Safer AI for a Safer World

Keith Harrison-Broninski, Collaboration Tools Ltd

0. Abstract

Systems Engineering techniques enable organisations to demonstrate defensible decisions, manage evolving risk, and maintain accountability chains in AI-enabled systems. A "Safety by Design" approach to AI usage embeds safety considerations into AI development as a primary driver. By prioritising holistic real-world concerns from the start, this approach reduces the need for rework at later stages, so rather than generating additional costs is instead likely to increase the efficiency and effectiveness of systems development. Further, it will have a deep positive social impact that goes far beyond AI: safety engineering can operationalise ethical intent by forcing developers to make claims, assumptions, hazards, mitigations, evidence, and residual risks explicit and reviewable.

1. Introduction

Standard AI assurance frameworks define how to govern AI responsibly. By contrast, Systems Engineering techniques can be used for the actual safety assurance work, to explain the argument, expose weaknesses, derive mitigations, assess compliance, and maintain an auditable safety case as the system changes. This disciplined approach is vital for AI-enabled systems, in which complex algorithms get even more complex over time as they learn - not only are their mechanisms impossible for humans to fully understand, but also their possible inputs and outputs are effectively infinite.

A Systems Engineering approach to managing AI risks focuses on assuring the safety of wider engineered systems that use AI, in a traceable and reviewable way. The emphasis is on the formal artefacts produced by safety engineering work: argument structures, evidence chains, bowtie diagrams, compliance assessments, and justification that risks are ALARP (As Low As Reasonably Practicable). The table below summarises how Systems Engineering reframes conceptual AI governance challenges as practical engineering tasks.

AI Assurance Challenge	Systems Engineering Task
Are we governing AI responsibly?	Ensure that AI-enabled systems have robust, current, and defensible safety cases.
Do we have policies, controls, oversight, documentation, and lifecycle risk management for all our AI activities?	Justify the claims made for AI-enabled systems, identify argument weaknesses, mitigate associated risks, and demonstrate compliance.
Do we manage AI activities in a trustworthy and compliant way?	Explain what AI-enabled systems do, and where the arguments are weak.

AI Assurance Challenge	Systems Engineering Task
Can we demonstrate governance and conformity of our AI?	Maintain the artefacts needed to make engineering decisions for AI-enabled systems.
Have we defined all our AI-related processes?	Provide the evidence underlying all decisions regarding AI-enabled systems.
Are we controlling every AI-related activity?	Identify which artefacts are compliant, which are missing, and which need amendment for AI-enabled systems.

This paper addresses the research question: how can AI assurance frameworks be operationalised in practice through Systems Engineering artefacts? It proposes an artefact-driven, lifecycle-based approach, illustrates it through reference to mature Systems Engineering methodologies and established Systems Engineering techniques, gives examples from smart city and defense operating environments, then draws conclusions about the holistic benefits of using safety concerns to drive technology development.

2. Standard AI Assurance Frameworks

Standard AI assurance frameworks such as NIST AI RMF, ISO/IEC 42001, ISO/IEC 23894, and the EU AI Act are primarily designed to help organisations **govern, manage, document, and oversee AI-related risk** across the lifecycle of an AI system. NIST AI Risk Management Framework[1] (NIST AI RMF) structures AI risk management, ISO 42001[2] institutionalises it, ISO 23894[3] guides it, and the EU AI Act[4] enforces it:

- NIST AI RMF: structure and manage AI risks (voluntary framework)
- ISO/IEC 42001: establish and run an AI management system (governance standard)
- ISO/IEC 23894: guide AI-specific risk management within existing processes
- EU AI Act: impose legal obligations based on AI risk level (regulation)

NIST AI RMF is a voluntary U.S. framework for managing AI risk. Its core structure is four functions: **Govern, Map, Measure, Manage**. The point is not certification or legal compliance; it is to help organisations manage risks to people, organisations, and society while supporting trustworthy AI.

ISO/IEC 42001 is the international standard for an **AI management system**. It specifies requirements for establishing, implementing, maintaining, and continually improving an AI management system across an organisation. It is about governance, accountability, process discipline, and continual improvement.

ISO/IEC 23894 is narrower: it gives **guidance on AI-specific risk management**. It is for organisations that develop, deploy, or use AI and want to integrate AI risk management into

[1] https://www.nist.gov/itl/ai-risk-management-framework
[2] https://www.iso.org/standard/42001
[3] https://www.iso.org/standard/77304.html
[4] https://artificialintelligenceact.eu

their existing activities and functions. In practice, it complements broader governance approaches like ISO/IEC 42001.

The EU AI Act is law, not guidance. It uses a risk-based regulatory model: some AI uses are prohibited, some are regulated as high-risk, and some have lighter transparency obligations. The European Commission describes it as the first legal framework on AI, and high-risk classification includes AI used as a safety component of certain products or AI systems listed in Annex III.

3. Making AI Compliant

A growing body of research has explored how AI systems can be made safe, accountable, and trustworthy.

One important strand focuses on the use of assurance cases and structured argumentation for AI and machine learning (ML) systems, extending established safety-case approaches to address challenges such as model opacity and uncertainty.[5] [6] These approaches demonstrate how claims about system behaviour can be supported by structured arguments and evidence, but can remain focused on the construction of individual arguments rather than their maintenance across the system lifecycle.

A second strand addresses the safety of machine learning systems, highlighting issues such as non-deterministic behaviour, dependence on training data, and vulnerability to distributional shift.[7] [8] This work has significantly advanced understanding of AI-specific risks and failure modes, but tends to concentrate on model-level properties and technical mitigation strategies, rather than on how these risks are integrated into system-level assurance artefacts and decision-making processes.

A third strand focuses on algorithmic accountability and auditability, emphasising transparency, explainability, and the ability to justify decisions to stakeholders.[9] [10] [11] While this literature provides valuable frameworks for governance, auditing, and socio-technical analysis, it often treats accountability as an external process - such as audit or oversight -

[5] Hawkins, R.D. and Kelly, T.P. (2010) 'A systematic approach for developing software safety arguments', Journal of System Safety, 46(4), pp. 25–33.
[6] Calinescu, R., Weyns, D., Gerasimou, S., Iftikhar, M.U. and Habli, I. (2022) 'Engineering trustworthy self-adaptive software with assurance cases', *IEEE Transactions on Software Engineering*, 48(7), pp. 2291–2311.
[7] Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J. and Mané, D. (2016) *Concrete Problems in AI Safety*.
[8] Varshney, K.R. (2019) Engineering Safety in Machine Learning.
[9] Kroll, J.A., Huey, J., Barocas, S., Felten, E.W., Reidenberg, J.R., Robinson, D.G. and Yu, H. (2017) 'Accountable Algorithms', University of Pennsylvania Law Review, 165(3), pp. 633–705.
[10] Raji, I.D., Smart, A., White, R.N., Mitchell, M., Gebru, T., Hutchinson, B., Smith-Loud, J., Theron, D. and Barnes, P. (2020) 'Closing the AI accountability gap: Defining an end-to-end framework for internal algorithmic auditing', in Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency (FAccT). New York: ACM, pp. 33–44.
[11] Selbst, A.D., Boyd, D., Friedler, S.A., Venkatasubramanian, S. and Vertesi, J. (2019) 'Fairness and abstraction in sociotechnical systems', in Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT)*. New York: ACM, pp. 59–68.

rather than as something embedded directly within the engineering artefacts used to design, implement, and operate systems.

Taken together, these strands highlight critical aspects of AI assurance but leave a gap between high-level governance and technical implementation. In particular, there is limited focus on how assurance is operationalised through the generation, interrogation, and evolution of artefacts (such as safety arguments, hazard models, and compliance assessments) that underpin engineering decisions over time.

This paper addresses that gap by framing AI assurance as a Systems Engineering problem, consistent with emerging practice in safety assurance for AI-enabled systems. It builds on existing work in assurance cases, machine learning safety, and algorithmic accountability, but differs in focusing on how these concerns are integrated and maintained through lifecycle-oriented, artefact-driven processes. By applying established Systems Engineering techniques (Goal Structuring Notation, Bowtie analysis, and iterative assurance via the V-Model) it shows how governance requirements can be translated into traceable, continuously updated safety cases for AI-enabled systems. This perspective is consistent with emerging assurance methodologies for AI-enabled and autonomous systems, which operationalise similar principles in practice.

4. Alignment with Existing Methodologies

The approach described here is strongly influenced by established methodologies for assuring AI-enabled and autonomous systems, particularly the University of York's Safety Assurance of Complex Systems Engineering (SACE)[12] and Assurance of Machine Learning for Autonomous Systems (AMLAS),[13] as well as Nagy's Level of Rigor framework for Artificial Intelligence Development (LOR).[14] These methodologies provide concrete, practice-oriented frameworks for constructing safety cases and assurance artefacts, and can be seen as specific instantiations of the systems engineering perspective advocated here.

SACE operates at the whole-system level, providing a structured process for developing a safety case for complex autonomous systems in their operational context. It emphasises the need to consider the system, its environment, and their interactions holistically, and to generate a coherent set of artefacts that together justify safety. This aligns directly with focus here on assuring the wider engineered system that uses AI, rather than treating AI components in isolation. The emphasis in this paper on explicit argument structures, evidence chains, and lifecycle assurance is consistent with SACE's approach to constructing and maintaining a defensible system-level safety case.

---

[12] Hawkins, R., Osborne, M., Parsons, M., Nicholson, M., McDermid, J. and Habli, I. (2022) *Guidance on the Safety Assurance of Autonomous Systems in Complex Environments (SACE)*. University of York, Assuring Autonomy International Programme.
[13] Hawkins, R., Habli, I., Kelly, T., McDermid, J. and others (2021) *Assurance of Machine Learning for Autonomous Systems (AMLAS)*. York: Assuring Autonomy International Programme, University of York.
[14] Nagy, B. (2022) *Level of Rigor for Artificial Intelligence Development*. NAWCWD Technical Publication 8864, Naval Air Warfare Center Weapons Division.

AMLAS complements this by addressing the machine learning component level, defining a lifecycle for the development and assurance of ML models, including stages such as assurance scoping, safety requirements, data management, model learning, verification, and deployment. This closely matches the treatment here of AI-specific challenges such as model uncertainty, data dependence, and continuous change. In particular, structured argumentation via GSN, hazard modelling via Bowtie Diagrams, and iterative compliance assessment via the V-model are practical mechanisms for generating and maintaining the evidence and reasoning required at each stage of the AMLAS lifecycle.

LOR provides a complementary, development-stage-oriented perspective, defining 14 assurance tasks applied across five stages – requirements, architecture, algorithm design, algorithm code, and test and evaluation – with increasing levels of rigor corresponding to increasing confidence in deployed system behaviour. This aligns closely with the emphasis here on lifecycle assurance and iterative compliance assessment. In both cases, confidence in system behaviour is directly related to the degree of rigor applied across development stages, reinforcing the principle that assurance must be embedded within the engineering lifecycle rather than applied retrospectively.

Across all three methodologies, there is a shared emphasis on traceability, structured artefacts, and explicit reasoning. SACE and AMLAS rely on structured safety cases and evidence chains, while LOR requires systematic documentation and traceability between requirements, data, algorithms, and test outcomes. This is consistent with the focus here on generating audit-ready assurance artefacts that link claims, evidence, and reasoning in a traceable and reviewable form. In all cases, assurance is achieved through the disciplined creation and maintenance of engineering artefacts, rather than through high-level governance processes alone.

These methodologies also reflect a common recognition of AI-specific challenges, including data dependence, uncertainty, model variability, and the need for explicit treatment of assumptions and operational context. The mechanisms described in this paper – GSN for exposing assumptions and argument structure, Bowtie Diagrams for modelling hazard pathways and control effectiveness, and the V-model for iterative lifecycle assurance – provide general-purpose techniques that align with and complement the more specific practices defined in SACE, AMLAS, and LOR.

The primary differences are in scope and level of abstraction. SACE and AMLAS are focused on autonomous and ML-enabled systems, providing detailed guidance and patterns for safety assurance, while the LOR framework is a prescriptive, acquisition-oriented methodology developed for defence contexts, with defined tasks and confidence levels. By contrast, this paper presents a more generalised systems engineering perspective on AI assurance, applicable to any AI-enabled system and aligned with multiple governance frameworks. These methodologies can therefore be understood as concrete instantiations of the generic principles described here, demonstrating how lifecycle-based rigor, traceability, and structured assurance artefacts can be operationalised in practice.

Taken together, SACE, AMLAS, and LOR provide strong supporting evidence for this paper's central claim: that effective AI assurance requires the integration of governance,

engineering, and assurance through structured, lifecycle-oriented artefacts and processes. They demonstrate that the combination of explicit safety arguments, rigorous lifecycle practices, and systematic evidence generation is both feasible and necessary for achieving high levels of confidence in AI-enabled systems.

The systems engineering techniques described in the following sections provide general-purpose mechanisms for implementing these principles across a full range of AI-enabled systems.

5. The Systems Engineering Approach to AI Assurance

Systems Engineering is a disciplined, interdisciplinary approach to designing, integrating, and managing complex systems throughout their lifecycle. It focuses on defining requirements, understanding interactions between components, and ensuring that the system as a whole delivers its intended outcomes safely, reliably, and efficiently. Rather than treating elements in isolation, it considers the full context – technical, human, operational, and environmental – and uses best practice methods to balance trade-offs, manage risk, and maintain coherence from concept through design, operation, and change.[15]

A Systems Engineering approach to AI risk focuses on **assuring the safety of wider engineered systems that use AI** in a traceable and systematically reviewable way. The emphasis is on the formal artefacts produced by safety engineering work: argument structures, evidence chains, bowtie diagrams, compliance assessments, and justification that risks are ALARP (As Low As Reasonably Practicable). The Systems Engineering techniques listed below provide practical mechanisms that use these artefacts to address the challenges of AI-enabled systems - challenges that are not fully addressed by traditional assurance approaches, such as uncertainty, dynamic outcome generation, model dependence on data, and continuous change. Each approach is described in more detail in the sections that follow.

Argument Explanation via Goal Structuring Notation supports the explicit handling of uncertainty and model assumptions by requiring that all claims are justified with evidence and framed within clearly defined context, assumptions, and justifications. This makes implicit assumptions about training data, model behaviour, and operational conditions visible and open to challenge, enabling more robust and defensible safety arguments.

Risk Mitigation via Bowtie Diagrams provides a means to model machine learning failure modes within a system context. By linking threats, top events, and consequences to preventive and mitigative barriers, it allows ML-specific risks (such as model misclassification, data drift, or adversarial inputs) to be represented as part of a structured hazard and control model, with explicit consideration of how controls may fail or degrade.

Iterative Compliance Assessment via the V-model enables continuous assurance under model change. As AI models are updated, retrained, or exposed to new data, the corresponding assurance artefacts can be re-evaluated and revalidated at each stage of the

[15] https://www.nasa.gov/reference/2-0-fundamentals-of-systems-engineering

lifecycle. This ensures that changes in model behaviour are systematically reflected in requirements, design decisions, verification activities, and safety arguments, maintaining a coherent and up-to-date assurance case over time.

Together, these techniques provide a lifecycle-oriented approach to addressing the distinctive risks of AI systems, by embedding uncertainty management, failure modelling, data quality issues, and continuous assurance within standard Systems Engineering practice.

5.1. Argument Explanation via Goal Structuring Notation

AI assurance frameworks typically require safety management and documentation, but do not themselves explain how to expose the internal logic of a safety argument in a way that shows weak claims, hidden assumptions, or missing evidence. A well-established safety engineering notation for this purpose is the diagrammatic Goal Structuring Notation (GSN).[16]

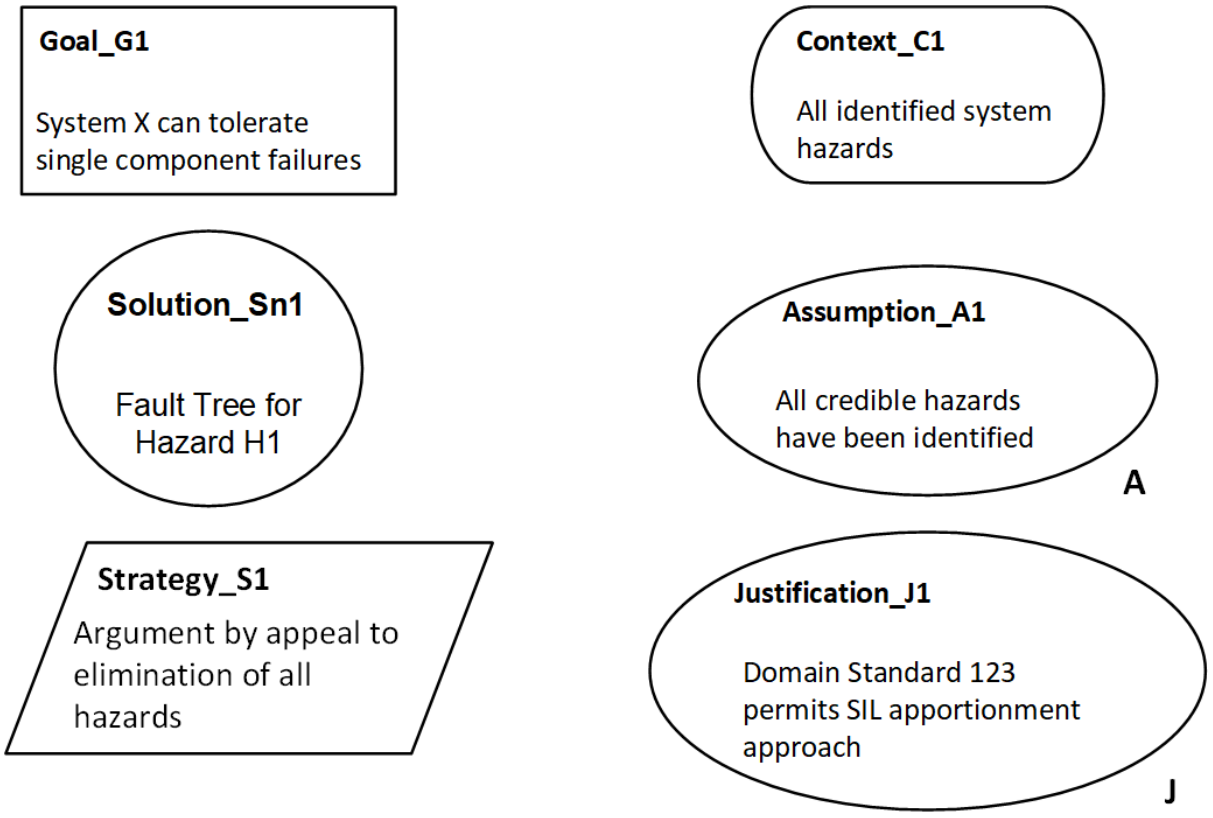

Figure 1: Goal Structuring Notation (GSN)

GSN makes explicit the reasoning chain that justifies safety, by structuring it as a clear, visual argument:

- **Goal (Claim):** A statement that must be justified - e.g., “System is safe for operation in environment X”

[16] Hawkins, R.D.; Kelly, T.P., ibid.

- **Strategy:** The reasoning approach used to break the claim down - e.g., “Argument by hazard analysis and mitigation”
- **Sub-goals:** Decomposed claims that collectively support the top-level goal - e.g., “All hazards identified”, “All risks reduced to ALARP”
- **Solutions (Evidence):** Concrete artefacts that support each claim - e.g., test reports, analyses, inspection records
- **Context / Assumptions / Justifications:** The conditions under which the argument holds - e.g., operating envelope, design assumptions, regulatory basis

GSN turns a safety case from a document into a structured, inspectable argument:

- You can trace every claim to evidence
- You can see exactly how conclusions are derived
- You can challenge the logic, not just the content

Because the argument is explicit, problems become visible:

- Weak claims: goals with insufficient or low-quality evidence
- Hidden assumptions: unstated conditions that should be explicit
- Missing evidence: nodes with no supporting solution
- Logical gaps: unclear or unjustified strategies linking claims
- Overconfidence: conclusions not proportionate to evidence

AI assurance frameworks tell you to manage and document system safety. GSN shows whether or not your safety argument is actually valid - making the logic, assumptions, and evidence explicit, and revealing where it is weak, incomplete, or unjustified. For example, it can be used to:

- Define appropriate objectives with manageable evaluation and safe exploration with robustness to distributional shift;[17]
- Manage uncertainty about assumption validity, knowledge completeness, evidence strength, and other areas of residual doubt;[18]
- Ensure procedural regularity, fairness, and inherent accountability;[19] and
- Demonstrate use of inherently safe (interpretable and causally grounded) design, safety reserves that account for uncertainty, safe fail (reject options, escalation to humans, or manual examination when uncertainty is high), and procedural safeguards (e.g., audits, training, warnings, user-experience design, and openness of code or data where appropriate).[20]

[17] Amodei et al, ibid.
[18] Hawkins, R.D.; Kelly, T.P., ibid.
[19] Kroll et al, ibid.
[20] Varshney et al, ibid.

5.2. Risk Mitigation via Bowtie Diagrams

AI assurance frameworks typically require that risks are assessed and mitigated but do not explain how controls interact across the full hazard pathway. A well-established method of risk management in Systems Engineering is Bowtie Diagrams.[21]

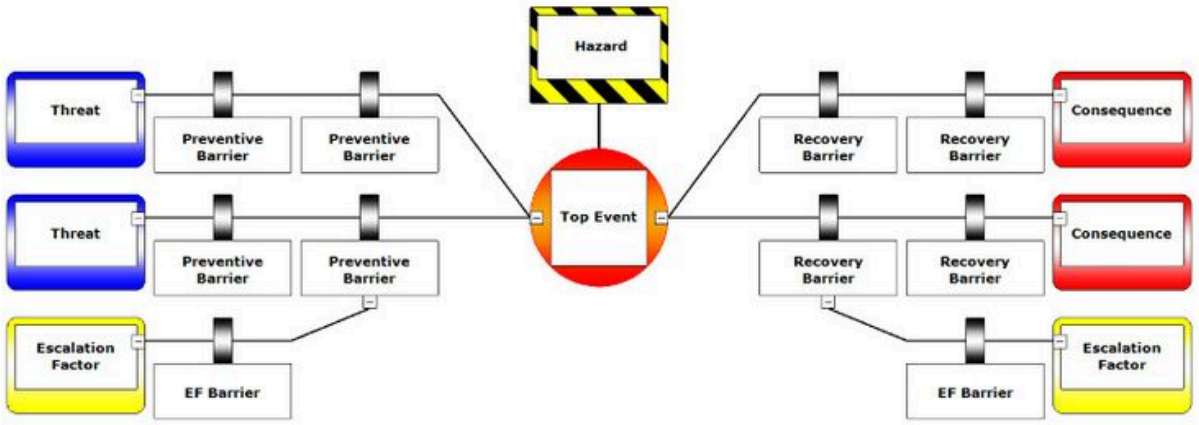

Figure 2: A Bowtie Diagram

Bowtie Diagrams turn risks into an explicit hazard logic model, showing how hazards arise, how they escalate, and how they are controlled, with threats, consequences, barriers, and escalation factors all tied back to evidence.

At the centre of a bowtie is the **Top Event** – the moment control over a hazard is lost. From there, the model expands in two directions:

- On the left side, **Threats (causes)**. What could lead to the top event - e.g., component failure, human error, environmental conditions.
- On the right side, **Consequences (outcomes)**. What could happen if the top event occurs - e.g., injury, system loss, environmental damage.

Between these are barriers (controls):

- On the left side, **Preventive barriers**. Stop threats from causing the top event - e.g., design safeguards, procedures, alarms.
- On the right side, **Mitigative barriers**. Reduce the severity of consequences - e.g., containment systems, emergency response.

Overlaying this are:

- **Escalation factors.** Conditions that weaken or defeat barriers - e.g., maintenance failure, operator fatigue.
- **Escalation controls.** Measures that protect the barriers themselves - e.g., inspections, redundancy, training

[21] de Ruijter, A. and Guldenmund, F.W. (2016). “The bowtie method: A review.” Safety Science, 88, 211–218.

A Bowtie Diagram transforms identified and treated risks into a clear, testable model of control in which:

- You can see exactly how each hazard is managed end-to-end
- You can verify that every threat has a barrier
- You can check that barriers are not vulnerable to escalation
- You can link each control to evidence of effectiveness
- How weaknesses are exposed

Because the structure is explicit, gaps become obvious:

- Uncontrolled threats: no preventive barrier
- Unmitigated consequences: no recovery measures
- Single-point failures: over-reliance on one barrier
- Fragile controls: barriers vulnerable to escalation factors
- Evidence gaps: controls not supported by test, inspection, or analysis

Bowtie is widely used in high-hazard industries (oil & gas, rail, aviation, defence) because it enables:

- Holistic risk visibility – from cause to consequence in one model
- Barrier assurance – focus on whether controls are effective, not just present
- Operational relevance – directly links to procedures, monitoring, and maintenance
- ALARP justification – shows how risk reduction is achieved and where further measures may be needed

AI assurance frameworks tell you to identify and mitigate risk. Bowtie Diagrams show whether hazards are actually controlled, with how and where the system can still fail. They can be used to avoid:

- Negative side effects, reward hacking, unscalable supervision, unsafe exploration, and robustness to distributional shift[22]; as well as
- The traps of framing (the failure to model the full system over which a social criterion like fairness is supposed to hold), portability (the mistaken assumption that a fairness solution developed in one context can be transferred cleanly into another), formalism (the belief that social concepts can be fully specified by a technical formalisation), ripple effects (the way interventions in one part of a sociotechnical system can create downstream effects elsewhere), and solutionism (the tendency to prefer narrow technical fixes to deeper process or institutional change).[23]

5.3. Iterative Compliance Assessment via the V-Model

AI assurance frameworks require accountability, documentation, and traceability. In practice, this is realised by engineers translating high-level requirements into audit-ready assurance artefacts with evidence-backed reasoning and source-level provenance. These artefacts – safety cases, compliance matrices, verification reports, hazard logs, and assurance

[22] Amodei et al, ibid.
[23] Selbst et al, ibid.

arguments – are what regulators and assessors actually review to determine whether a system is acceptable.

From engineering documents and data, the artefacts summarise:

- **Claims** – what is being asserted (e.g. system is safe, requirement is met)
- **Evidence** – tests, analyses, inspections, simulations
- **Reasoning** – how the evidence supports the claim
- **Traceability** – explicit links from claims to evidence, and back to requirements, hazards, and standards

Crucially, this is not just informal documentation – it is formal justification that must withstand scrutiny. The process is inherently complex because:

- **Standards are interpreted, not executed.** Regulations (e.g. IEC 61508, ISO 26262, Def Stan) define what must be achieved, not exactly how
- **Context matters.** The same requirement can be satisfied differently depending on system architecture, operational environment, and risk profile.
- **Evidence is heterogeneous.** Data comes from multiple sources – models, tests, logs, expert judgement – with varying levels of confidence.
- **Judgement is required.** Engineers must decide whether evidence is sufficient, relevant, and proportionate (e.g. ALARP).
- **Assurance evolves over time.** Changes to the system require reassessment of impacted claims, evidence, and compliance status.

There is no single template for compliance because:

- Different domains prioritise different hazards and controls
- Different regulators expect different forms of argument and evidence
- Different systems require different levels of rigour and independence
- Different lifecycle stages (concept, design, operation) require different artefacts

As a result, assurance of compliance is case-specific, iterative, and judgement-driven, not a checklist exercise. This requires skill, experience, and effort. Engineers deliver the traceability and accountability mandated by policies, regulations, and standards through artefacts that are interpreted, justified, and defended in context.

This is not a one-off exercise, but continuous throughout the lifecycle of an engineering project. The standard Systems Engineering way to show how safety and compliance assurance should be built into engineering development rather than bolted on at the end is the V-Model.[24]

[24] Mooz, H. and K. Forsberg. 1991. "The Relationship of Systems Engineering to the Project Cycle", Joint Conference of INCOSE and the American Society for Engineering Management, 21-23 October 1991, Chattanooga, TN.

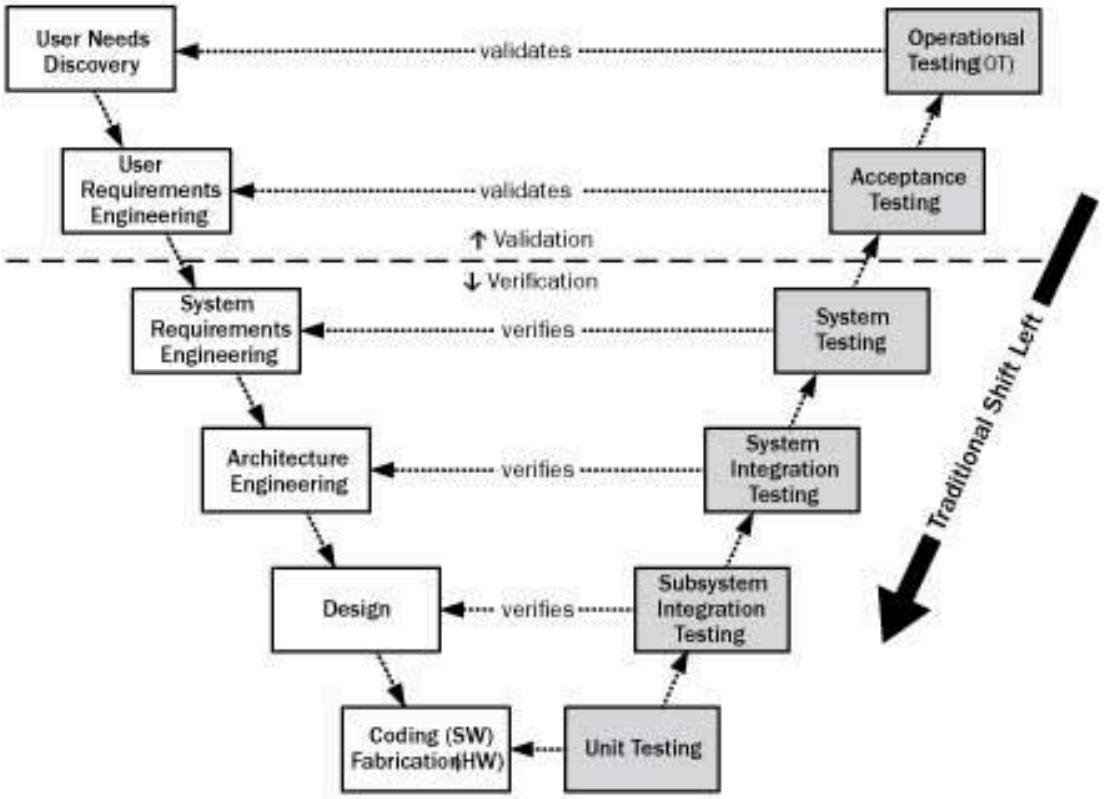

Figure 3: The Systems Engineering V-Model

On the left-hand side, the system is progressively defined and decomposed: stakeholder needs and the product concept are translated into system requirements, then into architecture and detailed design. On the right-hand side, implemented elements are assembled and integrated back into higher-level aggregates, with verification and validation activities demonstrating that each level meets its specifications and, ultimately, that the overall system satisfies stakeholder needs. The point of the V is to make the logic of assurance explicit: what is defined on the left must be proven on the right, and verification planning should be established during requirements and design, not improvised during testing.

In modern Systems Engineering, the V-model is applied iteratively: artefacts are refined as understanding improves, and assurance evidence is repeatedly re-established as change occurs. This supports a practical pattern for document summarisation and compliance assessment at every stage of development. At each stage, summarisation distils the current engineering baseline into audit-ready assurance artefacts (claims, evidence, reasoning, and traceability). Compliance assessment then evaluates those artefacts against applicable policies, regulations, and standards to identify:

- Areas of potential non-compliance,
- Conditions for compliance (what must be demonstrated or controlled), and
- Mitigations to close gaps.

These outputs can then not only be used to revise the current stage's artefacts but also carried forward as explicit constraints and work packages for the next stage. This ensures that requirements, design decisions, implementation tasks, and verification activities are all driven by a continuously updated understanding of what is needed for compliance.

This lifecycle-wide use of compliance to drive engineering can be applied to AI-enabled systems to make them "Safe by Design", in the same way that the UK Government requires systems to be "Secure by Design" (SbD).[25] In an SbD approach, safety controls and safety justification are integrated from the earliest concept and maintained continuously as the system evolves, rather than being treated as an afterthought or an end-stage compliance exercise.

SbD explicitly frames assurance as a continuous activity across the service life cycle, including continuous assurance and safe handling of change. The same framing applied to safety means that compliance conditions and mitigations are identified early, embedded into requirements and design, and then systematically evidenced through implementation, integration, testing, and operation. The result is a proactive, repeatable, and auditable assurance trajectory - not merely demonstrating safety at a gate, but:

- Continuously sustaining defensible safety arguments as AI systems change in data, models, context, and use;[26]
- Using the evolution of safety arguments to empower optimally effective and efficient design decisions; and
- **I**mplementing end-to-end algorithmic audits that distinguish engineering reliability from social acceptability;[27]

6. Illustrative Operational Examples

Given below are two examples of how this approach can be applied, illustrating it with reference to AI models created for Smart City Flood Risk and Air Defence Decision Support. The techniques referred to above, and described in the examples below, are also demonstrated in worked examples, for which links to video demonstrations are provided.

6.1. Smart City Flood Risk

As an example of this approach, consider an AI-enabled flood-risk model used by a city operations centre to support drainage management, pump activation, road-closure decisions, and the prioritisation of infrastructure inspections. This is a realistic smart-city use case: AI and real-time urban data are increasingly used to improve services such as water management, mobility, and wider infrastructure operations, but their effectiveness depends heavily on data quality, reliability, and the management of interdependent urban systems.

The model has been trained on historical rainfall, water-level, drainage, land-use, and traffic data, and validated against a defined urban operating context. During deployment, however, the city experiences a change in conditions: new building development alters runoff patterns,

[25] https://www.security.gov.uk/policy-and-guidance/secure-by-design/principles
[26] Calinescu et al, ibid.
[27] Raji et al, ibid.

several sensors are replaced, drainage assets deteriorate, and unusually intense local storms become more frequent. The model continues to operate, but the live input distribution no longer matches the assumptions under which its safety and performance claims were originally justified. This is not simply a forecasting issue. It is an assurance issue, because the operational context has changed in a way that may invalidate the original case for safe and reliable use.

In GSN terms, data drift weakens the argument, not just the model. A top-level claim such as “the flood-risk model is acceptably safe for operational use in city environment X” depends on assumptions about the representativeness of training and verification data, the stability of the built environment, the reliability of sensors, and the validity of intervention thresholds. When drift emerges, those assumptions become open to challenge. Evidence that previously supported the claim – validation results, test reports, and acceptance criteria – may remain factually correct, but no longer be sufficient for the current context. GSN makes this visible by exposing weakened sub-claims, challenged assumptions, and stale evidence, showing exactly where the safety argument has become fragile and what new evidence is required to restore confidence.

In Bowtie terms, the hazard is not “use of AI” in the abstract, but a specific loss of control such as “mis-prioritised flood risk presented to city operators.” On the threat side, causes may include changed runoff behaviour, degraded or mis-calibrated sensors, altered drainage capacity, corrupted feeds, or retraining on unrepresentative data. Preventive barriers may include drift detection, sensor-health monitoring, challenge datasets, configuration control, independent model review, and threshold-based human oversight. On the consequence side, the effects may include delayed pump activation, failure to close a vulnerable road, missed inspection of a critical asset, avoidable local flooding, or erosion of trust in the city’s decision-support system. Mitigative barriers may include manual override, degraded-mode operation, reversion to simpler rule-based thresholds, temporary withdrawal of the model from service, and rapid revalidation using updated urban data. Bowtie therefore shows not only that a risk exists, but how drift creates a concrete hazard pathway and where controls may already be weak or vulnerable to escalation.

In V-model terms, drift is treated as a lifecycle assurance event. On the left-hand side, requirements and architecture should already have defined the model’s operating envelope, monitoring thresholds, retraining triggers, data-quality constraints, fallback modes, and verification strategy. On the right-hand side, operational monitoring, regression testing, and re-verification are used to determine whether the original claims still hold. If they do not, the outputs of compliance assessment become actionable engineering inputs: the affected artefacts are identified, areas of non-compliance are made explicit, conditions for continued use are defined, and mitigations are assigned into the next cycle of work. These may include narrowing the operating domain, increasing operator review, collecting new data, revising hazard controls, updating verification cases, and refreshing the safety argument before full operational confidence is restored.

This example shows why a systems engineering approach is operationally useful in Smart Cities contexts. What first appears to be a model-performance issue is, in practice, a change to the validity of the safety case. GSN shows where the argument has weakened, Bowtie shows how the hazard pathway has opened, and the V-model ensures that the response is

structured, traceable, and built back into the lifecycle rather than handled as an ad hoc patch.

6.2. Air Defence Decision-Support

As a further example of this approach, consider an AI-enabled decision-support model used to prioritise sensor tracks for analyst review in an air-defence or Intelligence, Surveillance, and Reconnaissance context. The model has been trained on historical signature data and validated against a defined operational envelope. During deployment, however, it begins to encounter a new pattern of inputs caused by adversary adaptation, novel environmental clutter, seasonal weather effects, and changes in own-sensor configuration. The model continues to operate, but its live behaviour no longer matches the assumptions under which its safety and performance claims were originally justified. This is not simply a technical accuracy issue. It is an assurance issue, because the operational context has changed in a way that may invalidate the original case for safe use.

In GSN terms, data drift weakens the argument, not just the model. A top-level claim such as “the decision-support model is acceptably safe for operational use in environment X” depends on context and assumptions about the representativeness of training and verification data, the stability of the operating environment, the validity of performance thresholds, and the effectiveness of human oversight. When drift emerges, those assumptions become open to challenge. Evidence that previously supported the claim – test reports, validation results, and acceptance criteria – may remain factually correct, but no longer be sufficient for the current context. GSN makes this visible by exposing weakened sub-claims, challenged assumptions, and missing or stale evidence. The effect is to show precisely where the safety argument has become fragile and what new evidence is required to restore confidence.

In Bowtie terms, the hazard is not “use of AI” in the abstract, but a specific loss of control such as “mis-prioritised threat assessment presented to the operator.” On the threat side, causes may include adversary signature change, degraded sensors, altered data-fusion inputs, corrupted metadata, or retraining based on unrepresentative data. Preventive barriers may include drift detection, confidence thresholds, challenge datasets, independent model verification, configuration control, and operator training. On the consequence side, the effects may include delayed review of a genuine threat, unnecessary escalation of benign tracks, unsafe allocation of scarce defensive resources, or loss of trust in the decision-support chain. Mitigative barriers may include mandatory analyst confirmation, degraded-mode operation, reversion to simpler rules, withdrawal of the model from service, or rapid data refresh and revalidation. Bowtie therefore shows not only that a risk exists, but how drift creates a concrete hazard pathway and where controls may already be weak or vulnerable to escalation.

In V-model terms, drift is treated as a lifecycle assurance event. On the left-hand side, requirements and architecture should already have defined the model’s operating envelope, monitoring thresholds, retraining triggers, data-quality constraints, fallback modes, and verification strategy. On the right-hand side, operational monitoring, regression testing, and re-verification are used to determine whether the original claims still hold. If they do not, the outputs of compliance assessment become actionable engineering inputs: the affected

artefacts are identified, areas of non-compliance are made explicit, conditions for continued use are defined, and mitigations are assigned into the next cycle of work. These may include narrowing the operating domain, increasing human review, collecting new data, revising hazard controls, updating verification cases, and refreshing the safety argument before full operational confidence is restored.

This example shows why a systems engineering approach is operationally useful in Defence contexts. What first appears to be a model-performance issue is, in practice, a change to the validity of the safety case. GSN shows where the argument has weakened, Bowtie shows how the hazard pathway has opened, and the V-model ensures that the response is structured, traceable, and built back into the lifecycle rather than handled as an ad hoc patch. The result is not merely a better model, but a more defensible operational decision about whether, how, and under what constraints the model should continue to be used.

6.3. Worked Examples of the Techniques

Worked examples of implementing each of these techniques in engineering practice have been created, using tool-supported approaches to streamline preparation of the assurance artefacts.[28] Video demonstrations of the examples are available online:

- Argument Explanation via Goal Structuring Notation[29]
- Risk Mitigation via Bowtie Diagrams[30]
- Iterative Compliance Assessment via the V Model[31]

7. Limitations and Challenges

While Systems Engineering techniques provide a structured and practical approach to AI assurance, their application to AI-enabled systems introduces several challenges that must be recognised.

First, **machine learning systems may lack stable specifications**. Traditional Systems Engineering assumes that system behaviour can be defined and verified against explicit requirements. In contrast, ML models are often specified indirectly through training data and performance metrics, and their behaviour may evolve over time as data, models, or operating contexts change and Agentic AI systems set their own goals.[32] [33] This makes it more difficult to define precise claims and acceptance criteria within assurance artefacts such as GSN.

Second, **evidence for AI systems is frequently statistical rather than deterministic**. Verification activities may rely on testing across datasets, simulation, or probabilistic performance measures rather than definitive proof of correctness. As a result, assurance arguments must accommodate uncertainty, confidence levels, and potential gaps in

[28] https://dedoctive.ai/#safety-navigator
[29] https://youtu.be/ors615rN7Kk?si=qYvYmgwjMy2QesXT
[30] https://youtu.be/z6lDaNdC7lQ?si=QcexrI1aAh9WB4tb
[31] https://youtu.be/Id3bufZfwXc?si=pLn3I2HR2tdMoqaP
[32] Amodei et al., ibid.
[33] Varshney, ibid.

coverage, requiring careful judgement about what constitutes sufficient and proportionate evidence.[34]

Third, **assurance artefacts may become complex at scale**. Large AI-enabled systems can generate extensive networks of claims, evidence, hazards, and controls. While structured representations such as GSN and Bowtie improve transparency, they can also introduce challenges in terms of manageability, navigation, and cognitive load. Maintaining clarity and usability of these artefacts over time requires disciplined structuring and appropriate tooling, a challenge also recognised in broader work on assurance cases and socio-technical systems.[35]

Finally, **effective application depends on tooling and expertise**. The creation, maintenance, and interpretation of assurance artefacts require both domain knowledge and familiarity with Systems Engineering methods. Without appropriate tools to support traceability, versioning, and automated analysis, and without skilled practitioners to interpret results, the approach may become resource-intensive. This aligns with findings in the algorithmic accountability literature, which emphasises the organisational and procedural demands of meaningful auditability.[36] [37]

These challenges do not undermine the value of a Systems Engineering approach to AI assurance, but they highlight the need for **careful adaptation, appropriate tooling, and skilled application**. In particular, they reinforce the importance of treating assurance as an iterative, judgement-driven process that evolves alongside the system it is intended to justify.

8. Discussion

AI assurance frameworks provide essential governance, but governance alone does not guarantee that AI-enabled systems are safe, that risks are understood, or that decisions are defensible. A Systems Engineering approach closes this gap by operationalising those frameworks through structured, inspectable assurance artefacts - making the reasoning, evidence, and residual risk explicit.

Techniques such as Goal Structuring Notation, Bowtie Diagrams, and Iterative Compliance Assessment move assurance from documentation to justification, enabling engineers to expose weaknesses, validate controls, and maintain ALARP arguments as AI-enabled systems evolve.

Together, these approaches are complementary: frameworks define what must be achieved, while Systems Engineering provides tried-and-tested techniques for ensuring that it is achieved. In an era of rapidly evolving AI-enabled systems, this combination is critical to delivering accountable, trustworthy, and practically assured outcomes at the pace required.

Proven methodologies are available to harness Systems Engineering techniques for AI assurance in a structured manner - in particular, SACE / AMLAS and LOR. However,

[34] Calinescu et al., ibid.
[35] Selbst et al., ibid.
[36] Raji et al., ibid.
[37] Kroll et al., ibid.

significant time and resources are required to implement either of these approaches. Further, neither methodology is complete in itself. Rather, they complement each other, with SACE / AMLAS providing a systems perspective in which to embed the detailed ML techniques of LOR.

In 2022, the author worked with Richard Hawkins, Bruce Nagy, and others to start fusing AMLAS and LOR into a comprehensive approach to AI assurance via Systems Engineering. It was not possible to complete this work in the time available, but the output (see Figure 4 below) illustrates the potential for such a fusion. It should be noted that a fused methodology would likely require even more time and resources to implement than AMLAS or LOR alone.

For organisations or projects that find themselves under pressure to deploy AI-enabled systems without having the time or resources to implement a holistic methodology, this paper offers techniques that require a more manageable level of effort:

- **Argument Explanation via GSN** can be used to define appropriate objectives with manageable evaluation and safe exploration with robustness to distributional shift;[38] manage uncertainty about assumption validity, knowledge completeness, evidence strength, and other areas of residual doubt;[39] ensure procedural regularity, fairness, and inherent accountability;[40] and demonstrate use of inherently safe design, safety reserves that account for uncertainty, safe fail, and procedural safeguards.[41]
- **Risk Mitigation via Bowtie Diagrams** can be used to avoid negative side effects, reward hacking, unscalable supervision, unsafe exploration, and robustness to distributional shift[42] as well as the traps of framing, portability, formalism, ripple effects, and solutionism.[43]
- **Iterative Compliance Assessment** can be used to update assurance cases as the system changes;[44] evolve safety arguments to empower optimally effective and efficient design decisions; and implement end-to-end algorithmic audits that distinguish engineering reliability from social acceptability.[45]

Taken together, these approaches provide some of the benefits of methodology implementation for significantly lower cost.

[38] Amodei et al, ibid.
[39] Hawkins, R.D.; Kelly, T.P., ibid.
[40] Kroll et al, ibid.
[41] Varshney et al, ibid.
[42] Amodei et al, ibid.
[43] Selbst et al, ibid.
[44] Calinescu et al, ibid.
[45] Raji et al, ibid.

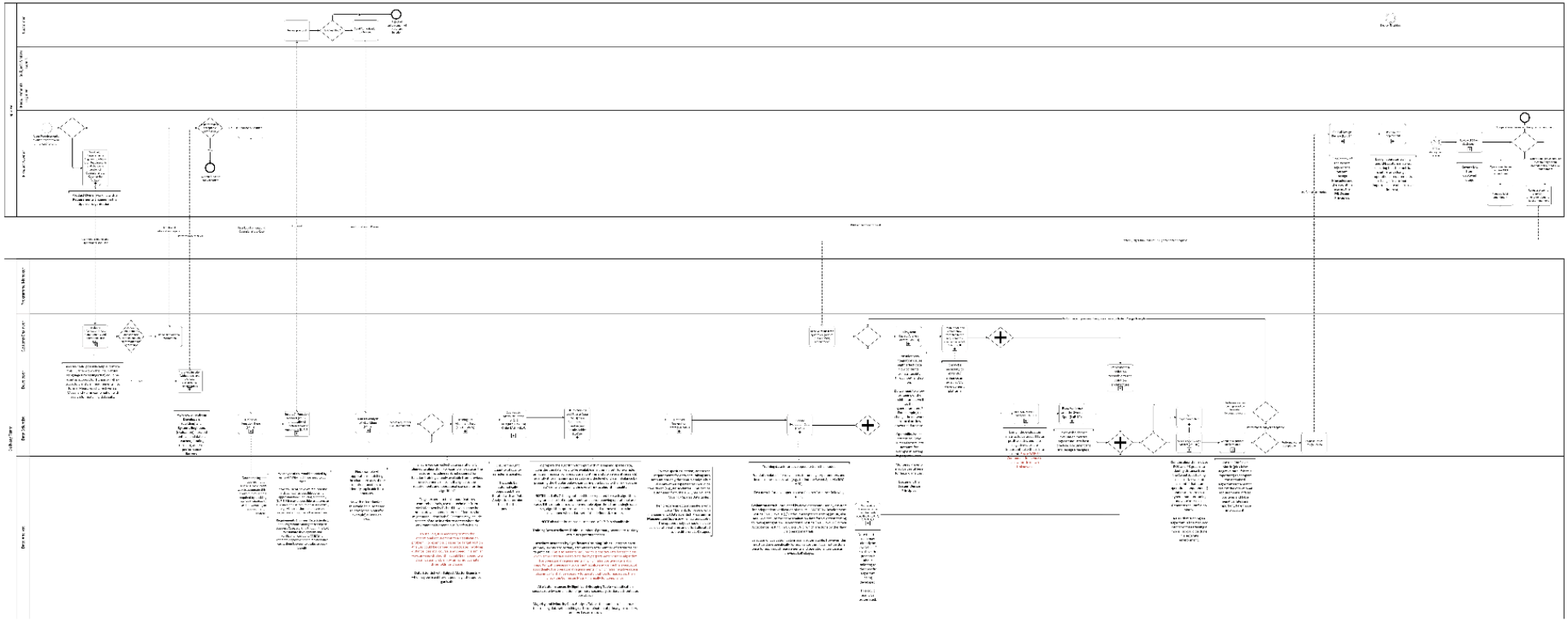

Figure 4: A Partial Fusion of AMLAS and LOR (Harrison-Broninski, K., Nagy, B., Hawkins, R., 2022)

9. Conclusions

This paper does not recommend a piecemeal approach over mature, systematic approaches such as SACE / AMLAS and LOR. Such methodologies represent the distillation of expert thought and practice, and typically deliver significant return on investment. Where sufficient time and resources are available to implement an engineering methodology for AI assurance, this paper describes practical techniques that can support implementation by streamlining and optimising production of key assurance outputs.

In other cases, AI-enabled systems may be developed under pressure to deliver at scale and pace with only limited effort. If real-world constraints mean the investment required for methodology adoption is unavailable, this paper proposes practical techniques that can help mitigate the associated assurance risks.

Either way, using the key Systems Engineering techniques outlined in this paper will enable organisations to demonstrate defensible decisions, manage evolving risk, and maintain accountability chains in AI-enabled systems. This “Safety by Design” approach embeds safety considerations into AI development as a primary driver. By prioritising holistic real-world concerns from the start, this approach reduces the need for rework at later stages, so rather than generating additional costs is instead likely to **increase the efficiency and effectiveness of systems development**. Further, it will have a deep positive social impact that goes far beyond AI.

The development of any new science and technology, including but not limited to the AI-enablement of infrastructure, defense, and other critical systems, should not be treated as ethically inevitable. The claim that “if we do not do it, someone else will” does not remove responsibility from researchers, engineers, or industrial decision-makers; it merely shifts attention away from the conscious choices they still make about purpose, acceptable harm, operating boundaries, and conditions of use. In this sense, accountable innovation requires more than following market incentives or strategic pressure. It requires explicit judgement about which capabilities should be pursued, under what constraints, and with what evidence that their benefits justify their risks.

Safety engineering can provide much of the practical discipline needed to support such judgement. It does not replace ethics as a moral philosophy, but it can **operationalise ethical intent** by forcing developers to make claims, assumptions, hazards, mitigations, evidence, and residual risks explicit and reviewable. Structured argumentation makes the case for acceptability visible; hazard analysis identifies who or what may be harmed and how; lifecycle assurance ensures that decisions are revisited as systems and contexts change.

In this way, safety engineering can serve as the operational backbone of an ethical framework for science and technology development: not by deciding values on behalf of society, but by turning ethical commitments into disciplined, auditable engineering practice.

10. Author

Keith Harrison-Broninski is an author, researcher, and keynote speaker specialising in cross-boundary collaboration, community antifragility, and technology for good. He has led social enterprises that won awards from Gartner and the NHS. Keith’s first book "Human Interactions" (2005) was described by Information Age as "the overarching framework for 21st century business technology". Keith's other books include two for Springer and three for the Workflow Management coalition. Keith’s most recent book "Supercommunities" (2021) was described by the Chief Executive of the RSA in his foreword as “authoritative and highly readable” and by Vint Cerf, co-inventor of the Internet, in a second foreword, as “a path away from social and economic meltdown”. Keith currently focuses on using trustworthy AI to create apps for the “Internet of Communities”.