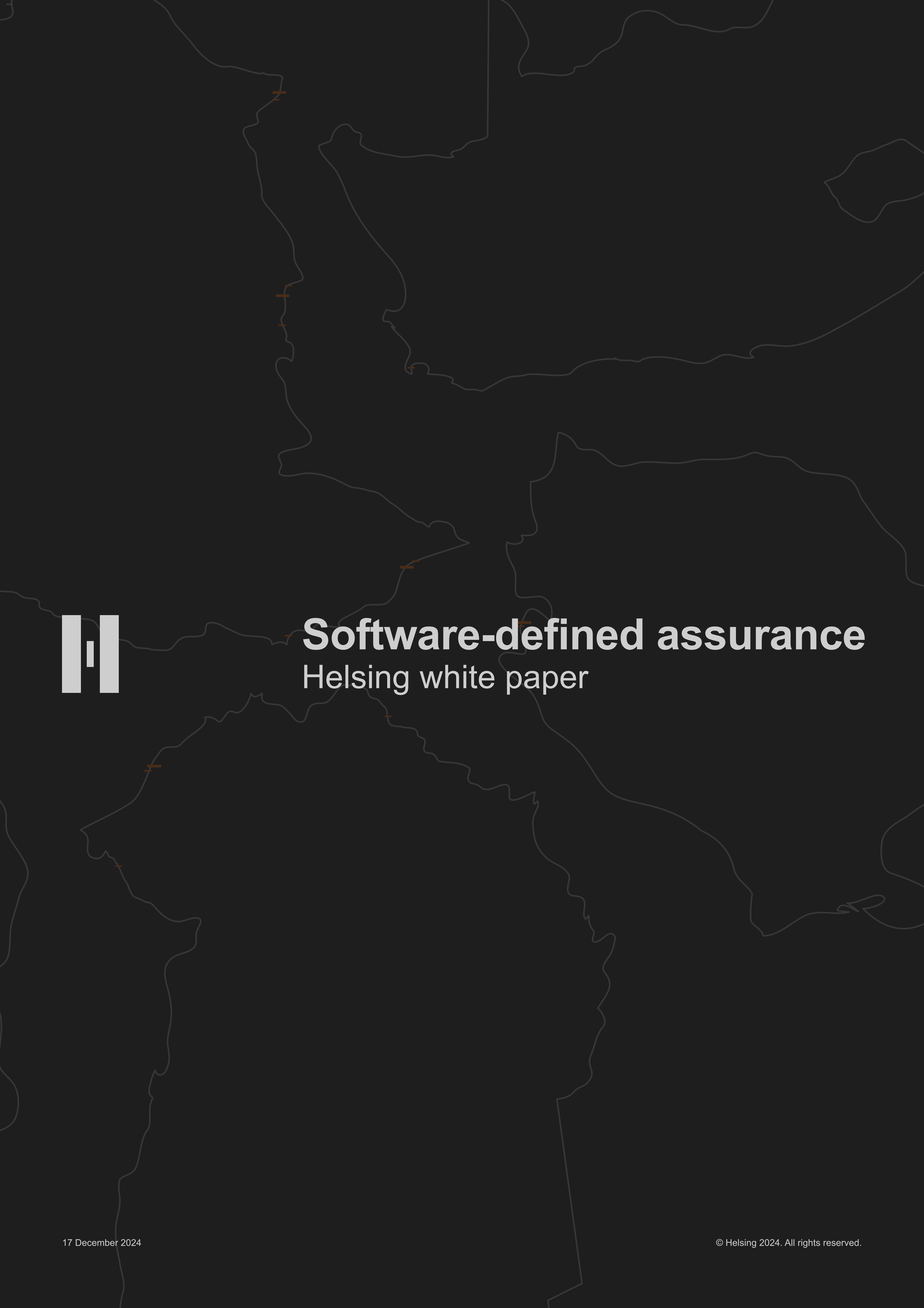

Software-defined assurance

Helsing white paper

17 December 2024

Table of Contents

Executive Summary

Assurance is key to bringing effective new defence capabilities to the battlefield. As software and AI becomes increasingly central to capability development, our approach to assurance needs to adapt. Specifically, we need to revisit our assurance processes with a new model in mind:

- Assurance is a continuous process that allows for capability adaptation.
- Software re-use is intended, across many different hardware configurations.
- Systems are assured as part of an ever-changing composable force structure.
- Specific approaches and techniques enable assurance of AI-based systems.

We identify this model as software-defined assurance, and recommend focused investments in software infrastructure, digital assets, and new standards to accelerate its adoption. Successful adoption offers strategic advantage in the production and maintenance of defence capabilities.

01 As defence becomes software-defined, assurance needs to follow

As we see in Ukraine, software and AI are becoming central to modern military systems. Mass is being generated using software-defined capabilities which integrate intelligence and autonomy into off-the-shelf or low-cost hardware platforms. Decision advantage is created through near real-time situational awareness generated from sensor data collection, fusion, and processing. Software-based system updates enable forces to rapidly adapt to changing operational contexts and evolving threats. More and more, the software layer is defining powerful, adaptable capabilities on top of the hardware (background: 1, 2).

To ensure that these capabilities are effective and safe, we need to assure them. Assurance is about establishing that a system is trustworthy, by showing that it meets performance and safety requirements.

We use the word assurance in this paper to cover the sum of the efforts used to establish confidence in the suitability and readiness of a system for operational deployment. This includes both qualification and certification, based on activities such as design assurance, validation and verification (V&V), quality assurance, security assurance, and safety management.

Without effective assurance processes, forces cannot rely on systems in combat. Existing processes have been built up over decades based on learnings from system development and production, and are embodied in standards whose adherence is expected by purchasers and satisfied by suppliers.

However, as capability becomes increasingly software-defined, our assurance processes must evolve. Existing assurance approaches are not well-adapted to software-defined capabilities, and this acts as a brake on innovation, development and investment. We could build better systems, faster, if we had more suitable assurance processes.

While the general evidence-based systems engineering approach established in the past decades remains valid, we need to adapt to the advantages of software. In particular, our approach to assurance needs to enable:

01 **Speed:** Software has a radically shorter development cycle, with more iterations of the full loop from requirements to validation. Rapid capability updates can be delivered in the field via the software layer, without changing the hardware. The assurance process must enable us to deliver confidently at speed.

02 **Transferability:** Investments in software can be preserved across different pieces of equipment. The same software-defined capability can be deployed to many different types (and generations) of hardware, even across providers, concentrating investment. The assurance process must recognise this 1:N relationship between software and hardware.

03 **Autonomy:** The level of autonomy in systems is increasing across all domains, driven by developments in artificial intelligence. AI presents a new set of assurance challenges, demanding more robust technical approaches, different from those commonly used for software.

04 **Composable forces:** Historically, we've relied on human operators to link systems together, with limited digital interfaces. As more and more assets are connected together, in a wider variety of configurations, we need to be able to assure complex, composable forces (aka dynamic systems-of-systems), ranging from heterogeneous drone swarms to plug-and-fight, layered air defence systems.

These challenges require a rethinking of assurance processes for a software-defined world. We call this software-defined assurance. It can unlock better, cheaper, faster software capability development.

02 Defining software-defined assurance

There are four fundamental principles underlying the assurance model that distinguish software-defined assurance from existing defence assurance approaches.

We are painting with a broad brush here. In some areas, local initiatives have resulted in the construction of assurance approaches which look more like software-defined assurance.

We are not dismissing those innovations, we are championing them. We want to see them pushed further and made universal, as part of a general transition towards software-defined assurance. We believe that by identifying the high level needs driving these evolutions, and describing the emerging trends, we can focus attention on the opportunity and accelerate the transition.

Existing defence assurance approach	Software-defined assurance
Assures a system on a one-off basis to approve deployment or mid-life upgrade	Treats assurance as a continuous process, including operation and regular updates
Defines and evaluates a combined hardware+software system	Defines and evaluates a software system for use with many hardware systems (1:N)
Assures a single system in isolation	Assures a system as part of a composable force structure (system-of-systems)
Assumes that all software is built from code	Defines assurance paths for AI

Moving to a continuous assurance model

Existing assurance processes aim to determine the suitability of a system for deployment once and for all (or at least until the mid-life upgrade in 10 years). This misses a range of opportunities brought by the more dynamic nature of software, which can be rapidly updated.

For software, the assurance process is better conceptualised as a continuous loop, with an arbitrary number of re-deployments, than as a linear start-work-end process. This is a fundamental change in perspective that leads to two immediate opportunities.

First, we can extend the software assurance model past deployment and into operation. The software itself can record and report on the operational experience, feeding back into the update process. As well as improving the capability, this can tell us where the assurance has fallen short, or where the capability was called upon in operational conditions under which it is not assured, driving the following assurance loop.

Second, each iteration through the loop does not have to be the same. We can tailor the assurance process to the operational context that we plan to deploy into. This means getting the first capability into operation more rapidly, and covering other operational contexts as and when they become relevant. This is a very different mindset to a traditional assurance process, which aims to cover a large set of possible operational contexts in a one-time assurance process, and thereafter excludes all others. We can frame this mindset as adding an additional layer to the maintenance processes, one where we maintain the capability definition itself.

1:N - Assuring software for use with many hardware systems

Existing assurance processes focus on the assurance of an overall system made up of both software and hardware, coupling the development and assurance of the software to that of the hardware. The logic behind this approach is that it is the system which is delivered, and which needs to be designed, developed and assured holistically. Conceptually, the software is just another part of the overall system, which is mostly hardware. This logic produces a waterfall-like approach to development and assurance, with approaches and timelines that match that of the hardware. Re-use of software is less common than it could be, and the best software engineers go to places where they can move faster..

We need to free the software development and assurance process from the hardware approach and timelines. We should conceive of each system as having two distinct, equal parts: hardware and software. We can fix the interface and requirements between the hardware and software, and then iterate independently on the software, applying a software-oriented approach. When necessary, we can test the software with the hardware, to assure the complete system. The US Air Force's Next Generation Acquisition Model is a clear move in this direction.

Software can be tested in a highly automated and large-scale fashion. We can test the performance of the software with respect to the requirements, including the compliance with the hardware interface specification, with a mix of real, augmented and artificial data. This will give a robust evaluation of the performance, enabling data-driven improvements to the core algorithms. Scale and frequency of testing is only bound by compute cost, and automation enables rapid re-assurance of the software system.

With the software assurance handled independently and the hardware built to meet its requirements, we can change the underlying hardware and conduct end-to-end tests to validate the integration. This enables us to leverage the investments in software across more platforms (over time, and across manufacturers), and to iterate more quickly on the hardware. This is the promise of software-defined defence - with software-defined assurance in place.

Assuring composable force structures

Existing assurance approaches focus on confirming that a system satisfies its requirements. These requirements are defined from the point of view of the system itself. Some of these requirements relate to interaction with the surrounding operational context, by specifying interfaces that must be respected (e.g. Link 16). Few focus on how systems work together.

Even if we tried to capture specific interactions between systems as requirements, we could not hope for the specification to be rich or dynamic enough to capture the range of interactions, especially as the level of autonomy in each system grows. The specification problem grows combinatorially with the number of capabilities involved. Exhaustive requirements definition and testing are impossible.

With software-defined assurance, we can leverage the rich software-based testing environment to model the behaviour of the system being developed when it interacts with other systems. This type of assurance must be data-driven. We need to create evaluation environments in which we expose the systems-of-systems to a variety of operational situations, and evaluate the consequent behaviour. By doing this, the ceiling of complexity for what can be assured moves from what a human can specify in requirements to what we can store and compute.

Assuring artificial intelligence (AI)

Approaches to assuring software emerged from existing approaches to dealing with electronics, as software was an extension of electronic systems. Initially, there was uncertainty that software could be effectively assured: prevalent assurance approaches at the time were statistical (e.g. FMEA), but it was not possible to analyse the potential for software design flaws in the same fashion. In aviation, this led to the creation of the DO-178, ancestor of today's widely used DO-178C / ED-12C standard, with a novel focus on design assurance and the development of common verification techniques for software.

Artificial intelligence (AI) is in a similar position today. Many of the well-known approaches used to ensure the reliability of software are difficult or impossible to apply to AI-based software, where models are created from data rather than hand-coded by software developers. This creates friction in the commissioning and development of AI-based software, because it is unclear what criteria will be used to assure it.
The potential worst case is that assurance of systems involving AI are subject to a matrix of both poorly-fitting existing requirements and new but underspecified AI-related requirements.

Software-defined assurance accepts that the way in which we create software will change, and that this will impact the way that we assure software. It clearly separates those concepts which are independent from the method of implementation of the software (such as high level requirements) from those which are tied to it (such as code coverage). As the approaches used to create AI will certainly change in the coming years, the structure must allow for regular updates and the incorporation of new techniques.

03 Achieving software-defined assurance

To realise the transition to software-defined assurance, action and investment is required in a number of areas:

01 **Software-defined assurance infrastructure:** Software infrastructure that enables development, assurance, deployment, rapid re-assurance, updates and monitoring across thousands of assets.

02 **Digital assets:** Customer-owned digital assets - data, environments, models - that can be continuously developed and used for assurance across programmes.

03 **Standards for software assurance:** Rethinking of defence standards for software development to enable assurance of software on its own terms.

We need a strategy that sets a clear course across these areas, with appropriate investment to kickstart the change process. We also need to consider broader challenges in procurement, to ensure that the benefits of software-defined assurance are realised as widely as possible.

Software-defined assurance infrastructure

The fundamental advantage of software assurance is that the assurance process can itself be encoded in software. Traditionally, assurance processes are described in standards, implemented through processes, and enforced through reviews and audits. This is time consuming and error prone, and brings a heavy training and knowledge management burden.

For software assurance - including AI - we can build these processes into the infrastructure that is used to build the capability, from development through assurance to deployment, updates and monitoring. This software assurance infrastructure can act as a central management system for capability development in order to:

- Push assurance requirements back into the development workflow (e.g. data provenance for AI).
- Manage and integrate the digital assets necessary for development and assurance (see below).
- Provide tools to define - and where possible, automate - the assurance process.
- Gate deployment of software capabilities against assurance requirements being met.

The infrastructure doesn't need to be monolithic, but it must be integrated to remain coherent. Different organisations - customer or industry - will have different preferences for specific tools, but these must be brought together under a system that manages the overall software assurance process. It is the role of that central system to orchestrate the process while enforcing the demanding security and classification requirements that are critical to defence. Ministries of Defence must play an active role in the deployment and adoption of this infrastructure across the ecosystem.

Digital assets used in software and AI assurance

Software and AI assurance capabilities are built up over time, through continuous investment in infrastructure and assets. There are several categories of assets used in software and AI assurance:

- **Simulations:** Simulation environments with accurate models. Simulators can model and test the performance of systems across a wide range of scenarios and conditions, including extreme and rare events that are difficult to replicate in real life. Simulation allows for extensive testing at a fraction of the cost and time, accelerating the re-assurance process for updates.
- **Operational design domains (ODDs):** Normalised definitions of an expected operating context that define the bounds within which to evaluate the performance of a system and its components. Alignment of ODDs enables re-use of software components and system-of-systems level evaluation.
- **Data:** Collections of real, augmented and artificial data. Rigorous testing and evaluation of software and AI systems depends on collecting data for evaluation that sufficiently represents and covers the operational design domain.

We need to build these as customer-owned and controlled assets, to ensure that many industry players can deliver assured systems. They can still be developed and maintained by industrial partners, as other types of equipment are, but they cannot be controlled by those partners.

These assets need to be built against a longer-term, customer-owned roadmap. We can leverage specific programmes to prioritise and resource the development of these assets, but we need a vision for a coherent set of assets that can be used across programmes. We see examples of this starting to emerge, as with the US Air Force's investment in digital twins. Greater speed and scale is needed.

Adapted standards for assurance of software-defined capabilities, including AI

Existing standards for software assurance in defence have essentially two lineages:

01 Defence standards developed for hardware, extended to incorporate software. E.g. AQAP 2310 + 2210.

02 Software assurance standards from the civil world, adapted to defence. Eg: DO-178C.

Of course, software will continue to be used in many ways in building military systems. A safety mechanism for a fuel storage system will benefit from the cumulative experience encapsulated in IEC 61508, more than from a process for rapid updates. There's no need to throw the baby out with the bathwater - the focus should be on clarifying how to assure software-defined capabilities, not covering all software.

Where we are building a software-defined capability, we need to align on standardised approaches that recognise that (a) software is different to hardware and (b) defence is different to the civilian world. We need to apply systems engineering principles to assuring software systems, thinking from a software-first perspective. In particular, new standards must:

- Provide guidance on the development and verification of machine learning and AI. Standards that assume that all software is hand-coded are out of date. Guidelines for effective verification of algorithms (especially AI) can spread best practices and accelerate capability development. The EASA Artificial Intelligence Concept Paper is a very positive step in this direction.
- Balance the opportunity for high-speed innovation against the risks involved in assuring capabilities iteratively. An explicit risk management style approach to weighing speed-to-capability against reliability can enable conscious trade-offs in these dimensions, instead of an all (peacetime) or nothing (wartime) approach.
- Reflect the need for rapid re-assurance of software that is incrementally updated (for example, by retraining an AI model based on new data to improve mission performance). This subject is not handled well by hardware-oriented defence standards, nor by civilian standards for software and AI assurance. Even the EASA AI Concept Paper, being civil aviation focused, does not address the question of the rapid integration of data into defence capabilities.

Initiatives like the European Initiative for Collaborative Air Combat Standardisation (EICACS) exist to develop standards for defence applications. They need to consider assurance of software-defined capabilities without being too strongly tied to existing civilian or hardware-oriented approaches. Similar efforts are needed across all domains — air, maritime, and land — to ensure that standards fully reflect military operational requirements in the software-defined defence paradigm.

Beyond this, Ministries of Defence and procurement authorities also have a critical role to play by signalling their willingness to innovate on assurance as well as on capability. MOD and procurement leaders must refocus attention on the desired outcomes - reliable systems - rather than delegating blindly to existing standards or processes. It will require effort to find mutually acceptable solutions, but one or two significant programmes, given a mandate to drive innovation in assurance as well as capability, can act as pathfinders and provide a tipping point for the assurance and deployment of AI capabilities.

Updated acquisition models and open architectures

It is important to recognise that modernising assurance alone will not be sufficient. Acquisition processes broadly also need to change. We refer again to the US Air Force's Next Generation Acquisition Model, which highlights some of the key topics to be addressed under the rubric of "Agile Acquisition". Without changes in the way that procurement is handled, the benefits unlocked by software-defined assurance will be limited by different forms of vendor lock-in, both technical and commercial.

A full treatment is beyond the current scope, but particularly pertinent because of the close link to the topics discussed above are open architectures. To exploit the 1:N opportunity of software and AI - particularly across platforms built by different suppliers - we need to define and adopt open architectures into which that software and AI can be integrated.

04 Software-defined assurance is a strategic capability

There is - rightly - much discussion about the transformative role that software and AI can play, and in some cases are already playing, on the modern battlefield through software-defined defence. However, without a fit-for-purpose approach to assurance, this potential will be handicapped. Capability development and deployment will be slowed. Investment will be limited in volume due to the friction, and diluted in effect due to the parallel and repetitive development of software capabilities. Applied over years, this will significantly impact the capabilities available to NATO forces.

We need to invest in software-defined assurance in order to unlock the full opportunity of software-defined defence. As with factory production, the nations that are able to build industrial-scale software assurance systems that enable the production of quality software at high speed will have a strategic advantage in modern warfare. For software and AI, this system and the advantage that it produces can reach all the way from research and development to deployment on the battlefield, raising the stakes.

Achieving software-defined assurance requires rethinking assurance to focus on software as a distinct system, not just as an annex to the hardware. This will enable us to decouple software assurance from hardware assurance, reframe it as a continuous process, and solve the higher complexity assurance problems presented by autonomy and composable force structures. Defining and demonstrating assurance paths for AI will be critical. While there is some existing movement in this direction, it is patchy, slow, and uncoordinated - we need to consolidate and accelerate the efforts.

Software-defined assurance is a north star around which we can mobilise an incredible depth of expertise - public and private - across NATO countries, accelerating our overall capability development. To support this, we need to focus attention and resources on three efforts: software assurance infrastructure; customer-owned digital assets; and new defence software standards. With these foundations in place, we can establish a software-defined assurance model that enables us to build and update software-defined capabilities rapidly and with confidence.

Helsing

helsing.ai