Bridging the Valley of Death: How Assurance Takes Us from Proof of Concept to Minimum Viable Product

Abstract

Between proof of concept (PoC) and minimum viable product (MVP) lies the infamous *Valley of Death*, where promising innovations falter before reaching the market. While technical maturity is often cited as the barrier, the real gap is just as much about *assurance* as engineering. Customers, regulators, investors, and the public all demand confidence that a product is safe, secure, reliable, and responsibly managed.

This blog explores assurance cases as a structured way of bridging that gap. By framing claims, arguments, and evidence, we can move beyond tacit assurance and activity-based requests ("just do a pentest") toward explicit, defensible cases. I propose treating vulnerability management outputs as **assurance fragments**, each finding linked to risk treatment and embedded in a broader assurance framework. In cyber-physical domains such as automotive, heavy goods vehicles, and OT systems in energy, this approach represents an innovation: bringing structure to what is often a patchwork of reports and risk registers.

The result is not just better compliance or more persuasive audits. It is a pathway to building products that cross the Valley of Death with trust intact, products where assurance is visible, living, and as essential as the engineering itself.

ncc group

Bridging the Valley of Death: How Assurance Takes Us from Proof of Concept to Minimum Viable Product

Introduction

I am, in many ways, a product of the UK's engineering doctoral process (EngD). Innovation was not presented to me as a corporate slogan or a line on a strategy slide, it was drilled into my day-to-day thinking, shaping the way I approach every problem. For me, innovation has always meant the *application of knowledge from different domains to address a real challenge*. That mindset became second nature: every new project was a chance to connect the dots across disciplines, to test, to prove, to demonstrate.

I still occasionally joke about being a "former academic, for my sins." But the truth is, that experience doesn't weigh me down, it sharpened how I work as a consultant. It trained me to think systematically, to value structured reasoning, and to search for evidence that can carry an argument. It also revealed how fragile the journey is between an exciting idea scribbled on a whiteboard and a product that survives in the world.

That fragility has a name: the *Valley of Death*. It's the chasm between a **proof of concept (PoC)** and the **minimum viable product (MVP)**; between something that *can* work in principle and something that *does* work in practice. Many ideas never make it across: some run out of money; some lose stakeholder trust; some fail to prove themselves outside the lab.

What's striking, at least to me, is that the Valley of Death isn't just about technical readiness. More often, it's about *assurance*. Customers want confidence that your product is reliable. Regulators want evidence of compliance. Investors want risk managed. The public wants to know they can trust you. These are all forms of assurance, and they matter just as much as whether the technology itself functions.

But what exactly is *assurance*? Colloquially, I like to describe it as *what lets you sleep at night*. It's the combination of confidence, structure, and evidence that makes you believe a system will behave as expected tomorrow, not just today. In more formal terms, "assurance is ***confidence*** that something will behave as ***intended***" and 'structured assurance' describes the process of making claims about a system, supporting those claims with arguments, and underpinning them with evidence. Whether you are designing an aircraft, deploying an industrial control system, or releasing a new piece of software, assurance is what convinces people, including yourself, that the risks have been thought through and are being managed responsibly.

And importantly assurance also means being *ready for when things do go wrong.* If resilience is your driver, then incident response and readiness must be part of the picture. In the real world, no system is flawless, but what lets you sleep at night is knowing that when the unexpected happens, you are prepared.

In short: this is about showing how structured assurance can be the shift that gets more ideas across the Valley of Death.

Living with the Valley of Death

The *Valley of Death*, is not just a turn of phrase. It describes a brutally familiar reality for engineers, researchers, and innovators: most ideas never make it out the other side.

On one edge, you have the **Proof of Concept (PoC):** a prototype, a demo, or a sketch on the back of a napkin that proves *this can work*. On the other, you have the **Minimum Viable Product (MVP):** something robust enough that *this does work* in practice, in the hands of real users, under real constraints.

The problem is not that the gulf is imaginary — it's that it is wide, resource-hungry, and unforgiving. Funding dries up. Organisational priorities shift. Regulators aren't convinced. Early champions lose interest. Technical brilliance alone isn't enough.

We often try to capture this journey with **Technology Readiness Levels (TRLs).** TRLs are useful shorthand:

- **TRLs 1–3:** principles, concept, early lab work.
- **TRLs 4–6:** prototypes and validation.
- **TRLs 7–9:** full systems, qualified and in service.

But TRLs mostly measure *technical readiness*. You can reach TRL 6 or even 7 with a product that works technically but is still unconvincing to customers, regulators, or the public. That's because what's missing is *assurance.*

Crossing the Valley of Death requires more than working technology. It requires building **confidence**:

- Confidence for a customer that they can **trust it**.
- Confidence for a regulator that it **meets the rules**.
- Confidence for investors that it can **survive in the market**.
- Confidence for the public that it **will not harm or betray them**.

An example: Autonomous Vehicles in the UK

A vivid example of the Valley of Death is **autonomous vehicles.** Proofs of concept have been around for years — from carefully choreographed demo cars to limited pilot schemes. Yet the journey to an MVP that genuinely works at scale has required *hundreds of millions, even billions, of pounds* in funding.

- Since 2015, the UK government and industry together have invested **around £600 million** in more than 100 Connected and Automated Mobility (CAM) projects.
- In 2023, the government announced a further **£150 million** to push the technology toward viable deployment.
- In 2025, an additional **£18 million** was awarded across 37 CAM projects.
- UKRI has also contributed **£28 million** to AI and control systems for self-driving vehicles.
- Industry players, like Wayve, have raised hundreds of millions in private capital, often alongside partnerships with global firms such as Uber.

All this funding is directed toward bridging the Valley: moving from PoCs and trial runs to regulated, commercially viable services.

But money doesn't just fill the Valley of Death, it changes the *shape* of it.

- **At its best**, sustained investment strengthens assurance. It provides the breathing room for multiple trials, iteration, and regulator engagement. It creates space for building the evidence and arguments that underpin confidence.
- **At its worst**, investment warps incentives. Government funding may prioritise industrial strategy outcomes like job creation or global competitiveness. Industry funding often chases first-mover advantage. Both can create pressure to deploy quickly and demonstrate progress, sometimes at the expense of assurance.

The lesson here is that money alone cannot guarantee trust. It may accelerate the journey, but without structured assurance it risks producing MVPs that technically function yet either:

- **fail to convince** regulators, customers, or the public when they *should* be convinced, because the evidence is fragmented or poorly presented, or
- **convince them when they shouldn't be**, because superficial demonstrations and glossy pilots hide gaps in resilience, safety, or security.

And that's the key point: **money may bridge the Valley in appearance, but assurance is what bridges it in substance.**

Assurance as the Bridge

If Technology Readiness Levels describe how technically mature a system is, a structured assurance case, written for a specific stakeholder, can be a shorthand for how *trustworthy* it is. The two are connected, but they are not the same. You can have a system at TRL 7, technically proven in a demonstration environment, that still fails to satisfy a regulator, an insurer, or the public.

This is where assurance comes in. At its simplest, assurance is about confidence: confidence that the system will work as intended, that risks are understood and managed, and that failures will be met with resilience. Broadly what this looks like involves:

- Capturing the explicit claims and distilling implicit claims about a system/service.
- Supporting those claims with arguments.
- Underpinning those arguments with evidence.

I use the **Claims–Arguments–Evidence (CAE)** model here because it helps tell a clear narrative. But this isn't the only way to express assurance. The same principles can be represented in **Goal Structuring Notation (GSN)** or other formal approaches. What matters isn't the diagram style — it's the discipline of thinking structurally, of making explicit what is often left implicit.

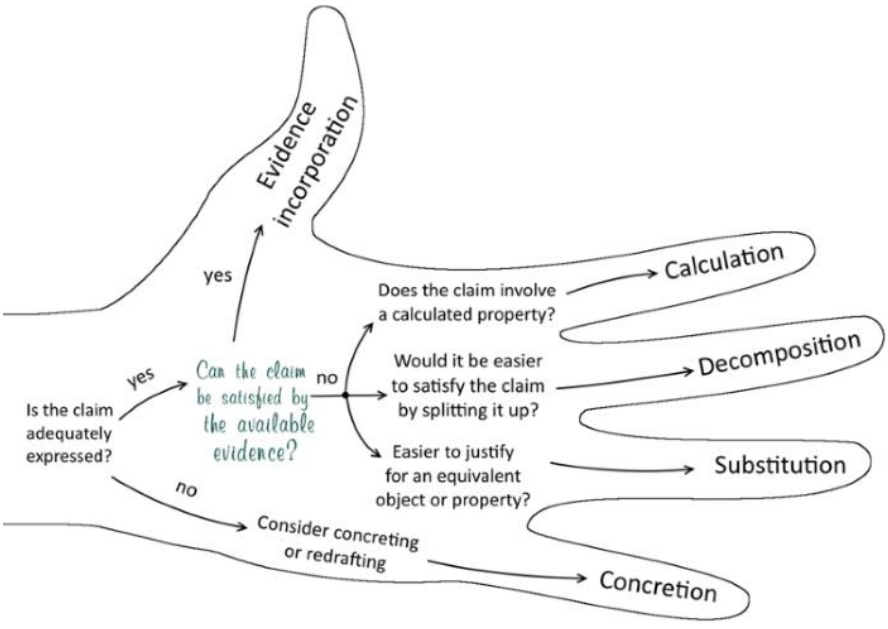

Figure 1: CAE Helping Hand (https://claimsargumentsevidence.org/notations/helping-hand/)

The key is that assurance reframes the journey across the Valley of Death. Instead of asking, *"Can we prove this works in a lab?"*, assurance asks:

- *"What do we need to claim to our customers, our regulators, our investors, and the public?"*

- *"What arguments will make those claims persuasive?"*
- *"What evidence will underpin those arguments?"*

This turns progress from a scatter of trials, engineering work products and reports into a deliberate framework. Assurance doesn't just measure whether the technology works, it considers how to build confidence in the stakeholder group that the product is safe, secure, and resilient in the real world.

In practice, that means assurance becomes the **bridge** across the Valley: the structured pathway that connects a proof of concept to a minimum viable product. It's not a compliance afterthought, bolted on when launch looms. It's a living discipline, developed in parallel with engineering, and tuned to the needs of different stakeholders.

Assurance is Not One-Size-Fits-All

A common mistake is to think of assurance as producing a single airtight argument that convinces everyone. In practice, different stakeholders respond to very different kinds of assurance, and what persuades a regulator may not persuade a customer, an investor, or the general public.

- **Customers** are often reassured by real-world demonstrations, usability trials, or endorsements showing that a product behaves reliably under normal conditions and that edge cases have been treated robustly.
- **Regulators** look for formal conformance: compliance with standards, accredited testing, and alignment with legal benchmarks.
- **Investors** typically want clarity on risk — whether that means resilience in the market, clearly defined liability, or evidence of continuity planning.
- **The public** tends to place its trust in visible signals of safety such as ethical frameworks, transparent reporting, or independent validation. Exposure to a technology/system/process can help here.

For this reason, a robust assurance case cannot follow a single linear path. It has to branch into a network of arguments, each one crafted to meet the perspective of a different audience, while still rolling up to a coherent overall claim.

The UK's **Automated Vehicles Act 2024** provides a useful illustration of how complex this can become. The Act sets a simple benchmark: automated vehicles must be at least as safe as a "careful and competent human driver." On the surface, this is an accessible comparison, easy for policymakers to articulate and for the public to understand. But it also risks narrowing the assurance conversation too far. Humans may be the obvious baseline for comparison, yet they are poor benchmarks for accountability. In accidents, human drivers are inconsistent, biased, or motivated to misattribute blame. Their accounts are rarely reliable, and this complicates insurance, liability, and compensation.

This is where automated systems have the potential not just to match human performance but to exceed it in areas where humans are inherently weak. Chief among these is **explainability**. An autonomous vehicle can provide logs, telemetry, and decision traces that make its behaviour transparent. Where a human might lie, forget, or misjudge, the AV can present a structured account of what it sensed, what options it evaluated, and why it acted as it did.

That ability to explain is not just a technical detail; it creates tangible assurance value across multiple stakeholders:

- **Insurers** can more easily calculate liability and damages.
- **Regulators** can verify that safety principles were applied consistently.
- **Courts and investigators** can attribute responsibility based on evidence, not testimony.
- **The public** can gain confidence that when accidents do happen, they will be explainable and resolvable, rather than lost in a fog of contested narratives.

This is why assurance for AVs must go beyond the language of being "as safe as a careful and competent driver." That phrase may work politically, but for assurance it misses the point. These systems should be designed, and assured, around the qualities that matter to stakeholders: not just functional safety, but explainability, attribution, and accountability. In practice, that means assurance cases that explicitly include claims about the system's ability to explain its decisions; arguments linking explainability to outcomes such as insurance, liability, and trust; and evidence in the form of telemetry logs, simulation results, incident analyses, and regulatory reviews. Only when these elements are made explicit does assurance provide a bridge that all stakeholders can rely upon.

The Tacit Case and the Pentest Trap

In many organisations, there already exists what you might call a *tacit assurance case*. It isn't that people blindly assume their product is safe, secure, or reliable. Rather, the structure of *what they believe and why* is internalised. Engineers may lean on test coverage, product managers on customer feedback, compliance staff on standards checklists, executives on incident-free track records. Each has part of the case in their head — but rarely do these parts align into a coherent, explicit whole.

This internalisation creates two problems. First, it means that different silos often talk past each other, because their arguments and evidence are not shared or harmonised. Second, when external stakeholders ask for assurance, the organisation struggles to articulate a clear case. What emerges instead is a familiar request: *"Can we get a pentest?"*

For those of us in security consultancy, this request is both common and understandable. A penetration test is often the starting point for deeper conversations, and there's nothing wrong with that. But what it really represents is a proxy: the customer knows they need evidence of security, but instead of articulating the claims and arguments they want to support, they ask for the activity they know.

Good Consulting vs. Shallow Consulting

This is where good security consulting makes a difference. Digging just a little deeper behind the initial ask can completely change the value of the final deliverable. Two pentests can be run against the same scope, with the same techniques, and yet land very differently:

- In one case, the report drops on the table and goes nowhere. It documents issues, but it doesn't resonate, because it never touched the unstated objectives of the customer.
- In the other, the consultant took time to understand the *why* behind the request. The same test was run, but the report was written to support the customer's internal assurance case, even if that case was never formally articulated. Findings were framed in the language of the customer's stakeholders, mapped to risks they actually care about, and presented in a way that made action obvious.

Both tests were technically sound. But only one *built assurance.*

The Procurement Problem

This difference is also what makes procurement so challenging. On paper, *pentest == pentest*. Company X offers a pentest, Company Y offers a pentest. Both tick the same compliance boxes. Which one should you choose? Usually, procurement falls back to price or to those who don't seem to be asking all the difficult questions.

The trouble is that two suppliers may approach the engagement in very different ways. One might simply produce a list of vulnerabilities, while another might structure the findings to support decision-making across engineering, compliance, and management stakeholders. If you don't have a way to articulate what you actually need from the engagement, beyond "run the test", then those differences are invisible.

A formal assurance framework helps solve this problem. By making the requirements explicit — even the intangible ones, like *"we need findings disseminated in a way that supports communication to non-technical stakeholders"*, you

create a basis for comparing suppliers in terms of value, not just cost. Without that, you end up in a commoditised market where a pentest is just a pentest, and the cheapest bid wins.

The Pentest Trap

This is the essence of the pentest trap. Pentesting itself isn't the problem. The trap lies in treating the pentest as *the assurance case*, when at best it should be *one piece of evidence within a case.* Left unstructured, it risks producing either under-assurance (failing to answer the right questions) or over-assurance (creating false confidence in areas the test never covered).

The way out is to make the tacit case explicit: to turn internalised, siloed reasoning into a shared structure of claims, arguments, and evidence. Once that structure exists, pentesting can be framed as exactly what it should be — a valuable form of evidence that strengthens the overall case, not a substitute for it.

Reframing Pentesting as Assurance Evidence

Penetration testing isn't the assurance case it is an activity that produces the **evidence inside** an assurance case. Treated that way, a pentest becomes powerful: it stops being a one-off report and starts strengthening a structured argument about how risks are governed.

In mature organisations the realistic, if often unwritten, claim is simple:

Claim (pragmatic): *Findings identified through penetration testing are addressed within the organisation's risk management process.*

That claim reflects the real world: there will always be vulnerabilities; what matters is **how you manage them**. Once you say this out loud, the rest of the argument practically writes itself:

- **Argument A — Scope matched the perceived threat & risk.**
  The test covered what matters, in a way that aligns with risk appetite, sector baselines, and credible threat methods.
- **Argument B — Findings were governed proportionately.**
  High-impact findings were remediated; where remediation wasn't feasible, **effective compensating controls** were put in place; where neither made sense, **explicit risk acceptance** was justified and owned.
- **Argument C — Decisions remain valid over time.**
  Because systems and threats change, a **review loop** revisits earlier cost–risk trade-offs and updates them when conditions move.

And then you attach **evidence** that makes those arguments bite:

- For **Argument A**: signed SoW and scoping notes that reflect business priorities; mapping of scope against sector/regulatory baselines; tester competence and methodology; threat-driven techniques that minimise blind spots.
- For **Argument B**: triage and treatment records; re-test/verification artefacts; compensating-control design and validation; approvals and ownership for any acceptances.
- For **Argument C**: documented review process and schedule; decision logs showing changes over time; integration with vuln/change management and threat intel; proof that stale acceptances trigger re-assessment.

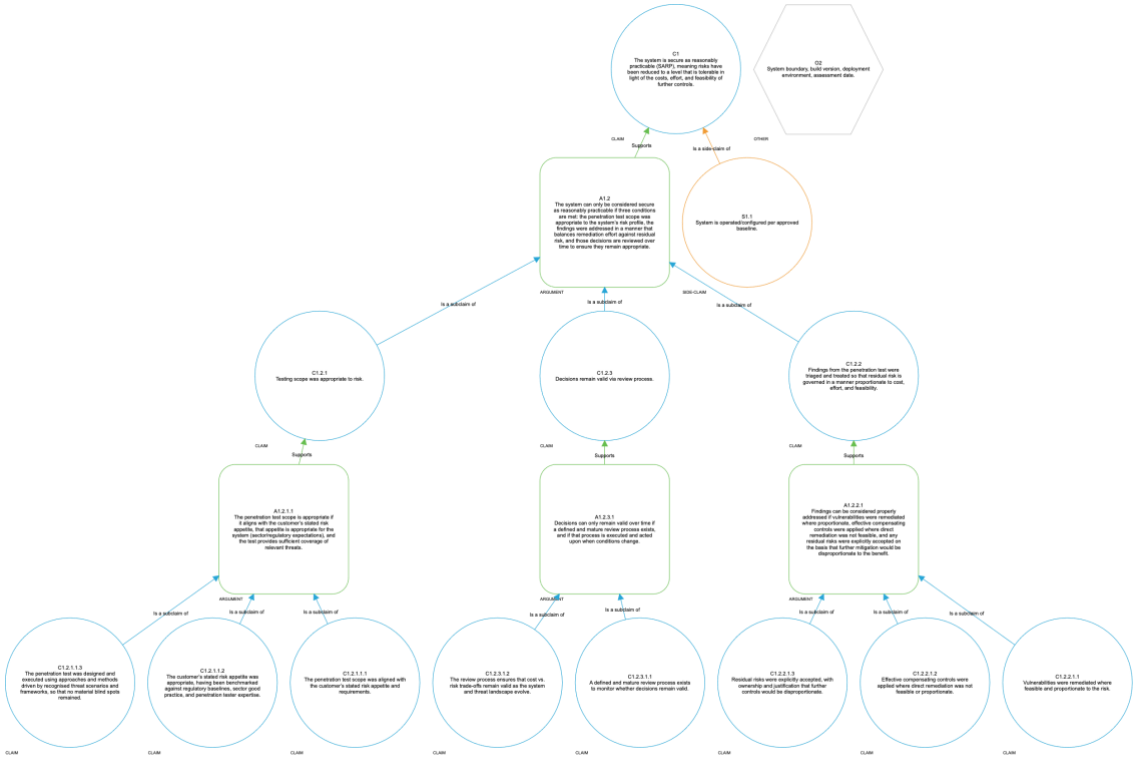

Figure 2: Shows how penetration testing evidence supports the broader claim that a system is secure as reasonably practicable (SARP).

Crucially, each finding becomes a fragment with its own mini-case (claim → argument → evidence). Risk owners "own" the fragment for their asset; fragments roll up into the programme-level assurance. That's what turns a pentest report from a static PDF into living assurance that can survive audits, handovers, and incidents.

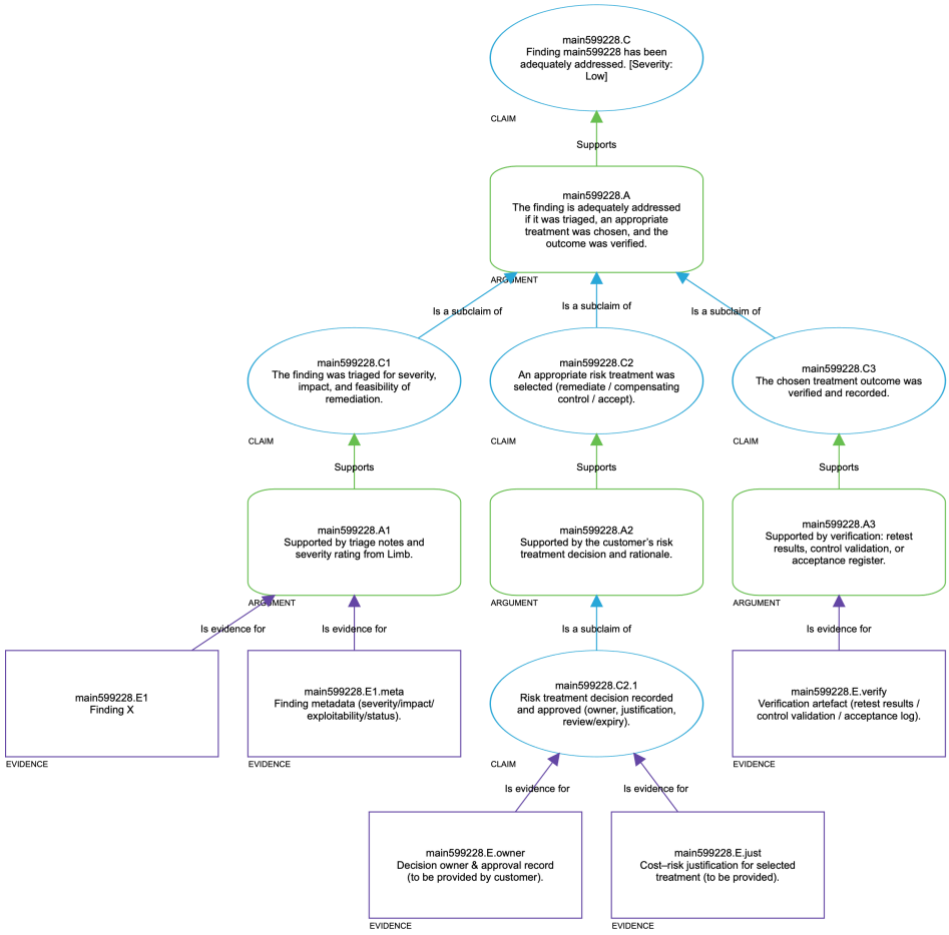

Figure 3: Shows how a single pentest finding is triaged, treated, and verified with supporting evidence.

This framing also fixes the "report that lands and sits there" problem. When a test is designed and reported **to support the case**, it lands with the right audience, in the right language, with the right next actions. Same technical work, radically different assurance value.

Example: Avoiding the Pentest Trap in R155

In automotive type approval, regulators are familiar with **witness testing**: physically observing emissions or safety tests and verifying reported results. That works for repeatable, measurable phenomena like emissions levels, but it does not translate well to cybersecurity.

If applied literally, it would mean regulators watching penetration testers in real time — an exercise that is expensive, slow, and of little value for managing risk. Penetration testing is not a one-off emissions check; it is part of a living security process.

A stronger approach is to keep the regulator engaged **throughout the process**, but through evidence, not observation. Instead of sitting in the lab, the regulator reviews structured artefacts that show the Cybersecurity Management System (CSMS) at work:

1. **Pentest undertaken.** The test is executed against the vehicle or ECU in scope.
2. **Findings identified.** Issues are recorded and linked to assets or functions.
3. **Triage performed.** Each finding is assessed for severity, feasibility of remediation, and proportionality of response.

4. **Evidence captured.** Risk treatment decisions, approvals, and re-test results are documented within the CSMS.
5. **Regulator reviews the trail.** Rather than watching every exploit attempt, the regulator inspects the evidence that these steps occurred, confirms governance was applied, and checks outcomes against type-approval requirements.
6. **Narrative demonstration.** A pre-/post-patch example can illustrate the journey without requiring constant oversight.

This approach acknowledges the regulator's role but avoids importing an emissions-testing model into cybersecurity. What matters is that the **vehicle type undergoing approval is governed by the CSMS** and that the organisation can show it.

The Inverse: Certificate of Compliance Reassessment

The same logic applies when demonstrating **ongoing compliance under UNECE R155**. Once vehicles have been type approved, organisations must continue to manage cybersecurity across the operational lifecycle and demonstrate this during **Certificate of Compliance (CoC) reassessments**.

Here, the playbook is similar, though asynchronous:

- Security testing and monitoring activities (including pentests) continue during operation.
- Findings are triaged and treated within the CSMS, with evidence logged and tracked.
- Regulators expect to see proof that these processes are active, not just one-off events.

And this is where the **counterexample** comes in. A pentest is one type of assurance evidence — but not the only kind. In practice, the evidence for assurance may also be drawn from:

- **SIEM/SOC events:** showing that attacks are detected and triaged in real time.
- **Incident response records:** demonstrating that incidents were managed to closure with lessons learned.
- **Patch and vulnerability management:** evidence that exposures are tracked, prioritised, and addressed.
- **Audit logs and monitoring outputs:** providing traceability of ongoing governance.

Together, these operational artefacts show that the CSMS is not a one-off project but a **living system of assurance** — one that supports both type approval and ongoing reassessment.

Beyond Pentests: The Assurance Evidence Portfolio

Pentesting is a useful starting point because it is tangible, familiar, and produces a report people can point to. But in practice it is only one piece of the puzzle. An assurance case worth relying on must draw from a much broader portfolio of evidence, one that spans the entire lifecycle of a system. It's not enough to show that something can withstand attack on a Tuesday afternoon in a test lab — we need to know that it was built securely, that it is being operated as intended, that issues are detected and managed, and that suppliers and partners are feeding into the process.

One important strand comes from **build reviews**. Here, there are really two different questions to answer. The first is: *is this configuration as secure as it could be?* This is the hardening perspective, where you compare what you have against best practice benchmarks and standards. The second is: *is this configuration the same as what the configuration management system says it should be?* That is a question about drift — about whether the live system still matches its approved baseline. Both questions generate valuable evidence. The first reassures you that the

system is designed to resist attack; the second reassures you that what you signed off on is actually what's running in production.

Another source of assurance evidence comes from **operational monitoring**. Security Information and Event Management (SIEM) tools and Security Operations Centres (SOCs) generate a stream of data about how the system is performing in practice. Detected incidents, triage records, alerts investigated, false positives reduced — all of this tells a story about whether the system is being watched and cared for. The claim here is not that "nothing happens," but that when events occur, they are noticed, triaged, and acted upon. That is lived assurance: evidence that your security processes work under real conditions, not just in design documents.

Closely related are **incident response records**. Every organisation knows that incidents will happen. What distinguishes mature organisations is not the absence of problems, but the way they respond. Post-mortems, after-action reports, and lessons-learned exercises all provide assurance evidence that incidents are managed to closure and that improvements are fed back into the system. In this way, incident response becomes assurance: not proof of perfection, but proof of resilience.

Supplier assurance is another vital strand, especially in cyber-physical systems where products are built on complex supply chains. Here, evidence might be as simple as patch notifications received or as complex as audit reports on supplier practices. But the real value lies in the response: what did you do when the patch notice arrived? Did you triage, test, and roll it out in a timely way? Did you track it in your risk register and verify that it worked? The assurance is not just that your supplier delivered updates, but that you integrated those updates into your own risk management system.

Finally, there are **certifications and governance artefacts**. Standards such as ISO 27001, IEC 62443, or an approved CSMS do not guarantee security by themselves. But they do provide evidence of a mature system of management. They reassure regulators, investors, and customers that there is a structured governance backbone — one that embeds assurance as a repeatable process rather than a series of one-off heroics.

When you bring all these strands together — build reviews, monitoring, incident response, supplier assurance, certifications — you create a portfolio that transforms assurance from a snapshot into a **living picture of security across the lifecycle**. Pentests still matter, but now they are one voice in a larger choir. And it is that harmony, not a single instrument, that persuades stakeholders to trust that a system is safe, secure, and resilient.

CAE in Practice: Why White Box Testing Adds More Value

One of the clearest places to see the impact of Claims–Arguments–Evidence thinking is in how penetration testing is scoped.

When no explicit assurance case is on the table, the default request often falls to **black box testing**. The implicit claim is: *"Independent experts spent X days trying to break in, exfiltrate data, or deny service, and here is what they found."* That is still evidence, but in our experience, it is weak assurance. It gives a narrow snapshot, says little about layered defences, and under budget constraints cannot realistically emulate a determined adversary.

That doesn't mean black box testing is useless. It can have value in certain circumstances, for instance, when an organisation's case (implicit or explicit) is really about **inherited risk**: *"we accept that attackers may find things we haven't anticipated internally, and we want to know what that looks like".* It is also valuable when an organisation has a mature assurance culture, and is explicitly looking for that one missing puzzle piece. In that context, the evidence it provides can be appropriate. But for most organisations, it falls short of supporting the stronger claims they need to make to customers, regulators, or investors.

This is why, in the absence of a clear assurance case, we advocate for **white box testing**. By giving testers knowledge of the system's architecture, configuration, and intended behaviour, you make better use of limited time and resources. The assurance case then reads more like this:

- *Claim:* "Our system's security posture has been independently reviewed against a defence-in-depth model."
- *Argument:* External perimeter controls were verified to behave as intended; internal layers were assessed under the assumption of compromise; and the rationale for resilience was evaluated, not just resistance to a few sampled attacks.
- *Evidence:* Test results, configuration and code reviews, architectural assessments, and documented treatment of findings.

This produces both more actionable outputs and stronger assurance. Instead of arguing only from activity, *"we tried for X days"*, it argues from structure: *"here is how the system is defended in depth, here is what experts validated, and here is the evidence of how findings were managed."*

A **more mature assurance case** takes this further. It doesn't just decide between black box or white box; it scopes testing as a targeted exercise against *specific subsystems*, *specific threat actor profiles*, and *specific objectives*. The claim might be: *"Given attacker profile X, targeting subsystem Y with objective Z, our system is resilient within acceptable bounds."* The argument is then validated by third-party experts emulating that capability. This approach avoids wasted effort, concentrates resources on what matters most, and produces evidence that is directly relevant to the risks the organisation has chosen to manage.

For most organisations, white box testing is the better starting point because it builds a structured case where none exists. As maturity grows, assurance cases allow you to move from "general proof of security posture" to **precision validation of risk scenarios**. That is where penetration testing achieves its maximum value: not as a generic activity, but as evidence precisely scoped to strengthen a defined assurance argument.

Closing Thoughts: Building with Claims, Arguments, and Evidence

If I could leave readers with one piece of advice, it would be this: try using **Claims, Arguments, and Evidence** in how you talk about what you are doing.

As an industry, we put a lot of weight on slogans like *Secure by Design* or *Secure by Default.* These are valuable aspirations, but they often stay at the level of principle rather than practice. What matters is the ability to show, in a structured way, how those principles are being realised, not just in your head, but in a form others can scrutinise and trust.

Design engineers already work with goals, requirements, and verification steps. Goal Structuring Notation (GSN) is one way of showing those relationships explicitly. But whether you use GSN, CAE, or another method, the point is the same:

- You have **claims** that the system, product, or service must meet across its lifecycle.
- You construct **arguments** to show that the design and verification approach is sufficient.
- You gather **evidence** from unit tests and pentests to incident logs and supplier audits to support those arguments.

Thinking this way forces you to consider that what feels like a strong argument to you may not persuade others. A developer may see unit test coverage as compelling assurance. But to an incident responder, assurance comes from resilience under attack. To a regulator, it may be certification or governance. To a customer, it may be usability or transparency. Different silos, different stakeholders, different assurance needs, all of which must be surfaced and reconciled.

And this is not just about new products inching toward Minimum Viable Product. *Every organisation, every programme, every project is at some point in its own Valley of Death.* It's the moment where trust is fragile, where stakeholders are not yet convinced, and where tacit cases aren't enough. For a start-up, that might be scaling beyond a prototype. For a national infrastructure operator, it might be proving to regulators that legacy systems can still be trusted under NIS2. For a logistics operator, it might be demonstrating resilience in the face of ransomware.

In all of these cases, assurance is the bridge. Explicitly making your claims, arguments, and evidence visible transforms isolated activities into a narrative of trust. It reduces assurance debt, strengthens credibility, and makes it far more likely that your ideas, products, or services will survive the crossing.

The Valley of Death is not unique to innovators in shiny labs. It exists wherever confidence is fragile and the burden of proof is high. By treating assurance as a living discipline — one that is just as essential as the engineering or the business case — we give ourselves a better chance of getting to the other side with trust intact.