundless. The architect cannot account for an infinite amount of information for an infinite number of possible classes. Hence, lateral thinking emerges to mitigate this limitation and increases the architect's chances of predicting as many required classes as possible, thereby enhancing the resilience of the solution over time and in the face of changing complexity.

Another source of ignorance can stem from the lack of situational awareness about what the minimum viable set of informational classes (systems, components) is that need to be considered.

The architect relies on the quality of their predictive thinking and knowledge base about any observed complexity field to make engineering decisions about what to include in the training dataset. Training datasets influence the reliability of an ML model in any given real-world scenario. The victim's knowledge base and predictive thinking also influence their behavioural decisions in a particular scenario. Since all of those knowledge systems are finite at any given time, the perception of each agent can be described to be inherently deficient. This deficiency leads to the inevitable emergence of epistemic ignorance, which in turn results in catastrophic failures. Note how all are interconnected, and also note that we said "influence" rather than "control".

With that in mind, we can understand the requirement for successful and safe operations among the architect, the training datasets and the victim as follows:

From the victim's perception perspective: In order for the victim to safely cross the road, they need to **control** their body movements to **influence** the road traffic correctly. To control their body movement and make the right decision on when to cross, they need to **appreciate** the general pattern of traffic (how many cars pass by at a time) and whether incoming cars are autonomous or human-driven. The reason they must appreciate the nature of incoming traffic is that the victim **has no influence or control** over the type of vehicles. In order to reduce the appreciation epistemic uncertainty, they need to be trained on the fact that the road now has a mix of traditional and autonomous cars. Note how the mitigation of appreciation uncertainty of failures involves knowledge about some new scenarios. This is fundamentally different from mitigation of control failures, for example, a control failure of the victim would be unable to control their foot movement, which may be due to faulty shoes or muscle spasm.

Did Uber designers consider a scenario where a person crossing the road might experience an unexpected muscle spasm while J-walking? If not, why not? This is an example of a Black Swan scenario. Note how thinking in AIC helped predict a potential hazardous operational scenario when considering modes of appreciation. Indeed, the machine and the architect can only appreciate people's physical states by considering such scenarios in their risk assessment and mitigation plans.

From that, we can define the following failure modes experienced by the victim:

- The victim failed to appreciate the nature of vehicles driving before crossing the road. This is caused by Uber's failure to inform the victim or have guards in place to clearly notify pedestrians in time for them to recognise that autonomous vehicles are driving. Note that the car itself had no sign to indicate that it was in self-driving mode.

- Failure of appreciation has led to failure to influence the traffic correctly. Note, it was not the victim's failure to control their mobility that led to the accident, but a failure to correctly appreciate. Uber also failed to appreciate the importance of providing pedestrians with a means to inform them that their vehicles are in self-driving mode. **Why did Uber design team miss such a critical safety requirement?**

Furthermore, if you look at the footage from the front camera of the Uber car, you see the front lights were illuminating far enough, and the road was not sufficiently lit either. This adds another factor involved in the failure mode:

- The road infrastructure did not appreciate the requirements of proper lighting to reduce the risk of computer vision false negatives.
- It also means that Uber training missed out on considering poorly lit roads and how the vehicle front light should have operated in such conditions. That is a missing training requirement and a safety requirement:
    - In the event of a poorly lit environment, the front light shall operate in full-beam conditions.
    - The ML perception system shall be trained on various front light illumination operations, and during partial or full beam illumination.

From the machine's perception perspective: For the ML model to influence the safe driving of an autonomous system, it must accurately control the speed and steering of the car in a given scenario. The ML model has **no influence or control** over what scenarios they may face in **an open, complicated environment**. Therefore, the ML model does not control or influence the type of scenario faced; it can only correctly **appreciate** it. For an ML model to accurately comprehend the variety of scenarios, it must be trained on scenarios that encompass this diversity, meaning its training dataset must include these scenarios. The ML model has no influence or control over the training dataset on which it is trained. The architect and the data-gathering process wholly control these datasets. This means that the ML model can only correctly control the vehicle by properly appreciating the knowledge stored in its datasets, which in turn enables the model to recognise (appreciate) the architect's engineering judgement.

From that we can deduce the following failure modes:

- The machine failed to appreciate a situation where a person was occluded by the light

Now, suppose that the systems engineering approach used to design the system only focuses the architect's attention on “failed control actions.” What is the likelihood that the architect would also consider failed appreciation actions?

Most current failure analysis techniques (such as HAZOP, STAMP, FRAM, etc.) reduce the complexity of the problem domain to a control issue, leading architects to focus on direct and easily identifiable control-related behaviours. We realised that this approach can be inefficient when addressing intricate open problems, as illustrated by the Uber case study; other general modes of action are not control-related. These include influence and appreciation. In Chapter 3, we observed how HAZOP, FRAM, and STAMP directed our thought processes to concentrate solely on a limited number of general failure modes (guide words) and on control actions. Hence, we view Uber’s systems engineering approach as also a party of this tragic accident; it failed to adequately enable the design team the discovery non-control failures, like appreciation.

E.1.2 AS open-ended belief state uncertainty

Why did we say “AS open-ended belief state”?

This is because when we teach an ML model to identify a person (for example), we don’t specifically tell it what a person exactly is (in a similar fashion as we would specify an acceptable voltage value for a deterministic controller). We give it an open-ended characterisation, open to infinite possible changes of what a person is. The pictorial definition of a person depends entirely on the context within which the ML model operates. The challenging part here for the architect is predicting all possible contexts that the problem domain may change into to account for them in the training process (more detailed articulation can be read in E.1).

In other words, we provide many possibilities of what a person “could be”, but ultimately, we expect the trained ML model to decide what constitutes a person when it observes one, hence it is always a probability of what it has learned in general through our datasets. Thus, “openness” can be understood as the boundlessness of possibilities, some of which are “meaningful” or “intended through design”, others are “meaningless” or “unintended as a by-product of intended design”. By “meaningful,” we mean we train a model to understand something significant to us (solving a particular problem) and expect it to follow through on its own.

So, how do we understand the parts of the problem articulated by SACE?

Well, it seems there are several key concepts involved which we need to make sense of and work out a general solution for them as a whole:

- Openness of the operational environment and open-endedness of machine understanding of the operational environment.
- Real-world states.

- Machine's belief states.
- How does uncertainty behave in a problem involving an open, complicated environment and an open-ended belief and decision maker (other than a human)?

We further add another component to the above articulation, which is essential to the general systems approach solution:

- The architect's epistemic uncertainty about the open environment and the autonomous system behaviour in response.

We look at the components of the problem from the standpoint of "Architect vs Problem" , so we understand the situation as:

- Architect predictive capability problem (ArcPC).

In the above articulation, we recognise three main problems to manage:

- The operating environment is open and complicated with a multitude of real-world state types (dominated by hidden Black Swan real-world states).
- We expect the engineered AS to behave or execute its designed belief states as expected in those black swans.
- The architect's belief system that predicts and defines those states.

In this context, the **architect's belief system** refers to the internal mental model, predictive assumptions, and reasoning framework the architect uses to understand, anticipate, and design for both the real-world operational environment and the behaviour of the autonomous system (AS). This internal mental model can be in two distinct situations when facing complexity: **Clear** or **Confused**. These two situations determine the quality of the engineering judgement and design decisions related to the final AS design.

It encompasses the architect's current knowledge, expectations, and uncertainties about how the AS will perceive and act in open, unpredictable scenarios, particularly in relation to unknown or Black Swan events. So, from an architect's belief-system development perspective, the more categories of parts and the more parts in each category or type are involved in the open environment and the behaviour of their engineered system, the faster the architect's epistemic uncertainty about the robustness of their assumptions compounds. Hence, we use the term "Compounded Uncertainty Problem".

The complexity of the physical environment includes inherent randomness, such as irregular lighting conditions due to wind and rain (read section 2.1.1.5 for background literature on uncertainty and ignorance). Such randomness is a form of aleatoric uncertainty that cannot be physically reduced with more added elements and factors. The physical environment uncertainty is then translated into informational uncertainty in ML training and validation images at the pixel level. However, the more examples we capture, the more sources of environmental aleatoric uncertainty are captured, and the less epistemic uncertainty (about the environment) becomes apparent in the training and validation datasets. However, epistemic uncertainty in training and validation datasets depends entirely on the architect's epistemic uncertainty about the environment and abstraction/design skills. While we cannot influence or control the open environment, we can influence and enable the architect to make the most out of their predictive capability using an ontology and process that helps to minimise sources of epistemic ignorance (a risk associated with epistemic uncertainty).

In other words, physical environment randomness (physical aleatoric uncertainty) is translated into pixel-level epistemic uncertainty in images, which impacts the architect's uncertainty about the image's fidelity to capture the real world accurately and sufficiently. At the same time, the physical aleatoric uncertainty of the environment (how random it becomes) directly impacts the architect's predictive capability to know what to include in the training and validation of the ML component. This reflected the coverage and nature of information captured in training and validation datasets (source of the machine’s epistemic uncertainty). Hence, to assure coverage in the dataset, we must ensure that the architect's systemic perception of the open operational environment is comprehensive and accurate.

Here, where architect's epistemic uncertainty becomes the ultimate receiving end of physical environment randomness. In section 4.8, we define a categorical system that captures Operational Design Domain (ODD) sources of aleatoric uncertainty (randomness) based on environmental conditions that we believe involve more challenging situations that can negatively impact the architect’s confidence in assumptions made about the operational domain, which ultimately challenges the reliability of the training process of ML components. Hence, we proposed that the safety of autonomous systems in open, complicated environments is the architect's predictive capability problem. Based on that, we develop Predictive Thinking techniques (SECoT).

From an ML training syllabus development perspective, specifying or designing pixel-level uncertainty (writing traceable requirements) is impractical. Hence, we must pool them and abstract them enough in the form of epistemic pictorial training classes. With training classes being the architect's design problem, aleatoric uncertainty becomes a question of balancing the right granularity of epistemic uncertainty by the architect. Here, architects' knowledge (chosen assumptions) of the real-world complexity field and their abstraction skills are crucial in managing the risk of ignorance. Hence, we proposed the CuneiForm as a traceable pictorial abstraction method to design those pictorial training classes.

E.1.3 Epistemic Uncertainty's Impact on ML Safety: Why Should We Care?

Epistemic uncertainty, the uncertainty arising from incomplete knowledge about the adequacy of the data-generating process, poses a critical yet often underappreciated threat to the safety of ML components. In the decision-theoretic framing of safety, Möller et al. emphasise not only the reduction of expected risk but also the minimisation of unexpected harms, i.e., those arising from "unknown unknowns" in the system's behaviour [5]. While classical risk minimisation tackles only the probability of known harms, epistemic uncertainty encompasses the potential for entirely unforeseen operational scenarios by the architect and the training set, which, when severe enough, constitute genuine safety hazards.

One primary source of epistemic uncertainty is the mismatch between training data and operational conditions. When samples {(xi, yi)} are drawn from a distribution that differs, either subtly or drastically, from the true deployment environment, the learned model may exploit spurious correlations, leading to harmful mispredictions under domain shift [6] [7]. In safety-critical settings such as autonomous vehicles or medical devices, even rare shifts (e.g., unusual lighting conditions or atypical patient demographics) can precipitate catastrophic outcomes. Moreover, because epistemic uncertainty is reducible (in principle) through additional data or model constraints, failure to acknowledge it means foregoing crucial opportunities to bolster safety via targeted data collection or robust learning formulations.

A second manifestation of epistemic uncertainty arises from sparse coverage of the feature space. In regions of low sample density, the inductive bias encoded in the hypothesis of some class H dominates the learned decision rule, rendering model behaviour effectively unpredictable to human overseers [8]. Unlike aleatoric uncertainty, which remains irreducible noise, even an arbitrarily powerful learner cannot "learn" in areas where no data exists. This is especially problematic in cyber-physical systems, where the input–output space is high-dimensional and safety-relevant failure modes may reside in remote corners of that space. Without explicit demonstration that epistemic uncertainty is rigorously addressed during design time, ML components may make overconfident assertions that extend beyond their domain of competence, violating the "fail safe" principle and exposing users to undue risk.

Furthermore, the distinction between actual risk and operational risk underscores the amplifying effect of epistemic uncertainty on individual cases. Statistical learning theory assures convergence of empirical risk, Emp(h), to the actual risk R(h) only in the limit of infinite data [9]. In real systems, however, a finite and sometimes small number of critical decisions may be made under conditions where the realised (operational) risk vastly exceeds the ML model's expected performance. Each outlier misprediction, driven by epistemic gaps in training, can directly translate into a safety incident, ranging from a misrouted surgical robot tool to a failure in a collision avoidance system.

In light of these considerations, any ML application aspiring to safety must explicitly model and mitigate epistemic uncertainty. Strategies such as inherently safe design (e.g., interpretable models that expose regions of high uncertainty), safety reserves via robust optimisation over uncertain parameters, and safe-fail mechanisms (e.g., uncertainty-aware reject options) all hinge on reliable estimates of epistemic risk. Procedural safeguards, such as virtual-environment hazard simulation, likewise depend on identifying where a model's knowledge is weakest. ***Our approach*** attempts to qualitatively reduce epistemic uncertainty by enhancing the architect's predictive thinking. The virtual environment that the literature discusses will require an architect with deep and objective predictive capability to make the virtual environment as comprehensive as possible (high fidelity with the problem domain).

Thus, to fulfil the dual objectives of reducing both expected and unexpected harms, the thesis of this work posits that accounting for epistemic uncertainty is not optional but fundamental to ML safety.

E.1.4 When Mediocristan systems safety tools lack in the face of Extremistan risks

We cannot proceed to evaluate existing safety and systems engineering approaches without first highlighting fundamental concepts for understanding the risks associated with probability distributions of events in complex systems. A fundamental concept underpinning our evaluation approach to existing methodologies (such as STAMP, FRAM, and HAZOP) is understanding how these approaches prompt the user or architect to consider the problem or system complexity. To build such intuition, we needed to understand the opposing variety of possible views an architect may have about any safety problem, where traditional approaches fit within this mindset, and how our solution fits into the picture. Since we view the problem of understanding risks in a complexity based on being an uncertainty problem, we needed a well-founded theory that we can use as our basis. This is Taleb's theory for understanding risk for complex problems (see section 2.1.1.4).

Based on Taleb's distinction between Mediocristan and Extremistan, we can recontextualise these domains as metaphors for how safety engineering approaches conceptualise complexity, risk, and hazard emergence in engineered systems. In Mediocristan, events are governed by thin-tailed distributions, risks emerge incrementally through the accumulation of small, predictable failures. This domain aligns with traditional risk assessment paradigms, such as HAZOP (Hazard and Operability Study), which assumes that hazards can be systematically identified through structured deviations from expected operational conditions. Such methods imply a Mediocristan worldview: risks are tractable, distributions are regular, and failures result from known causes that can be identified through thorough decomposition and analysis.

By contrast, Extremistan describes systems governed by long-tailed distributions, where rare, high-impact failures dominate system behaviour and elude prediction. These methods acknowledge the Extremistan nature of modern socio-technical systems, where wild randomness, high interdependence, and scalable impacts (e.g., a software bug leading to an airline disaster) undermine the assumptions of predictability and control.

One important implication of engineering Mediocristan systems in Mediocristan environments is that it is reasonable to rely on statistical tools, such as the mean, standard deviation, linear regression, and principal component analysis, as robust methods to estimate risks. However, in engineering Extremistan systems in Extremistan environments, those tools are not robust to handle variations in the complexity field to trust risk estimates. This is where we need to introduce another dimension to reduce risks. That dimension is the enhancement of practitioners' or problem solvers' predictive capability to predict harder-to-predict risk factors.

In general, safety approaches prompt practitioners to adopt either a Mediocristan or Extremistan view of engineered complexity. Traditional, control-based hazard analysis methods assume a Mediocristan landscape, where risk is a sum of small parts, failures are mild, and system behaviour is decomposable. Conversely, holistic systemic approaches should reflect an Extremistan mindset, recognising that emergent behaviour, complex couplings, and disproportionate event impact are not only possible but expected in modern systems.

E.1.5 The impact of a control-biased Mediocristan approach to safety

How do systems engineering approaches tacitly encourage the architect to view a problem as a normal-distribution probability problem? And why is such an approach alone not enough in solving CUPs?

Systems engineering approaches do so by defining a specific (limited) set of types of failures, events or behaviours to consider and prompting the architect to think of what can make logical sense. For example, HAZOP prompts the architect to consider a set of deviations, such as guidewords, and expects the architect to make an engineering judgement (constrained by expertise and legacy experience) to choose what makes logical sense and ignore what makes no sense. By ignoring a subset of deviations, the architect, essentially, predicts that the ignored subset of deviations poses 0 risk to the system. This activity prompts the architect to consider a subset of events as the primary concern and assume others as negligible. Which is similar to tacitly thinking;

Let's assume that all safety problems can be reduced to two types of events: those we consider logically important and those we consider negligible, such that if their impact or probability of occurrence is assumed to be 0 or minimal, no significant effect is expected on the overall safety of the system.

That is the characteristic of viewing the complexity of the problem as a normally distributed set of events. However, CUPs, where we are dealing with different types of non-deterministic systems operating in an open non-deterministic environment, is not the right place to make such simplification. In fact, we believe such simplification is the initial sin in engineering judgment that an architect or a project makes.

HAZOP, and STAMP utilise this technique by defining some key considerations and assuming that the system is mainly a single type of cybernetic problem: a control-only problem. We experienced this phenomenon when we applied HAZOP in Chapter 3, we realised that our predictive thinking prioritised only the guidelines that we found easier to make sense of (See Chapter 3, and Appendix G).

Why do we think so?

The fundamental aspect of any normal distribution is that there is a finite set of the most repeated or frequent pins or classes of information, among some number of pins (pins, in this case, would mean types, categories, or things). Hence, if the engineering approach encourages the focus on specific types of failures, and allows for labelling certain types as “irrelevant” or “not-applicable”, it indirectly encourages the architect to assume a finite number of types of events. However, in a non-linear distribution of events and categories type of complexity field, such bias entails neglecting hidden, unorthodox kinds of events or “real-world states” misjudged as “irrelevant”.

Thus, pre-set-based approaches, such as HAZOP and FMEA, may not be able to capture essential types of events solely through the use of guidewords. Additionally, the system of interest’s behaviour is also unpredictable. It is influenced by unknown variables, emergent interactions, and unseen factors, often involving multiple autonomous agents trained on confusing datasets. As a result, traditional methods that rely on predefined categories, likelihoods, or deterministic assumptions are insufficient or inadequate in certain areas, necessitating new models of predictive cognition, adaptive reasoning, and exploratory systems thinking even to begin engaging with the problem space. Hence, AIC Systems Approach is designed to propose a process to solve such problems.

E.1.6 Impact of lateral thinking process on safety in CUP

Part of the safety assurance process is identifying hazards and defining mitigating requirements (Safety Concept). Along with predicting hazards, the architect or safety engineer must conduct some form of qualitative or quantitative risk assessment. This includes assessing the likelihood and the impact of a given hazard. Usually, with established safety analysis and well-known hazards, risk assessment of failures enjoys a considerable amount of legacy and a body of knowledge. However, when dealing with CUP where the problem domain is dominated by novel hazards, including soft hazards (see section E.3 for more details on the types of hazards in CUP), due to the novelty of such hazards, often there would be minimal or no legacy body of knowledge to back up the safety engineer's risk assessment.

For example, a moon appearing like a traffic light for an autonomous car or a drone. Suppose the architect predicted that such a hazard (associating the moon with traffic lights is a bizarre association to most people). How could they justify the association, let alone justify any form of risk assessment in the absence of logical rigour or a legacy body of knowledge? It is an analogical association that is hard to logically connect (hence systematic thinking engineers may miss such associations). Here, the lateral thinking process can provide a systematic justification that explains why such a rare example is impactful.

Bear in mind that the risk assessment of CUP differs in expectation from that of a regular system. The likelihood of an event, in a complexity where all events are assumed to be equally likely and their impacts also the same, does not make sense to quantify likelihood and impact. Risk assessment of hazards and failures for CUP can be approached holistically based on the concept of demonstrated reduction of epistemic uncertainty. This means that risks are reduced to ALARP based on the deliberate pursuit of predictable Black Swans, and the architect utilised both lateral and vertical predictive thinking techniques to arrive at as comprehensive a set of hard and soft hazards as practically possible. This notion will become important when assessing risks associated with training coverage.

E.1.7 Compounded Uncertainty of Open Complicated Problems

"The complexity of an object is in the eyes of the observer" Klir [1]

SACE (Safe Assurance of Autonomous Systems in Complex Environments), guidance suggests the following observation about the relationships between autonomous systems and an open operational environment:

> "The different possible scenarios and environmental states that relate to the decision can then be identified by considering the real-world state and the belief state of the AS at the point at which the decision is made, along with each of the possible options. The real-world state represents the actual state of the operating environment, whilst the belief state represents the understanding that the AS has of the state of the operating environment"

We can summarise the observation as a general expression:

The open real-world states influence the AS open-ended belief state about them, and vice versa.

Why did we say "open-ended"? This is because when we teach an ML model to identify a person (for example), we don't specifically tell it what a person exactly is (in a similar fashion as we would specify an acceptable voltage value for a deterministic controller). We give it an open-ended characterisation, open to infinite possible changes of what a person is. The pictorial definition of a person depends entirely on the context within which the ML model operates. The challenging part here for the architect is predicting all possible contexts that the problem domain may change into to account for them in the training process.

In other words, we provide many possibilities of what a person "could be", but ultimately, we expect the trained ML model to decide what constitutes a person when it observes one, hence it is always a probability of what it has learned in general through our datasets. Thus, "openness" can be understood as the boundlessness of possibilities, some of which are "meaningful" or "intended through design", others are "meaningless" or "unintended as a by-product of intended design". By "meaningful," we mean we train a model to understand something significant to us (solving a particular problem) and expect it to follow through on its own.

So, how do we understand the parts of the problem articulated by SACE?

Well, it seems there are a number of key concepts involved which we need to make sense of and work out a general solution for them as a whole:

- Openness of the operational environment and open-endedness of machine understanding of the operational environment.
- Real-world states.
- Machine's belief states.
- How does uncertainty behave in a problem involving an open, complicated environment and an open-ended belief and decision maker (other than a human)?

We further add another component to the above articulation, which is essential to the general systems approach solution:

- The architect’s epistemic uncertainty about the open environment and the autonomous system behaviour in response.

We look at the components of the problem from the standpoint of “Architect vs Problem” , so we understand the situation as:

- Architect predictive capability problem (ArcPC).

In the above articulation, we recognise three main problems to manage:

- The operating environment is open and complicated with a multitude of real-world state types (dominated by hidden Black Swan real-world states).
- We expect the engineered AS to behave or execute its designed belief states as expected in those black swans.
- The architect's belief system that predicts and defines those states.

In this context, the **architect's belief system** refers to the internal mental model, predictive assumptions, and reasoning framework the architect uses to understand, anticipate, and design for both the real-world operational environment and the behaviour of the autonomous system (AS). It encompasses the architect’s current knowledge, expectations, and uncertainties about how the AS will perceive and act in open, unpredictable scenarios, particularly in relation to unknown or Black Swan events. So, from an architect's belief-system development perspective, the more categories of parts and the more parts in each category or type are involved in the open environment and the behaviour of their engineered system, the faster the architect's epistemic uncertainty about the robustness of their assumptions compounds. Hence, we use the term “Compounded Uncertainty Problem”.

Before we define the Compounded Uncertainty Problem, readers need to understand core concepts related to systems engineering, particularly our approach to complexity and the idea of complicatedness. By the end of this subsection, articulating these concepts will make it more straightforward why such problems are unique and require a different approach. It will serve as an extension to existing systems engineering approaches rather than a replacement for them.

Section 2.1.1.7 describes the current state of the literature on the topic of complexity. We have noticed a general theme across the board regarding definitions:

- Complexity is a feature of an observed phenomenon.
- It is directly somewhat related to randomness in phenomena.
- Complexity increases as randomness increases.

Such characterisation is closely associated with aleatoric uncertainty (see section 2.1.2). where aleatoric uncertainty is characterised with the following properties:

- Related to the observed system.
- Related to randomness in the system.

- As randomness increases, so does aleatoric uncertainty.

Thus, we can somewhat see a link between the standard definition of complexity and aleatoric uncertainty. They may be the same or at least dependent on each other. However, if this is the case, what happens when randomness decreases? Does complexity decrease? The literature assumes it does. In other words, if we add more elements to the observed complexity, we also increase its randomness; hence, it is considered that aleatoric uncertainty is irreducible, meaning that the more we add, the worse the problem becomes (adding more fuel to the fire of aleatoric uncertainty).

Where there is randomness, there is probability, and where there is probability, there is "predictability". Hence, some authors have characterised complexity in terms of predictability, such that the more unpredictable the system is, the more complex it is. A concept with which we can intuitively agree. From that, we conclude that the literature suggests:

- Adding more parts to a system increases the risk of high randomness.
- An increase in randomness means an increase in aleatoric uncertainty.
- An increase in aleatoric uncertainty indicates degraded predictability, which in turn implies an increase in complexity.
- The opposite is also true: removing parts may decrease randomness, which may decrease complexity.

However, if we have an entirely predictable system, in a closed predictable environment, and systematically add more elements to it (for example, adding more propellers to a drone, where the drone is tethered to some power source, or more tyres to a car driving in circles around an indoor ring), do these additions relatively increase the unpredictability of the car or drone system? We know precisely how a propeller is connected, and we can comfortably predict what can go wrong or right. In this case, increasing the number of predictable elements in a predictable system within a predictable environment may not lead to an increase in unpredictability. Thus, a predictability-based complexity definition may not hold in such a scenario. This means the following:

- Predictability does not seem to align consistently with changes in general complexity.
- A larger, predictable system is not more complex than a smaller, predictable one. In fact, it is the dream of any architect to develop a system that remains explainable and predictable, regardless of its size.

There may be a missing distinctive factor in this observer-independent approach to complexity, where the probability of events and predictability have a clearer impact!

Technically, adding more predictable elements to a predictable system does not necessarily (assuming a perfect world) change its predictability. That is the whole point of systematic increments, in systems engineering, which controls the predictability of systems as more elements are added.

This raises the following questions:

- What is the holistic quality of an organisational state-of-disorder that directly depends on its scale and frequency of events, regardless of whether these events are predictable or not? This is a probability problem.
- What is the holistic quality of an organisational state-of-disorder that directly depends on its predictability, regardless of its scale or frequency of event changes? This is a predictability problem.

The literature on the probability and predictability of systems or organisations is vast. The reader is welcome to explore the field. Below is our re-interpretation in the context of predictive architecting and problem solving of the literature about those two concepts:

E.1.8 Probability View of Complexity

Changes in scale and frequencies of occurrences, care less about whether they are predictable or not predictable by the predictive observer (the architect). So, its measurement does not rely on how predictable they are for an observing architect. In other words, it is a perspective that is more aligned with the measure of likelihood and probability's which can be thought of in two ways:

- **Frequentist:** Probability is the long-run relative frequency of an event in repeated trials. An objective measure, independent of the observer's knowledge.
- **Bayesian:** Probability is a degree of belief about the likelihood of an event, given an observer's prior knowledge (subjective).

A probabilistic view of problems or systems involves considering all possible types of outcomes and their corresponding frequencies of occurrence. to construct a probability distribution, you need to do two activities:

- Select the relevant pins (categories) for the things or types of scenarios that are currently occurring, may occur in the future, or have happened in the past. For example, the HAZOP process uses a set of "guide words" to categorise types of possible failures.
- Estimate the frequency of their occurrence. For example, in the HAZOP process, the practitioner is given a choice of what they consider the most suitable category if failure were to occur for a given interaction or event in a system.

- The most frequent categories are the ones that are more important and worth attention. Least frequent categories are negligible since they are the least likely types of events to occur in a system.

For example, in the HAZOP process, the practitioner's bias towards the most obvious category for a given interaction or event in a system often results in a biased distribution of frequency, where a small subset of guide words is disproportionately considered over others. The HAZOP process does not require consideration of all possible guide words for every event or interaction in a system. See Chapter3 for more details on how we applied HAZOP.

If you follow the frequentist approach, you need to consider what you can think of when estimating the most suitable option and record the final frequencies of the options you choose. A frequentist approach, such as HAZOP and FMEA, falls into this category, as they are “guide words” category-based thought processes. If you follow Bayesian sense, then you need to perform an iterative, hierarchical approach, based on continuous refinements of a set of beliefs. SHARCS and STAMP/STPA processes follow such an approach.

In either case (frequentism or Bayesian), generally speaking, probability is not necessarily about our capacity to accurately predict what will happen; it is a measure of how likely an event, failure, or state may occur in a system, either objectively or subjectively. Hence, most probabilistic complexity-inspired approaches (HAZOP, FMEA, etc) come with a probability-based risk register.

However, non-deterministic engineered systems, such as ML-based systems, operating in an open and complicated environment, exhibit emergent behaviour that is heavily dependent on possible interactions with the environment, as they adapt dynamically rather than deterministically. Furthermore, they accept infinitely many possible inputs (like an ML-based computer vision system is expected to handle any possible variation of a target object of interest, such as a drone). In such a type of complexity, Black Swans and rare events cannot be ignored as negligible.

Those characteristic differences require the consideration of an infinite number of possible types of categories of events and failure modes to ensure completeness and negligible events. Such a task means probabilistic-based approaches to the problem may not effectively handle such vastness. As pre-determined guide words may bias the practitioner towards a limited set of predictive thoughts, this may lead to missing valuable failure modes or requirements.

In other words, it is more effective to consider ML-based systems to be a “predictability” problem that requires a predictive thinking model to discover categories of interactions as it is applied iteratively and dynamically.

E.1.9 Probability distribution of events-based classification of systems

From an architect's epistemic uncertainty perspective, the architect can engineer a complex into one of the following systemness types. For each type of system, there are different problems associated with uncertainty, risk quantification and impact evaluation, confidence in prior beliefs, confidence in assurance of requirements and whether expected behaviours do emerge as intended or not in unforeseeable scenarios:

- **Normal-system:** A normal-system is a complex whose uncertainty distribution of its emergent behaviours is assumed to follow a normally distributed profile or tendency. In such systems, there exists exactly a finite subset of events that can be described as the most likely events to occur. Assurance in such systems would be based on proving that indeed those subsets of events are the most likely and we can precisely verify this using mathematical proves. In normal-systems it is reasonable to assume complexity to be Mediocristan.
  - In normal distribution, there is a clear boundary between a range of bins [-b to b] where the probability of any event outside the range is precisely 0.
  - Normal systems are easier to trust the assurances made by the architect of the number of emergent behaviours in deterministic means. Hence, they are relatively easier to have confidence in.

This means that the Predictive Problem Solver can be confident that only a specific or desired set of events will likely occur, whose impact is worth their attention and mitigation. Any other rare but negligible events. Examples of normal-systems such as a door, a regular car, or a computer. The architect assumes the normal system to be ideal. Domino-like deterministic systems can be viewed a special case where the bell curve is very narrow, since there can only be a specific subset of events that can occur in those systems. So the architect’s uncertainty can be reduced to near 0 given enough time to learn how they work. However, this cannot be said for ML-based CPS; there is always residual uncertainty and surprise.

- **Abnormal-system:** A clear complicated complex whose uncertainty profile or probability distribution of its emergent behaviours does not follow a normal distribution (bell-shaped curve). In other words, it may follow a heavy-tailed distribution. For such systems, the architect can only possess a priori knowledge of the most likely events to occur (the head events); however, this prior knowledge is not fixed and will be updated and refined over time to determine whether that subset of events is exhaustive or if other events should be included.

  - In contrast to a uniform and normal distribution over a range of bins [−b,b], the P(x) never converges to 0. This indicates that there are no clear-cut boundaries for systems whose emergent behaviours follow a long distribution profile.
  - On their own, with an inherent boundless categories of emergent behaviours (bins in histogram), they are challenging to trust assurance claims of their designed behaviours and that they will only behave in a way that were intended for and never more.

The approach to make them more transparent and trustworthy, abnormal complicated complexes is:

- Defining loosely bounded abstract concepts (like purpose) which are open-ended to accept more variations of unaccounted designed behaviours. For example, instead of concrete specification of what a human look like, use an abstract image of a typical human for a dataset for a vision system. This way more variations of what a human is can be validated to be acceptable.
- Clearly demonstrate that abnormal predictive thinking is employed as much as practically possible to predict long-tailed emergent behaviours. Abnormal thinking is another word to "lateral thinking" which allows to accept loosely concrete strong traceability of specification. For example, a picture of a human in a painting may also be an acceptable interpretation of a requirement that asks for including a human in the training dataset. Another example may be considering reflections of light on raindrops over tree leaves as a potential source of failure for an ML component.

For example, an ML-based perception system trained to detect drones in specific open green fields. We can say that the events in the training dataset are those we are reasonably sure will be detected if the perception system encounters them in the real world. However, we also expect that other unseen images are not part of the training dataset, and therefore not included in our prior subset of events. For instance, images from different open green fields or even cities. As you can see, an abnormal system also has a set of unseen events that contribute to its capabilities. In abnormal systems, it is not reasonable to assume the complexity to be Mediocristan, and it is safer to assume it to be Extremistan.

Such systems are characterised by sensitivity to butterfly effects and Black Swan scenarios. In other words, perceived rare and insignificant events cannot be overlooked (these are the Long-tailed events). Open complicated environments, neural network ML-based autonomous systems, and the combination of both are examples of abnormal complexes. Abnormal complexes can be described as “Abnormal-systems” when an architect models their behaviour and describes how they work and what to expect to observe about them.

- **An ideal system:** A clear complicated complex, a special case of normal systems, its emergent behaviours entirely predictable, complex in all possible world views for any Predictive Problem Solver. You can imagine the bell curve of an ideal system to be completely flat at the far ends of the bell, and a very narrow and long bell.

Ideal systems operate in “complete knowledge” of an operational domain, such that epistemic uncertainty (residual epistemic Ignorance) equals zero. The randomness of the observation is entirely predictable. For example, domino-like deterministic systems (like a comprehensively modelled and explained analogue controller or a regular computer) are a good approximation of ideal systems in theory because the system is predictable and explainable. A fully automated factory in a highly controlled environment (factory condition) is an ideal system since it is predictable.

- **Stochastic Confusion:** A confusing, complicated, complex, an extreme form of abnormal complex**,** A completely unpredictable VUCA complex or one that is completely confusing in all possible world views for any Predictive Problem Solver. This corresponds to “complete ignorance” of an operational domain, such that architect’s epistemic uncertainty = 100%. The randomness of its emergent behaviours is entirely unpredictable and range of possible types of events are boundless. It is a nightmare for any general systems architect.

With the above classification of types of complexes, we can categorise four possible approaches that an architect can take regarding CUP, which also determines the types of safety paradigms (see sections 2.1.3 and the introduction to Chapter 5). Figure 4.1 differentiates the types of available safety approaches for an architect to deal with CUPs in engineering based on the general attitude of the approach towards Black Swans (dominate feature of CUPs). Formal methods (like Event-B) and HAZOP can be classified as the initial phase of safety, where the system's complexity and its environment is thought of as a set of controllable interactions in a domino-like fashion. This would be an ideal view of the problem and is much suited for highly controllable environments and deterministic systems. This paradigm considers Black Swans (rare and impactful events) negligible.

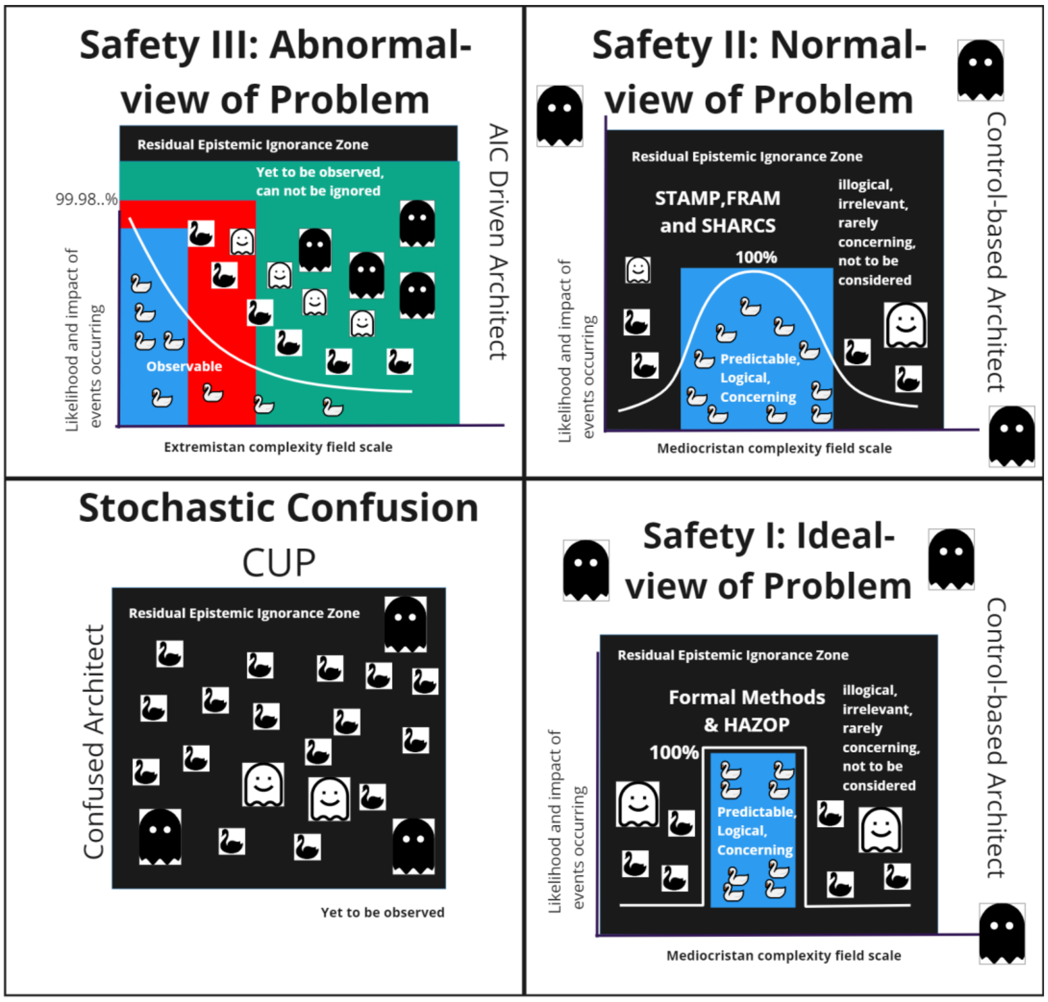
The same attitude towards Black Swan was inherited by the follow-up generation of safety approaches (Safety II), STAMP, FRAM and SHARCS. Our experience in applying STAMP and FRAM in Chapter 3 confirmed this belief (see Chapter 3 and section 9.2). Safety II looks at the safety problem as a hierarchical functional control structure. With such a view of complexity, there is an expectation of a rigid boundary where some information is deemed irrelevant or not applicable. For example, whether the operator of a control switch is wearing a white or black shirt is irrelevant information about why a switch fails. Tacitly, calling such a situation irrelevant is an indirect dismissal of causal impact due to the rarity of a shirt colour having anything to do with a failed switch. Such a mindset can not be effective when dealing with CUP, because, architect's assumptions at any given time and scale of complexity are never solid and rigid—only temporary correctness.

Hence, we need an approach that believes in the following principle: "There is no such thing as an irrelevant, and there is no such thing as a rigid boundary. Here, where control only view of stochastic conclusion becomes a liability for a problem solver because they will evidently miss out on those rare and impactful events as irrelevant or not applicable.

Here, where AIC (Appreciation, Influence and Control) comes in to add three graded of rigidity in boundary definition:

- **Control:** direct coupling between two components. Hard-bounded with rigid (unchanging) exchange of effect. For example, the operator physically flips the switch on or off.
- **Influence:** indirect effect between two components. Softer-bounded functional exchange, rigid (unchanging) exchange of effect. For example, the operator's manager demanded that the person wear formal attire that day, but the operator did not obey this directive, which forced the manager to fire the operator that day.
- **Appreciation:** the absence of control or influence from one component on the other. The exchange can change dependents on the appreciated system behaviour, whereby the appreciating has no power to control the appreciated behaviour. For example, the country's current economic situation impacts the operator's ability to find another job.

If we applied a control-driven mindset, there is no reciprocal control or influence between the operator and the economic situation of the country, where a simplistic feedback control loop model (as per STAMP principle) would not necessarily be helpful if we are dealing with complicated problem in which there is no rigid assumptions can be trusted to remain true for a long time. Such an attitude may lead to unwanted surprises for open, complicated environments and non-deterministic engineered agents.


Safety III: Abnormal-view of Problem
Residual Epistemic Ignorance Zone
Yet to be observed, can not be ignored
99.98..%
Likelihood and impact of events occurring
Observable
Extremistan complexity field scale
AIC Driven Architect
Safety II: Normal-view of Problem
Residual Epistemic Ignorance Zone
STAMP,FRAM and SHARCS
illogical, irrelevant, rarely concerning, not to be considered
100%
Predictable, Logical, Concerning
Likelihood and impact of events occurring
Mediocristan complexity field scale
Control-based Architect
Stochastic Confusion
CUP
Residual Epistemic Ignorance Zone
Confused Architect
Yet to be observed
Safety I: Ideal-view of Problem
Residual Epistemic Ignorance Zone
Formal Methods & HAZOP
illogical, irrelevant, rarely concerning, not to be considered
100%
Predictable, Logical, Concerning
Likelihood and impact of events occurring
Mediocristan complexity field scale
Control-based Architect

Blue: thinking of control only (no appreciation), Red: thinking of partial control and partial appreciation (influence) perspective, Green: thinking of no-control (appreciation), **Black:** Residual Epistemic Ignorance Zone

Figure E.1 Types of architect approaches to complexity of in CUP. Compounded Epistemic Uncertainty Problem (CUP)

The swans and ghosts icons represent pictorial scenarios in different visibility levels to the architect's perception of the engineered complexity problem[2].

For full theoretical detail behind Predictability, Probability distributions, Complexity, Complicatedness, Epistemic Uncertainty, Aleatoric uncertainty, System, Stochastic Confusion (non-system), see section E.1 (Appendix E). Given our evaluation of the topic in section E.1, we define the following key concept:

> A **Compounded Epistemic Uncertainty Problem** is a Stochastic Confusion[3] (non-system) Involving agents, deterministic systems and an open, complicated environment, where any changes in any part of the problem lead to unpredictable whole

[2] See sections 4.9 and 9.13 for a more detailed description of what the icons mean
[3] See the definition of Stochastic Confusion in this section.

for the predictive observer. The epistemic uncertainty of the predictive observer is understood to compound nonlinearly as the complexity of the problem domain changes unpredictably.

There is a set of properties in such problems which can be sources of risks:

1. Any prior beliefs about such problems by the observer can be volatile. There is a very high likelihood that prior beliefs will not prevail over time.
2. Confusing, purposeless, uniformly or non-normally distributed complexity of events and behaviours.

It is a scenario when unpredictable changes in an observed complexity (aleatoric uncertainty/randomness) cause a non-linear (compoundedness) increase in architect epistemic uncertainty (lack of knowledge) about the complexity field, making the problem fundamentally harder to predict (confusing). In general systems wisdom, concepts like VUCA (volatility, uncertainty, complexity & ambiguity) [4] and Complex Adaptive Systems [5] are other terms that describe problems that can become a Compounded Epistemic Uncertainty Problem (CUP) for a predictive architect. To resolve such problems:

The best an architect can do to deal with such problems is to deliberately combine lateral and vertical thinking strategically. Approaching such problems from a control-theory biased perspective may lead to low resilience and inefficient solutions.

These problems arise in open, unstructured, and dynamic environments where the architect cannot confidently define or rank the likelihood of possible events or interactions, as the uncertainty profile is unpredictable due to changing complexity. Reducing such complexity based on the premise that "we shall assume the probability distribution of events to be normally distributed" may be an incorrect characterisation of the problem. With a normally distributed probability distribution of events, narrowing the issue into a pre-set of most likely or exactly a finite number of general scenarios or situations is possible. However, the by-product of such a belief system is that any rare or improbable events can be assumed to be negligible or not concerning.

In other words, this uncertainty profile means that relying on assuming normal probability distribution, which would constrain the analysis to a finite set of likely scenarios and dismiss rare (Black Swan) events as negligible, is an inadequate characterisation for open, unstructured, and dynamic environments.

E.1.10 Predictability: Architect's ability to forecast scenarios in a complexity

Predictability refers to how well we (or any predictive model or architect) can forecast the outcome of an event given a certain amount of information or a specific model of the system [2]. Predictability of an observed organisation of things depends on the predictive observer's knowledge (for example, belief system) and predictive modelling approach (the reasoning approach) [3]. The primary function of the predictive observer is to apply reasoning methods to infer patterns that align with the rules that govern the complexity field of an observation. For example, in abductive reasoning, the predictive observer infers the most plausible general rules or patterns that explain some observed states or behaviours [4].

Regardless of the scale of complexity of a system or the likelihood of particular events within a system, a few fundamental rules or principles typically govern its behaviour, leading to the observed frequency of events. The process that enables the architect to infer or design these rules forms the foundation of the architect's mental model for predicting how the system will behave under various scenarios. In predictability problems, the architect's goal is to develop a set of fundamental rules and continually refine or update them as time progresses. The architect assumes that, regardless of the probability distribution profile observed or to be observed, there are underlying rules that lead to such a distribution.

How do we find and be sure of them? This is the job of the systems approaches!

In Bayesian terms, the architect's prior knowledge of those rules reflects their understanding of how observed complexities behave and thus predictions are made accordingly. As new observations about the system accumulate, these priors are updated, effectively refining the architect's predictive model. Hence, systematic processes (like event-B and SHARCS) increase the architect's trust in their belief about the complexity of the system designed with every refinement introduced. Such a belief is observer-independent, meaning anyone who knows SHARCS ontology well can verify SHARCS models and arrive at the same conclusion. However, systematic processes (akin to vertical thinking-based processes) are highly sensitive to initial assumptions to remain preserved. For example, an electrical current can only run in one direction across a wire. If, for argument's sake, we have a situation where the electrical current becomes sentient and starts to pick and choose which direction to flow, systematic models collapse in their correctness.

Therefore, even though the frequencies of expected outcomes inform us about the likelihood of behaviours to occur given satisfaction of guard conditions, a Bayesian perspective on predictability would interpret those observed frequencies within the context of obeying specific structural rules or mechanisms that shape the system's probabilistic behaviour. Systematic approaches provide the architect with a way to infer such rules. In conclusion, systematic solution approaches like HAZOP, SHARCS, STAMP, etc, are Bayesian-based architect predictive thinking approaches that heavily depend on the architect's confidence about the probability that assumptions are valid forever.

The architect's perception and confidence in those rules cause many design mistakes (especially for autonomous systems). Open complicated problems with free-will, constrained-will agents, deterministic systems and an open environment are not the same as electrical current flowing across a copper wire. To minimise design errors caused by engineering misjudgement (in such problems), we cannot avoid managing the architect's perspective about the general rules that govern complicated open environments and how the autonomous system will interact with unforeseen complexities. Traditional systems engineering approaches needs to be adapted to such problems.

- So what? What have we learned from the above articulation?
    - When the predictive architect analyses a complexity, the architect's primary purpose is to determine the simple, general rules that govern the complexity's behaviour or the system as a whole, part of a larger whole, and the interactions among its constituent parts. The architect goes beyond merely observing the types of events or the frequency of repetitions; the focus is on identifying whether there are general rules that govern the occurrences and repetitions.
    - When the predictive architect designs complexity, they use simple rules of interactions to create a predictable complexity. For example, the architect may use AIC rules to design the holistic behaviour (system-level behaviour) of a system of interest or explain why a particular behaviour occurred in some complexity.

Suppose a system or an organisation of things and events is highly deterministic, and the architect can specify the governing rules and the required initial conditions precisely. In that case, we might say it is highly predictable (even if it has nontrivial probabilities for various outcomes when considering partial information). Such probabilities of events would occur predictably to the architect.

- For example, A deterministic system operating within a closed environment, such as an automatic building door.
- If the architect faces a problem involving one door or a thousand independent doors, from a probabilistic perspective, the frequency of failures will drastically differ between the two. If we use a probabilistic sense of complexity, more frequencies indicate greater complexity, while fewer frequencies and parts imply less complexity.

- From a predictability perspective, having a thousand predictable doors or just one door does not change the predictability state of the problem. Both problems, regardless of scale, are predictable organisations. Meaning there can be only a finite set of categories of failures that will repeat over and over again. The architect would have predicted and accounted for all these failures in his design.
- Because it is a predictable problem, it makes sense to ignore any modes of failures that are very unlikely or rare.
- In such a context, the architect may assume that the predictability of the problem can eventually be reduced to a normally distributed set of events, whereby there are a handful of most likely categories of events and failures that remain constant as the number of doors increases or decreases. For such problems, HAZOP, FMEA, STAMP, and SCHARCS can be effective.
- So what? well, those approaches come with the tacit assumption that problems can be reduced to a normally distributed set of events, whose risks can be objectively quantified. This can not be the case with Compounded Uncertainty Problems.

Suppose a system is subject to chaotic dynamics. In that case, small changes in uncertainties in assumptions or even missing ones can lead to significant differences in predicted outcomes, making the process less predictable. In such problems, any pre-set category of events or failure modes (such as those expressed as guide words) may not be sufficient to predict enough events or hazards.

- For example, an AI-based system operating in an open, complicated environment, such as an ML-based perception system operating in an open train tracks zone.
- The probability distribution of events in these types of problems is characterised as being heavy-tailed distributed [5].
- Rare events are non-trivial and cannot be ignored automatically, as they have a significant impact. Hence, if an approach that tacitly accepts the focus on a set of the most likely events and ignores less likely events is used to solve such problems, the results will be an insufficient characterisation of the problem.
- Heavy-tailed distributed complexity is dominated by low-probability, high-impact events, commonly referred to as Black Swans. Risk registers that classify risks based on likelihood and impact can be misleading methods (on their own) for such problems, as these scoring techniques accommodate the concept of a “low-probability, low-impact event" [6] [5]. Such risks often lack a statistical basis to support their acceptability level argument for Black Swan scenarios.

From above, we recognise the dependency between randomness (probabilistic uncertainty) and inability to trust predictions (predictability of the observed randomness). The literature delineates two types of uncertainty, which we view as fundamental concepts to our main topic "understanding architect predictive capability in facing complexity fields":

- **Involvement of uncertainty types:**
    - **Aleatoric (random) uncertainty** can render an event unpredictable in principle (e.g., the randomness of events during a storm concerning an ML-perception system).
    - **Epistemic (lack of knowledge) uncertainty** can also hamper predictability (e.g., a predictive architect not knowing specific parameters or types of factors involved in a storm).

The logical link between the two uncertainties can therefore be understood as:

> Lack of knowledge (ignorance) about the rules that govern a probability distribution leads to an observation's unpredictable probability distribution (and vice versa), which in turn leads to "stochastic confusion" for the predictive observer.

Now we have laid out the concepts of Probability and Predictability in the context of architect predictive thinking and the complexity of observation. We can then distinctly define the difference between the two fundamental concepts of our approach**:**

E.1.11 Complexity vs Complicatedness

Bogdanov [7] and Klir [1] share the principle that the complexity of the real world is influenced by the observer's perception of that complexity, who is capable of modifying it towards a particular desired purpose. Therefore, we believe that if we make the observer's perception "safe", we can make a safer, designed complexity. **Safe from what?** Safe from incomplete or misperception of real-world complexity, including other systems' purposes. We base our systems approach on targeting the observer's perception of the world to make engineered complexity safer for the observer and the rest of the environment. To be more specific, our approach targets the autonomous systems architect as the predictive observer; by enhancing the architect's predictive capability to form a clear picture of complexity, the architect can then make more sustainable and resilient design decisions, which in turn influence the safety of autonomous systems design.

From the above, we further define the following premise:

> *Complicatedness refers to the architect's epistemic uncertainty resulting from a potential increase in aleatoric uncertainty of the complexity of a complex.*

We use the term “potential increase”, which encompasses actual increase, perceived increase, or possible increase in randomness. Complicatedness can be in three modes of predictability:

- **Clarity:** where the complexity is somewhat predictable by the architect. Such complexity can be described as a clear “system”. By which the architect can determine the nature, time, and place of every event and state in the system.
- **Confusion:** the complexity is somewhat random and unpredictable. Such complexity (for systems engineering) can be described as a Stochastic Confusion or “non-system”. By which the architect cannot determine the purpose, nature, time, and place of any event or state in the observation.
- **Fuzziness:** Partially clear yet partially confusing. This indicates that the architect can confidently predict certain aspects of the problem and comprehend the general principles that govern its probability distribution. Conversely, there are areas where the architect experiences confusion, rendering them unable to confidently make accurate predictions.

It is essential to recognise that clarity and confusion depend on the observing architect's knowledge and mental model, which are often fuzzy. As for complexity, there are three holistic modes of complexity based on the probability distribution pattern of events and states:

- **Normally distributed complexity[4]:** the probability distribution of events and states is said to be normally distributed, where there exists a subset to is more likely to occur, and others least likely to occur. Rare events can be negligible, and most likely events can be biased by the problem solver's knowledge and mental model. An architect may design a normal complexity, and include constraints to ensure that the complexity remains normal around expected behaviours and that inputs also remain specifiable.
- **Abnormally distributed complexity:** the probability distribution of events and states is said not to follow a usual distribution trend. For example, a heavy-tailed distribution, but power-law distributions or light-tailed distributions can also be included. Such complexity is characterised by Black Swan events that cannot be neglected. They can be biased depending on the predictive architect's knowledge

[4] Note that we are also including a deterministic system in this normally distributed complexity category. In a highly deterministic system, such as a light switch, the likelihood of a single event occurring at a specific time and place is a special case of a normal distribution curve. Whereby the standard distribution curve would be 0 probability at all other categories of possible events to occur in any given time, and 100% on the triggered event. Like the histogram of a single colour image that has only one pixel type (0,0,0).

and the accuracy of their mental model that defines the types of things or events to be observed.

- **Uniformly distributed complexity:** The probability distribution of events and states is said to be uniformly distributed, meaning that all identified types of events and states are equally probable to occur at any time or place. They can be problematic (if confusing) and non-problematic (if an ultimate purpose can be identified and verified for such randomness to serve).

Based on the above classification of problem types, we can define 9 distinct modes of complexity and complication in problems.

Table E.1 Complexity and complicatedness-based classification of systems engineering problems

	Complexity Modes		
Complicatedness Modes	**Normal Complexity Field**	**Abnormal Complexity Field**	**Uniformed Complexity Field**
Clear	**Clear, purposeful, Normal Complexity (solution System, non-problematic)**	**Clear, purposeful, Abnormal Complexity (solution System, non-problematic)**	**Clear, purposeful, Uniformed Complexity (solution System, non-problematic)**
Fuzzy	**Fuzzy, Purposeful, Normal Complexity (evolving problem, partially problematic)**	**Fuzzy, Purposeful, Abnormal Complexity (evolving problem. partially problematic)**	**Fuzzy, Purposeful, Uniformed Complexity (evolving problem, partially problematic)**
Confusing	**Confusing, Purposeless, Normal Complexity (problem / non-system)**	**Confusing, Purposeless, Abnormal Complexity (problem / non-system)**	**Confusing, Purposeless, Uniformed Complexity (non-system, problematic)**

Table E.2 classifies the complexity of problems according to how confusing their probability distribution is concerning the predictive architect. The following are the definitions of each type, with an example complexity:

Table E.2 Problem types definitions

Problem Type	Predictability	Probability Distribution of Events	Architect's attitude towards the problem	Example
Clear, Normal Complexity (Solution System)	High (Predictable)	Normal	A deterministic system with fixed behaviour; all events and states are known and repeatable.	Light switch, electronic watch, non-autonomous car.
Clear, Abnormal Complexity (Solution System)	High (Predictable)	Abnormal (Heavy-tailed)	Rare or Black Swan events are explicitly modelled; the system is explainable and controlled despite irregular distributions.	AI training simulator for autonomous vehicles with rare Black Swan bias.
Clear, Uniformed Complexity (Solution System)	High (Predictable)	Uniform	Events have an equal likelihood; all categories of scenarios are known and predictable, with high epistemic confidence.	Cuneiform-based uniformly distributed AI image dataset with known variations.
Fuzzy, Normal Complexity (Evolving Problem)	Partial	Normal	Some frequent events are well-known, while rare events exist but are negligible; prediction accuracy improves with time and experience.	Automated delivery robots on campus , affected by minor anomalies (e.g., a dog chase).
Fuzzy, Abnormal Complexity (Evolving Problem)	Partial	Abnormal (Heavy-tailed)	Frequent categories are identified, but unpredictable high-impact events (Black Swans) exist and can't be ignored; they can be measured with some prior knowledge.	Autonomous cars or drones with machine learning (ML)-based components are tested under known distributions, but unpredictable events can occur in rare cases.
Fuzzy, Uniformed Complexity (Evolving Problem)	Partial	Uniform	Cannot identify any most likely scenarios; every type of event is equally probable, as the architect lacks	Drone swarm in post-disaster rescue, an unknown urban environment, with an

			reliable priors; the unpredictable part is unknowable in advance. but, some parts of the complexity are predictable.	unstructured and volatile operating domain.
Confusing, Normal Complexity (Problem/Non-System)	Low (Unpredictable)	Normal	The architect doesn't yet understand why a deterministic system behaves abnormally; root causes exist but are unclear, and rare causes are assumed to be negligible.	Warehouse robot fails to respond due to obscure or hidden issues in a closed environment.
Confusing, Abnormal Complexity (Problem/Non-System)	Low (Unpredictable)	Abnormal (Heavy-tailed)	Non-deterministic system; high unpredictability; causes unclear; rare events can't be ignored; some interactions are hidden or unknown.	An autonomous robot fails its mission in a semi-controlled environment, such as road traffic, despite being functional.
Confusing, Uniformed Complexity (Problem/Non-System)	None (Unpredictable & Unclear)	Uniform (Unknown)	Compounded Uncertainty Problem — complete unpredictability in open, unconstrained environments; the architect cannot prioritise risks; complete epistemic and aleatoric uncertainty.	Multiple autonomous systems trained on confusing datasets (where the architect has no grasp of the scenarios used in those datasets) operating in a busy marketplace or open airspace (e.g., above jungle or urban areas).

Table E.3 provides a structured classification of systems engineering problem types based on their predictability, the probability distribution of events, the architect's cognitive relationship to the problem, and real-world examples. It distinguishes between solution systems, evolving issues, and problem/non-system scenarios, highlighting how increasing uncertainty and environmental openness escalate the system's unpredictability and complexity. As problems transition from clear to fuzzy to confusing, the architect's ability to model, predict, and manage system behaviour diminishes, especially under conditions where Black Swan events or uniform uncertainty dominate. This classification informs the nature of challenges engineers face. It guides the selection of appropriate design, assurance, and reasoning approaches, ranging from deterministic control in clear systems to adaptive and exploratory strategies in complex, uncertain contexts.

E.1.12 Randomness, Predictability, Confusion and Eureka

Although we articulated earlier that it makes sense to believe the following organisational concept:

More aleatoric randomness means more potential unpredictability for a predictive observer. The impact of randomness on predictive observer confusion depends on how well the chosen mental model handles the relative outlier without requiring changes to the initial categorical system.

We defined the concept Stochastic Confusion as the state of disorder where the complex is random and thus unpredictable. The question then left would be: what would be the state experienced by the predictive observer, where it receives sufficient knowledge and experience with randomness, such that randomness becomes predictable? In other words, predictable randomness. Such a state is truly "confusing," because randomness is inherently unpredictable. This is a kind of positive (relative to the observer) type of confusion, where the observer is confused but in a good way. Maybe we can name it "Eureka"!. Eureka is a state where the predictive observer finally gains enough knowledge to predict the randomness of an observation. But Confusion is still present in Eureka!. It is understandable, because even If the observer gains more knowledge, the fundamental construct of knowledge remains "randomness" or "uncertainty" so it becomes a doubtful or temporary or false absence of confusion. Learning is a cyclic process between stochastic confusion and Eureka. When the architect reaches Eureka, the architect has defined a system.

E.1.13 Systemness of a General Complex

We start our articulation for a general definition of a system (which we need to base the definition of Compounded Uncertainty Problem) by defining a general fundamental term:

> **General Complex:** a set of coexisting elements and events (regardless of nature) that may or may not be interacting. The definition also includes all possible relationships or interactions that could emerge among those elements.

We reuse Bogdanov's definition of the complex as an entity, a combination of elements. We distinguish complexes by saying that not any complex of things is a system. However, the general systems scholars tend to define a system as everything is a system. George Mobus introduces the concept of Systemness [3]:

> "Systemness: Bounded networks of relations among parts constitute a holistic unit. Systems interact with other systems, forming yet larger systems. The universe is composed of systems of systems."

Now, the definition remains too abstract to be practical from an objective engineering perspective, where we need a quantifiable notion to assess as precisely as possible whether we are right or wrong about something. In our view of systems engineering, the problem boils down to the following general test:

> Have I produced a predictable and explainable set of interacting situations (a complex), regardless of any possible context this complex may find itself in an operational environment, such that my intended purpose is guaranteed for as long as the complex exists?

So what? well,

- Have I managed to predict all possible contexts of the operational environment?
- Have I made a predictable and explainable system of responses to that context?
- Have I guaranteed that the primary purpose of the engineered complex shall be preserved in all possible operational contexts?

All in all, we see ourselves repeating one primary concept, "predictability" of what? of a set of situations or events. Where there is predictability involved, there is "probability" lurking somewhere. Where there is probability, there is "uncertainty". Luckily, all concepts "Predictability", "uncertainty" and "probability" are measurable. Therefore, for us to base our systems approach on a quantifiable definition of Systemness, that we can re-use and validate our creations and models, it will have to be associated to:

- Predictability
- Uncertainty
- Probability

From that, as part of our foundational theory, we will make an assumption about the Systemness of a complex and say:

> ***Systemness of a complex*** *is the potential presence of predictable rules that govern the randomness in a complex. We also understand Systemness as the capacity (probability) that a complex verifiably adheres to constraining rules assumed by a predictive architect.*

To give an example of a complex that is impossible to predict its behaviour, a theoretical 100-rigid-rod pendulum with a self-determining black box ML component that makes random decisions on which rod it will influence its behaviour. The set of events (impulses delivered by the black box) is unboundedly random; no invariant statistical regularity emerges. The "architect" cannot posit any finite rule (deterministic or probabilistic) to summarise black-box decisions because each new trial generates counterexamples. Consequently, although the 100 rods and the black box coexist and interact, the capacity (probability) that the complex adheres to any constraining rule is effectively zero. There is no emergent law that constrains the randomness in a stable way.

While an example of a complex that has capacity to be studied and predicted would be an ideal gas piston. Although the ideal gas's microscopic evolution is random and chaotic, the emergent statistical rules (Maxwell–Boltzmann distribution, ideal-gas law, etc.) are robust. Known physics fixes the set of possible interactions (elastic collisions, boundary reflections), and there is no hidden "intelligence" reprogramming the particles on the fly. Therefore, unlike the 100-rod pendulum, an architect's hypothesised "constraint" (e.g., "every experiment at volume V and temperature T yields pressure p obeying pV=Nk_BT") is correct to within diminishing error margins.

Some Predictability of the set of events that will have an impact, and the predictability of the probability distribution of those events occurring in a complex of interacting situations that impact the belief state of some predictive observer's uncertainty (architect or machine). such that:

- ***Complicatedness*** *is a function of the predictability of a complex relative to prior knowledge of a predictive observer about the complex. We define Complicatedness as being directly related to the epistemic uncertainty of the predictive observer (the architect's knowledge or the training corpus for the ML model regarding the problem space).*

While epistemic uncertainty is reducible, complicatedness may be efficiently reducible (the rate of epistemic uncertainty reduction with respect to aleatoric uncertainty increase), depending on the nature of the approach's predictive thinking model.

- ***Complexity*** *is a function of the nature of the randomness distribution of events occurrences among the elements of a complex. We define Complexity to be related to the aleatoric uncertainty associated with the observation.*

Complexity cannot be reduced by collecting more data. However, the more information gathered, the clearer the nature of Complexity becomes for a predictive observer. However, it may or may not be manageable by the reduction of Complicatedness for an influential predictive observer.

We proposed those definitions at a conference to ISSS, which received an initial appraisal of their acceptance in our published work [6]. In this research, we will not attempt to formalise and prove such a definition; this will be done in future research. For now, we will accept as an assumption the following relationship between the uncertainty of the predictive observer, the predictability of an observed probability distribution.

We realised the definition above after observing the output of our AIC analysis in Stage 1 (see Section 9.3, Findings 5, 6, 7). Also, after conducting experiments in section 7.2, we trained an ML model and tested its confidence about a set of images that capture the observation of systems interacting. We noted that the ML model lost considerable confidence in Black Swan datasets (a testing dataset which was visually out-of-distribution (OOD) with its prior training and validation). We trained a model with nearly 85,000 images of drones[5], and when we tested it on a set of OOD images (which can be thought of as informational systems), the model failed drastically. But for us, those hard images for the ML are easy. This situation inspired us to realise that:

- Black Swan images had complexity. This complexity is somewhat related to the histogram distribution of a set of pixels that capture certain physical events. They appear meaningful to us, and we can easily predict what is in them.
- Those Black Swan images appear to be ***confusing*** for the ML, ***harder to predict***, but they are ***clear*** for us, ***easy to predict***.

Since the images are clear and predictable, we can clearly see the presence of a recognisable pattern of a system. For a sufficiently trained model, Black Swans were unpredictable; clearly, there is no system. So, the definition of “system” or “no system” was purely based on whether a probability distribution was predictable or unpredictable with respect to a predictive observer's prior experiences and intelligence (inference methods).

If the pictures were entirely composed of pure noise, with each image having a completely different distribution of noise and randomness, can we say that we recognise a pattern of a system? No, we can’t. That is an example of “confusion”, so total confusion is unpredictable by the observer and thus ***not a system***. Had it been predictable by a predictive observer, the total stochastic confusion would have been a ***system***. We elaborate further on this topic in Appendix E, section E.1.

[5] See section 6.9, H.10 for the experiments we conducted.

E.2 Ordered and Disordered AIC timing

When all AIC interactions occur spontaneously, the timing and order of AIC are said to be harmonic. This means the system spontaneously appreciates some systems, controls some, and influences some situations. This is the ideal timing when systems spontaneously perform AIC interactions. However, it is also possible that systems can perform ordered non-harmonic timing among AIC interactions. This entails the following logical non-harmonic AIC interactions: timing or order (the clockwise ordered AIC): ***agents must first appreciate to enable control to achieve the desired influence*** to avoid confusion.

In the context of predictive thinking, for a clockwise ordered AIC thought process, to resolve confusion, the architect's predictive thinking should first appreciate the observed complexity field and recognise control interactions to realise the intended or predicted influence. For engineering an agent, the architect must consider what appreciable factors (factors that cannot be influenced or controlled by the agent, yet can influence the agent), then define what controls are needed to achieve the desired influences. Figure E.2 describes the clockwise AIC timing for the complexes' behaviours. Epistemic uncertainty or uncertainty is always present, so confusion is always waiting to be experienced.

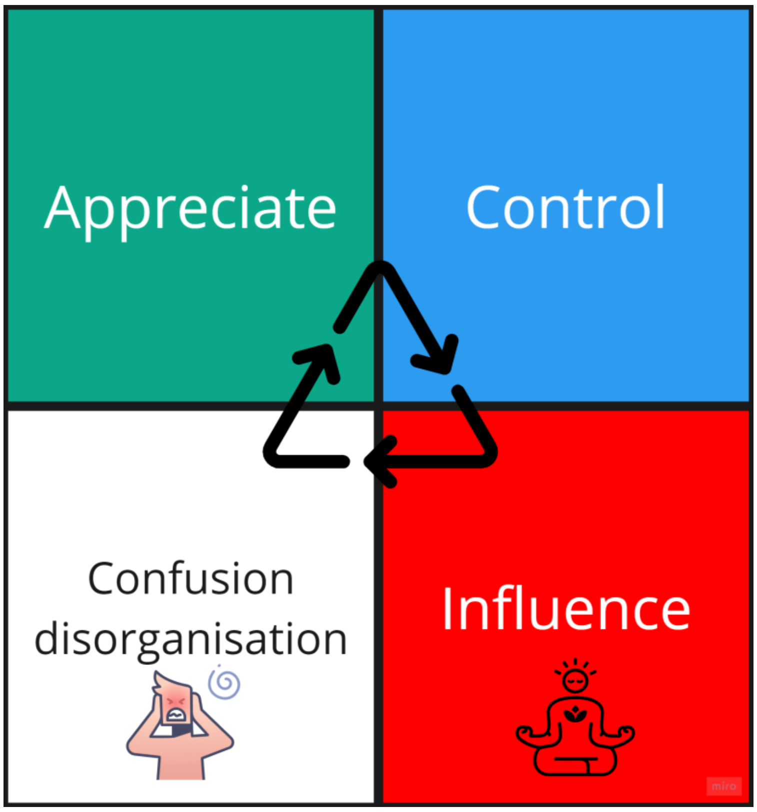


Figure E.2 ordered non-harmonic clockwise AIC timing.

Harmonic and Non-Harmonic Timing:

The following defines the nature of AIC timing variety for the potential emergence of complicated interactions among complexes in a complexity field:

1. **Emergent Harmonic Timing**:
    - All AIC interactions occur spontaneously and in balance.
    - The system seamlessly appreciates, controls, and influences as needed.
2. **Emergent Ordered Non-Harmonic Clockwise Timing (intuitive thought)**:
    - AIC interactions follow some chained rules and logical sequences, relatively enhancing system efficiency:
        1. **Control precedes influence**: Influence takes effect after control actions.
        2. **Appreciation precedes control**: Control actions happen after appreciation.
    - **Optimal Order**: Appreciation → Control → Influence (AIC clockwise).

We would consider models 1 and 2 as an "intuitive" sequence of events**.**

3. **Emergent Disordered Non-Harmonic Timing (counter-intuitive thought)**:
    - AIC interactions deviate from harmonic or ordered patterns, potentially causing confusion or inefficiency of cognitive energy to make most accurate prediction.
    - **Example:** A cameraman lifting a ribbon (control action) before noticing an approaching robot (late appreciation).

An analogy to make sense of the AIC timing orders: Consider a scenario of the Eagle Drone deciding to cross over train tracks:

The architect decides that clockwise timing[6] would be common sense to predict the chain of events and derive subsequent requirements:

- Appreciate the status of train movement across the train tracks.
    - **So what?** The architect must discover requirements for training scenarios involving train tracks and trains at various distances.
- Control safe crossing (avoid power lines or any other obstacles on the route)

[6] Think about the intuitive thought process you experience before you cross any street. What are the sequences of transitions that your mind mentally goes through? Analogously, we subconsciously do this when we cross the road. Perhaps there is a reason why our predictive thinking processes developed in such a sequence. We speculate that such an order would be the most efficient thinking sequence, ensuring minimal energy to ensure survival. After all, crossing first and then appreciating comes with a significant loss of energy and time and may lead to disasters.

  - **So what?** The architect must discover requirements for control decisions and actions to constrain the crossing or avoiding of train tracks.
- Influence the security on the other side of the train tracks.
  - **So what?** The architect must discover requirements to mandate the success of deterring intrusion into train tracks when such capability is demonstrated. Such requirements can be in the form of test scenarios.

Examples of AIC Timing:

- **Emergent Harmonic Timing**:
  - A self-driving car appreciates obstacles, controls its steering to avoid them, and influences a collision by preventing it.
- **Emergent Strategized Non-Harmonic Timing**:
  - An Autonomous System may alter the AIC order based on experience or the situation:
    - Steering into another lane (control) before appreciating an obstacle because of an emergent behaviour where the car constantly finds itself steering around at a particular time of the day. After a while, it learnt to do it after removing the object. This may lead to a safety hazard, an example of an emergent soft hazard.
    - Collecting environment data (appreciation) before taking control actions.
    - Such behaviour may be deemed as hazardous or a useful affordance by the architect.

Significance of AIC order and its usefulness:

- **Predictive Lateral Thinking:**
  - By reversing or disrupting AIC timing, architects could invoke lateral thinking and thus can anticipate potential emergent behaviours, black swan scenarios or failures and plan mitigations. For example, what can happen within the Eagle Drone system that would cause it to control before it appreciates?
- **System Behaviour Design:**
  - AIC timing rules provide a framework for imagining or predicting potential emergent behaviours (agent interventions) that lead to the definition of additional system requirements.

E.3 Safety and Security Concepts in the Context of AIC

Safety: When we think of safety, we think it of the following terms:

> Don’t cause unintended physical harm to yourself or others by effectively appreciating and controlling your own and environmental behaviours.

In other words, Safety refers to a system's ability to influence and prevent harm or danger to the primary purpose of the engineered system. In the definition, we also include the anticipation of causing harm or danger to other complexes' primary purposes within the complexity field. It often concerns preventing harm to humans, property, or the environment. It encompasses the system’s capability to operate safely under normal conditions and to mitigate risks arising from unforeseen circumstances. Safety is evaluated by identifying hard and soft safety hazards and implementing mitigating interactions or capabilities.

Safety Hazards: A direct, immediate harm within the system’s complexity field or external environment that physically harms the system or its surroundings [7] [8].

General modes of a Safety Hazard:

- **Loss of required Appreciation:** The system fails to recognise critical environmental factors or internal states, leading to unsafe behaviours (e.g., misperception of obstacles, sensor blindness in critical conditions).
- **Loss of required Control:** The system loses the ability to directly regulate its behaviour or an environmental system directly, causing unintended physical harm (e.g., an autonomous vehicle failing to brake due to control malfunction).
- **Loss of required Influence:** The system loses the ability to indirectly regulate other systems in its environment or itself (e.g. an autonomous car loses the ability to influence another autonomous car's decision to not overtake in a particular situation [like a sleeping driver]).
- **Emergent Risk:** Unintended system-environment interactions create new hazards, even when no immediate failure occurs (e.g., an AI-driven surveillance drone unintentionally distracting drivers and causing accidents).

Safety Risk: The likelihood of a safety hazard occurring [9], considering both current and anticipated complexity field structures.

Security: When we think of security in general, we think of the following:

> Securing the longevity of a desired autonomous system's appreciation, influence and control effectiveness (over changes in the complexity field) from unaccounted-for emergent affordances or intentional unauthorised influence or control over the system.

Security may focus on protecting the system's ability to govern (control) intended situations from malicious attacks, unauthorised access, or intentional threats [9]. It ensures the system's integrity, availability, and confidentiality, safeguarding it from being controlled by undesirable influences (human or otherwise). While Safety is a broader concept related to protecting the system's primary purpose and others, security is more focused on securing the system of concern's ability to exert governance and control through its auxiliary control goals and actions.

Security Threats: Any emergent affordance (unaccounted for during design time), unauthorised influence, or adversarial control that compromises an autonomous system's ability to appreciate, influence, or control its operational environment, leading to potential degradation, exploitation, or loss of system integrity over time.

General modes of a Security Threat:

- **Appreciation Compromise**: The system's ability to recognise and interpret its environment is manipulated, blocked, or deceived (e.g., sensor spoofing, adversarial perturbations).
- **Influence Disruption**: The system's ability to modify or interact with its environment is hijacked, altered, or constrained (e.g., jamming, misinformation, false data injection).
- **Control Hijacking**: The system's ability to execute autonomous decisions is designated, overridden, or exploited (e.g., hacking, unauthorised access, AI model poisoning).
- **Machine's emergent defiance**: This is when the machine makes an artificially intended decision that does not align with the architect's intent and perspective of what is right or wrong based on the machine's self-assessment of what is best for itself or others.

A security threat does not necessarily cause immediate harm but introduces vulnerabilities that could be exploited over time, leading to unintended system behaviours, failure modes, or adversarial control. Security threats can originate from both external actors (cyber-attacks, adversarial entities) and internal system weaknesses (unsecured affordances, design oversights).

Situations that compromise the system's control are often caused by vulnerabilities in the system's complexity field. These vulnerabilities can be attributed to the architect's lack of knowledge or foresight on how the whole complexity field (the system being part of it) behaves over time. This accumulates as a lack of appreciation of the whole.

- **Example**: A hacker can intercept and control an autonomous delivery robot with a vulnerable wireless communication system. The system's appreciative interaction with communication systems (e.g., GPS) becomes a security vulnerability.
- **In the context of AIC**: Such vulnerabilities threaten the system's influence and control, potentially leading to obstructive forces taking over. Appreciative interactions are the apparent interactions that can open complexes to exploitation if other actors control the appreciated complexes.

Security Risk: The likelihood of a security threat manifesting, considering current and future complexity field structures.

Negative Affordances and Potential of Security Threats:

Constraining unknown possible affordances could be considered a straightforward theme differentiating security from safety. For example, the Eagle Robot flying over train tracks may be secured and safe. However, the nature of having a flying drone over train tracks behind local houses provides several side-affordances that may not be part of the design intent:

- It provides excellent training content for adversarial counter-security drones. For example, it can be filmed and photographed easily and used for training adversarial platforms.

- It provides local news with content that can be exploited for political campaigns or other social engineering purposes (an example of a soft hazard).

These cases are not safety concerns; they do not cause direct physical harm to the system, which makes them a distinct security concern. Therefore, we believe the "exploitation of possible affordances" is purely security-related. We capture those affordances in the HazTops method (section F.5) under the "Opportunities/Affordance" concept.

Key Distinctions Between Safety and Security

We distinguish some key conceptual differences between Safety and Security. We believe that confusing security requirements with safety requirements and vice versa could introduce inefficiencies in the design. For example,

- **Safety requirements:** The Eagle Drone must not fly too close to train catenary power lines to prevent an electrical short circuit that could cause fire.

- **Security requirement:** Jamming Attack: If adversaries use radio-frequency (RF) jamming, they could disrupt the Eagle Drone's GPS navigation, causing it to lose control and potentially crash or exit the secured perimeter.

A jamming attack is not a safety concern, as it may not directly cause physical damage. Given enough gust of wind energy, patrolling very close to powerlines will most likely cause a collision and fire. If patrolling next powerlines is misclassified as a security concern, the architect might assume the risk only arises from an intentional attack, such as adversarial hacking or deliberate sabotage, rather than an operational safety risk inherent to normal environmental conditions (e.g., wind gusts, loss of stability, miscalculated flight paths). This may reduce the likelihood of it occurring in the architect risk analysis thought process, thus assigning a lower priority for mitigation. Therefore, we attempt to explain the difference between Safety and Security below in order to help the architect make better engineering judgements:

Table E.3 The difference between Safety and Security Engineering

Safety	**Security**
○ Deals with preventing direct physical harm, regardless of whether the system is reliable or not. ○ A broader concept concerns the impact on the system's primary purpose and its impact on other complex primary purposes within the complexity field as a whole. ○ **Example:** Ensuring a drone avoids collisions in urban environments. ○ A broken-down micro drone landing in an area that does not present a direct hazard to the environment can be a safe drone for itself and its environment.	○ Deals with protecting the system from malicious or unauthorised influence or control. When an authorised or vetted controller becomes an appreciating stakeholder of the engineered system, the unauthorised controller or user becomes an appreciated system. ○ Deals with preventing negative affordance exploitation by users or unauthorised entities. ○ **Example:** Preventing hackers from intercepting and taking control of a drone's navigation system. ○ A broken-down micro drone landing in an area that does not present a direct environmental hazard is not necessarily secured because it is vulnerable to theft. Anyone can pick it, take its parts, reverse engineer it and reuse them for something else.

Interrelation between safety and security: A lack of security can compromise safety [7]. For example, a hacked autonomous vehicle could become dangerous, blending security and safety concerns. Conversely, a system can be secure but unsafe or safe but insecure, depending on the context. The following is an illustrative example of how Safety and Security can interplay during the design process:

Table E.4 Illustrative Scenarios in Autonomous Systems

Safety & Security integration type scenario	**Potential conflict of interest or contradiction examples**
Unsafe but Secure	**Scenario 1:** A drone operates based on a secured training process but has a hidden uncertainty in the training set of its obstacle-avoidance algorithm, risking collisions. 1. A potential appreciative interaction between a trained model and the hidden biases in the training data. 2. The architect identified a problem concerning an insecure training dataset, which led the architect to define a limitation on access to the training set. This security requirement may weaken dataset validation, resulting in hidden biases that could lead to failure to avoid issues and obstacles. **Scenario 2:** The lock on an autonomous car breaks down while people are inside**.** The car is secure but potentially unsafe in case of fire. **Explanation**: Security measures prevent external interference, but internal faults make the system unsafe. So, if the security requirement imposes stronger lock mechanisms, this may mean jeopardising people's safety in case of fire due to a fault in the LiOn battery sub-system.
Insecure but Safe	**Scenario 3**: An autonomous car operates safely under some environmental conditions but is vulnerable to theft. A car may not be operational (safe), but the door is unlocked as per

	a safety mitigation requirement to make the vehicle unlocked when broken down, or the car could be towed away. **Explanation**: The system is safe but remains insecure due to potential theft or undesirable relocation. Another example is smoke inside the autonomous car. The internal car perception system identifies the presence of a child, deactivates the child, and the child opens the door while the car is moving. The car is safe but insecure.
Unsafe and Insecure	**Scenario**: Cyber-warfare compromises a fleet of drones, causing them to lose control and collide with obstacles. **Explanation**: The drones are both insecure (due to the attack) and unsafe (causing harm).
Secure and Safe	**Scenario**: An autonomous vehicle employs advanced encryption and intrusion detection while operating safely under rigorous testing. **Explanation**: The system maintains security and safety, minimising risks to users and the environment.

E.3.1 The Intrinsic Hard Hazard of Appreciation

All appreciative interactions are the Complexity field's weakest links for maintaining a primary purpose.

By "weakest link", I mean the likelihood of a complexity losing purpose may come from its appreciative interactions. This is because the nature of appreciation is basically "complete dependence with no possible capability to influence the behaviour of the appreciated". This is a very useful notion because the architect must think of appreciative interactions to enable control, and by naming those appreciative interactions, the architect can immediately anticipate hazardous scenarios originating from the appreciation.

The hazard is related to the potential safety and security vulnerability associated with appreciation. When system A is in an appreciative interaction with some unavoidably influential appreciated situation D, whoever controls or influences D unavoidably influences or controls system A. This means that for any appreciative relationship, the appreciative system must also appreciate other situations that can influence or control its appreciated situation. For example, an autonomous car (system A) appreciates visible traffic signs (situation D); however, whoever or whatever influences or controls the clarity and visibility of a traffic sign unavoidably influences or controls any passing autonomous vehicle. This is an important concept, especially in cyber-security, as whenever an appreciative interaction is defined, a cyber-security or safety threat is inherently associated with appreciative interactions. If appreciative interactions are exploited, the system is open to whatever exploiting system's goal or PrimeP.

E.3.2 Soft Hazards in Complexity Fields

A **Soft Hazard** is a situation that impacts the realisation and maintenance of an ideal operational environment system where an action, decision, or inaction may not directly threaten or harm the immediate safety or functionality of the system of interest. Instead, Soft Hazards pose intermediate or long-term risks or consequences that negatively impact the success of the deployment mission, broader environment, or social context, where autonomous systems may eventually be affected by it. For example, we consider "Ethical Threats", a re-wording of the concept "Ethical Hazard" [10] as an example of Soft Hazards; identifying such hazards is often philosophical and subjective depending on the designer's ethical awareness and culture, as they may involve decisions about intent, fairness, justice, privacy, or the unintended side effects of the deployment of the autonomous system.

The main point here is that we are considering an interaction within the overall problem complexity field which the autonomous systems are part, not necessarily where the autonomous systems are directly involved. An example of a soft hazard is an ethical hazard. Below is a figure that explains a soft hazard:

Let's take the Eagle Drone, for example; a road rage incident between a car driving over the bridge and a pedestrian may not be of Eagle Drone's direct concern. However, this ethical hazard impacts maintaining an ideal operational environment, which includes autonomous systems. Perhaps it may affect the success of drone deployment in the long term, such as blaming the drone's presence at some part of the train track zone, which causes pedestrians not to pay attention while crossing the road, thus leading to question the overall safety of local people or even be subject to criminal prosecution. It may also lead to bad press for the organisation operating the drone and, thus, potential early decommission and deployment failure. The thinking type that helps discover such scenarios is the main problem solver for long-tailed complexity problems.

Soft hazards can refer to the potential of becoming a physical hazard in the future due to a change in the system of interactions, even if it's not currently a hazard. In simpler terms, they are potential future dangers or causes of harm.

In the context of autonomous system design, Soft Hazards arise when the system behaves in a way that, while safe or functional in isolation, has negative implications for other stakeholders or broader systems. For instance, a security drone may patrol a train track safely. Still, its surveillance capabilities could unintentionally infringe on privacy rights or create social inequities, thus posing a Soft Hazard even though it performs its intended function without safety issues.

Example:

In the design process described, the AIC mental model originally conceptualised a police officer performing a control action **"remove"** on trees, which was initially seen as a neutral effect action concerning the trees. However, upon reflection, we realised it is better to assume the worst in this case to have some mitigation in place, so we updated the effect to be "obstructive" representing a potential the police may have ill-intent towards local trees as they prioritise the success of the autonomous systems deployment over the wellbeing of the natural environment which affects local people in return and the problem domain as a whole. This is an example of a Soft Hazard; it emerges from an interaction between two complexes in the problem domain, where the system is not a party. Even if this ill intent does not impact the safety or operation of the system itself, it could negatively affect the broader context, such as environmental sustainability or social equity. This scenario highlights the role of ethical considerations in identifying and mitigating soft hazards.

E.3.3 Soft Hazard vs. Hard Hazard

We delineate two types of hazards to be considered in AIC systems Approach:

Table E.5 Types of Hazards in AIC Systems Approach

Hard Safety Hazard	**Soft Safety Hazard**
Directly threatens the operation or users of the system, potentially leading to physical harm or autonomous systems failure.	Soft hazards are risks to the realisation and maintainability of ideal operational environment dynamics, which autonomous systems are part of. These may include aspects such as societal, environmental, or ethical norms, such as privacy violations, fairness, bias, or negative long-term consequences, but they do not immediately impact the safety of the system itself.

	o A form of Black Swan hazards, which are scenarios that are deemed to be unlikely or rare or hard to believe to be of concern, yet if they do occur, will have a surprising impact on safety and security. o A potential future harm or threat that could evolve into a hard safety hazard due to changes in the complexity field. This may not pose an immediate danger, but it requires foresight to anticipate and mitigate. **Soft Hazards can emerge from** interactions within the complexity field that have to appreciate the system of interest behaviour, and the architect may miss their impact in the future. This can be predicted using an AIC perspective shift.
Example: An autonomous vehicle fails to detect a pedestrian due to hidden bias in its object detection algorithm, creating an immediate risk to the pedestrian, vehicle occupants, and nearby property. The hidden bias becomes a source of malfunction within the system.	**Examples**: ▪ **Wildlife Interaction**: Animals near a train track zone monitored by a security drone may not present an immediate hazard but could become one if deployment conditions change and animal habits change with it. ▪ **Weather Conditions**: Adverse weather outside a drone's immediate operational zone might impact future deployment. For example, an erupted volcano in a different country could alter natural light conditions across an area where autonomous systems (trained on normal conditions) operate.

	▪ **Human Factors**: Local population temperament could evolve into a safety threat in specific contexts. ▪ **Human or free-will agents' intent** is also a soft hazard since it can only become evident over time as their complexity changes. This also include the architect intent.

Properties of Soft Hazards for Safety Design Consideration:

- **Subjectivity:** Soft Hazards are often subjective, depending on the ethical values and bar set by the architect. What one may consider a Soft Hazard (e.g., excessive security drone surveillance) may not be regarded as a hazard by others.
- **Mitigation:** By identifying these Soft Hazards early in the design process, more constraints can be placed in the operational design domain, and the system can be modified to mitigate them, contributing to creating a **Trustworthiness Case**. This case demonstrates that the system's deployment is functional, safe, secure and ethically sound.
- **Time Transcendence:** Soft hazards can happen in the future. During design time, the complexity may not pose any dangers to the robot; however, some aspects of the operational domain complexity may become a hazard in the future. For example, drones flying across the train track zone over time may impact the presence of bees and local birds, triggering resistance from the community to terminate the robot operation, thus impacting the system.

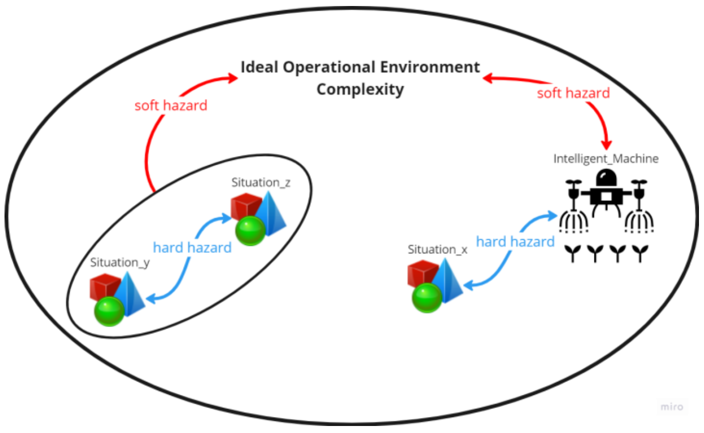


Figure E.3 Soft Hazard Model

Figure E.2 illustrates the dynamic interaction between Soft and Hard Hazards within an Ideal Operational Environment of a complex, intelligent system. The model demonstrates how soft hazards arise from broader environmental, ethical, or social interactions that may not pose immediate threats to the safety or operation of the system (e.g., an autonomous drone), but can negatively affect the long-term viability of its deployment mission or the societal context in which it operates.

In the figure, Situation X and Situation Z represent distinct operational scenarios within the system's environment. Hard hazards, which include immediate operational threats (e.g., physical collisions), are marked by blue arrows, indicating direct risks to the system's functionality. In contrast, red arrows symbolise soft hazards, reflecting broader, less immediate concerns—such as ethical or societal issues—that emerge from the system's indirect involvement in or impact on the environment.

For example, soft hazards may arise from intended or unintended consequences of the system's behaviour, such as a surveillance drone infringing on privacy. This could lead to a loss of public trust or regulatory backlash over time. This model emphasises that while soft hazards may not directly affect the system's immediate operation, they pose a significant risk to maintaining an ideal operational environment. Therefore, it is crucial to identify and develop mitigation strategies for these hazards early in the design process.

E.4 Lateral and Vertical Predictive Thinking thought experiment

We will use a thought experiment example (Autonomous Drone Scenario) In order to examine predictive thinking as an intellectual phenomenon:

Imagine an architect tackling the problem of an autonomous security drone tasked with monitoring a train track zone. In a vertical approach, the architect may start with a rule (requirement): "Any object on the track that is not a scheduled train is a threat." With this in mind, consider how a vertical thinking architect might proceed and how a lateral thinking architect might approach such an initial thought.

How a typical architect or designer may proceed as a vertical thinker: We are assuming the following as a typical design thinking process based on my own thought process. Further research and validation will be explored In future research.

1. **Initial Assumption/Rule**:
    - The architect begins with the general common belief: "Any object on the track that is not a scheduled train is a threat."

2. **Observation**:
    - The architect may imagine a drone detecting an object on the track. This object may be a piece of equipment left by a maintenance crew.
    - The architect follows a deductive reasoning method.
3. **Contradiction**:
    - The architect may think that the initial assumption fails because the imagined object is not an actual threat but is misclassified as one due to the rigid rule.
4. **Response to Contradiction**:
    - The architect refines the initial assumption or adds new theorems:
        - **Refining the Rule**: Modify the assumption to say: “Any object on the track that is not a scheduled train or authorised maintenance equipment is a threat.”
        - **Adding Theorems**: Introduce a new theorem: “Maintenance equipment can be authorised by tagging it with specific markers detectable by the drone.”
5. **Iterative Process**:
    - The architect continues iterating, refining rules, and adding exceptions to handle future observations, gradually building confidence in the updated model.
6. **Outcome**:
    - A refined understanding emerges: “Authorized objects, such as maintenance equipment, should not be treated as threats,” with clear rules for identification. The system becomes more accurate, but it relies on predictable patterns and known categories.

How the architect may proceed as a lateral thinker. I am assuming the following as a typical lateral thinking process based on my thought process**:**

1. **Initial Approach**:
    - The architect approaches the problem without assuming the existence of a general rule, such as “Any object on the track is a threat.” Instead, they explore the track zone as a complex field, acknowledging that each observation may reveal new and unique patterns.
2. **Observation**:
    - The architect may imagine a drone detecting an object on the track. The architect does not categorise it immediately as a threat but instead considers multiple analogies of “object on tracks”:
        - Is this object similar to maintenance equipment seen in other contexts?

        - Could it be an unauthorised object accidentally resembling maintenance equipment?
        - Could it indicate an entirely new type of situation, like an animal or a weather-related phenomenon?

3. **Exploration of Patterns**:
    - Instead of refining a single general rule, the architect generates multiple hypotheses:
        - “Objects on tracks could be threats, maintenance equipment, or neutral entities.”
        - “How do different categories of objects behave or interact with the track environment?”
    - They analyse and compare the new observation to diverse scenarios without attempting to converge on a fixed definition or rule.
4. **Expanding the Knowledge Base**:
    - The lateral thinking architect adds this observation as a new training class or pattern to the system, allowing the drone to recognise it as distinct in the future without prematurely classifying it as a threat.
5. **Outcome**:
    - The Predictive Problem Solver learns to handle ambiguity and counterintuitive scenarios by associating new patterns with prior observations. The approach remains open to discovering unexpected situations without oversimplifying or rigidly defining them.

Thought experiment observations:

- **Vertical thinking**: Systematically refining rules and addressing contradictions, aiming for a stable and predictable system. It excels at resolving known issues but may struggle with unexpected scenarios or patterns. The approach is somewhat fundamentally deductive reasoning-based.
- **Lateral Thinking:** embraces uncertainty and explores new possibilities through analogies and associations. It generates diverse solutions and is better suited for handling counterintuitive or entirely novel scenarios. However, compared to vertical methods, it may lack precision and stability. The approach is somewhat fundamentally inductive reasoning-based.

The characteristic difference between vertical (sequential or vertical) and horizontal (lateral) thinking can be reframed using concepts from probabilistic distributions. A vertical or algorithmic thought process quality is sensitive to initial assumptions and sequentially or systematically builds up confidence in the prediction based on the correctness of a single assumption. If you build a vertical axiomatic system with thousands of propositions proven by theorems based on a few axioms, then a small change in the assumptions will lead to the collapse of the proof system. This issue sounds like a butterfly effect. Therefore, we believe that the vertical thinking approach or proof system is essentially a predictable chaotic system, in the sense that a small perturbation (such as changing one aspect of the foundational axioms) will lead to the collapse of the entire system's correctness [11]. At least, due to precision and domino-like vertical reasoning, the architect may be able to predict and understand failures very well.

In other words, we view the stability of the correctness of a vertical axiomatic system as being, at best, “metastable” [12] or semi-robust because its stability or robustness is very sensitive to the slightest changes in the real-world complexity field that may contradict its founding assumptions.

Why is this important to know?

The architect’s knowledge in developing any axiomatic system is finite; therefore, comprehension is inherently deficient and possesses some level of residual epistemic uncertainty. This Residual Epistemic Ignorance can be minimised if the perception of the modelled real-world complexity is compared to reality; however, every complexity field is subject to change at any given time. So, suppose the initial perception of founding assumptions is not resilient to change (meaning it does not resist change in various changes of complexity and time). In that case, a vertical thinking process (such as a vertical formal model) cannot be expected to inspire comfortable confidence from its architect as the complexity field evolves in the real world.

When a contradiction is found, either the initial assumptions or subsequent theorems, or both, must be revised. However, vertical thinking is the most valuable predictive thinking approach when it is used to discover contradictions. Every contradiction exposes a knowledge gap that can be fed into the axiomatic system. The main risk is “accumulative bias"; as vertical thinking progresses, the modeller’s bias converges to a single view of the complexity field.

That is the weakness of vertical thinking when used to predict Black Swan scenarios (aka solving long-tailed, power-distributed real-world complexity). Bias creates an invisible barrier between the Predictive Problem Solver and the long-tailed, counter-intuitive scenarios (which are Black Swan scenarios). The vertical thinker's tacit assumption is that the world or any problem domain can be simplified into an elaborate domino puzzle. Such oversimplification may lead to undesirable surprises when approaching abnormally distributed complexity fields.

Generally speaking, in a vertical thought process, we might proceed as follows:

1. **Initial Assumption:** Street lamp posts are always fixed permanently away from the main street and cause no obstacles for passing cars. This is the starting point for our reasoning.
2. **Observation:**
   Suppose an autonomous car drives through a Zebra crossing, causing an accident, and an accident report claims that the failure is due to an evasive manoeuvre around a faraway lamp post. This observation contradicts our initial assumption.
3. **Response to Contradiction:** In vertical thinking, contradictions like this are seen as errors. To resolve the issue, we revise either:
   - **The Initial Assumption:** We might amend our assumption to say: "Street lamp posts are usually fixed on the ground but can be displaced under extreme conditions like storms or earthquakes."
   - **Subsequent Theorem:** Alternatively, we could add a new theorem to account for the observed situation: "External forces such as storms can cause fixed objects to become displaced."
   - At no point could we imagine a scenario in which a fixed lamppost causes an autonomous vehicle to perform an evasive manoeuvre around it when it is not on the way. It is just a counter-intuitive thought.
4. **Iterative Process:** The thought process continues iteratively, systematically refining the assumptions and theorems until the contradiction is resolved and the model aligns with all observations. Each step is based on building confidence that the refined assumption or theorem is correct.
5. **Outcome:**
   In this example, the vertical thinking process would now result in a modified, refined understanding of the system: "Street lamp posts are fixed on the ground but can become dislodged under certain conditions." This becomes the new basis for future reasoning.

Now, horizontal or lateral thinking is an orthogonal way of thinking to a vertical thinking approach. In lateral reasoning, the fundamental premise is that:

> *The nature of the Solution domain is a uniformly distributed domain where all events are acceptable, including contradictory ones. This means every possible situation involves some unknown training class. All varieties are probable and reasonably acceptable, and there is no such thing as a contradiction. 1=0, and not(1=0) are both analogically correct instantons to some pattern yet to be known.*

In lateral thinking, prediction is not a matter of evaluating differences or residuals between an established baseline (rule A) and an observed outcome (theorem B). This is the basis of most AI theories, including re-enforcement learning (which also, fundamentally, bases its learning on minimising errors or penalties). Instead, it involves continuously generating, adding new classes, and exploring multiple analogous patterns ***without*** anchoring to a single "correct" reference point. Rather than starting with a known rule and checking how far off the observed scenario is (how big the residual is), lateral reasoning treats every scenario as a unique instance potentially related to other new scenarios through analogy or association. So, in lateral thinking, there is no convergence to a single precise result., but more like diverging to multiple precise (based on some sets of axioms or rules) results. We believe that lateral thinking is a more effective thinking approach to solving epistemic uncertainty and reducing Residual Epistemic Ignorance of the Predictive Problem Solver.

Nature of Lateral Prediction: Lateral prediction is **analogy-based and generative**, rather than difference-based. It does not rely on a fixed reference from which to calculate deviations. Instead, it assumes that any given pattern or outcome could relate to others unexpectedly. Each new observation is not judged against a single rule; instead, it is connected, compared, and reinterpreted in light of a broad range of potential analogies. Every scenario is a new canvas that could invoke a previously unseen pattern, no matter how counterintuitive or contradictory it might seem.

E.4.1 Where is lateral predictive thinking more effective?

Where is lateral predictive thinking more efficient, valuable, or critical than vertical Predictive thinking?

Generating Comprehensive AI Training Datasets

Lateral predictive thinking is invaluable in creating robust AI training datasets because it allows architects to grasp the complexities of diverse scenarios intuitively. While vertical thinking focuses on refining specific errors within the dataset, it often misses rare or unconventional scenarios (**Black Swan events)**.

- **Example**: Consider training a self-driving car's AI system. Vertical thought processes might address obvious edge cases within the dataset, such as poor lighting or extreme weather. However, lateral thinking introduces counterintuitive scenarios, like a situation where the moon, reflected in a car's windshield, is misinterpreted as a traffic signal. Vertical thinking may overlook such a possibility because it falls outside of linear assumptions, whereas lateral thinking anticipates unconventional analogies and anomalies.

Discovery of Black Swan Scenarios

Black Swan scenarios—rare, high-impact events—are often counterintuitive and defy the expectations set by traditional models. Vertical Predictive thinking struggles here because it operates within defined boundaries and assumptions, while lateral thinking thrives in ambiguity.

- **Example**: Imagine a drone tasked with monitoring wildlife in a dense forest. A vertical Predictive approach would focus on refining detection algorithms based on common animal behaviours. In contrast, lateral thinking might uncover an unlikely event, such as a group of animals mimicking a predator's movement to deter threats. This discovery requires a creative leap that vertical methods may not naturally pursue.

Solving Compounded Epistemic Uncertainty Problems

In systems where uncertainties are interdependent and do not follow normal distributions, lateral thinking becomes critical. Vertical thinking is often constrained by its reliance on predefined probabilities and logical consistency, limiting its ability to address complexities that emerge from nonlinear interactions.

- **Example**: A Predictive Problem Solver is tasked with modelling a climate prediction model attempting to forecast weather patterns in the Arctic that might rely on vertical thinking to model known factors like temperature and wind interactions. However, lateral thinking introduces the possibility of unexpected cascading effects, such as minor shifts in ice reflectivity amplifying heat absorption in previously unmodeled ways. This generative approach broadens the architect's predictive scope, capturing possibilities that vertical thinking may miss.

E.4.2 AIC Balanced Predictive Thought Process

Regarding AIC, Appreciation is a method of appreciating a pool of infinite possibilities. Control is a method to increase the probability of implementing finite actions based on an appreciated possibility. A bias towards control tends to have a less influential thought process in terms of discovering Black Swan scenarios. A bias towards appreciation renders less influential thought processes in analysing the vertical turn of events given one possible assumption occurs. We need a holistic thought process that helps balance between appreciation and control to be more effective. The architect's predictive thought process is challenged to think in appreciation and control in order to ensure influence occurs.

Balancing lateral and vertical thinking is critical to designing robust predictive systems approaches that address various scenarios, including Black Swan events for dynamically evolving complexities. The Appreciation-Influence-Control (AIC) framework is uniquely suited to achieving this balance by integrating the exploratory and generative aspects of lateral thinking with the focused, error-resolution-based precision of vertical thinking. Below, we present an argument demonstrating how AIC achieves this balance.

Integration of Divergent and Convergent Thinking:

Lateral thinking emphasises the exploration of possibilities, embracing contradictions, and considering counterintuitive scenarios. Vertical thinking, by contrast, narrows focus, builds on established assumptions, and resolves contradictions to maintain logical consistency. AIC combines these two modes by:

- **Influence (I):** Functioning as the desired emergent value or property contextualising appreciation and control. Influence captures the interplay between lateral exploration and vertical refinement, ensuring that desired output values are both adaptive (due to appreciation) and practical (due to control action). For example, alerting a human security guard for a potential threat in a train track zone.
- **Control (C):** Imposing structure and refining the desired influence into direct actions, akin to vertical thinking. For instance, controlling the drone orientation concerning train track zone.
- **Appreciation (A):** Encouraging the architect to explore infinite possibilities broadly and generate new categories of thought, akin to lateral thinking. For example, in an autonomous system like a drone, appreciating unexpected environmental factors (e.g., unusual weather conditions or unforeseen obstacles) expands the system and the architect's understanding of the operational complexity.

Handling Contradictions:

Vertical thinking treats contradictions as errors to be resolved, so naturally, when the architect thinks in a vertical manner, subconsciously, their mind will be blinded to consider what might be counterintuitive or contradictory. Meanwhile, lateral thinking views contradictions as clues for expanding understanding. AIC addresses contradictions in the following ways:

- **Influence** emerges from balancing the A and C, it is appointed in the middle between soft appreciation and hard control. It is an indirect control, which entails a tangible action that doesn't directly cause a change because it is the control action that does this. Influence allows the architect to understand the extracted value of control and appreciation and how they feed into the whole through desired influence. With this, the architect can test the validity of A and C. For example, drone perception influences drone track and trace functionality. A potential contradiction at the influence level can be discovered.

- Through **Control**, the architect may harden the vertical rigour and only consider what control actions are required to support influence action. Thus, contradictions are resolved by refining influence and by adding rules to align observed scenarios with system goals. This might involve updating the drone's object detection algorithm to handle edge cases more differently.
- Through **Appreciation**, the architect may soften the vertical rigour. Contradictions are welcomed as opportunities to explore alternative scenarios. In this thought process, the architect drops the vertical thinking hat and opens their mind to welcome the unexpected. For example, if a security drone misidentifies an object on a train track as a threat, appreciation would prompt the architect to consider various analogies or new classifications for the type of objects that may lead the drone to fail, including unlikely ones like a "picture of a clown".

Mitigating Bias:

Vertical thinking's reliance on predefined rules and assumptions can accumulate bias, narrowing the system's perspective and reducing its ability to address Black Swan scenarios. Lateral thinking minimises bias by treating all possibilities equally valid but risks decision paralysis due to a lack of convergence. AIC mitigates these issues by:

- Allowing **Influence** to dynamically adjust the balance between appreciation and control, enabling the architect to understand why appreciation and control are needed. Thus, it adds a layer of questioning regarding the relevance of A and C.
- Applying **Control** to focus the architect's attention on the most relevant actions, ensuring that the architect remains goal-oriented and objective.
- Using **Appreciation** to explore diverse perspectives and uncover hidden biases in the architect's assumptions about the control action and desired influence.

Resilience to undesirable surprises:

Real-world complexity fields are inherently dynamic, requiring systems and their architects to be resilient to unforeseen changes. Vertical thinking struggles with such changes due to its reliance on fixed assumptions, while lateral thinking risks losing focus in the face of overwhelming possibilities. AIC enhances resilience by:

- Encouraging **Appreciation** of the full range of potential future scenarios, including counterintuitive and low-probability events.
- Leveraging **Control** to maintain direct relevance to the present reality, stability and coherence in the face of current uncertainty.

- Using **Influence** to ensure that the system remains extracting value from A and C, balancing exploration and exploitation as conditions evolve.

Example: Autonomous Drone in a Train Track Zone

An autonomous drone tasked with securing a train track zone provides a practical illustration of how AIC balances lateral and vertical thinking:

- **Appreciation (lateral thinking):** Appreciating what is inside and outside the problem domain. The architect considers a wide range of possible objects on the track, including non-obvious threats (e.g., maintenance equipment, debris, or holographic illusions). This lateral exploration ensures that no potential scenario is dismissed prematurely.
- **Control (sequential/vertical thinking):** Control what is inside the problem domain. The architect applies predefined requirements and rules to categorise objects and take appropriate actions, such as stopping the train or sending alerts. Vertical reasoning ensures that the system remains operationally effective.
- **Influence (Lateral and Vertical thinking):** Consider the indirect impact on other complexes within the problem domain. The architect can meaningfully adapt drone behaviour to a desired influence by integrating appreciation and control. For example, if a drone encounters an object that defies existing classifications, it uses appreciation to generate new hypotheses while using control to prioritise immediate safety actions.

E.5 General Lateral Predictive Thinking Process

The primary motivation for this process stems from the increasing number of bizarre cases where ML systems (in this case, perception) fail to operate reliably in relatively everyday situations. One instance is Tesla cars mistaking the moon's appearance for a red traffic light, halting unexpectedly in the middle of the road [2]. One of the most critical factors associated with ML trustworthiness in safety-critical applications is how the architect can trust its performance during Black Swan scenarios. To predict such scenarios, we need something more than computation thinking; we also need lateral thinking. This section presents a structured framework for applying lateral thinking to predictive systems engineering. The reader can refer to the following supporting section: 4.2.1. The process is directly used in section 6.2.2.

Unlike traditional vertical thinking, lateral thinking emphasises exploring broader possibilities, embraces irrelevance and unrelated interactions, accepts some form of ambiguous reasoning, and uncovers counterintuitive or unforeseen scenarios. The process outlined here aims to equip system engineers with a systematic approach to generating novel insights, operational scenarios and expanded knowledge bases. It can be articulated using the SECoT (Systems Engineering

Chain of Thought) format or a more generalised process framework, offering flexibility while retaining analytical rigour. In this section, we presented the thought process in a generic format for variation.

E.5.1 Step 1: Broaden base perspective

Assume a broad perspective without anchoring to existing logical rules of interactions.

The first step involves broadening any initial perspective by deliberately avoiding reliance on fixed common-sense assumptions. This step challenges traditional views by considering even seemingly irrelevant or stationary elements as potential contributors to emergent interactions. You can use general systems hypothesis or philosophical intuitions to guide the broadening of the base perspective to make a prediction. For example, suppose you want to consider the operational environment for autonomous airliners using machine learning (ML) perception at cruise altitude. It is clear that there is not much going on at 40,000 feet in terms of visual complexity (in comparison to ground level); nonetheless, you wish to discover Black Swan events. Then, start by adopting a broader perspective, considering broader possibilities, and intuition (assume the operational domain as a complex of uniformly distributed events where any possible event is a highly probable concern). For example, while all objects may differ, some may appear visually similar to a predictive observer.

Based on such intuition, the architect might ask themselves: What objects could deceive the perception of the airliner and cause it to resemble other objects? The architect may ponder, what if a cloud, under certain lighting conditions, resembles the tip of a mountain? Bogdanov's Ingression Principle inspires the earlier rule: *There are always intermediate Complexes linking multiple Complexes, allowing for understanding through a continuum of similarities* [3].

- **Thought Step 1.1**: Begin by defining a context without preconceptions. For example, avoid assuming that "a stationary bird standing on a stationary lamp post cannot directly influence a car". Instead, acknowledge that such an object might exert indirect or unforeseen impacts through environmental, contextual, or systemic relationships.
- **Thought Step 1.2**: Analyse both direct (control) interactions, such as physical impact, and indirect (appreciation or influence) interactions involving environmental elements or other actors in the system. For instance, a lamp post might indirectly influence a car by altering the behaviour of pedestrians or casting shadows that affect the car's sensors.

This step establishes a foundation for exploring the latent complexity of interactions within a system.

E.5.2 Step 2: Generate analogies

Generate analogies across different contexts

Building on the broader perspective established in Step 1, this step focuses on exploring alternative patterns of influence by drawing parallels across different domains and considering counterintuitive scenarios.

- **Thought Step 2.1**: Develop analogies from both similar and unrelated domains. For instance, compare the influence of a stationary lamp post on an autonomous car to how static objects in ecosystems (e.g., a tree in a forest) influence the behaviour of dynamic entities (e.g., animals).
- **Thought Step 2.2**: Explore surprising or humorous scenarios to provoke unconventional thinking. For example, use thought-provoking memes or counterfactuals like, "It would be funny or hilarious if [some unexpected interaction] occurred." This approach encourages the exploration of interactions that defy common expectations and reveal hidden dynamics within the system.

This step broadens the scope of analysis, fostering creative and innovative approaches to understanding system behaviour.

E.5.3 Step 3: Formulate predictions

Formulate hypotheses and generate new classes of operational scenarios (predictions)

This step formalises the observations and analogies from previous steps into actionable hypotheses and operational scenarios, enabling the system to adapt to new patterns of complexity.

- **Thought Step 3.1**: Treat each observation or analogy as an opportunity to hypothesise new operational scenarios. For example, hypothesise how a lamp post might indirectly influence a car's behaviour under varying conditions, such as poor lighting or sensor malfunction.
- By focusing on these hypotheses, the architect can identify emerging classes of interactions that might not have been previously considered, paving the way for more coverage of the infinite possible input acceptable problem through a comprehensive understanding of its operational domain.

This step transforms abstract insights into concrete scenarios that guide system design and testing.

E.5.4 Iterate your thought process

Iteratively Build a Broader Knowledge Base

Throughout the process, the focus is on expanding the system's understanding by incorporating new observations, interactions, and insights into its knowledge base. This dynamic process ensures that the system remains adaptive and can address evolving complexities.

- Instead of refining or narrowing assumptions, this step encourages continuous reinterpretation and reassociation of observations. For example, link newly observed patterns to prior analogies or hypotheses, even if they initially appear contradictory.
- Integrating diverse perspectives will allow the system to accommodate a growing variety of possibilities, thus enhancing its ability to generalise and adapt.

This iterative process fosters the development of a robust and expansive knowledge base, enabling the system to address increasingly complex and unforeseen scenarios. We can describe the process using GSN to describe how the predictive thought can be modelled:

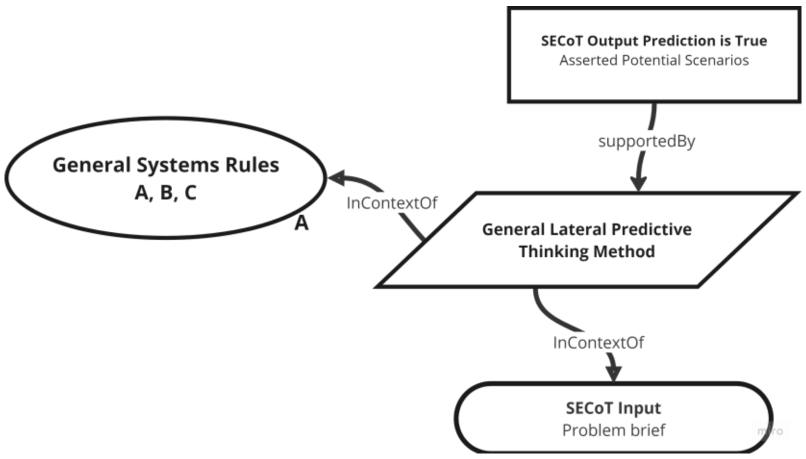


Figure **Error! No text of specified style in document.**.4 Abstract Lateral Predictive Thinking model using GSN language

Figure 5.10 describes the lateral thinking process using the GSN format. The thinking steps are modelled as an inference strategy that asserts the predicted scenario to be true. At the same time, the input problem is considered as a context where the thinking attempts to resolve into a set of predictions. The predictive thinking bases its assumptive (axiomatic) perspective on the nature of the problem and potential solution complicatedness based on the context of General Systems Rules:

General System Rule A: The nature of problem domain distribution of events is heavy-tailed, which means Black Swan events cannot be ignored. (as observed now).

General System Rule B: The nature of the solution domain is a uniformly distributed complexity field whereby all events are plausible and cannot be ignored (as I will assume it to be).

General System Rule C: All events are probable and reasonably acceptable, and there is no such thing as a contradiction.

The **General Lateral Predictive Thinking Process** provides a structured framework for exploring the latent potential behaviours of complicated complexity fields. By shifting perspectives, generating analogies, formulating hypotheses, and iteratively expanding the knowledge base, system engineers can uncover novel patterns, identify new operational scenarios, and predict efficiently to ever-changing complexity. This approach complements traditional vertical thinking by embracing ambiguity and exploring the unpredictable, ensuring robust and resilient complexity field designs in dynamic and uncertain environments.

E.6 Architect's Predictive Thinking Process (ArcPT)

> ***Predictive Thinking*** *is the process that reduces the complicatedness of the complexity of a complex to update some initial priors, by anticipating unforeseen scenarios before they actually happen, and demonstrates reduction of uncertainty, ALARP justifiably.*

The output of ArcPT is a cognitive component of the Architect's Predictive Mental Model (ArcPM):

> *A* ***Predictive Mental Model*** *a set of general rules and types about how a general complex behaves, which justifies the prediction of unforeseen behaviours before they actually occur used to update initial priors.*

It also included a structured representation, governed by the abstract rule set and categories of aspects, that enables an architect to forecast types of information in a complexity field. For example, the AIC approach to complicatedness resolution, STAMP, SHARCS, and FRAM guidewords and categories of actions, failures, and events.

Predictive Mental Models are responsible for the architect's engineering judgment and decisions in the face of aleatoric uncertainty. While aleatoric uncertainty cannot be reduced (as more things are involved), epistemic uncertainty can be reduced through an enhanced Predictive Thinking Process. In the context of designing a mental model, epistemic uncertainty would mean:

The residual doubt about the choice of a mental model design:

- *Will be able to meaningfully classify all possible types of information in any complexity, expected or unexpected.*
- *Will help to make predictions about the complexity accurately.*
- *Will be consistent and verifiable when instantiated.*
- *Will or will never require adjustments over time and experience in dealing with any complexity.*
- *Residual doubt should be reducible over time and variety of experiences with complexity.*

Predictive Systems Thinking is the topic that provides the tools and techniques to enhance an architect's predictive thinking by minimising sources of epistemic ignorance. Epistemic ignorance is an inherent deficiency associated with the architect's limited ability to account for all possible sources of randomness in an open operational environment complexity field when perceived by the architect. Therefore;

The architect's epistemic perception is inherently deficient

Based on the above principle, we understand that the purpose of the systems approach and systematic design processes is to reduce the architect's perception deficiency. Since epistemic uncertainty is reducible by acquiring more knowledge, the architect needs a process of acquiring knowledge and making epistemic inferences.

Epistemic inference is an inference from a set of beliefs to new or similar beliefs about an observation that either updates or validates the original set [10]. Shannon defines information as a message from some source to a receiver, and that information can be characterised as the removal of uncertainty for the receiver [11]. However, the receiver who lacks (thus experiencing an epistemic need) the knowledge carried by the information needs to make sense of the message first. Suppose the source of information is an observation of some complexity, and the message processor is the observing architect, who has some epistemic uncertainty about the observed complexity. Based on their existing belief system, the architect's epistemic inference minimises their uncertainty[7]. Through epistemic uncertainty reduction processes, the architect makes a series of inferences, which can be thought of as predictions to be added to the architect's knowledge store (or, dare we say, memory).

If we revise the above description, we realise that making a prediction truly starts from some level of ignorance (lack of knowledge) realised due to some shift in perspective that may be induced by a new experience or evoked by some relative change. This means a sense of "epistemic need" may be necessary to initiate architect predictive thinking process. The more various the information that is required to be processed by the architect, the bigger the lack of knowledge, the stronger the urge for the architect to minimise such lack of knowledge, which leads to the emergence of "epistemic need" in the architect's predictive thinking process.

We define "architect prediction" as the architect's epistemic inference driven by epistemic processing of information received about some complexity, underpinned by some set of beliefs or axioms. We see this epistemic inference processing of information as the Architect's Predictive Thinking Processes (ArcPT), which underpins the architect's predictive capability, a general problem-solving function of the architect.

Paradoxically, perhaps, to store knowledge, which serves as a remedy for the problem of insufficient knowledge, the information conveyed about the situation's complexity must evoke a heightened sense of ignorance within the architect. The output of such a predicament process is

[7] The core concepts are rooted in epistemology and the formal belief revision theory. We contextualise them in predictive architecting context.

either a re-confirmation (re-validation) of what is already known or a eureka (a new knowledge realised). Among the nine categories of systems engineering problems, in Table E.1, Stochastic Confusion presents the least available information to the architect, resulting in the highest degree of epistemic need. In other words;

> *The more the architect realises their ignorance, the more knowledge can be sought after and thus potentially gained. Consequently, the stochasticity of the problem will eventually become less confusing for the architect, and they will become more certain (and therefore confident) in their design decisions, despite an exponential increase in uncertainty.*

Given the evident influence relationship between the stochastic nature of a problem and the level of epistemic uncertainty that the architect possesses concerning that problem, it follows that to address a Stochastic Confusion issue effectively[8] It would be more efficient to approach the problem of uncertainty, which compounds very fast, not as a standalone issue - an "observer-reasoner independent"- but rather as an "observer-reasoner dependent problem," where the architect takes a predictive observing problem solver role.

Therefore, a solution approach for the Stochastic Confusion problem would focus on carefully and rigorously challenging the architect's situation of knowledge from multiple perspectives and making them question what they learned at different stages of the solution approach. This way, the systems approach would evoke the architect's predictive capability to look at the problem domain from varying perspectives of world views. In doing so, the architect starts to appreciate yet-to-be-thought-of unknowns (Black Swans) that may lurk under the hood of the heavy tail of uncertainty. Consequently, the systems approach only needs to focus on reducing the architect's epistemic confusion (ignorance or need) through more comprehensive predictive thinking processes. Hence, we operated in this PhD on the perspective that the CUP is "an architect's predictive capability problem".

To make a prediction[9], considering the Bayesian perspective, the architect needs to have the following activities present:

1. **Prior Belief System and Method of Inference:** The architect must have a prior set of beliefs in two tiers of abstractions:

[8] Such as the development of a safe and autonomous system comprised of non-deterministic components within an open, non-deterministic environment.
[9] Our elaboration is fundamentally driven by modern interpretations of predictive processing in cognitive science, which posit that the brain continuously generates and updates a model of its environment to minimise prediction error[?]. In these models, encountering new information or surprising events induces a sense of uncertainty that then drives mental-model updating.

- **Abstract general beliefs:** They apply to all possible complexities. For example, all things are systems and thus have parts.
- **Concrete beliefs:** these instantiations of applying the general core beliefs in the context of a specific complexity field. For example, a drone is a thing, thus a system. Therefore, it has parts. Which provokes a follow-up question: What are those parts?

Evidence
When we applied HAZOP, STAMP, and FRAM, our predictive thinking process was constrained by the fundamental definition of what a clear system is, which they indirectly or directly instil. FRAM assumes that systems are sub-organisations of control functions within larger organisations of systems, and some control interactions resonate with others, leading to an emergent amplification of impact. HAZOP assumes failures occur in systems, such as a traceable analogue control system, in which only a subset of deviations, subject to the observer's expertise, can occur. STAMP assumes complexity as a first-order cybernetics problem. A problem of hierarchical control feedback loop. Those prior beliefs about the general rules of complexity directly impact our overall predictive performance.
Another piece of evidence we learned from our experiments is that attempting to generate a reliable dataset strategy can enable ML models to perform as expected in cases of relatively surprising data shifts (away from the distribution of training datasets) in the operational domain (Black Swan). The nature of the distribution of the training and validation dataset (whether typical operations or a mix of random images or Black Swan scenarios) really dictated the reliability of the model. The nature of dataset distribution can be understood as our engineered priors; an ML model ought to believe about the real-world operational complexity. If we supplied the model with pictures of a drone purely outside the context of the operational domain, it performed poorly over Black Swan datasets (in-context Black Swan). See section H.10. This demonstrates that the nature of the dataset context dictates the reliability of the ML model's predictive capability in the face of problem complexity.

2. **Realisation of Ignorance:** A sense of lack of knowledge (epistemic need) induced due to the uncertainty of disbelief in the validity of some prior belief system in experienced complexity[10].
   - Such a situation may be triggered due to an unforeseeable change in the circumstances around the problem. For example, the drone incorrectly detected

[10] This is why we understood Bogdanov perspective about systems being an "organisational experience" between the observer and the observation.

a Target Object of Interest (TOI). If the incorrect detection is a thing, (Which provokes a follow-up question), then what are its parts? However, detection of TOIs is intangible; (provoking another question) how can it have tangible parts?.

- or, evoked by another architect or system approach that makes the architect realise some potential inconsistency in their existing belief systems about the problem. For example, a systems approach step prompts the architect to consider some empty space, encouraging the architect to question the original set of belief systems; a space is also a thing. How does it have parts?

Evidence
When we applied HAZOP, STAMP, and FRAM, our predictive thinking process was prompted by constantly asking ourselves questions about how we would interpret the deviation guideword in an interaction context. This delta in epistemic lack helped us predict different failure scenarios between one guideline and another. The realisation of lack of knowledge (ignorance) in applying abstract general rules (guidewords) was a triggering factor in predicting a failure scenario.

3. **Making a prediction (epistemic inference):** The first two activities constituted a chain of conditions we experienced while implementing the processes. The realisation of ignorance made us wonder how a general rule like a guide word may manifest in the real world. This predictive thought process led us to recognise a lack of knowledge, or ignorance. Subsequently, a series of rationally interrelated questions emerged that provoked some inferences. At the end of this thought process, we arrived at a conclusion, which is the prediction.

Evidence
When we applied HAZOP, STAMP, and FRAM we were unaware of the failure events we discovered later after applying the processes. We predicted more ways the problem complexity may scale up and increase in complicatedness. Our epistemic ignorance was reduced every time we predicted new interaction.

If we combine the generic predictive thought process with concepts of aleatoric uncertainty, we realise the following assumptions:

- The solution of aleatoric uncertainty (randomness in the observed complexity) is determinism of the processes among complexes, such that emergence is predictable and verifiable (independent of the observer). For a CUP, this is a very hard ask.
- The solution of epistemic uncertainty (unpredictability of the observed complexity) is the ability to be certain in predictions made about involved components and the observed

processes' output based on an epistemic mental model of the observed complexity, which is underpinned by general belief systems about how complexity in general behaves.

Therefore, Prediction or inference, in general, minimises architect epistemic uncertainty and prompts the reason to continue the cycle of predictive thinking.

Hence, we believe that Architect's Predictive Thinking Processes (ArcPT) provide a structured approach to refining a Predictive Problem Solver's capability, enabling them to navigate and minimise Epistemic Uncertainty effectively. By combining systematic and systemic thinking approaches, ArcPT aims to enhance the architect's capacity to predict complicated behaviours and uncover unforeseeable interactions. We realise that for a predictive thought to iterate or be powered to carry on generating predictions, there needs to be the following key activities to challenge the architect's situation of knowledge continuously:

- Set the general belief system: General, Concrete or both.
    - We opted to use general systems theories as the basic belief system for the architect to base their prediction-making process on.
    - We also used AIC (Appreciation, Influence and Control) to be part of the belief system behind the reasoning for prediction.
- Evoked a sense of ignorance (lack of knowledge) through questions.
    - We opted for a predictive question method based on the set belief. we named such methods as Predictive Questions.
- Remedy the sense of lack of knowledge using a guiding prompt.
    - We opted to use a prompt like general guidance to guide the architect's predictive thinking process towards making an informed and informing prediction.
- Strategically shift the basis of a belief system that was initially set, to stretch their imagination and consider more Black Swans.
    - We opted for a perspective-shift process to deliver the core value of lateral thinking.

This section introduces the Predictive Systems Engineering Chain-of-Thought (SECoT), a process inspired by chain-of-thought prompting techniques used in large language models. SECoT represents a comprehensive, step-by-step framework that guides architects in systematically applying predictive thinking grounded in General Systems Theory (GST) and systems thinking frameworks. Its ultimate purpose is to improve the architect's ability to balance lateral and vertical thinking, ensuring a rigorous, transparent, and creative problem-solving process.

E.6.1 Predictive Systems Engineering Chain-of-Thought (SECoT)

Earlier, we examined a systematic process for predictive thinking. We realised that each step builds on the step before it in a chain reaction triggered by a sense of ignorance. This chain reaction of continuous inquiry into the nature of observation (executed by the architect's predictive mind) is analogous to the idea of "Chain-of-thought" (CoT) associated with large language models. This time, we refer to it as predictive architect chain-of-thought. The nature of a predictive CoT really can be determined by two main aspects:

1. Primary indicator: The nature of the belief system (and how correct they are).
2. Secondary indicator: The nature of the reasoning method used to correctly validate the set of beliefs in a context of observed complexity to infer concrete beliefs.

and?

Suppose the set of beliefs is general systems theories, and the reasoning philosophy is based on a systems approach (like AIC, for example). In that case, the chain of thought is classified as a predictive system thinking process. When those Predictive CoTs (or let's call them PCoTs) are used in the context of Systems engineering, they become Systems Engineering Chain-of-Thought (SECoT). If PCoTs are based on any other belief system and use any reasoning, the PCoTs become predictive thinking of the general topic that belief and reasoning systems are part of. For example, suppose the belief system is related to the mechanical aspect of physical systems. In that case, the PCoT can be classed as a predictive mechanical engineering chain-of-thought. Suppose the set of beliefs on some ethical general rules, and the output is an engineering judgment that leads to a design consideration for an autonomous system. The PCoT can be classed as a predictive ethical engineering chain of thought. For example, the latter may help ensure ethical consideration in designing autonomous systems.

So What?

Therefore, SECoT is a process that involves a systematic and rationally related chain of predictive systems thinking steps (systems thoughts) based on systems thinking frameworks or GSTs. Implementing a SECoT aims to transition ArcPT objectively from a state of confusion to one of clarity. Effectively shifting the view of complexity from being a harder-to-predict-and-ascertain Stochastic Confusion to the situation of an easier-to-predict-and-ascertain system to manifest the PrimeP. In doing so, SECoT aims to indirectly enhance ArcPT's problem-solving capability, enabling it to predict or make assertions that can be explained and qualified.

By doing so, we can probe the design process deeply in case mistakes occur during the deployment of intelligent behaviours by understanding exactly how the architect predicts certain behaviours or overlooks considerations of some behaviours. SECoT provides evidence to support claims of justification for design decisions for safety cases. For example, part of the SACE safety case framework is an artefact called Sn5.1[AS design justification], which means an argument that justifies key design decisions taken at any stage in the design process (see Appendix L, section L.4.5.2). This approach further provides evidence of the claim of alignment to human values and allows for the implementation of human values into developing system requirements.

The cognitive process that SECoT performs on Architect Predictive Thinking is what we refer to as:

T-Thinking-based Prediction-Provocative Prompting

In this process, SECoT encourages the architect to think about general aspects of what they are trying to evaluate or create. The "Lateral" part of T-thinking is experienced by considering a general rule learned from various universal systems when making an inference. This is accomplished through a thought-provoking question and a guiding prompt on how to answer the question. The expected output of SECoT is an architect prediction or assertion in the following format: "The architect asserts that …". For example, based on implementing AIC Complicated systems principles, we can derive the following SECoT:

Table **Error! No text of specified style in document.**.6 Example lateral rule and predictive question

General Systems Axiom (Lateral Rule)	**SECoT thinking step**
AIC systems principle: Primary Purpose No observed set of coexisting complexes can be described as a "system" without the servitude of at least one shared Primary Purpose (PrimeP) to manifest. A set of coexisting elements that do not satisfy this principle is considered a "Stochastic Confusion"	**Predictive Question:** Given the observed phenomenon, given principle 1, What is the complex of coexisting complexes of concern and their PrimeP? **Guiding Prompt:** Identify a list of complexes of concern that share a common PrimeP and define each complex's primary purpose. If you find complexes that can be grouped to serve a clear PrimeP, identify the encompassing whole, not each element. For example, a tracking zone, a train, a train driver, and train passengers can all be grouped under one supra-system, which can be termed a "train network".

Table 4.1 presents an example of how a General Systems Axiom (Lateral Rule) is systematically translated into a thinking step to guide the architect in making structured predictions about a given observation. It applies a key AIC systems principle, the Requisite Primary Purpose. It situations that a set of coexisting elements can only be considered a system if they serve a shared

Primary Purpose (PrimeP). If no such shared PrimeP exists, the observed elements are merely a "Stochastic Confusion", not a system.

The Predictive Question is derived from the central concept of the general principle and re-wording it to evoke the architect's perception to notice some behaviours and define their primary purposes. The predictive question intends to ensure that the architect does not describe elements in isolation but rather identifies whether these elements function together toward a shared objective (PrimeP) or are a disjointed, Stochastic Confusion. The purpose of this question is to prompt a structured analysis where the architect must evaluate:

- What elements are interacting in the observed problem domain?
- Are they serving a unifying purpose that justifies treating them as a system?
- If no common PrimeP is found, then what missing factors might be causing the confusion?

The Guiding Prompt further refines the prediction process by directing the architect toward a practical lateral rule application process.

E.6.2 Predictive SECoT Structure

We define a particular structure of fully elaborated SECoT, which comprises the following elements captured in a table:

- **SECoT Title:** Define a particular descriptive title.
- **SECoT Input:** Define the input information needed to implement the CoT. A description of some phenomenon (an interaction or a scenario).
- **SECoT Process:** Describe the steps of the thinking process that prompt the systems thinker to consider the intuition described in general systems basis, in the format of a question and a guiding prompt. The SECoT process can be organised and structured into the following types of activities:
    - **A Predictive Thinking Pipeline:** A conceptually interrelated series of predictive thinking steps. A Predictive Thinking Pipeline title represents the primary prediction goal, expected to be asserted by the interrelated constituent thinking steps.
    - **A Thinking Step:** is a thought-provocative prompt that influences the architect's predictive thinking processes to make an informed prediction based on some general system perspective. It intends to focus (influence) the architect's attention to synthesising a potentially complicated interaction from some observed context by applying general rules about systems. It contains the following thinking activities:
        - **Step's input:** an input of information that requires processing.

        - **Step's process,** which includes the following activities:
            - **General Systems Rules:** A set of general rules about any system dynamics. In other words, it is a general rule that makes any complexity behave like a desirable system. The output of the thinking process must preserve those rules. They help define what acceptable output looks like and doesn't. In this section, various types of axioms can be used to guide the thought process of the designer:
                - Hard systems axioms, such as those related to physical systems dynamics.
                - Soft systems axioms, such as those related to soft concepts like ethical or human value consideration.
            - **Predictive Question:** Prediction-provoking Question
            - **Guiding Prompt:** Prediction-Guiding Prompt. Include a notion of completion, in other words, how a step is qualified as complete.
            - **Step completion criteria:** describe how the step is judged to be completed.
        - **Step's Output Inference:** The output inferred realisation or assertion.
    - **A Predictive Thinking Pipeline prediction:** Set of all inferences.
- **SECoT Output Prediction:** The series of predictions made across the steps from all thinking pipelines that define new anticipated situations or scenarios.

The above construct is captured in the format of a table. For example:

Table **Error! No text of specified style in document.**.7 General SECoT example

SECoT Title	AIC perspective shift SECoT.
SECoT Input	Security drones inhibit human intrusion into train track zone.
SECoT Process	**Predictive Thinking Pipeline 1:** Perspective Shift by Altering Interaction's AIC Type. **Thinking step 1:** identify AIC interaction of concern. **Thinking step 2:** Alternate the goal's AIC type. **Thinking step 3:** describe a scenario that captures the change in complicatedness. **Predictive Thinking Pipeline 2:** Perspective Shift by reversing effect direction. **Predictive Thinking Pipeline 3:** Perspective Shift by alternating effect type. **Predictive Thinking Pipeline 4:** Perspective Shift by alternating multiple perspective shift types.

SECoT Output Prediction	Four different unforeseeable scenarios for every given system phenomenon.

Table 4.2 presents an example of a SECoT (Systems Engineering Chain of Thought) titled "AIC Perspective Shift SECoT", which illustrates its purpose, input, process, and output. The primary purpose of this SECoT is to assist in predicting unexpected and unforeseeable emergent scenarios resulting from anticipated systems interactions. The input focuses on security drones inhibiting human intrusion into a train track zone. The process is structured into a predictive thinking pipeline with multiple steps designed to shift perspectives by altering AIC interaction types, reversing effect directions, or alternating effect types. Each step prompts the architect to think about interactions in new ways, ultimately leading to SECoT's output prediction: four different unforeseeable scenarios for every system phenomenon.

An example of a thinking step structure is that we will use Thinking Step 2: Alternate the goal's AIC type.

Table **Error! No text of specified style in document.**.8 Predictive Thinking step example (a variation of the general SECoT)

SECoT Title	AIC perspective shift SECoT
Input	Some AIC interactions, for example, Eagle Drone denies local people's intrusion into the train track zone. Eagle Drone has a control goal over local people's intrusion.
General Systems Rules	**Axiom A:** In any AIC relationship, a new complexity may emerge from the interactions of coexisting elements; when an element changes its AIC goal, the direction of impact is reversed.
Thinking Process	**Predictive Question:** What would happen if the interaction flow and effect type remained the same, but the goal's AIC type and action type were altered in the future? **Guiding Prompt:** Review the AIC dynamics of the observed systems and alter the nature of the goal and action. Then, define an appropriate action to bear the new AIC types and describe a scenario in the shifted context. **Step completion criteria:** the step is completed when a scenario is graphically modelled and the scenario is written.

Output prediction	**Architect assertion:** This is the output prediction of the thinking step. The assertion is to be written in the following format: “The architect asserts that”. For example, the architect asserts that there may be a potential unforeseeable emergent situation where the drone appears to attract unwanted intrusion by local people into the train track zone. In this case, the original perspective of the Eagle Drone being in control of local people's access has shifted to be appreciated.

Table 4.3 presents an example of a thinking step structure focused on "Thinking Step 2: Alternate the goal’s AIC type”. The input involves a specific AIC interaction, such as an Eagle Drone designed to deny local people's intrusion into a train track zone, with a control goal over this access. The thinking process employs a general systems rule (Axiom A) that predicts new complexities emerging when an element's AIC goal changes, reversing the direction of impact. The predictive question asks what would occur if the interaction flow and effect type remain constant but the goal's AIC type and action are altered in the future.

A guiding prompt encourages the architect to review the AIC dynamics, alter the goal's nature, and define a new scenario under the shifted context. The step is completed when the scenario is graphically modelled and elaborated in detail. The step's output takes the form of an architect’s assertion, such as predicting a potential unforeseeable situation where the drone unintentionally attracts local people to the train track zone, shifting the drone's original role from control to appreciation. This structured thinking approach highlights how altering goals can reveal emergent scenarios.

E.6.3 Engineered datasets’ epistemic risks

An interesting observation made by Joseph Nelson, Roboflow cofounder and CEO, about the approach of most ML perception developers, saying [15]:

> “We seek to build computer vision models that generalise to as many real-world situations as we can, even when we cannot anticipate them. It's a bit of a catch-22: build deep learning models that predict a world that you may not have anticipated. But that's what is at the crux of preventing overfitting. Unintentionally, we build models that best predict our training data, but not the real world (or the test set).”

Nelson describes the general attitude of ML software engineering in practice when it comes to training safe and reliable models. He emphasises key factors or aspects that impact trust in any ML model training and dataset gathering processes:

- The real challenge is that ML models are expected to generalise well in scenarios that the developer cannot anticipate (not just hard for them to predict).
- Predicting a real-world complexity that the developer may not have previously predicted requires them to consider real-world scenarios, not just clever techniques to circumvent the burden of thinking harder and deeper to predict more possible scenarios.
- The prevention of overfitting due to hidden biases in the dataset, due to a skewed or biased dataset gathering and development process (which is the task of the developer).
- ML developers can be prone to unknowingly or unintentionally having a mindset to focus their model training and validation around predicting the training data, but not the real-world.

These challenges all stem from a common root:

> **ML Developers have not fully engaged in a balanced, objective and comprehensive thinking process that meaningfully reduces their epistemic uncertainty about real-world conditions.**

Which in turn leads to gaps in the dataset's epistemic coverage, what we can call:

Engineered dataset epistemic risk.

Because developers often assume that their training data implicitly captures the diversity of real-world scenarios, they fail to ask, "What don't we know about the environments our model will face?" As a result, they collect and curate data that reflects their expectations and biases rather than the true breadth of possible situations, which hides risks of overfitting to familiar patterns. When hidden biases remain unmitigated during dataset curation, models appear to perform well on held-out splits but falter in unanticipated conditions.

In essence, skipping the deeper cognitive work of identifying and filling knowledge gaps about the world means datasets lack critical examples (epistemic coverage is incomplete), and the model cannot be trusted sufficiently to generalise in unforeseen scenarios. This engineered dataset epistemic risk manifests whenever developers over-trust or over-rely on clever tricks to patch errors, instead of systematically expanding their understanding of what the model might encounter outside the lab, through augmentation alone. In a safety-critical application, a safety quality regulator should care more about "has the developer deeply thought about the real-world operational domain" rather than "has the developer gathered sufficiently-sized datasets and applied clever automated augmentation techniques".

> **In other words, a regulator should care more about “has the developer’s predictive thinking process enabled them to predict those hard-to-predict scenarios?”**

The ML architect engages in epistemic inference through the Architect’s Predictive Thinking Process (ArcPT), a reasoning mechanism that updates prior beliefs about any complexity in response to new information observed or discovered. However, when the information about an observed complexity (such as a dynamic environment for a vision-based AI system) is sparse, ambiguous, or inherently stochastic, the architect’s epistemic need (a drive to reduce uncertainty) becomes more noticeable. As described in ArcPT, this need initiates a predictive reasoning chain, prompting the architect to ask: "What do I not know about this complexity?" The deeper this uncertainty, the greater the potential for unrecognised unknowns (Black Swan events) to emerge.

This cognitive gap reflects directly on how datasets are constructed; when architects and dataset curators fail to recognise or fully appreciate certain operational complexities (whether due to unfamiliarity with Black Swans, novel conditions, or failure scenarios), the resulting data will be **epistemically ignorant**[11] of those scenarios, which risks truing the completeness of datasets.

E.6.3.1 Epistemic Ignorance in Image-Based Datasets

This epistemic ignorance manifests tangibly in image-based datasets; in our case, we are concerned about training AI perception systems (e.g., for object detection, navigation, or collision avoidance). A dataset that does not give the architect sufficient confidence that it captures a variety of operational states, environments, lighting conditions, object variations, or anomalous behaviours introduces a silent gap: the AI system becomes blind to what it has never seen, and unexpected behaviours may blindside the architect.

> If software has “bugs” that can lead to harm, a formal methods approach reduces it. If a physical system has “flaws” or “faults” that can lead to harm, formal systems engineering approaches reduce them. Similarly, a dataset of images has “Epistemic Ignorance” which means the absence of valuable information that prevents harm[12], can be reduced by reducing the architect's epistemic ignorance using the AIC systems approach, traceability of images to requirements and the systematic prediction and mitigation of Black Swan pictorial scenarios.

[11] Epistemic Ignorance is a well-established concept in information theory (see section 2.1.15)
[12] As in harm-preventing information is absent.

Pictorial-driven hazards emerge from the *absence or biased presence* of critical visual information in the training data. Pictorial hazards are "soft hazards"[13] that pose indirect risks by denying the AI model the opportunity to learn from, detect, or anticipate certain operational states. For example, if the training dataset for a collision avoidance drone omits images of hot-air balloons at low altitudes, the resulting model will not be prepared to react appropriately in that scenario (despite its critical safety implications).

E.6.3.2 Linking Architect Confusion to Dataset Design Failures

The origin of these pictorial hazards lies in the architect's peripheral perception blind spots; a failure to foresee the whole landscape of possible environmental complexities (seeing the wood for the trees). This is compounded when:

- The dataset is designed from a static or biased worldview of problem domain complexity about typical operation scenarios,
- The architect relies on historic familiarity and incomplete concrete belief systems, and
- The epistemic need[14] to explore "unknown unknowns" is not actively evoked.

This misalignment leads to an unassured system: one that may perform well under common conditions but fails unpredictably in rare or complex situations (those most critical to safe autonomous operation).

E.6.3.3 Towards Mitigation: CuneiForm-based Epistemic Intelligence Development

In general, Intelligence is the reduction of ignorance or confusion

To mitigate epistemic pictorial ignorance, the CuneiForm development process enhances the architect's knowledge by systematically addressing their epistemic needs, challenging their assumptions, and broadening their worldview by seeing the operational environment from an intelligent machine's perspective. This process involves iterative predictive questioning, perspective shifts, and revision of the belief system to uncover invisible complexities in the operational domain. The aim is to ensure that the dataset design is not only comprehensive but also epistemically sound, reflecting not only what is known but also what may be unknown.

[13] See section E.3 for definition of soft-hazards.
[14] A need to acquire new knowledge

By systematically bridging the gap between architect epistemic uncertainty and dataset epistemic coverage, we can better anticipate pictorial hazards. In doing so, we shift from reactive data collection to proactive dataset engineering, ensuring that datasets become instruments for knowledge discovery.

E.7 The CuneiForm characteristics

To close the semantic gap between descriptive and pictorial representations of TOIs and behaviours, we define two main activities:

> Meaningful mapping of real-world TOIs and their environment's behaviours to pictorial models.

Meaningful mapping ensures that relevant elements of the textual description correlate with specific visual features or patterns in a CuneiForm, establishing a shared understanding between the CuneiForm creator and the observer/quality checker/validator. The mapping is based on mitigating potential perception failures due to its TOIs' pictorial behaviours. We define this mapping by identifying and mitigating the failure of a training dataset to enhance the perception model's robustness in recognising required TOIs in all possible scenarios.

E.7.1 Valuable Minimum Variety condition

Table 4.6 lists potential absence-of-harm-prevention-information (equivalent to failure) modes in developing training datasets, along with the corresponding mitigations in the CuneiForm design process. Note that we specified the "valuable minimum variety" condition as a mitigation for "insufficient variety", as the term "sufficient" cannot be measured. In contrast, the term "valuable" can be qualified by demonstrating that it meets expectations, in the context of "the added images contribute towards minimising epistemic uncertainty through meeting requirements", which makes them "valuable". In other words, a valuable image is:

- An image whose pictorial behaviour can be quickly and unambiguously traced to a descriptive parent requirement.
- An image that contributes towards minimising the epistemic uncertainty of a given class or feature related to the recognisability of TOI.

A valuable image adds value by:

- Assisting with the trustability of a claim made in the safety case.
- Contributing towards enhancing the computer vision model knowledge system.

A low-value image is an image that may add value to enhancing perception model capability, but:

- It is hard to understand how valuable it is in terms of meeting expectations.
- Also, it is hard to understand how it contributes towards minimising epistemic uncertainty since there is no specific class it belongs to.

For instance, consider an image of a dog swimming in a river, which is part of a dataset used for classifying dogs in a task. Suppose this image was not included in an explicit requirement. While such an image might be intended to enhance robustness, it ultimately lacks value in explaining how such an image is needed. What if adding such an image may reduce the robustness of dog recognition in a non-river-based scenario?

One way to think about the design assurance inefficiency introduced by low-value images is that they may cause more cognitive overload for human validators. Imagine a dataset containing thousands of low-value images used for safety-critical functionality; how can a regulator trust that safety requirements have been faithfully captured? Or even how can the regulator or design assurance authority trust that Black Swan scenarios have been extensively predicted and designed?

We also used "minimum" since variety alone is an immeasurable quality. However, with descriptors like "valuable minimum", we are tacitly saying this is a minimum variety, and there can be more, but this is as far as we can get for now. It can demonstrate how our process satisfies this quality and allows us to dictate a limited set of generic characteristic types that can be applied to any TOI and can be qualified against meeting expectations:

Perception training dataset development may fail to minimise AI epistemic uncertainty due to the following sources of pictorial aleatoric uncertainties:

Table **Error! No text of specified style in document.**.9 Pictorial Hazards Impact on Perception Reliability. The mitigation plans provide the characteristics that must be depicted in a CuneiForm.

Pictorial Hazard	**Mitigation:** CuneiForm characterisation must help **the** perception training architect to consider ...
Insufficient variety of TOI's visual aesthetic complexity, leading to the absence of information that could prevent harm.	A valuable minimum variety of TOIs' aesthetic features across perceived images.
Insufficient variety of TOI's visual complexity during motion and in	A valuable minimum variety of TOI's motion situations and optical effects across perceived images.

dynamic states, leading to the absence of information that could prevent harm.	
Insufficient variety of environmental objects, leading to the absence of information that could prevent harm.	A valuable minimum variety of pictorial environment objects across.
Insufficient variety of environmental objects' motion, leading to the absence of information that could prevent harm.	A valuable minimum variety of Pictorial Environment Objects, Motion, and Dynamic optical situations across perceived images.
Insufficient variety of TOIs' change of positions in the perception view, leading to the absence of information that could prevent harm.	A valuable minimum variety of TOIs' positioning across perceived images.
Insufficient variety of TOIs' change of 3D orientations in the perception view, leading to the absence of information that could prevent harm.	A valuable minimum variety of spatial orientations of TOIs in 3D space.
Insufficient variety of changes in TOIs' distance from the perception view, leading to the absence of information that could prevent harm.	A valuable minimum variety of Changes in the distance cause relative changes in TOI's image size concerning total frame size.

One important aspect we realised while developing the process was that the greater the variety of classes introduced for defining a characteristic for a single CuneiForm, the less valuable it becomes in terms of cognitive ease when validating or designing CuneiForms. Suppose a CuneiForm image contains too many variations of characteristic classes in a single characterisation. In that case, matching pictures to the parent abstract Cuneiform becomes inefficient during a quality check. This can result in mistakes and defeats the purpose of using Cuneiform. Cognitive overload during the design and validation process, which ML engineers typically perform, is a risk as it may lead to human errors. Therefore, in the process, we introduce some heuristic numbers to constrain the complexity of the designed variety in every CuneiForm.

E.7.2 Machine Training Classes topics

Considering the analysis in Figure 4.6, we define generic dimensions (analogous to topics about the world which we like to train the machine to recognise) that can be captured in a pictorial sense in the CuneiForm image. The main conceptual drive to represent each dimension is the assumption that aleatoric uncertainty is a source of architect and machine confusion, meaning randomness in the operational domain is visually captured in the dataset, thus leading to randomness in the pictorial dataset. This randomness, known as aleatoric uncertainty, can confuse the architect, resulting in a loss of potential knowledge that should be incorporated into the dataset used to train the machine on a specific task. Thus, the randomness in the pictorial dataset may influence the epistemic uncertainty of the architect and the machine. Each dimension aims to reduce the epistemic uncertainty of both the architect and the machine.

E.7.3 Visual Dimension 1: TOI's aesthetical features complexity topic

One source of pictorial hazards comes from the infinite acceptable input space of TOI's aesthetic variety of features.

The delineation of TOI pictorial features, particularly their aesthetic complexity, is a nuanced process that directly affects the reliability and safety of machine learning models. For example, a drone's shape, colours, and coating design. Since datasets impact designer comprehension, a challenging aspect is defining how many aesthetic design features should be included in a single CuneiForm. As the dataset increases in size, the number of required CuneiForms also increases in size. This increase affects the cognitive overload of dataset handlers (during the dataset design and validation processes). We recommend describing no more than 3 aesthetic features in a single CuneiForm. We recommend, based on our experience designing cuneiforms in our case studies, and inspired by the magic no.7 rule of thumb [17].

E.7.4 Visual Dimension 2: TOI's perceived motion situations and optical effects topic

One source of pictorial hazards comes from the infinite acceptable input space of TOI's motion situations and optical effects.

The TOIs' visual motion situations and optical effects are critical for accurately depicting how an object's movement can impact image capture quality and clarity. To describe this characteristic, the following minimum variety of aspects can be defined:

- **In-Motion states:** TOIs' motion patterns constitute significant variables impacting their detectability by perception systems. For simplicity, we define the following minimum variety of situations to be considered:
    - **Still State:** The TOI is stationary relative to the environment and the camera frame. This is the simplest scenario for object detection. For example, if the adversarial drone is hovering in place, it may appear static relative to the environment.
    - **Linear Motion:** The TOI moves in a straight line across the set of captured images. For example, the adversarial drone flies straight at a speed that may cause motion blur.
    - **Rotational Motion:** The TOI is spinning or rotating. This is captured in a series of images where the TOI exhibit a rotational motion. For example, the adversarial drone performs barrel rolls or spins.
- **Optical effects:** The dynamic situation of TOIs can affect the recognisability of the pictorial situation experienced by a perception system, such as motion blur. Motion blur is an Optical effect caused by the relative motion between the camera and the object if the object's speed is high compared to the camera's exposure time. For example, high-speed adversarial drones cause blur.

E.7.5 Visual Dimension 3: TOI's 3D spatial orientation topic

One source of pictorial hazards comes from the infinite acceptable input space of TOI's spatial 3D orientation.

The spatial orientation of TOIs in a three-dimensional space is an essential training class for computer vision perception systems. TOIs' 3D orientation can be represented by a discrete set of view angles. As a guideline, our process recommends a generic, relevant variety of no more than nine distinct view angles to describe these orientations comprehensively. However, this number depends on the application.

Additionally, our framework remains adaptable; the architect may determine a more appropriate set of 3D views tailored to the TOI's geometry. For example, in the case of a perfectly spherical object, the symmetry negates the need for multiple orientations, reducing it to a singular view. Figure 4.5 below depicts our recommended generic variety of 3D orientation views for any TOI. However, depending on the geometry of the TOIs, the architect may adapt our recommended variety accordingly. We define the following generic views for TOIs: front, side, top, front up, front up right and left, front down, and front down right and left.

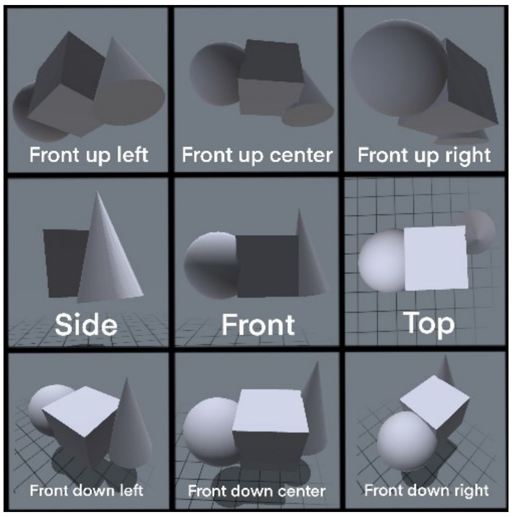


Figure **Error! No text of specified style in document.**.5 shows our recommended minimum variety of possible generic 3D orientations.

In the Eagle Robot case study, developing a CuneiForm representation for an adversarial drone encompasses defining its three-dimensional orientations. This approach entails generating a set of CuneiForm representations that encapsulate all possible perspectives or views of the drone, thereby facilitating its recognition under various orientational conditions. We choose an abstract depiction of a drone and then model its orientation roughly based on the generic definition in Figure 4.6.

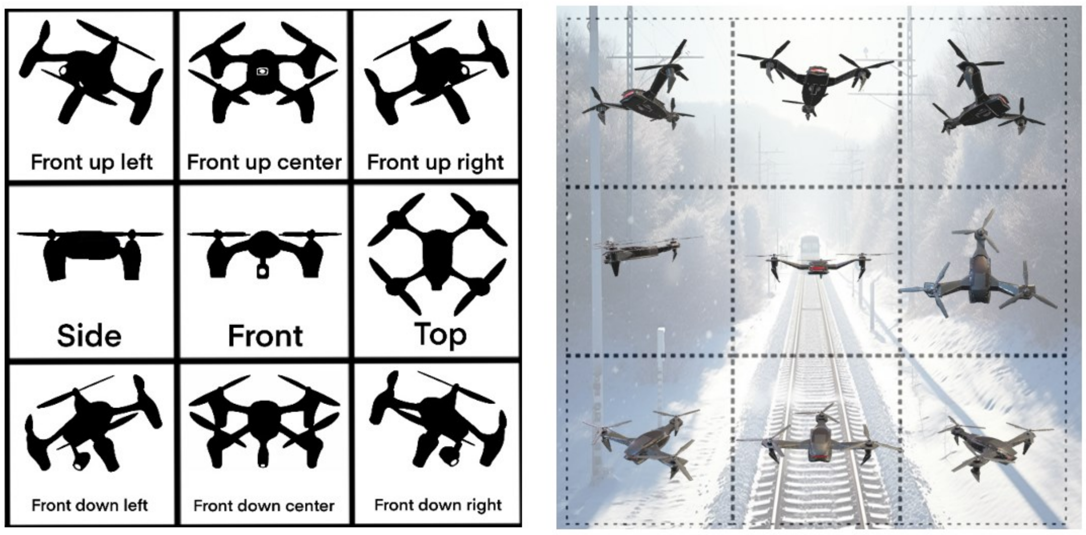


Figure **Error! No text of specified style in document.**.6 Minimum variety of 3D orientation CuneiForm representations with an example image of how an instantiation of them would look like in a dataset.

For the AVOIDDS (Vision-Based Dataset for Aircraft Detection) case study, we defined 18 generic orientations since the application involved detecting intruder aircraft to avoid mid-air collisions.

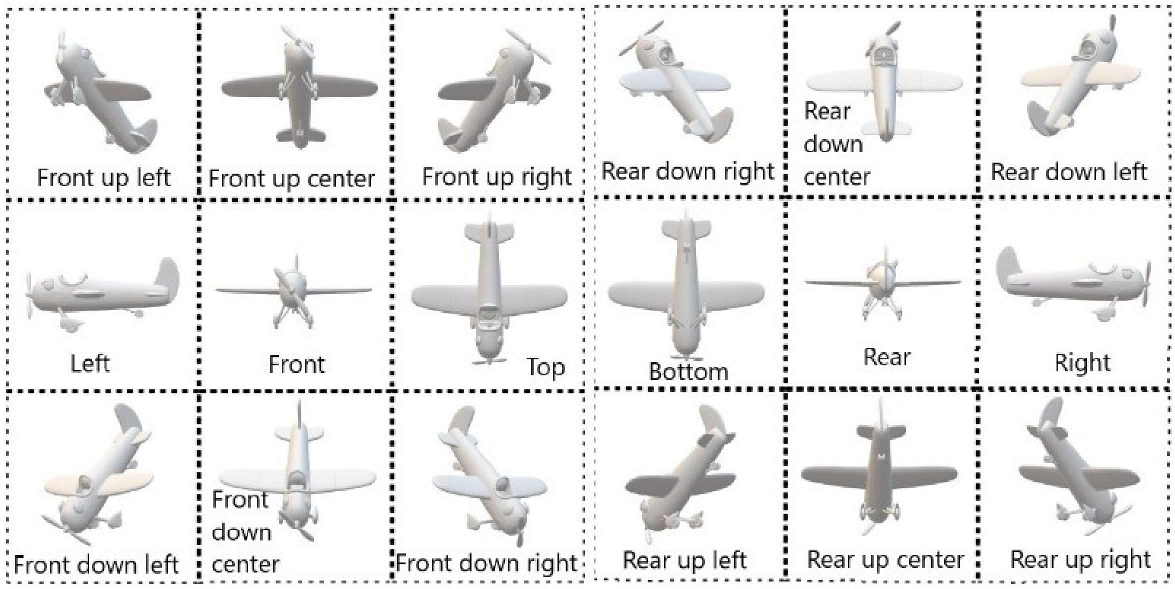


Figure **Error! No text of specified style in document.**.7 Generic 18 minimum variety orientations for TOIs in the AVOIDS case study

Figure 4.7 depicts 18 distinct spatial orientations used in the AVOIDDS case study to represent the three-dimensional views of target objects (TOIs), specifically intruder aircraft. These orientations were chosen to comprehensively capture various perspectives necessary for training the perception model to detect and avoid mid-air collisions. The views include standard angles such as front, top, and side and more nuanced perspectives like front-up-left and rear-down-right, ensuring robust coverage of potential visual scenarios. This extended variety of orientations allows the system to handle diverse real-world conditions and ensures that no critical viewpoints are omitted during training and validation.

To give another example, Figure 4.8 presents 18 distinct spatial orientations for a human Target of Interest (TOI), designed to comprehensively capture various 3D views required for training computer vision systems. These views include standard perspectives such as front, top, rear, and side, alongside more nuanced angles like up-front-left, bottom-front-center, and up-rear-right. This variety ensures the perception system can recognise and accurately interpret the TOI in diverse scenarios and orientations.

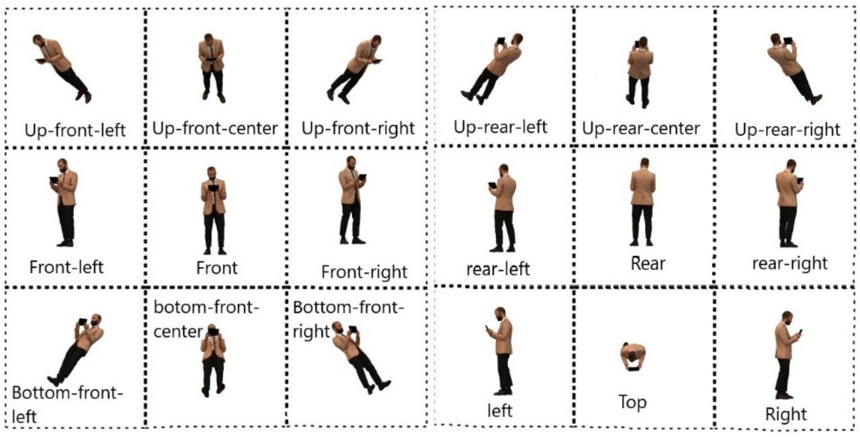


Figure **Error! No text of specified style in document.**.8 Generic 18 view orientations for a human TOI example.

E.7.6 Visual Dimension 4: TOI's pictorial positioning zones definition topic

One source of pictorial hazards comes from the infinite acceptable input space of TOI's pictorial positioning.

The positions of objects within images are an essential aspect of characterising the images as they can influence perception reliability (note that we said influence, not control). This approach, inspired by the zonalisation seen in ancient cuneiform tablets such as in Figure 2.9, applies a structured grid to categorise and standardise the spatial positioning of TOIs within an image. In our adaptation, we utilise nine distinct zones within each image to define the specific location of the TOIs. The rationale behind the choice of 9 We based on our experience designing cuneiforms in our case studies, and inspired by the magic no.7+/-2 rule of thumb [12]. By dividing the image into nine main zones, we can systematically describe and specify the location of TOIs. This zonal approach enables a more organised analysis of complicated visual scenes with multiple objects. It also helps reduce the ambiguity that might arise from overlapping or closely situated objects.

We created a CuneiForm development canvas to help with designing and characterising CuneiForm. The positioning is categorised into 21 types (9 main zone labels in yellow and 12 intermediate optional zones). Table 4.7 describes the possible categories:

Table **Error! No text of specified style in document.**.10 Generic TOI positioning zones labels

down right	**center right**	**up right**
down center	**center**	**up center**
down left	**center left**	**up left**
up left/center left	center left/center	center/down center
center left/down left	down left/down center	up center/up right
up left/up center	up center/center	Center/center right

down center/down right	up right/center right	center right/down right

Figure 4.9 illustrates the TOI positioning zones, which essentially act as a guide for perception architects. Each of the nine zones represents a potential location for a TOI, thus allowing for a detailed description of its position relative to the camera's perspective and other objects in the scene. We also included further intermediate zone labels to assist with describing Toi's position if they are found between zones. They are optional and depend on the need and application.

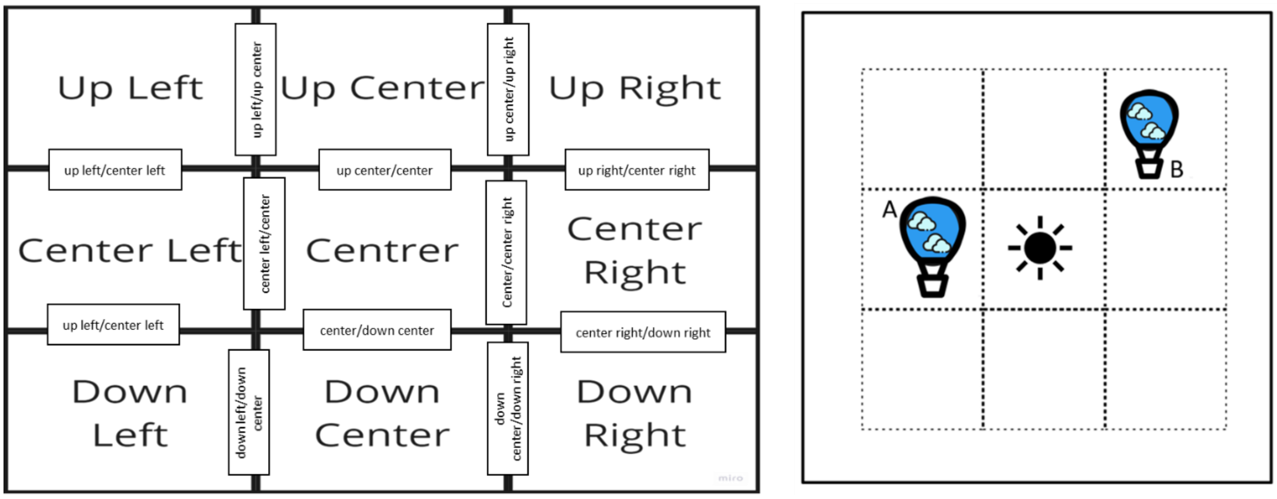


Figure **Error! No text of specified style in document.**.9 TOI generic positioning zones.

The right-hand depiction gives an example of cuneiform where TOI A occupies the centre-left, and TOI B occupies the up-right zones.

E.7.7 Visual Dimension 5: TOI's pictorial distance topic

One source of pictorial hazards comes from the infinite acceptable input space of TOI's pictorial distance.

Defining the distance of a TOI from a camera's focal point influences its recognizability in an image for a trained perception system. It is unclear what shape or size a TOI may be in the real world, nor are we clear about how far it will most likely be. In an open, complicated environment, all sizes and all distances are equally possible unless the operational environment is controlled. So, we had to consider a generic process specifying any TOI distance from any perception model. We define a process that quantifies this distance based on the intuition that the closer the TOI is to the camera, the larger its relative area compared to the total frame area. Thus, the pictorial area of the TOI within the camera's frame is compared to the image's total area. Depending on the requirement of how far the TOI should be recognised, the architect may specify a particular ratio.

For ease of communication and to add some precision to pictorial characterisation, we name a generic pictorial measurement of distance, represented in relative size, as *"Pictorial Nindan"* or just *"Nindan"* to describe a single unit ratio of distance as a function of a ratio of TOI's area to the frame total area. The term "Nindan" is an adaptation of a Sumerian unit that measures distance, and we reused the term in our approach for convenience and ease of specification. To measure the pictorial distance of a TOI, we don't care about how truly far it is; we only care about how the distance distorts the visual aesthetics of the object concerning the predictive observer's detection reliability. So, a Nindan captures the impact of physical distance on TOI's appearance.

So what? Let's have an example.

Suppose that we have a requirement to detect an adversarial drone at a long distance. If we are given an image with a TOI, and we need to calculate the pictorial distance of the TOI, we need a process to help us with estimating how many Nindans a TOI is far. Let's take the robot in the orange box (Figure 4.10):

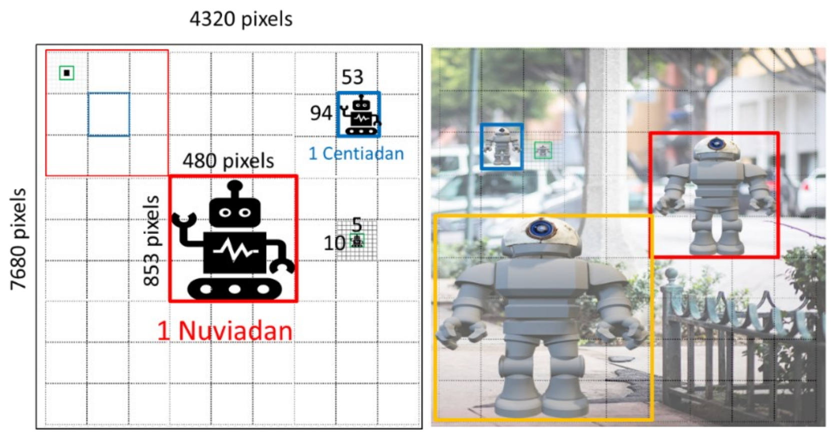


Figure **Error! No text of specified style in document.**.10 Defining pictorial distances in units of Nindans. 1 nuviadan = 9 Nindans. 1 Centiadan = 81 Nindans.

The simplest way to define standard pictorial distance measurement is to divide the image into a grid of squares. We use a 9x9 (regardless of image size), which results in 81 squares. To capture smaller TOIs we keep subdividing the squares into 9x9 grids. In the case of the orange box designated as TOI (Target of Interest), we apply this 81-square grid. One *Nindan* distance is equivalent to 81 squares.

Next, we determine how many squares the TOI occupies; in this instance, it occupies 25 squares. This indicates that the TOI is located further away than 1 Nindan.

To calculate the distance of the orange-boxed robot in Nindans, we divide the total number of squares (81) by the number occupied by the TOI (25):

81 / 25 = 3.24

Therefore, the orange-boxed robot is measured to be approximately 3.24 Nindans away. We must subdivide the image into an 81x81 grid for green-boxed TOI, which counts to 6561. Then, count the number of squares the green-boxed TOI occupies; in this case, we have 9. The pictorial distance of the green-boxed TOI is 6561/9 = 729 Nindans.

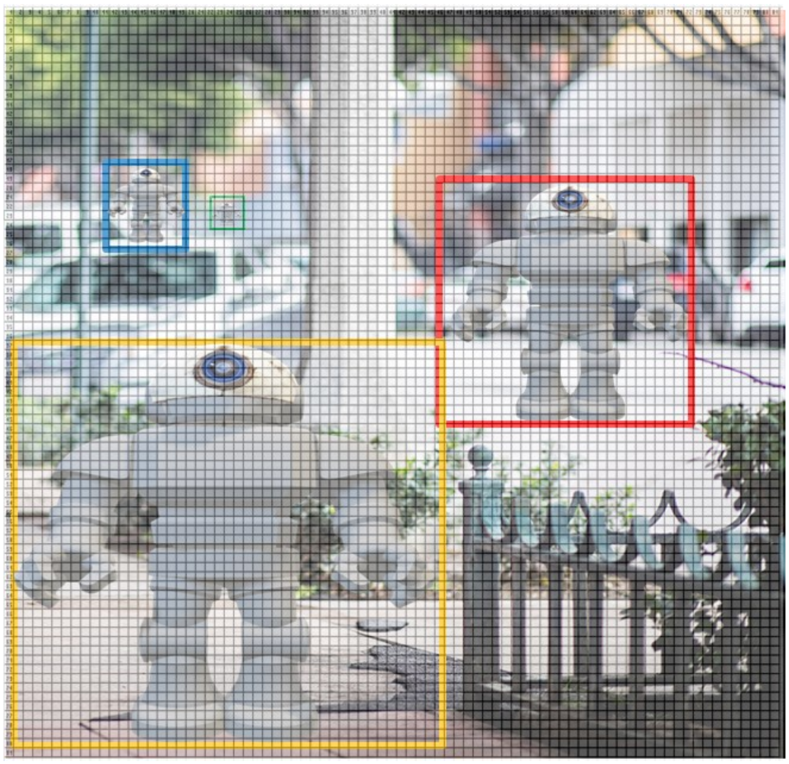


Figure **Error! No text of specified style in document.**.11 green boxed TOI appearing at a Miliadan pictorial distance = 729 Nindans pictorial distance

Let's take another example: the car that appears behind, which is partially occluded.

Figure **Error! No text of specified style in document.**.12 Partially occluded TOI (car)

Suppose the TOI is a car. In this case, if we are to measure how many nindans, we measure how many squares the bounding box occupies. In this case, we can manually determine the number of squares occupied: 16 (height) x 34 (width) = 544 scaled squares. To determine the pictorial distance in nindans: 6561/544 = 12 Nindans. It is worth noting that we assume the following concept:

A partially visible object is conceptually as unclear as if it were fully visible but that far

So, for example, let's take the same image in the Figure and determine how far the robot is conceptually. The whole robot was 3.24 nindans, not only part of its head is visible. In this case, we accept the convention of “the distance of the visible part, " meaning TOIs are conceptually as far as their visible part. We can manually count the number of squares, and in this instant: 5 (height) x 16 (width) = 80. Given 6561/80 = 82.0125 Nindans, which is nearly 25 times farther away conceptually than if it was visible entirely. See Figure 4.13.

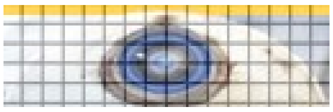


Figure **Error! No text of specified style in document.**.13 Partially visible robot TOI pictorial distance of 82.0125 Nindans

E.7.7.1 Standard pictorial distance categories topic

To simplify distance specification for a cuneiform design, we need to categorise various ranges of classes. Although the values of the pictorial distances for the classification may change from one TOI to another, also depending on the application, we will use the range specification for the aircraft mid-air collision avoidance dataset case study as an example.

Table 4.8 categorises pictorial distances (Dx) into five distinct ranges based on the recognizability of a Target of Interest (TOI) for the aircraft mid-air collision avoidance dataset. These categories span from "Extremely Unrecognisable TOI Distance" (Dx > 1600 Nindans), where the TOI is nearly imperceptible, to "Dangerously Close TOI Distance" (Dx ≤ 40 Nindans), indicating critical proximity requiring immediate action. The intermediate ranges—"Moderately Recognisable," "Recognisable," and "Clear Close" distances—highlight varying degrees of TOI clarity as perceived by a human verifier or system. This classification provides a structured framework to evaluate and design datasets, ensuring appropriate recognition capabilities for different operational scenarios.

Table **Error! No text of specified style in document.**.11 Classification of pictorial distance ranges for TOI recognition in the aircraft mid-air collision avoidance dataset.

Training class (category) (wrt human verifier)	**Pictorial Distance ratio (Dx) Range**	**Example TOIs at different distances**

Extremely Unrecognisable TOI Distance	Dx > 1600 Nindans	
Moderately Recognisable TOI Distance	729 > Dx ≤ 1600 Nindans	
Recognisable TOI Distance	300 > Dx ≤ 729 Nindans	
Clear Close TOI Distance	40> Dx ≤ 300 Nindans	
Dangerously Close TOI Distance	1≥Dx<= 40 Nindans	

E.7.7.2 Pictorial distance calculation and validation

We believe that validating AI-related artefacts requires direct human intervention rather than relying on another AI system to validate a safety-critical AI system. This is because AI Processes used for verifying AI development artefacts necessitate verification of correctness. Therefore, we developed a visual grid stencil-like process to define and manually validate the distance. Although this process can be automated for the most part, fundamentally, the dataset validator must manually check how conceptually far a TOI is in an image.

Furthermore, this process can be advanced further by relying on pixel comparison. Since it is a more deterministic process, it can validate an entire dataset and present the result as a histogram of how far TOIs are considered, thus revealing another potential layer of bias. This analysis involves processing labelled images to compute a measure of pictorial distance based on the number of pixels occupied by objects in the pictures. The goal is to determine how recognisable an object of interest (TOI) appears at different distances. The general steps for defining pictorial distance are as follows (using YOLOv8 labelling format as an example):

1. Extracting Object Annotations
   - Each image has a corresponding text file containing bounding box annotations.
   - Each bounding box specifies an object's center coordinates (x, y), width, and height in a normalised format (relative to the image size).
2. Converting Annotations to Pixel Values
   - The relative bounding box values are converted to absolute pixel dimensions using the image width and height.

   - The bounding box coordinates are transformed to represent the object's actual position in the image.
3. Computing Bounding Box Area: assuming a typical square box.
   - The total number of pixels inside each bounding box is computed (width × height in pixels)
   - This represents the portion of the image occupied by the object.
4. Calculating Pictorial Distance (Dx)
   - Pictorial distance is defined as the ratio of total image pixels to the bounding box pixels:

$$Dx = \frac{Total\ Image\ Pixels}{Bounding\ Box\ Pixels}$$

   - A higher Dx value indicates that the object occupies a smaller area in the image, making it less recognisable.
   - Conversely, a lower Dx value means the object is larger and closer, making it more identifiable.
   - If we flip the ratio, the result would be counterintuitive to the sense of far and close distance from the trainee ML model. For example, a TOI that occupies 10 pixels in a 1000-pixel image would count to 0.001. However, using the convention we propose would count to 100, which makes more sense regarding how far away from the observer.
5. Classifying Recognition Levels
   - The computed Dx value is used to categorise images into five distinct recognition classes:
     - Extremely Unrecognisable TOI Distance ($Dx > 1600$)
     - Moderately Recognisable TOI Distance ($729 < Dx \leq 1600$)
     - Recognisable TOI Distance ($300 < Dx \leq 729$)
     - Clear Close TOI Distance ($40 < Dx \leq 300$)
     - Dangerously Close TOI Distance ($Dx \leq 40$)

This process allows an automated, quantitative evaluation of object recognition based on the area occupied by the target in an image. The classification provides an objective measure of visibility, which can be used for further AI model training, dataset curation, and real-world object detection analysis. Table 4.9, Below is an example output of analysing the pictorial distance for the AVOID training dataset sample we used for our second case study (see Appendix I). Bear in mind that the categorisation and the range are customisable based on the application.

Table **Error! No text of specified style in document.**.12 Computing pictorial distance based on pixels

Image Name	Bounding Box Pixels	Total Pixels	Pictorial Distance Dx (Nindans)	TOI pictorial distance Training Class
0.jpg	1627	5621280	5621280/1627=3455	Extremely Unrecognisable TOI Distance
1.jpg	8801	5621280	5621280/8801=639	Recognisable TOI Distance
10.jpg	4969	5621280	5621280/4969=1131	Moderately Recognisable TOI Distance

E.7.8 Visual Dimension 6: TOI's environment complexity topic

Considering the analysis in Figure 4.6, we recognised that the image background's complexity influences the TOIs' recognisability. This section addresses the integration of environmental characteristics within the CuneiForm images. The focus is on how background objects and their dynamic situations affect a perception system's recognisability of the TOI.

E.7.8.1 Impact of background objects and environmental scenery on TOI recognisability

One source of pictorial hazards comes from background objects' infinite acceptable input space.

Background objects influence the perception system's ability to accurately identify and classify TOIs. Background objects can vary in complexity, ranging from simple, uniform textures to highly cluttered scenes, as perceived by an observing human architect. Such variability may obscure or confuse the perception system, leading to misidentification or non-recognition of the TOI. The challenge lies in ensuring that the TOI remains distinguishable despite the presence of varying background elements. This necessitates carefully selecting background features in CuneiForm images that represent real-world scenarios yet do not overwhelm the architect's ability to comprehend that a designed CuneiForm captures a requirement, and the instantiating images capture the valuable minimum variety needed.

E.7.8.2 Dynamic situations of background objects topic

One source of pictorial hazards comes from the infinite acceptable input space of Dynamic situations of background objects.

In addition to their static aspects, the dynamic situations of background objects can challenge a perception system. Dynamic backgrounds, such as moving foliage or other objects, can introduce additional uncertainty for ML components to handle. Therefore, when constructing CuneiForm images, it is critical to incorporate a range of dynamic situations for background objects. This will train the perception system to focus on the TOI and accurately interpret its features despite its presence.

E.7.8.3 Pictorial Horizon topic

One source of pictorial hazards comes from the infinite acceptable input space of Digitally Perceived Pictorial Horizon Attitude.

In the context of a camera system, the Pictorial Horizon refers to the apparent boundary where the sky meets the Earth or sea, as seen through the camera lens. The inclination and attitude of the horizon are influenced by the camera's position, 3D attitude, and environmental factors, including atmospheric conditions. Figure 4.16 illustrates the various horizons relevant to a camera system. There are two types of horizons [18]:

- The astronomical horizon is the theoretical line where the Earth and sky would meet if there were no obstructions, considering the Earth's curvature.
- The true horizon is the theoretical at sea level, unaffected by terrain or other local obstructions.
- The Pictorial Horizon is the pictorial line that separates the earth and sky impacted by objects.

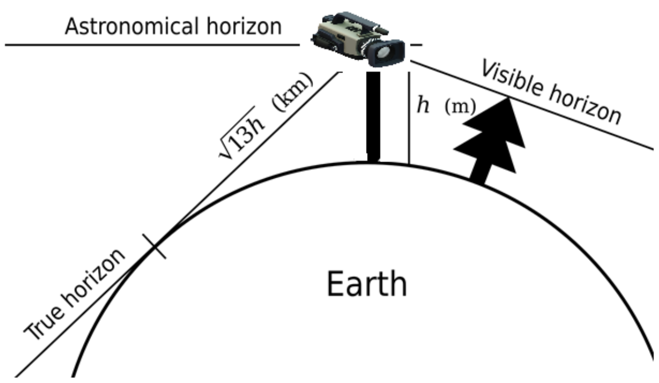


Figure **Error! No text of specified style in document.**.14 Types of horizons for cameras taken from [18]

Understanding the orientation of the horizon within an image is critical for accurate perception and decision-making. The horizon attitude in an image can be manipulated through the camera's Frame Roll and Frame pitch. To standardise this for any set of pictures in a computer vision training set, we could create a specification that dictates the Frame Roll and Frame pitch settings for different horizon attitudes. The justification for defining the horizon in a pictorial sense stem from the fact that the sky and earth have two distinct visual complexities.

E.7.8.4 Specifying Perceived Pictorial Horizon Attitude

To objectively define a process to specify the horizon from an image or a video frame, we need to establish the following:

- A universal pictorial horizon for any digital image. Let's call it an "absolute pictorial horizon reference".
- We need a measurement tool to measure the relative horizon in the image or video frame itself relative to the absolute pictorial and digital horizon reference.

We define a general horizon reference as the horizontal line that splits any picture into exact halves. The actual horizon in the image or video frame itself can then be compared against our standard horizon to determine the Perceived visible frame horizon metric using the aircraft attitude indicator instrument metric to specify the variety of frame horizon attitudes. The adapted instrument is a Pictorial Horizon Attitude Indicator (PHI).

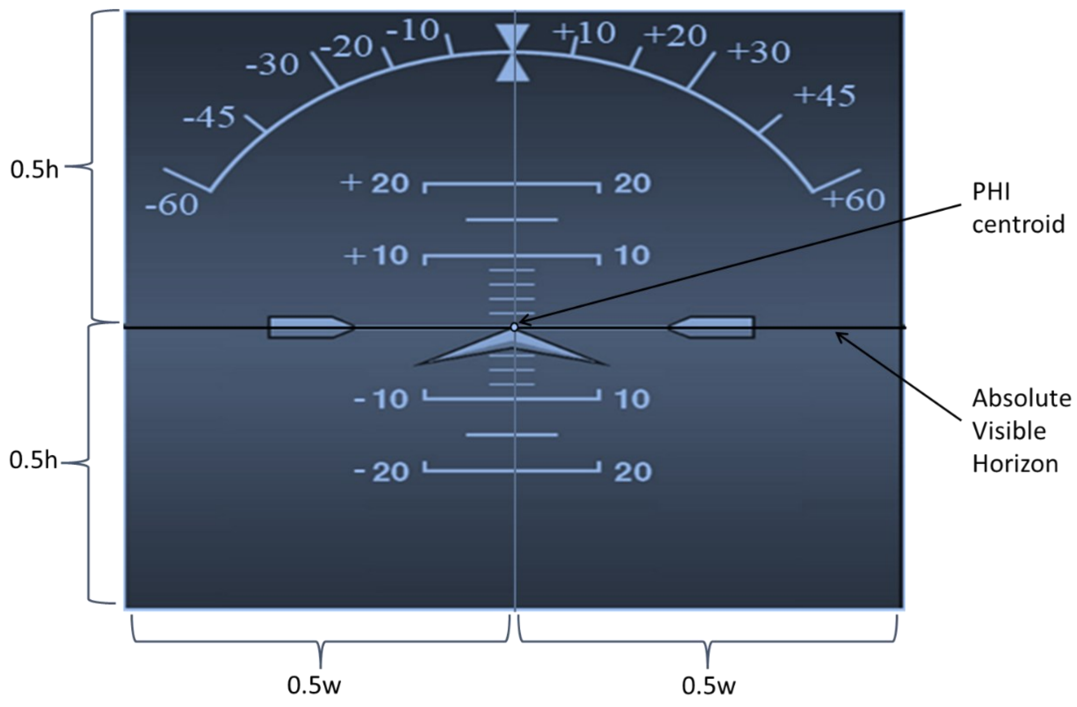


Figure **Error! No text of specified style in document.**.15 General illustrations of the Pictorial Horizon Attitude Indicator (PHI) stencil tool inspired by flight attitude indicator instrument metrics

Figure 4.15 illustrates the Pictorial Horizon Attitude Indicator (PHI), inspired by flight attitude indicator metrics, designed to manually and objectively define and measure horizon attitudes in images or video frames. The absolute Pictorial Horizon is a horizontal line splitting the image into two halves, ensuring a universal reference point for horizon measurements. The PHI centroid acts as the focal point for measuring relative horizon attitudes, enabling precise alignment and quantification of frame roll and pitch. This tool provides a structured framework for specifying and analysing the perceived Pictorial Horizon, aiding in consistently characterising horizon-related attitudes across digital images and training datasets.

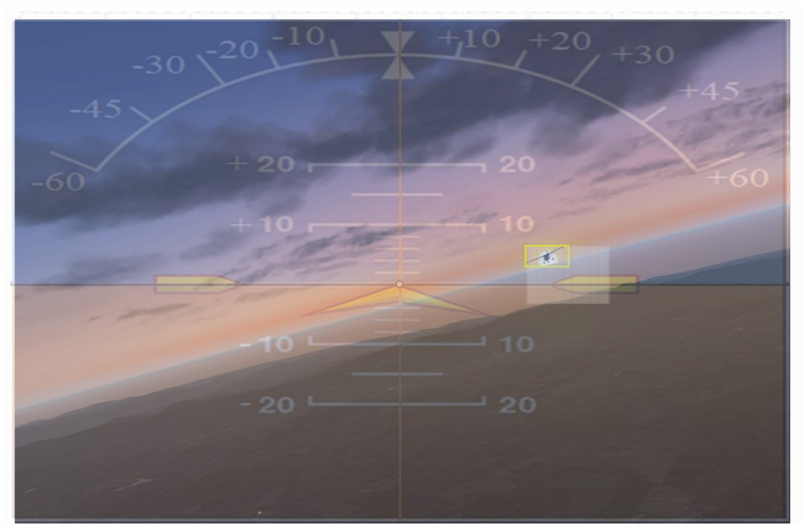


Figure **Error! No text of specified style in document.**.16 Perceived visible frame horizon attitude metric.

In Figure 4.16, We placed the matrices to match the frame size and made the metric transparent.

E.7.8.5 Measuring the Roll and Pitch of the frame horizon using PHI

We employed a manual measurement process using visual tools like a ruler to validate or define the roll and pitch angles of the horizon in images. This process involves aligning the PHI with the horizon in the image and measuring the inclination. Below are the steps for measuring both the roll and pitch of the horizon:

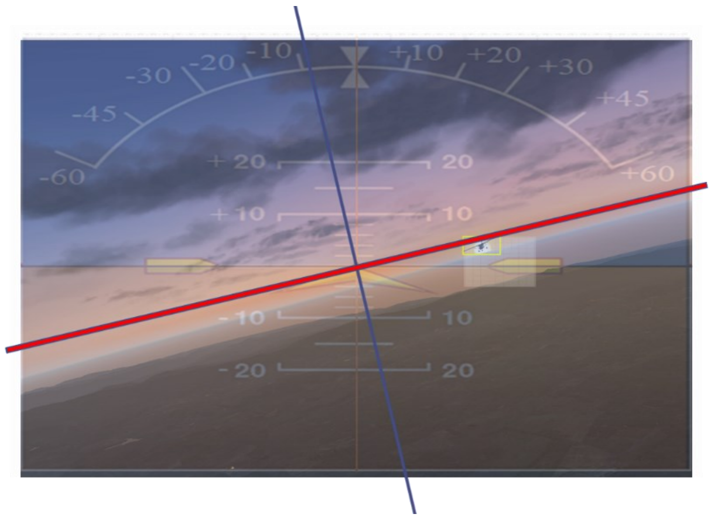


Figure **Error! No text of specified style in document.**.17 We placed the cross ruler at the centroid to measure the roll angle of the horizon attitude

Measuring the Horizon Roll Angle Process:

1. **Visual Ruler Setup**: Use a red horizontal line as a ruler to capture the horizon's inclination. A perpendicular blue line helps align the PHI centroid for accurate measurement.
2. **Placing the Ruler**: Position the centre of the ruler at the PHI centroid to capture the roll angle relative to the horizontal axis.
3. **Measuring the Roll**: Observe the inclination angle of the horizon roll as indicated by the blue line. For example:
   - In Figure 4.17, the horizon roll is at approximately -10 degrees.

Measuring the Horizon Pitch Angle

1. **PHI Rotation**: Rotate the PHI metric image to align with the ruler's roll angle for ease of pitch measurement.
2. **Ruler Placement**: Place the horizontal line of the ruler on the Pictorial Horizon.
3. **Measuring the Pitch**: Measure the vertical distance or inclination of the horizon relative to the ruler's position. For instance:
   - In Figure 4.18, the PHI is aligned, and the horizon pitch is measured as approximately -20 degrees.

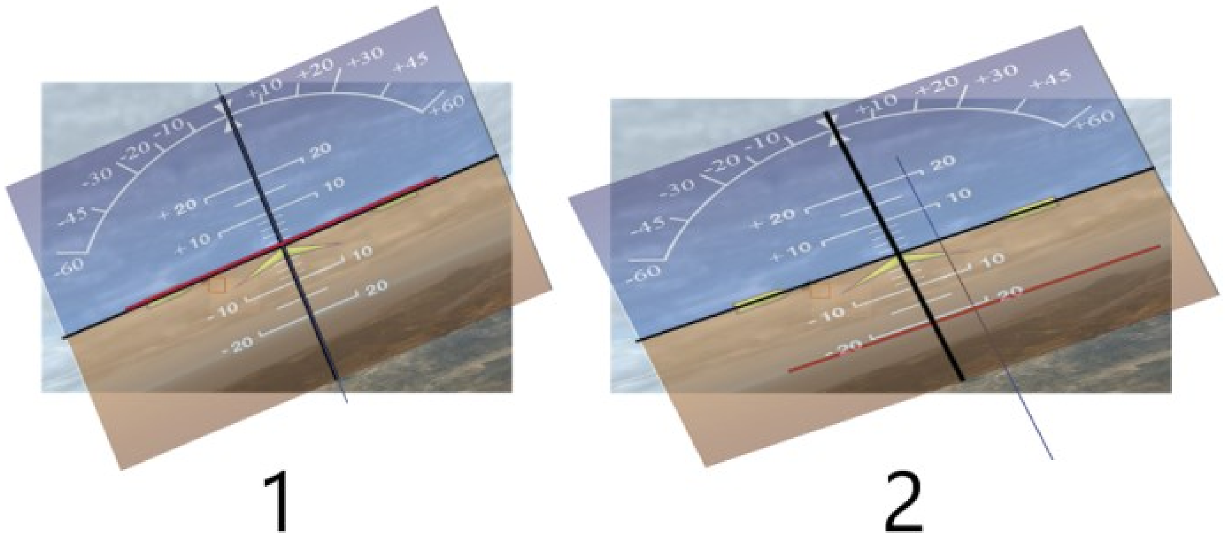


Figure **Error! No text of specified style in document.**.18 1) We rotate the PHI to align it with the ruler, then 2) We place the ruler's horizontal line on the horizon.

This manual process provides an intuitive way to validate the roll and pitch angles of the frame horizon in training datasets or image analysis. It ensures that horizon attitudes are accurately recorded for perception model calibration. The process can be adapted for automated systems or tools for greater precision in high-volume image datasets.

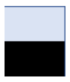
Captured image rotational movements definitions:

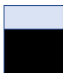
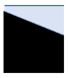
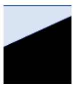
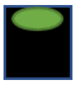
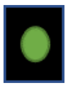
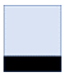
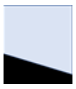
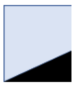
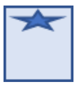
- Frame Roll (Camera Rotation around the front-to-back axis):
    - **Positive Frame Roll:** The camera tilts to the observer's right, raising the image's left side and lowering the right**.**
    - **Negative Frame Roll:** The camera tilts to the observer's left, raising the image's right side and lowering the left.
- Frame pitch (Camera Rotation around the side-to-side axis):
    - **Positive Frame pitch:** The camera tilts upward, lowering the horizon line in the image.
    - **Negative Frame pitch:** The camera tilts downward, raising the horizon line in the image.

Horizon Minimum Variety of Attitude Categories:

Table 4.10 categorises horizon attitudes based on defined ranges of roll and pitch values, capturing a variety of camera orientations in images or video frames. The categories include neutral views like "Level Horizon" (minimal roll and pitch) and more dynamic perspectives such as "Positively Tilted Elevated Horizon" and "Bird's Eye Ground View," which represent extreme tilts and elevations. These classifications provide a structured framework for standardising and analysing horizon orientations, which is essential for training and validating perception systems in diverse operational scenarios. Including categories like "Rocket Sky View" highlights the adaptability of this framework for applications requiring extreme camera angles.

Table **Error! No text of specified style in document.**.13 Horizon attitude categories are defined by roll and pitch ranges.

Horizon attitude training class (category)	**Roll Range**	**Pitch Range**	**Visual Description**
Level Horizon 	−1≤ROLL≤1	−1≤PITCH≤1	The horizon appears flat and straight across the center of the image.
Positively Tilted Level Horizon	1<ROLL≤90	−1≤PITCH≤1	The horizon tilts downward from left to right, remaining near the center vertically.
Negatively Tilted Level Horizon	−90≤ROLL<−1	−1≤PITCH≤1	Horizon tilts downward from right to left, centred vertically.

Elevated Level Horizon 	−1≤ROLL≤1	1<PITCH≤45	Horizon appears level but shifted upward in the frame, sky dominates the view.
Positively Tilted Elevated Horizon 	1<ROLL≤90	1<PITCH≤45	Horizon tilts left-to-right and is elevated in the frame.
Negatively Tilted Elevated Horizon 	−90≤ROLL<−1	1<PITCH≤45	The horizon tilts from right to left and sits high in the image.
Acute Angled Bird's Eye Ground View 	−90≤ROLL≤90	45<PITCH≤80	Ground dominates the frame, seen at a steep angle, with no horizon visible. Only looking at the ground from some angle.
Bird's Eye Ground View 	−90≤ROLL≤90	80<PITCH≤90	The image appears to be almost straight down; the horizon is not visible, only the terrain below.
Lowered Level Horizon 	−1≤ROLL≤1	−45≤PITCH<−1	Horizon appears flat but is low in the frame; more ground or cockpit visible.
Positively Tilted Lowered Horizon 	1<ROLL≤90	−45≤PITCH<−1	The horizon tilts from left to right and is lowered toward the bottom of the image.
Negatively Tilted Lowered Horizon 	−90≤ROLL<−1	−45≤PITCH<−1	The horizon tilts from right to left and is positioned low in the frame.
Acute Angled Rocket Sky View 	−90≤ROLL≤90	−80≤PITCH<−45	Sky dominates, no horizon visible and an upward pitch gives a “rocket” view climbing at an angle.

Ascending Rocket Sky View ★	−90≤ROLL≤90	−90≤PITCH<−80	Looking nearly straight up into the sky, no horizon is visible. Like climbing up in a rocket.

To put table 4.10 in context, the following is a histogram of how images captured in AVOID dataset (our case study in Chapter 7, Appendix I) performs in terms of coverage of all possible categories of horizons:

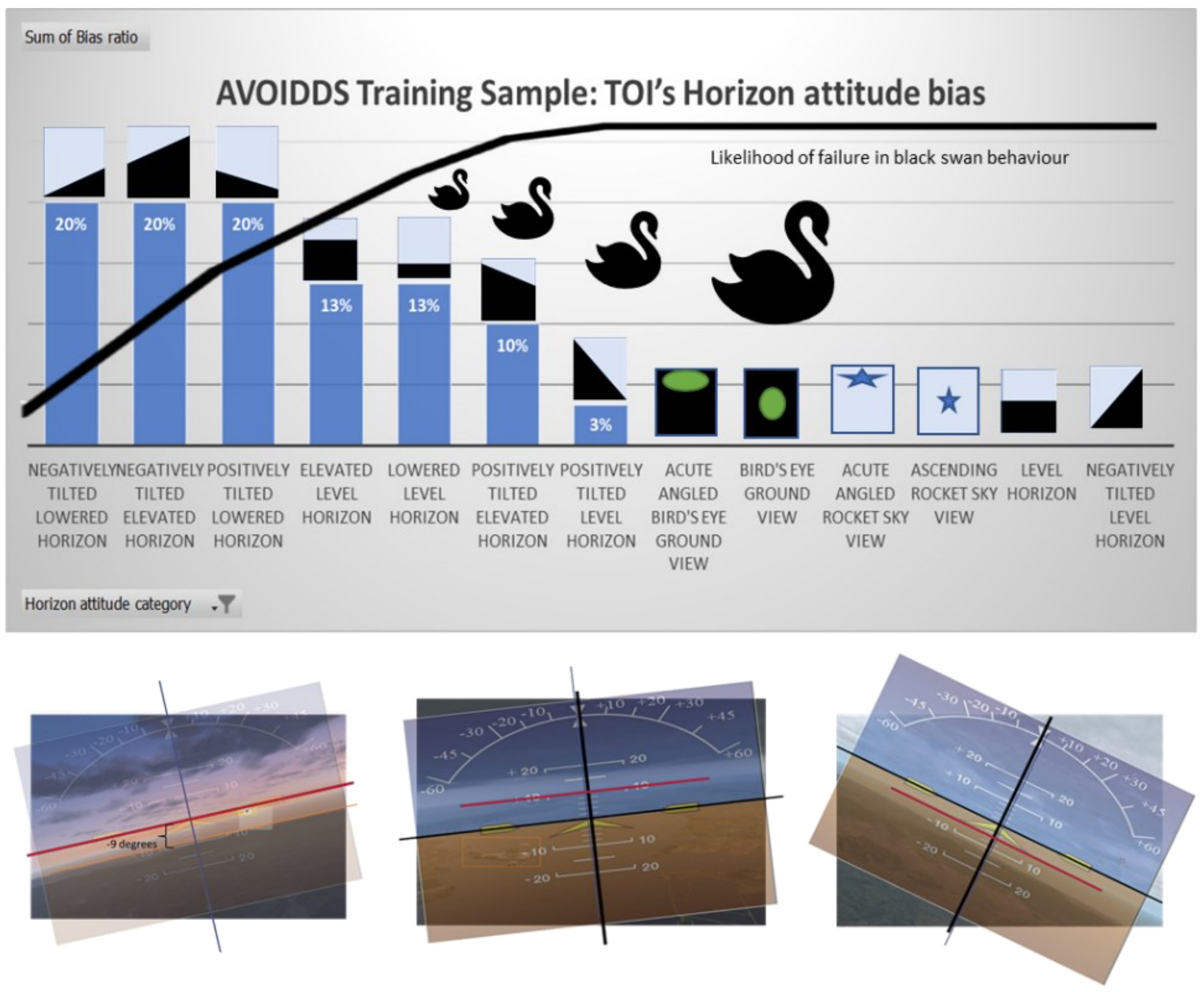


Figure **Error! No text of specified style in document.**.19 Pictorial horizon coverage for AVOIDDS sample training

Figure 4.19 illustrates that several horizon categories are not represented in the AVOID sample training dataset, which comprises 30 images selected randomly from the training dataset. Furthermore, we utilise the Black Swan symbols to highlight the pictorial hazards associated with potential gaps in coverage, indicating that the dataset's reliability is compromised, particularly in its ability to manage black swan scenarios within those absent categories. Consequently, from this perspective, the AVOIDDS training dataset may be deemed untrustworthy. The black curve shown in the graph represents an indicative curve that indicates the level of distrust; we refer to it as the “architect distrust curve”.

E.7.9 Traceability of images to their original descriptive text.

The concept of traceability in the CuneiForm-based system is foundational for linking pictorial elements in images to their descriptive counterparts in text. These annotations serve as a conceptual bridge, connecting each pictorial aspect depicted in a CuneiForm image to its original descriptive representation. The process involves assigning a unique numeric annotation to every feature visualised in the CuneiForm image. This annotation directly correlates with the specific aspect of the written description it represents. Conceptual annotations are captured in between braces {x}. Sometimes, more refining aspects are depicted related to some parent aspect. The annotation may follow the format of {x.y} to capture refinements. A potential validation process can be done by agreeing or disagreeing with what is being referenced in the CuneiForm, correctly captured in the description.

E.8 Comprehensive Multi-Environment Operational Environment Definition (Multi-ODD)

The comprehensive definition of a Multi-Environment Operational Design Domain (Multi-ODD) is critical in reducing architects' epistemic uncertainty and enhancing lateral and vertical predictive thinking in the context of autonomous systems. By establishing a structured taxonomy of natural and man-made features, the multi-ODD framework provides a systematic approach for identifying and contextualising environmental variables that interact with Autonomous Systems (AS). This standardisation aims to support the architect in objectively exploring and specifying a broad spectrum of scenarios.

To address the limitations mentioned in the literature review section 2.1.7, a general ODD definition process may be helpful to standardise defining operational domain taxonomies. We needed to describe a general process to help us standardise multi-environment ODD for our train tracks case study. We did not find a standard that considers both air and train tracks operational domain. In general, we envisage expanding the multi-ODD for various types of AS; those operating in sea, air, space, or land domains should encapsulate a range of abstract features to ensure comprehensive coverage. These features are categorised into two main taxonomies: Natural and Man-made related features.

E.8.1 General ODD Definition Process

Here is a general ODD characterisation process that defines the minimum information needed for a comprehensive ODD:

Step 1: Define natural surrounding ODD taxonomy:

1. **Dynamic or cyclic changing natural classes**: Factors that include elements that move or change position. Also, it includes cyclic state-based phenomena.

Examples: birds, fish, wind currents, seasonal changes (spring, summer, autumn, winter), tidal changes.

2. **Static natural classes:** Classes of factors that include relatively stationary natural elements complex.

Trees, rocks, and coral reefs are examples. In the space domain, this may include the sun, moon, or any relatively stationary celestial objects (including far stars in the background).

3. **Natural surrounding noise and interference classes:** Natural ambient noise and interference refer to background sounds and signals originating from natural sources that can impact an AS's functionality. These noises and interferences are typically omnipresent and can vary significantly depending on the operational domain.

Example: an underwater drone encountering communication difficulties due to dolphin echolocation clicks or the noise from underwater volcanic activity.

Step 2: Define manmade surroundings taxonomy:

4. **Informational environment classes:** Classes of factors that include the set of informational signals and data streams generated by human-made systems that assist or influence the operations of an is.

Example: a GPS navigation system providing turn-by-turn directions to a vehicle.

5. **Dynamic or changing engineered classes:** Classes of factors that are human-made elements that move or change position.

Example: traffic signals (changing from red to green), cars, trains, aircraft, cultural events.

6. **Static or static engineered classes:** Classes of factors are human-made elements that remain stationary.

Examples: bridges, buildings, roads, dams.

7. **Classes of man-made surrounding noise and interference:** Classes of factors that include man-made engineered noise and interference encompass unwanted signals or disruptions generated by human-made systems that can impact the functioning of an AS.

Example: These can include electromagnetic interference (EMI), radio frequency interference (RFI), acoustic noise, and other disturbances.

These categories ensure that relevant aspects of an AS's environment are considered, allowing for safer and more effective operation within its designated domain.

E.8.2 Comprehensive Multi-ODD that combines Land and Air-based AS applications

Traditional ODD frameworks for autonomous road vehicles often overlook the complexity and variability of other environments. By systematically defining dynamic, static, and noise-related factors in both natural and manmade contexts, we ensure that a wider range of potential challenges and operational constraints are addressed. This comprehensive consideration is crucial for designing safety into Autonomous Systems, as it allows for anticipating and mitigating a broader array of risks.

For complete taxonomy, refer to Appendix A. The general ODD process outlined above was systematically applied to create a comprehensive ODD framework encompassing land and air-based AS. This comprehensive ODD framework ensures that all relevant aspects of the operational environment are considered, allowing for the discovery of failure modes and Black Swan Scenarios and, thus, safer and more effective AS operations within their designated domains. The following elaboration details how each step of the general ODD process was applied to combine land and air-based AS applications.

E.8.2.1 Step 1: Define Natural Environment ODD taxonomy:

Dynamic or cyclic changing natural classes: The process involved identifying natural factors that are constantly moving or periodically evolving to account for dynamic or cyclic changing natural classes. These factors are crucial for understanding the environmental variability that AS must adapt to.

- **Weather Conditions Variety:** This includes a comprehensive range of weather conditions such as temperature, precipitation, wind, humidity, pressure, and visibility. For instance, temperature variations can significantly impact the performance and efficiency of AS, particularly those with sensitive electronic components or battery dependencies.

- **Time of the Year Seasons-specific Changes**: Seasonal changes, including spring, summer, autumn, and winter, bring distinct environmental conditions that affect AS operations. For example, winter may introduce snow and ice, obstructing pathways and reducing sensor accuracy, while summer may present heat waves affecting cooling systems.
- **Illumination Variety Definition**: This includes variations in natural light conditions such as sunrise, sunset, twilight, and night. Additionally, it considers the positioning of celestial bodies like the moon, which affects visibility and navigation for both land and air-based AS, particularly during night-time operations.

Static, unexpected, or non-cyclic changing natural classes: These classes include natural elements that remain relatively stationary or do not follow a predictable cycle but still influence AS operations.

- **Landscape Types Variety**: The variety of landscapes, such as urban, suburban, rural, mountainous, forest, desert, coastal, water, industrial, and arctic landscapes, is considered. Each landscape type presents unique challenges and considerations for AS, such as the need for urban navigation algorithms or rugged terrain handling capabilities in mountainous regions.
- **Geographical Region-specific Natural Phenomena**: This includes phenomena unique to specific regions, such as atmospheric phenomena (e.g., northern lights, monsoons), hydrological phenomena (e.g., floods, tsunamis), geological phenomena (e.g., earthquakes, volcanic eruptions), and biological phenomena (e.g., animal migrations).
- **Perceived Pictorial Horizon Attitude**: This involves the perception of the horizon as captured by AS sensors, which includes image rotational movements like frame roll and pitch. Understanding how the horizon is perceived helps calibrate sensors and ensure accurate navigation and stability.

Natural surrounding noise and interference classes: Natural ambient noise and interference can significantly impact the functionality of AS. These factors are typically omnipresent and vary depending on the operational domain.

- **Natural Auditory Noise**: This includes noise from wind, animal sounds, vegetation movement, and thunderstorms. Such auditory noise can interfere with acoustic sensors and communication systems, necessitating robust filtering and noise-cancellation techniques.
- **Natural Non-audio Interference** includes electromagnetic interference (EMI) from natural sources such as solar flares and terrain-induced interference. Such interference

can disrupt electronic systems and communication links, requiring AS to be equipped with shielding and error-correction capabilities.

E.8.2.2 Step 2: Define Manmade Environment Taxonomy

Dynamic or Changing Engineered Classes: This step focuses on human-made elements that move or change position, affecting AS operations.

- **Road, Train, and Air Traffic Conditions**: Dynamic traffic conditions require AS to adapt to varying speeds, densities, and flow patterns in real-time.
- **Dynamic and Impermanent Road, Train, and Aerospace Objects of Interest**: This includes temporary obstacles, construction zones, and other transient objects that AS must detect and navigate around.
- **Socio-Political Factors**: Events like cultural festivals, political gatherings, and public demonstrations can create dynamic changes in the operational environment, requiring AS to adjust routes and operational plans.

Informational Environment Classes: This step involves identifying the set of informational signals and data streams generated by human-made systems that assist or influence AS operations.

- **Road Transport Connectivity**: This includes Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication systems that provide real-time traffic information, navigation guidance, and safety alerts.
- **Railways Transport Connectivity**: This includes communication systems between trains (Train-to-Train) and between trains and wayside infrastructure (Train-to-Wayside) for coordinated operations and safety management.
- **Air Travel Connectivity**: This includes communication systems for airborne vehicles, such as Airborne Vehicle-to-Infrastructure (V2I) and Airborne Vehicle-to-Airborne Vehicle (AV2AV) communications, essential for air traffic management and collision avoidance.

Static or Static Engineered Classes: This step includes human-made elements that remain stationary and form the consistent landscape within which AS operates.

- **Road, Train, and Aerospace Physical Infrastructure**: Permanent structures like bridges, buildings, roads, dams, rail tracks, and airports provide a stable framework for AS navigation and operations.

- **Static and Permanent Road, Train, and Aerospace Objects of Interest**: This includes road signs, signal lights, and fixed landmarks essential for navigation and operational reference.

Manmade Surrounding Noise and Interference Classes: This step addresses unwanted signals or disruptions generated by human-made systems that impact AS functionality.

- **Land-based Noise and Interference Factors**: This includes electromagnetic interference (EMI), radio frequency interference (RFI), and acoustic noise from urban environments, industrial zones, and transport systems.
- **Air-based Noise and Interference Factors**: This includes interference from radar systems, communication signals, and other airborne sources that can disrupt the operations of UAVs and other airborne AS.

We have created a comprehensive framework that covers relevant aspects of land—and air-based AS operations by applying the general ODD process. This framework ensures that the unique requirements and challenges of these domains are addressed, enhancing the effectiveness of ODD for Intelligent Systems design in such domains.

E.8.3 ODD Aleatoric Uncertainty grading

In section 4.1 we explained the direct relationship between randomness in the physical environment (aleatoric uncertainty) and impact on epistemic uncertainty (ignorance) for the architect and the machine.

To effectively integrate safety into the training and testing of autonomous systems, we require a straightforward process for identifying what constitutes an "unsafe" operational design domain (ODD). This understanding will enable us to evaluate the criticality of various environments and inform the design process accordingly.

The rationale behind the grading system for ODD is based on the recognition that environments are complicated complexes where confusion and uncertainty can arise for autonomous systems and their architect. As the ODD conditions become hard to predict (confusing complicatedness), the likelihood of encountering unexpected environmental behaviours also grows. In other words, the more energetic the ODD, the greater the aleatoric uncertainty, the greater the architect's epistemic uncertainty about the resilience of their assumptions, heightening the potential for unforeseeable events and safety challenges (Black Swan events). See Appendix B for a full definition of the uncertainty Grading System.

E.8.4 Breakdown of the Grading System for ODD Uncertainty

Appendix B outlines an Operational Environment Uncertainty Grading System designed to support ML component safety training for autonomous systems like the Eagle Robot. The framework categorises operational environments into five grades (A to E), ranging from Grade A ODD Uncertainty (Ideal, assuming normal-complex conditions) to Grade E ODD Uncertainty (Not Favourable, assuming Stochastic Confusion conditions)**.** These grades are based on various environmental characteristics, providing a systematic approach to evaluating and modelling an ODD's uncertainty levels for AS design and validation purposes.

E.8.4.1 Key Components of the Grading System:

The following is a structure of how grading criteria can be integrated as part of ODD definition:

1. **Solution System and Operational Space**:
   - Example: The Eagle Robot's typical area of operation (e.g., a train track zone) serves as the reference operational space.
2. **Environment Characteristics**:
   - Environments are characterised by natural and contextual factors such as lighting, weather, and geographical features.
3. **Grading Schema**: The following are the assumptions made about the degree of aleatoric uncertainty involved. The more energetic and extreme the ODD, the more uncertain and chaotic it becomes. The more potentially chaotic the ODD is, the more potential for Black Swan scenarios, thus increasing the likelihood of facing an utterly Stochastic Confusion field for the architect and their trainee machine.
   - **Grade A ODD Uncertainty**: Ideal conditions assuming relatively minimal uncertainty, such as sunny weather, clear visibility, and low environmental variability. The least ODD uncertainty of the grading system.
   - **Grade B ODD Uncertainty**: Favourable conditions with slight variations, such as partly cloudy skies or moderate wind speeds.
   - **Grade C ODD Uncertainty**: Partly favourable conditions with noticeable variability, such as mixed sun and clouds or moderate precipitation.
   - **Grade D ODD Uncertainty**: Less favourable conditions with significant uncertainty, such as overcast skies, strong winds, or low visibility.
   - **Grade E ODD Uncertainty**: Not favourable conditions where operational challenges and risks are maximised, such as severe storms or foggy conditions. The highest ODD uncertainty in the grading system.
4. **Environmental Factors and Grading Criteria**:

Grading can be used to describe a change in ODD conditions.

- Natural Lighting Conditions: Transition from sunny (Grade A) to overcast or hazy (Grade E).
- Weather Conditions: Graded by precipitation, wind speed, humidity, and visibility.
- Time of the Year: Seasons-specific environmental characteristics.
- Landscapes type variety definition.
- Geographical region-specific natural phenomena.
- Time of the Day.
- Perceived Horizon Attitude.
- Sun sphere positioning.
- Moon sphere positioning.
- Specialised zone features.

E.8.4.2 Purpose and Application

The Uncertainty Grading System serves as a foundational tool for:

- **Designing and Training ML components specialised in a particular grade:** Ensures datasets capture a wide range of environmental uncertainties.
- **Risk Assessment and Safety Assurance**: Enables systematic evaluation of environmental factors to anticipate potential operational challenges.
- **Operational Context Definition**: Establishes a clear framework for characterising the complexity and uncertainty within the ODD.

E.9 Assuring the reduction of risk during unforeseeable scenarios

Developing an ML model through a training pipeline is one thing; developing a safety-defensible model with empirical evidence about the safety of its emergent properties, such as intelligence in unforeseeable scenarios, is another.

The primary reason for using ML models (such as in perception) is to leverage their powerful capabilities to potentially operate correctly in unforeseeable operational scenarios that may arise during the development process. Here is the catch:

Part of being "unforeseeable" is demonstrating that a scenario is correctly unforeseeable! And subsequently, the model has been correctly trained for it.

The second catch related to this challenge is to assurance of safety:

ML model's emergent generalisation shall reliably and correctly[15] emerge safely during those unforeseeable scenarios.

Traditionally, in deterministic safety-critical systems, we manage the risk of missing requirements through rigorous systematic approaches (such as SHARCS) to achieve completeness. However, when we are developing ML models, our intuition drastically changes; we do not worry about the majority of missing operational requirements, because we expect the model to perform as intended for unaccounted-for operational requirements, at least those that are somewhat similar to the training and validation data. Nonetheless, we still need to demonstrate that safety properties correctly emerge during those unforeseen scenarios.

Similarly, it is one thing to develop datasets and ensure that they enable the safe emergence of expected behaviours during unforeseen scenarios (those within the context of the problem domain but not mentioned in the training, validation, or testing) from the trained model; it is another. In consideration of unforeseen scenarios, we account for both adversarial and natural data shifts resulting from changes in operational domain complexity.

Assurance refers to the systematic process that guarantees to an authority that the desired properties are reflected in the system's actual, concrete, and tangible behaviour. However, the assurance of emergent intelligence considerably relies (for the most part) on the intangible information stored in its dataset in pictorial format. Here, the topic of certifiability of required safe emergent intelligence becomes directly relevant and highly challenging when it comes to convincing a regulator to trust a safety-critical AI system.

Hence, the topic in this section directly relates to the challenge of ML-model robustness certifiability in the face of unforeseen perturbations relative to training and testing datasets, for example PROVEN framework [22]. PROVEN technically aims to prove that, if we only consider random perturbations from a known distribution, we can often obtain much larger certificates in the sense that the model will be correct with probability ≥ some value over a certain measurable distance from a specific reference or threshold.

White-box Certified Robustness Approaches include two main types [24]:

- **Robustness Verification:** Certifies that a model's prediction remains unchanged within a perturbation bound.
- **Robust Training:** Trains models to maximise this certified robustness.

[15] By correctly, we mean to correctly manifest the desired properties (the architect's intent).

In our black-box approach to certifiability of robustness, we rethink the problem in a different light, and view it as:

> **Defensibility of a design and validation evidence about ML model behaviour in unforeseeable scenarios to reduce a competent human regulator's uncertainty and increase their trust.**

If we view it this way, it means that any systematic approach in demonstrating emergent properties, regardless of how mathematically rigorous, will help reduce a regulator's uncertainty and increase their trust. We would welcome a multi-dimensional approach, combining white and black-box processes to achieve this purpose. In our research, we focus on providing a systematic and holistic approach to achieve this goal. In the context of pictorial datasets, we propose some holistic terms encompassing the types of unforeseeable scenarios, which help us define different strategies to validate performance. We summarised the types of unforeseeable pictorial datasets in our conclusion chapter.

Our primary concern is to provide a demonstrable and defensible systematic approach for compliance with regulatory standards (for example, CAA SORA [19]) when making claims about the extension of safety properties beyond the performance over test datasets (i.e., unforeseen data). We approach the problem from a black-box perspective. We do not claim that this is the ideal approach; rather, our black-box approach complements any white-box approach to provide the necessary defensibility of claims regarding the provision of empirical guarantees of performance in a limited number of pictorial situations outside the test datasets.

Concepts in this section are based on our experiments conducted in sections 7.2 and H.10. We learned from these experiments that to ensure the safety of ML components, we need to address the challenge of guaranteeing the emergence of safe intelligence in unforeseeable scenarios outside the scope of training and testing datasets. So, if we examine the case of ensuring emergent properties emerge in unforeseeable scenarios (we call them Black Swans), we need a development process that clearly demonstrates a systematic discovery of unforeseen scenarios and that they are mitigated in training and validation.

We learned the latter principle from our experiments in translating requirements into a pictorial training syllabus. This section provides a theoretical foundation that we used to develop the process in Stage 6 of the approach (see Section 7.2).

The most challenging aspect is constructing an objective, convincing process that can demonstrate model performance during unforeseeable scenarios. Our process in stage 6, and especially the approach in step 7 (section 7.2.8), aims to provide a proposed solution for ensuring the preservation of safety requirements during unforeseen scenarios. Our experiments in Section 6.8 taught us a great deal about what “unforeseen” means at a pictorial level. The concrete doing of the training syllabus happens at the level of images and the uncertainty of information. This is why we cannot simply rely on the claim that Stage 4 provides satisfactory evidence for deliberate risk reduction during Black Swan conditions.

Our approach towards objective evidence of reducing the risk of unsafe operations during unforeseeable scenarios (Black Swan operations) is based on the following principles:

E.9.1 Motivation: Mandatory CAA Assurance Robustness

The Civil Aviation Authority (CAA), through the JARUS Specific Operations Risk Assessment (SORA) framework, establishes a structured approach to evaluating the trustworthiness and safety assurance of unmanned aircraft systems (UAS). A central concept in this framework is “robustness”, defined as the combination of integrity (how effectively a risk mitigation or operational safety objective reduces risk) and assurance (the ***confidence*** that this effectiveness is reliably achieved). The SORA outlines three levels of robustness, low, medium, and high, based on a matrix that correlates integrity and assurance levels. Notably, robustness is constrained by the lower of the two dimensions; for instance, medium integrity combined with low assurance results in low overall robustness.

The expected assurance levels range from self-declared compliance with limited verification (low assurance), to evidence-backed claims using testing or operational data (medium assurance), to third-party validated compliance reports (high assurance). This structured expectation for development process rigour and evidence-based compliance reinforces the need for systematically demonstrating UAS safety and trustworthiness in complex airspace operations.

The key notion in SORA robustness requirements lies in the definition of “confidence,” which refers to the Regulator’s Confidence that the supplied evidence demonstrating compliance indeed achieves the expected quality (for example, high assurance). To be more precise, SORA explicitly associate certainty to the level of assurance,

“The degree of certainty with which the level of integrity is ensured”

Achieving certainty in the effectiveness of a risk mitigation process, in this context, is rather ambiguous and hard to quantify since it depends on the assessor’s point of view. Another challenging quality SORA introduces is the quantification or qualification of “how good”;

“Level of integrity (i.e., how good the mitigation/objective is at reducing risk)”

Ensuring that risks are minimised during Black Swan operations, along with other scenarios within the operational domain of the machine learning (ML) model that were not considered during the design phase, presents the architect with two main challenges when it comes to providing evidence of compliance with a high level of robustness.

1. The challenge is to convince the regulator's sense of confidence (certainty) that the training process indeed developed and assessed the model during the Black Swan scenario. This can extend to some residual scenarios that are not part of the training and testing.
2. Convince the regulator that the CuneiForm syllabus are good at reducing the risk during Black Swan scenarios.

For that, we understand the above challenges as follows:

1. **Satisfying Regulator's sense of Certainty in the effectiveness of training process to enable safe ML behaviour during Black Swan operations:** We understand the term "certainty" to be related to reducing the assessor's epistemic uncertainty so that there is minimal doubt about the developer's approach to reducing a particular risk or achieving a specific safety goal. To achieve a convincing argument, the architect can follow the following principles:
    d. Black Swan scenarios have been separately considered and predicted**.**
        - **So what?** We dedicated Stage 1 to distil the problem domain (prior to the introduction of the solution system into the problem complexity field). Then, in Stage 4, we further enhance the approach to discover more Black Swan scenarios, with the solution being part of the problem domain.

    e. The process of predicting those Black Swan scenarios is scientifically rigorous.
        - **So what?** We introduced SECoT processes to justify the architect's engineering judgment of why a Black Swan scenario is such and how to tackle it. SECoT demonstrates a clear application of lateral and vertical thinking to predict Black Swans.

    f. They are traceable to the datasets.

- **So what?** We introduced CuneiForm syllabus development and validation to demonstrate that datasets can be traced to safety requirements.

2. **Demonstrating to the regulator how good the mitigation is at reducing risk during ML-training process (at dataset level):** to elevate this challenge, the architect would need to demonstrate that the risk of failure during Black Swan operations has been reduced through a systematic, evidence-based trial-and-error approach**:**

    d. The architect needs to demonstrate to the regulator that the model training process has been systematically training and testing, thus improving an initial configured model on a series of Black Swans and that the model clearly has improved in performance during Black Swan operations.

    - **So What?** We adapted the **SHARCS** process principle of a systematic and incremental approach to introducing capability into systems design. We treated Black Swan scenarios as additional capabilities for the system and incrementally introduced an initially trained and tested model to Black Swan scenarios.
    - We assessed the model's ability to handle such scenarios, and in the case of success or failure, we demonstrated deliberate training and assessment. With this approach, we demonstrate that discovered Black Swan scenarios are indeed challenging situations for a model configuration, and that our training log shows consistent improvement in various Black Swan scenarios.

To support the above approach, we need to clarify a number of key concepts:

E.9.2 Unforeseeability and Data Distribution Shift

In a pictorial sense, we interpret **a data distribution shift** as meaning the training and validation strategy has never encountered a similar (exact copy or somewhat the same) scenario. This can be demonstrated by showing that there is no pictorial depiction in the training and validation sets that statistically resembles the pictorial scenario in question (OOD OOD). An **unforeseeable scenario** means that the architect has not accounted for it during stages 3 and 4 of the development process. For example, the architect missed the specification of a CuneiForm requirement where an adversarial drone (TOI) was positioned precisely in the sphere of the sun relative to a camera. Alternatively, the architect may deem that in the context of a train track zone operational environment, it is **unforeseeable** that a perception model will encounter a scenario where a toddler holds an adversarial drone. This scenario would constitute context-shit since it is out-of-contex1twith respect to the operational domain defined by CuneiForms.

Technically speaking, unforeseeability means :

- The absence of a CuneiForm requirement.
    - To mitigate missing CuneiForms, the architect can cover all possible variations in CuneiForm classes.
- A pictorial image that is out of context relative to a CuneiForm requirement.
    - To mitigate this risk, the architect can include out-of-context (OOC) images.

So, how do we go about producing a defensible argument that can help a regulator trust that we have intentionally tackled in our design?

In Section 6.8, our experiment demonstrates that it is possible to provide a systematic and defensible argument that can demonstrate performance during white ghost scenarios. We need to demonstrate to the regulator that:

- We have followed an objective, systematic approach in predicting normal operational scenarios, and we used those to initialise training of our model (we use stage 3 to do so).
- We then followed a separate systematic process to predict Black Swan, non-typical operations (stage 4 of our process).
- We need to demonstrate that we tested the model over OOD data with respect to some typical operations, training, and validation (white swans) to demonstrate "data-shift of pictorial scenarios". These are the Black Swans, if they can be traced to Black Swan scenarios.
- We then need to show that we have systematically included those Black Swans into the initial white swan training process.
- We then need to test the new, trained model (White + Black swans) on another dataset that is in-distribution with CuneiForms.
- We also need to demonstrate that the model is trained and tested over unforeseeable scenarios. These are Out-of-Context (OOC) images.

Thus, in-and OOD similarity testing provides an approach for demonstrating training and performance during data shifts in Black Swan scenarios. Systematic audit train of constructing a dataset is a useful demonstration that at a given time of stepwise building of the dataset, it can be said that the training and validation have or have not seen an additional dataset distribution using

E.9.3 Reduction of epistemic ignorance ALARP

Now we understand the nature of the unforeseen scenario at a pictorial level. With that in mind, we need a systematic audit trail that clearly exposes those sources of ignorance and shows that they have been accounted for in the training process. With that in mind, we developed our approach, outlined in Table 7.12. The approach shows clearly that there have been a audit trail of discovering OOD related to real world data shift (Black Swan CuneiForms) and that a concurrent configuration trained model failed at it, but we included it systematically and now we have a model that clearly trained to perform and preserve safety properties in unforeseeable scenarios because we have a long history of training in those scenarios.

E.9.4 The risk of unmitigated OOD data

Unmitigated OOD data, which we understand to be associated with a model's epistemic ignorance, arises when a model is confronted with data or scenarios that were not represented during training. In such cases, the model's performance may degrade substantially, manifesting in underperformance or outright failure on OOD or OOC datasets. Demonstrating that a dataset is genuinely OOD is a necessary but not sufficient condition for establishing confidence in generalisation; it is equally important to incorporate OOC data, as this provides additional empirical evidence regarding the model's behaviour under unfamiliar conditions. However, one must be cautious not to rely solely on OOC datasets as a proxy for all unforeseeable scenarios, since the space of possible data shifts is effectively unbounded.

To provide empirical evidence of unmitigated data shifts, a systematic, incremental process is useful. Rather than aggregating all available images or samples at once, the training regimen should be structured so that data are introduced progressively. Maintaining a detailed training logbook that clearly shows a regulator how unmitigated OOD and OOC examples were taken into consideration becomes indispensable in this context, as it provides a chronological record of model iterations, hyperparameter configurations, and evidence of performance under unforeseen conditions. More importantly, it needs to document instances in which a candidate model fails to generalise, thereby revealing potential sources of data shift. In practice, this requires demonstrating empirically that a particular input distribution or environmental condition makes the model's predictions unreliable. A configured model that experiences such failures during testing can then be retrained on the newly identified distribution, retested, and the results recorded. Through this cycle of testing, failure analysis, and retraining, one can accumulate empirical evidence of epistemic ignorance and systematically expand the model's coverage of the data manifold.

From an assurance standpoint, it is crucial to retain instances of model failure on OOD or OOC data, as these failures serve as objective indicators of residual epistemic uncertainty. By cataloguing these occurrences, one can explicitly qualify the extent to which the training and validation sets were insufficient to capture the full variability encountered in deployment. In turn, this enables practitioners to articulate, in clear terms, where the model's ignorance lies and to what degree it has been mitigated. If repeated iterations demonstrate persistent failures in specific regions of the input space, despite progressive retraining, it becomes evident that simply augmenting the dataset may not be sufficient. Instead, this insight may motivate alternative approaches, such as revising model architecture, domain adaptation techniques, or targeted data collection campaigns aimed at previously underrepresented conditions. Ultimately, maintaining a transparent record of these failures underpins a rigorous argument regarding the remaining OOD risk and guides strategic decisions to fortify model generalisation.

E.9.5 Adapting the SHARCS process for Black Swan-driven training pipeline

Safety assurance processes aim to clearly and unambiguously demonstrate that errors in a practitioner's engineering judgment have been identified and remedied appropriately, as observed by a regulatory body. In other words, the ultimate goal of any safety assurance process is to reduce the following uncertainties:

- The architect's epistemic uncertainty.
- The qualifying regulator's epistemic uncertainty.

One profound approach for safety-defensibility of safety-critical software is SHARCS. It handles the reduction of epistemic uncertainty related to the preservation of safety properties during failures in a few ways:

- Using the STAMP-inspired deviation of the control actions process.
- Using Event-B proof obligations.
- Logical and provable traceability of subsystem-level requirements to system-level requirements.
- Provision of clarity through incremental increase of complexity (abstraction).

Using those four methods, SHARCS helps the assurance process in two dimensions:

- The architect realises any hidden ignorance about the design by discovering vulnerabilities in their design decisions due to incomplete engineering judgment.
- The regulator's ability to be in no doubt that the architect approached the design systematically, discovering flaws and mitigating them accordingly, thus increasing trust in the system.

Essentially, SHARCS prompts the architect to model a level of abstraction in control structure. By the end of that step, the architect ceases the abstraction modelling when they sense that they know enough about the system. SHARCS does not stop here; it then prompts them again to challenge their own epistemic certainty through the STAMP process of predicting deviations from expected control actions. The architect then realises their ignorance and that they may have overlooked further actions, and iteratively updates their knowledge about the system design and the problem. It seems as if SHARCS tacitly makes the designer predict what he thinks about what a complete solution should be, then it injures the architect's sense of confidence in their mental model using STAMP deviations guide words, and offers to remedy this broken sense of confidence with more requirements.

Therefore, in general, the assurance part of this process boils down to SHARCS' ability to provide a systematic audit trail, in which the practitioner realises where they were wrong or missed something, mitigates it, and incorporates it back into the design requirements. Eventually, those mitigation plans (in AMLAS terms called Safe Operating Concept) will manifest themselves at the concrete code or hardware level.

However, for that to happen, we need an ontology that bridges the gap between high-level STAMP deviations, mitigation, and code. Event-B provides such an ontology. In code, the same process applies; we have code checkers that discover semantic and syntactic bugs, which technically alert the software engineer to sources of pragmatic errors or oversights in their engineering judgment that they may have missed. In hardware engineering, HAZOP uses a similar approach through guide words. The practitioner’s ignorance is then exposed and remedied through mitigation plans.

From such articulation, we understand safety assurance in general at a concrete level to be a process that unambiguously demonstrates the following:

- Sources of ignorance had been exposed and validated to be legitimate errors.
- Remedies had been accounted for and unambiguously included in the design.

Therefore, if we wanted to concretely assure a model behaviour in Black Swan or unforeseeable operations domain, we need to be able to demonstrate the following:

- Sources of pictorial ignorance had been isolated and validated as legitimate missing information.
- Remedies had been accounted for in the dataset by their unambiguous inclusion in the training process.

The question then forces itself:

- How do we demonstrate exposure of pictorial ignorance at the dataset level?

Similar to what SHARCS does for the practitioner, we demonstrate the reduction of risk during Black Swan scenarios in the ML training process at the dataset development stage. While the architect (in SHARCS) seeks to discover ***deviations*** from ***expected*** control actions, the architect in our training approach seeks to identify those pictorial subsets ***whose distribution deviates*** from ***expected operations*** datasets, assess model performance to check how truly challenging those OOD subsets are, and then systematically and incrementally mitigate them by increasing the complexity of the training and testing dataset.

The following are the main concepts we learned through our experiments in section H.10 and H.11:

E.9.6 Systematic, incremental pipeline to Black Swan training and testing of ML models

The following are the main steps of our approach to demonstrate that a model has been trained to handle unforeseeable scenarios. We tailored this process in section 7.2:

1. **Initial Training and In-Distribution Validation ($A_1 \rightarrow A_2$):**

   1.1. Assemble $\mathbf{A_1}$, the in-distribution training set, by sampling data that typifies normal operations (the "White Swans"). This data originates from Stage 3 of our overall process, in which we enumerate standard operational scenarios.

   1.2. Train the model on A_1 and evaluate its performance on A_2, a disjoint holdout of the same distribution. Successful performance on A_2 establishes a baseline of competence under known conditions.

2. **Generation of the First Black Swan Test Set (B_1):**

   2.1. Identify a distinct operational scenario that the model has never encountered, one that lies outside the A_1/A_2 distribution. For example, consider a set of "CuneiForm" images that exhibit attributes radically different from those in A_1 (e.g., lighting, texture, or structural anomalies).

   2.2. Curate $\mathbf{B_1}$, a collection of such Black Swan images, explicitly designed to probe the model's capacity to handle rare or anomalous events. B_1 is traced to "Black Swan scenarios" as described in Stage 4 of our process.

3. **Isolated Black Swan Evaluation:**

3.1. Without mixing B_1 with the original in-distribution test set (A_2), evaluate the A_1-trained model on B_1. By preserving B_1 as a separate test set, we ensure that any performance degradation directly evidences an OOD failure rather than statistical noise.

3.2. Record all relevant metrics, accuracy, precision, recall, within the training logbook. This logbook entry documents the introduction of a Black Swan scenario and the scrutiny of the model's response (success or failure).

4. **Outcome Analysis and Iterative Refinement.**
   - **Case 4a: A_1-Trained Model Succeeds on B_1.**

     4a.1. If the model performs at or above an acceptable threshold on B_1, then we have empirical evidence that it can generalise beyond A_1/A_2 to at least this particular Black Swan scenario. One may record this result as partial compliance with robustness requirements for data-shifted situations.

     4a.2. To further reinforce robustness, we incorporate B_1 into the training set (forming $A_1 \cup B_1$) and retrain the model. This "inoculation" step ensures that the model internalises features of B_1 rather than merely demonstrating a fortuitous pass.

     4a.3. Generate a new test set, $\mathbf{B_2}$, composed of additional images reflecting the same Black Swan characteristics (e.g., another CuneiForm variant). Evaluate the B_1-trained model on B_2.

     - If performance on B_2 remains satisfactory, we consider the Black Swan class (as represented by B_1/B_2) effectively learned. Record these findings, along with versioned data, hyperparameters, and metrics, in the training logbook. Cease further iteration for this class and compile the evidence for compliance demonstration.
     - If performance on B_2 degrades below the threshold, we have detected a generalisation gap within the Black Swan class. We then include B_2 (or those B_2 samples on which the model failed) into the training corpus (now $A_1 \cup B_1 \cup B_2$), retrain, and generate $\mathbf{B_3}$ (a fresh batch of the same class). Repeat this cycle until the model meets performance requirements on B_n, at which point we conclude that the model has effectively learned the Black Swan class in question.
   - **Case 4b: A_1-Trained Model Fails on B_1:**

4b.1. Failure at this stage is equally informative: it indicates that the original training regime (A_1 only) did not capture this Black Swan phenomenon, thereby revealing an “unmitigated Black Swan” scenario.

4b.2. Proceed identically to Case 4a, Step 4a.2 onward: add B_1 to the training set, retrain, and evaluate on a new test subset B_2. Continue iterative refinement ($B_2 \rightarrow B_3 \rightarrow$ ...) until the model’s performance on the Black Swan class satisfies the predetermined robustness criterion.

5. **Extension to Context Shifts (OOC Testing):**

5.1. The procedure above demonstrates resilience to distributional shifts within the same operational context (i.e., variations of CuneiForms). To address **contextual shifts**, where the entire scenario differs in a semantically meaningful way, we perform a parallel set of experiments using OOC datasets. These datasets may be drawn from open-source repositories or synthetic scenarios that represent an entirely different operational domain (for example, an unrelated object class or a simulated environment with different lighting, background, or physical constraints).

5.2. As with Black Swan evaluation, maintain an OOC test set (C_1, C_2, ...) distinct from A_1/A_2. Evaluate the model on C_1; record successes or failures. If the model fails, incorporate C_1 into training, retrain, and generate C_2, iterating until the model reaches acceptable performance on C_n.

This loop, OOD Black Swan and OOC testing, retraining on similar (in-distribution) data, and re-testing, demonstrates a deliberate and methodical design. It shows that the model has been repeatedly exposed to unforeseeable inputs and then fine-tuned using similar examples. This process builds a defensible argument for extending the model’s safety claims beyond the original test set. It aims to provide regulators with evidence that the model isn’t only trained for known scenarios but also has built-in resilience for unfamiliar yet statistically similar ones.

If the model later fails in a live operational scenario, we can investigate whether the failed image is similar to any past training or test data. If it isn’t, this could reveal hidden biases or gaps in our dataset, which we acknowledge as necessary for future research, although it falls outside the current scope of this PhD.

In summary, although some of these "friendly ghost" scenarios haven’t been directly tested, our systematic Black Swan approach provides a defensible, evidence-based reason to trust that the model can handle them safely and as intended.

What gives us confidence in the good performance of the friendly Ghost Pattern?

The training syllabus enables the model to perform as expected on an in-distribution dataset in experiments 1, 2, and 9.1. In those example experiments, a considerable portion of the test dataset was in-distribution with the training and validation. We did not find any in-distribution dataset, in all of our experiments, where the trained model did not perform as expected. The challenge primarily involved OOD datasets. It needed a journey for us to build trust over such data shifts. This means, in theory, if we have a test set made up of OOD images, then we can also expect that any missing test data (Ghost Patterns) which are in-distribution with the testing images that are OOD with our training and validation, we can be reasonably assured that the model performance extends to them too, but only them!.

However, where there are friendly Ghost Patterns, there are also many unfriendly Ghost Patterns. These are images and scenarios that are OOD with our testing, training, and validation. We could start bagging or collecting those unfriendly Ghost Patterns by producing more data, conducting similarity tests with the entire existing dataset, and then testing the model over them. However, such research can be done in the future, not part of this PhD for now. Busting Ghost patterns would be an interesting investigation in the safety assurance of AI systems.

E.9.7 Safety Regulation Compliance Justification

This section demonstrates why the Systematic, Incremental Approach to Black Swan Training and Testing (Section E.9.6) constitutes a suitable objective evidence for claims about model performance and safety preservation under limited unforeseeable (OOD Black Swan and OOC) scenarios, and thus meets the assurance requirements of safety-critical regulations.

E.9.7.1 Premises: What the Regulator Requires

Typically, regulators in safety-critical domains insist on the following foundational assurance:

Traceable and Systematic Evidence: Every development and test activity must be fully auditable. Datasets, model versions, experiment configurations, logs, and performance metrics must be versioned, timestamped, and cross-referenced to specific design requirements or risk scenarios. For example: **UL 4600 "Standard for Safety for the Evaluation of Autonomous Products"** requires that "Traceability of training and testing data to ODD coverage". Also, in section 8.5.4 "Machine learning-based functionality shall be acceptably robust to data variation", UL4600 mandates the following:

" a) Description of the method for and results from evaluating the functionality robustness of machine learning-based aspects of item:

1) Mitigation of risks due to any lack of robustness

b) Description of method for and results from evaluating response to distributional shifts:

1) Mitigation of risks due to distributional shifts of data

c) Field engineering feedback regarding performance when "surprises" have been encountered"

So, how does our approach in E.9.6 clearly demonstrate compliance with the UL4600 requirement?

By beginning with **Step 1**, training on in-distribution data (A_1) and establishing a performance baseline on A_2, we create a **reference** against which any Black Swan dataset can be determined to be OOD using similarity measures. Subsequent steps (E.9.6.2–E.9.6.5) then ensure that any model failure on a Black Swan test is:

1. **Identified and Quantified:** Failures are logged with metrics, demonstrating where the model's epistemic blindness lies.
2. **Remediated Through Iteration:** Each failure triggers a retraining cycle with the newly discovered OOD samples until performance meets the robustness criterion.
3. **Documented in a Traceable Manner:** Every data collection, model build, and evaluation result is recorded in the logbook, satisfying the regulator's demand for auditable evidence.
4. **Extended to Contextual Variations:** By explicitly testing on C_j (OOC sets), we ensure that the model's resilience encompasses not only distributional but also contextual shifts. We can evidence that a set of images represents examples of context shift, by showing that OOC images do not comply with any CuneiForm.

Consequently, our "Systematic, Incremental Approach to Black Swan Training and Testing" provides a **comprehensive, evidence-based demonstration** that:

- The model's behaviour under unforeseeable scenarios is not assumed but empirically validated.
- Data collection, model versioning, and performance metrics are fully traceable.

- Risk mitigation measures for distributional and contextual shifts are systematically invoked and documented.
- The process aligns with UL 4600 requirements regarding robustness evaluation, risk mitigation, and field feedback for “surprises.”

Thus, this approach furnishes regulators with the **objective evidence** needed to substantiate claims of safety preservation and robustness under both known and unforeseeable scenarios.