[dstl]

The Science Inside

Defence Science and Technology Laboratory

Assurance of Artificial Intelligence and Autonomous Systems

A Dstl Biscuit Book

Ministry of Defence

Defence Science and Technology Laboratory

Assurance of Artificial Intelligence and Autonomous Systems

A Dstl Biscuit Book

DSTL/PUB136185

Front Cover image: greenbutterfly/stock.adobe.com

Contents

Foreword

Artificial intelligence offers great potential. So, too, do autonomous systems.

To benefit from this potential we need to have confidence. We need confidence in the artificial intelligence algorithms that solve complex, poorly defined problems. We also need confidence in the autonomous systems that turn these algorithms (and other components) into things we can use.

Overall, we need confidence that algorithms and systems are worthy of our trust, or that they are trustworthy. Even if they are, we wouldn't let them do just anything, without any form of supervision. We'd just be able to do a lot more, with a lot less risk, than we can do at the moment.

Assurance is about gathering evidence, often from different parties, and structuring it in a logical argument. This helps decision makers understand the level of confidence that can be placed in an algorithm or a system. Without sufficient assurance, we run the risk of using systems too soon when they are still unsafe, or using them too late and missing valuable opportunities.

Ultimately, assurance is the key that safely unlocks the potential of artificial intelligence and autonomous systems.

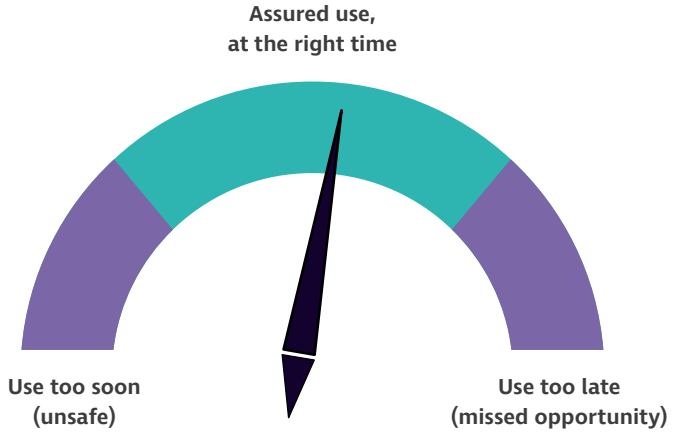


Figure 1. Assurance allow systems to be used at the right time

Introduction

This guide is what we call a Biscuit Book, something you can pick-up and 'dip into with a tea and biscuit'. The Biscuit Book is arranged as a series of easily digestible chunks that each cover a topic. It aims to provide the essential information, without ever being too technical.

Autonomous Systems and associated Artificial Intelligence (AI) algorithms can improve our world. From medical assistive robots to autonomous vehicles that work in dangerous locations, some of the greatest benefits come in safety-related applications. If we are to deliver a better future for all, finding a way to safely use AI algorithms and autonomous systems is not just desirable, it is essential.

To date, we have not fully satisfied this need. For example, we don't routinely see autonomous vehicles on our roads. A big reason for this is a lack of confidence in how the algorithm and the system will behave. Often, we're most interested in confidence that a system will behave safely, especially when it is faced with unexpected or unusual situations.

We need a way of generating confidence and describing how much we have. That's what this book is all about.

What do we mean by AI, Autonomy and Assurance?

In previous Biscuit Books[1, 2], Dstl defined AI, Autonomy and Autonomous Systems as follows:

Artificial intelligence
Theories and techniques developed to allow computer systems to perform tasks, normally requiring human or biological intelligence.

Autonomy
The characteristic of a system using AI to determine its own course of action by making its own decisions.

Autonomous systems
A system containing AI-based components that allow it to exhibit autonomy.

Although a number of alternative definitions could have been chosen, for convenience, we'll keep using these.

Assurance is the key focus of this book, there's a strong hint in the title. We can think of assurance as the structured body of evidence that gives us confidence that we can trust. More directly:

Assurance
Provides a compelling case for the trustworthiness of the system or the algorithm.

At this point, it is helpful to distinguish autonomous systems from their 'highly-automated' cousins. Unlike automation, autonomous systems can respond to situations that were not pre-programmed or anticipated. Being able to respond to the unexpected and determine their own course of action is precisely what makes these systems valuable.

Unfortunately, this also makes it hard to provide confidence in their behaviour, which can sometimes be unpredictable. This lack of confidence can undermine the trust of developers and users, as well as wider society. The very thing we prize, being able to respond to the unexpected, is the very thing that makes assurance difficult. Difficult, but not impossible.

Assurance uses evidence from a number of perspectives. Some of these relate to development processes. Others relate to the product itself. For large, complex systems, we need a way of organising this evidence into a logical structure, which is often called an Assurance Case or a Safety Case.

When developing this case we need to take a balanced approach. Clearly, we need supporting positive evidence that confirms the system can be relied upon and is trustworthy. We also need to look for undermining negative evidence, by trying (and hopefully failing!) to make the system falter and misbehave. If we don't find even a small amount of negative evidence, then we probably haven't looked hard enough.

There are a couple of similar sounding words, which describe things similar to assurance. To avoid confusion, we note these below.

- Ensurance (which is not really a word, but does get used) provides a guarantee of something. Assurance is different, as it does not guarantee trustworthiness. Nor does it guarantee complete safety, or anything else.
- Insurance provides protection (often financial) against the effects of a possible future adverse event. In contrast, assurance provides confidence that adverse events are unlikely to arise from unexpected situations.

Test
A test uses pass / fail criteria to check a specific, single function.

Evaluation
An evaluation combines multiple tests and experimental use, to provide an overall judgement.

Verification
Verification is the act of checking against a specification. It asks, 'did we build the thing right?' or 'are we doing the thing right?'

Tests are usually associated with verification.

Validation
Validation is the act of checking against an end user's need. It asks, 'did we build the right thing?' or 'are we doing the right thing?'

Evaluations typically relate to validation.

Assurance is underpinned by Test, Evaluation, Verification and Validation (TEVV) activities, which are described below. These activities are typically conducted throughout development. During use, autonomous systems will encounter unanticipated situations, so they may also require ongoing, or runtime, assurance.

There is a benefit associated with assurance. So, we should not be surprised that it also comes at a cost. Simplistically, the more assurance that is required, the more evidence we need and the greater the cost.

The level of assurance we need is closely related to the level of risk. Risk combines the likelihood of failure and the consequences should a failure arise: higher likelihoods and higher consequences both result in higher risks. It is often discussed in abstract terms, like the probability of failure per flying hour.

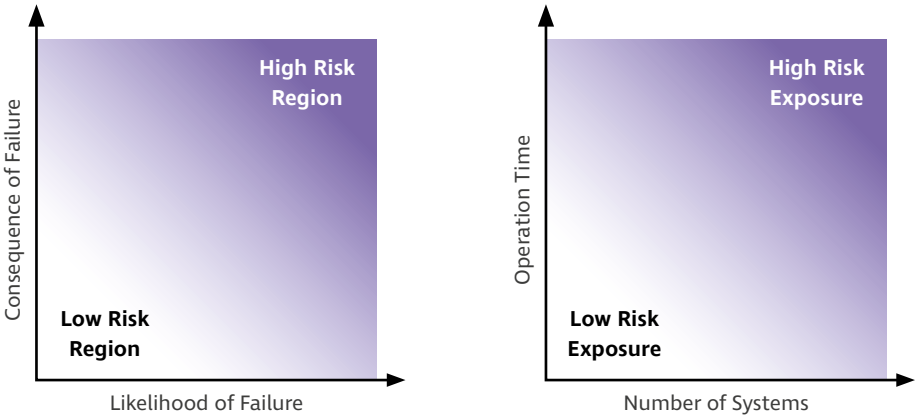


Figure 2. An illustration of risk and risk exposure

To understand what these risks actually mean, we need to think about our exposure to them. The longer and more widely a system is used, the greater the level of risk exposure. We can make it less likely that bad things will happen by reducing a risk, or by reducing our exposure to that risk (or, probably, both).

We have to place a large amount of trust in systems and algorithms that have high levels of risk exposure.

A large amount of confidence, and hence assurance, is needed when a system is deployed in its thousands and operated for years, or the likelihood and consequence of failure are high. Trans-Atlantic passenger aircraft are a typical example of this. Less assurance may be required when a handful of systems are only operated for a brief time, or the likelihood and consequence of failure are low.

Legal and ethical considerations of AI-based systems

Proxima Studios/stock.adobe.com

Legal and ethical concerns are important reasons for conducting assurance. AI is the key technology that allows autonomous systems to operate in complex environments. But practical, AI systems are a relatively new invention, and in many respects, are still 'getting there'. Similarly, the legal and ethical aspects of fielding AI-based systems are still developing.

Of course, the same laws apply to both AI-based systems and traditional ones. However, AI brings with it additional legal and ethical challenges. There are several reasons for this; for example, AI-based systems typically rely on large amounts of data and their behaviour can sometimes be difficult to predict (or explain).

If a human operator willingly uses (or misuses) any system to harm others then they can be held responsible for the outcomes. If a human is overseeing an autonomous AI-based system, which causes the same behaviour without it being the operator's intent then it becomes more difficult to trace moral and legal responsibility. Developers and users of autonomous systems need to think carefully about potential legal and ethical consequences of their actions or inactions.

Exactly which areas of law are relevant will depend on what the AI-based system is being used for. The UK Government Data Ethics Framework[3] provides guidance to help understand these issues.

The central concern of AI ethics is to understand and mitigate the negative impacts of AI on individuals and society, and in doing so promote responsible use of AI for optimal public benefit. More precisely:

AI ethics

AI ethics is a set of values, principles and techniques that employ widely accepted standards of right and wrong to guide moral conduct in the development and use of AI technologies.

Source: Understanding artificial intelligence ethics and safety: A guide for the responsible design and implementation of AI systems in the public sector[4].

These values, principles and techniques should guide and motivate morally acceptable practices throughout the lifecycle of design, development and use of AI-based systems. Values typically express our beliefs and our desire to 'make things better'. They help us reflect on our ethical goals as well as the potential problems we want to avoid. Principles, techniques and guidance take us a step further by providing the framework and tools we need to design and use AI ethically.

There have been many AI ethics principles published in recent years, by several different organisations. In general, they tend to focus on the following themes[5]:

- Transparency, so we understand why particular decisions were made.
- Justice and fairness, so we do not disadvantage specific individuals and groups.
- Non-maleficence, so we avoid doing harm.
- Responsibility, so we consult stakeholders and plan for the long-term.
- Privacy, so we respect people's rights over their data, during both development and operational use.

The real challenge is putting these principles into practice. There are several published frameworks associated with the legal and ethical adoption of AI-based systems. As well as the framework we noted earlier, the UK government has also published guidance relating to ethical AI in the public sector[6].

In addition, a National AI Strategy is due to be published around the same time as this Biscuit Book. This strategy will include a focus on ethical, safe and trustworthy development of responsible AI.

Aspects of assurance

In common with traditional systems, there are 2 main ways of providing assurance: process-based and product-based. An Assurance Case generally contains evidence from both approaches.

Process-based assurance

Process-based assurance gains confidence from the way in which something is developed.

The specific processes used for a particular system are of interest. These processes should consistently deliver a well-engineered product.

As well as having good processes, we need to show they have actually been used. Quality Assurance (or, strictly speaking, 'ensurance', as it provides a form of guarantee that processes were followed) activities are helpful here. The process-based nature of Quality Assurance distinguishes it from the related, but product-based, notion of Quality Control.

Limited assurance can also be gained from a supplier's track record and accreditation to professional standards. Whilst this is no guarantee of success, it can still be a useful part of a wider assurance argument.

Product-based assurance

Product-based assurance gains confidence in a final item, often through Test, Evaluation, Verification and Validation (TEVV). This can provide high-quality information, but may be limited in scope and time.

For example, some aspects of TEVV require access to a 'final product', although valuable insights can be gained throughout the systems engineering lifecycle. In service information, gained from operating the product can also be valuable.

Process-based and product-based approaches provide a theoretical basis for assurance. From a practical perspective, when assuring a system, we need to think about several things. We need to think about the algorithm that implements the AI, which enables autonomy. We also need to think about the data and the software toolkits used to develop this algorithm. Oh, and just to be on the safe side, the computing hardware that runs our algorithm also needs some thought.

Algorithms and data are important, but they only deliver benefit when they're part of a system. So, we need to think about how we integrate an algorithm (and its computing hardware) into a system.

Obviously, we need to think about the system itself, as well.

Our list of things to think about gets longer still. Unless our system will be used in complete isolation (and that's pretty unlikely), we also need to consider the other systems it interacts with. That is, we also need to consider the wider 'system-of-systems'.

Finally, like it or not, AI and autonomous systems make attractive targets for adversaries, also known as 'The Bad Guys'. Hence, assurance needs to consider how we protect against bad people causing bad things to happen. This applies across the piece, from data and computing hardware right up to the system-of-systems. In addition, there are legal and ethical considerations, which underpin everything we do.

For a single system, it is simplest to talk about a single integration (other people talk about this as an 'architecture') that contains a single algorithm and we mostly adopt that approach in this book.

In reality, several algorithms may be captured in a single integration: typically, some of these algorithms will be 'traditional', rather than AI-based. Likewise, a system may include several integrations. And, by definition, a system-of-systems has to include more than one system!

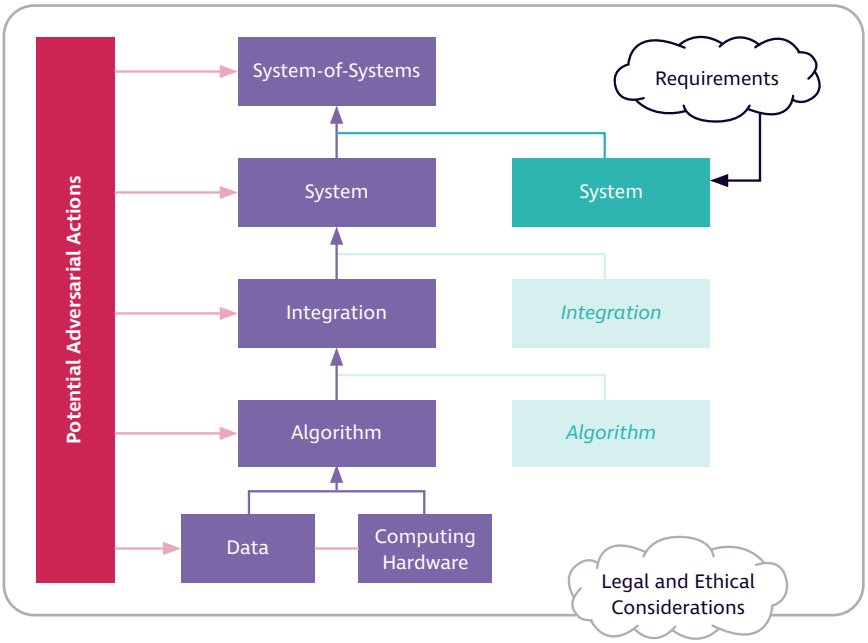


Figure 3. Aspects of assurance

Assurance of requirements

Requirements are at the heart of any system. Put simply, without them it is practically impossible to develop anything, or at least anything close to what was actually wanted or needed.

Even if something was developed, without requirements we cannot demonstrate confidence in the system's behaviour, as we don't know what it was intended for. In the case of 'traditional' systems, requirements are validated with the user (are we building the right thing) and verified in the design (have we built it right).

AI algorithms and autonomous systems are, by their nature, intended to work in uncertain environments. Just to be difficult, this may produce requirements that are vague or ambiguous. It is hard to show that these types of requirement have been satisfied.

In 'traditional' verification and validation, requirements flow down from concise higher-level intent (what the system should do) to precise lower level requirements (how it is to do it). The assurance process uses this flow down to relate low-level behaviour, which is easily tested, to higher-level system behaviour, which can then be evaluated.

In AI-based systems, this flow down is much less precise. This makes it harder to combine pieces of evidence into a compelling Assurance Case. While we can make some individual pieces, they just don't join together as well as they would for a 'traditional' system.

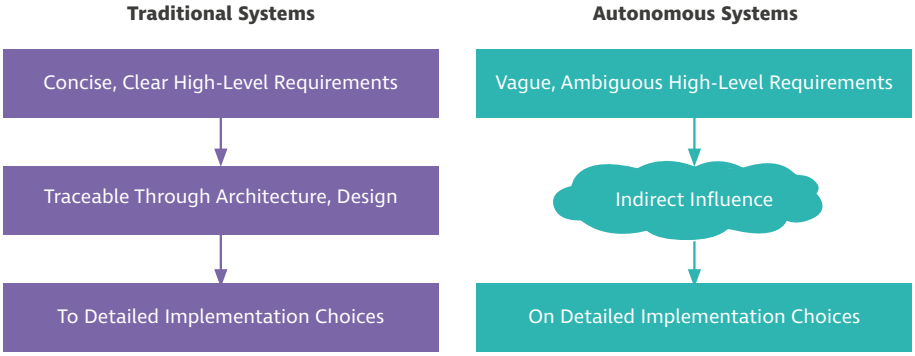


Figure 4. Requirements are different in autonomous systems

Assurance of computing hardware

AI algorithms run on computing hardware. This could be general-purpose hardware, like that in your laptop. Alternatively, it might be hardware specifically designed for a particular type of algorithm, such as a deep neural network.

Believe it or not, hardware is neither perfect nor precise. Hardware errors and lack of precision can cause an algorithm to give wrong or inaccurate outputs. So, any Assurance Case needs to consider the computing hardware. Assurance that the hardware is, and remains, correctly configured is important.

Understanding how different factors may change algorithm execution time may also be important. It is no good if an autonomous vehicle makes the right decision but at the wrong time: deciding to change direction 3 seconds after you have hit a tree is of little use!

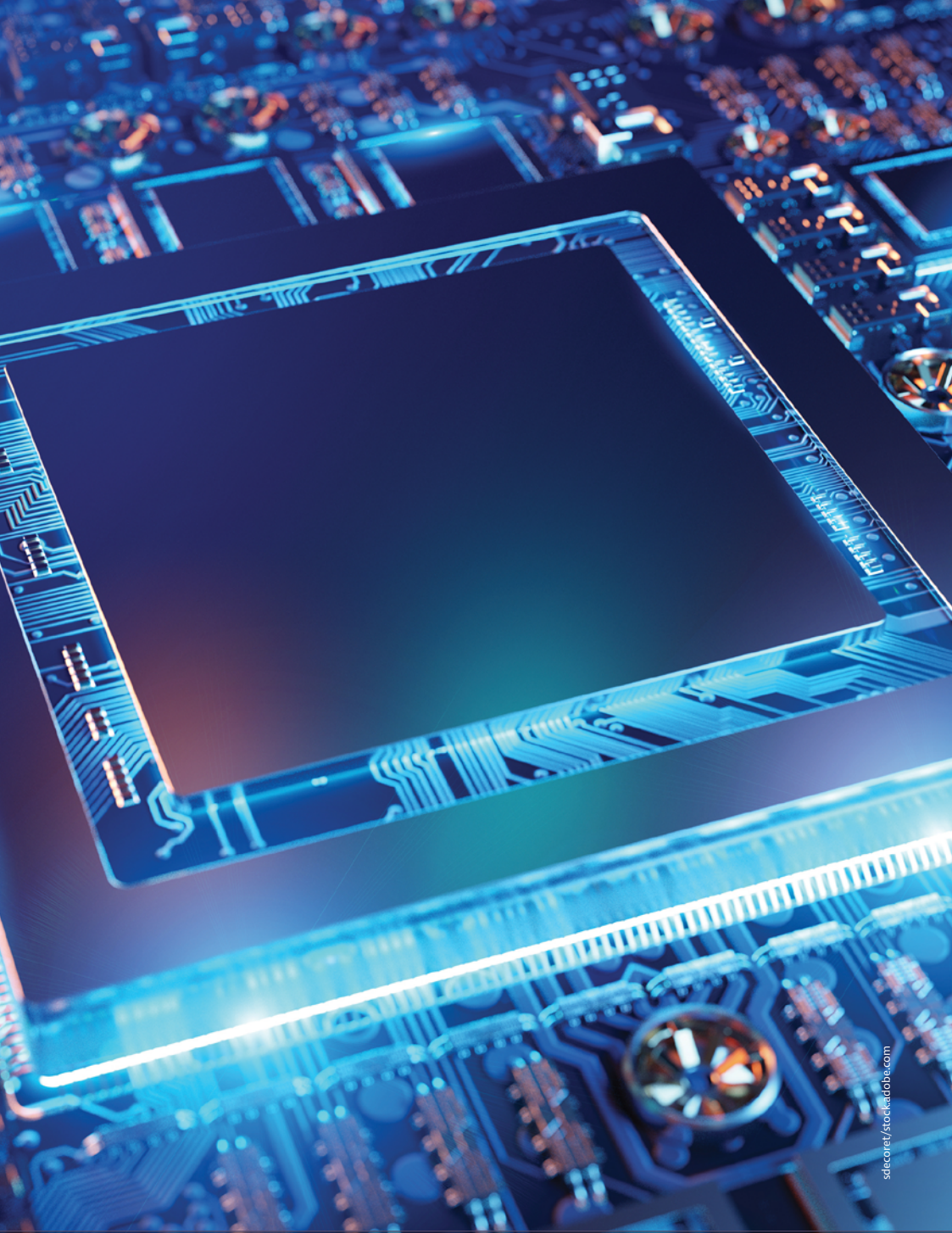
Like all assurance activities, computing hardware assurance relies on information. It can be difficult to get this information for hardware that is sold commercially in large volumes.

Access to designs and process based information can be particularly challenging. This can place greater emphasis on 'black-box' product based testing. For example, more algorithm inputs may have to be used in the testing process. It may also require tests to be deliberately designed to target hardware features, rather than algorithm ones. However, computing hardware is very complex, so this type of testing can be very difficult.

These limitations mean testing is most likely to provide confidence in the specific combination of the algorithm, the data and the hardware. This can only provide limited confidence for using the same hardware with different algorithms, or indeed the same hardware and algorithm but with different data (for example when the system is used operationally). Every time we change something, (whether data, algorithm or hardware), we need to keep re-testing.

It can take a lot of computing hardware resource to train an AI algorithm. Often, it is not possible to include all of this resource in an autonomous system. Consequently, training hardware and operational hardware are often different.

For example, to increase efficiency, operational hardware might represent numbers in less precise ways, which use less space in the computer's memory. Issues arising from different types of hardware need to be considered in the Assurance Case. If possible, it can help to conduct most (or all) of the algorithm tests on the operational hardware.



sdecoret/stock.adobe.com

Assurance of data

For simplicity, we are referring here to 'training data'. Training data is used during the development of Machine Learning algorithms, to create an approximate mapping between inputs and outputs. This could be a large data set of labelled images. Alternatively, training data could be generated 'on demand' from simulations in a synthetic environment.

Whatever form it takes, because of the approximate mapping it creates, training data partially expresses requirements. It may say, 'pictures like this should be classified as cats'. Or, it might say, 'when approaching an obstacle, slow down'. We want the AI to learn the general behaviour expressed by the specific examples in the training data.

Funtap/stock.adobe.com

If the data doesn't display the intended behaviour then the algorithm won't display it either. Likewise, relationships we don't want (for example gender or ethnicity based discrimination) that are in the data may be captured by the algorithm. For these reasons, any Assurance Case needs to pay particular attention to training data.

As well as data to train the AI, we also need data to test it. If we used the same data for both purposes, it would be like giving a student both the exam paper and the correct answers! Training and test data need to be related, though. Otherwise, it would be like asking a student who has studied French to pass an exam on the History of Art!

From the perspective of an Assurance Case, the data used in testing needs special attention. Badly formed test data can give us too much, or too little, confidence in the algorithm's performance. Neither of those outcomes is attractive.

As well as the data itself, the Assurance Case also needs to consider the way data has been collected and pre-processed. For example, if labels are being applied to input images for training purposes (this is a cat, this is a dog, this is a chocolate cake) then we need assurance that these labels are correct. We could do this by having the same set of images labelled by several different people and checking for differences. Obviously, this increases cost.

Alternatively, we could train several algorithms on different parts of the training data. Again, this increases cost, albeit cost of training time rather than cost of manual labelling. It also needs a reliable way of determining which algorithm is best. This is partly about algorithm assurance. It may also relate to integration (or architecture), where multiple algorithms are used in an ensemble. Both of these topics are discussed later in this book.

It can be difficult to collect all of the data we require. Sometimes specific items are too costly to obtain. Other times, they may not be obtainable in a safe manner. There are a number of ways of extending, or augmenting, a data set to address these issues. Simulations can be used to create artificial (or synthetic) data items and existing data can be manipulated to provide additional data. However it is done, we need to be confident that this augmentation does not introduce any unwanted characteristics into the data.

Collecting, pre-processing and augmenting data is an expensive business. This can make pre-generated data sets look very attractive and they can be used to good effect. The possibility of the data set containing malicious items needs to be considered, though. This is one reason why properly curated data sets, which come complete with their own Assurance Case, are very valuable artefacts.

Having spent so much in acquiring and preparing our data, we need to keep it safe. We need to protect it against unintentional and unauthorised changes. More specifically, we need to place it under configuration control. This is important because only a small amount of data needs

to be maliciously changed to produce a backdoor vulnerability into a model trained on the data.

And, if you're wondering, a backdoor provides an easy way for an adversary to maliciously change an algorithm's behaviour. Since these changes only happen in very specific circumstances, they can be very hard to spot. So, it's best to avoid situations where other people can put their own backdoors into our systems!

In addition to all of the points discussed so far, the data also needs to be relevant to the problem. An algorithm trained on images of cats is unlikely to be good at recognising road traffic signs. More generally, a data set that is relevant for one application is not necessarily relevant for another.

This limitation applies regardless of any assurance that can be made about the data set. No matter how much confidence we have in the data, applying it to the wrong problem will not be helpful.

For ethical and (potentially) legal reasons, the data should be free of unintentional bias. A number of tools have been developed to help identify bias: AI Fairness 360 is a good place to look. In some cases, the data may be personally or commercially sensitive, or sensitive for some other reason. If so, the data needs to be protected appropriately.

The possibility of an attacker being able to retrieve the training data from a deployed algorithm should also be considered. In some situations, analysing a surprisingly small number of algorithm runs can yield a worryingly large amount of data.

Overall, our Assurance Case needs to have a lot of data about our data!

Assurance of algorithms

For our purposes, an algorithm is something that maps an input to an output. It could be a deep neural network, which recognises pedestrians in an image. Or, it could be a 'random forest' that identifies which features in a data set are most likely to affect whether a loan is repaid.

Assurance of algorithms is based on a series of principles:

- **Firstly, understand what the system needs from the algorithm**. While this sounds simple, in reality it can be very difficult. A key strength of AI is its ability to solve open, poorly defined problems. So, as highlighted earlier, we might only have vague, ambiguous requirements

Rather than cover everything, it may be easier to focus on just those requirements associated with safe operation of the system.

Similarly, rather than describe everything that may be permissible, it may be easier to focus on those things that should not occur.

Blue Planet Studios/stock.adobe.com

- **Secondly, relate the system-level requirements to the implementation of the algorithm**. For traditional software, including automated systems, this relationship is a hierarchical, traceable decomposition. This decomposition provides a clear mapping between requirements at different levels, with lower levels having more detail. AI algorithms do not have the same level of traceability. Instead, a significant proportion of the requirements are imprecisely captured in the training data.

  Requirements can also affect the choice of algorithm: convolutional neural networks may be good for one type of problem; random forests may be good for another. They may also affect specific features of the algorithm, for example, the number of layers in a deep neural network.

  One key factor, which is often overlooked, is the relationship between requirements and the loss function, which helps a training algorithm decide which of 2 choices is better. For example, in some situations, all incorrect outputs are equally bad, whereas in other situations there are some incorrect outputs we definitely want to avoid: mistaking a stop sign for a 50 miles per hour speed limit sign is never a good idea. These 2 cases call for different types of loss function.

- **Finally, show the implementation is appropriate**. This may be influenced by the software toolkits that are used.

  Since these toolkits control the algorithm that is produced, we need to have confidence in them.

  Showing the implementation is appropriate typically involves some form of testing. Deciding how many tests need to be conducted is difficult. For the most critical algorithms, this testing should be done using independent data by an independent team.

  If all of the tests give the expected results then we're happy. If they don't then it can be difficult to know how the algorithm should be changed. This is, by the way, another significant difference between AI algorithms and 'traditional' software.

These 3 principles chain together. An appropriate implementation that is appropriately related to the system-level requirements ought to do what the system needs. In theory, that sounds straightforward. In practice, deciding what 'appropriate' means for any particular algorithm, used in any particular system, is still a big challenge.

One small part of 'appropriate' is showing that enough tests have been run. Here, it can help to distinguish between different types of input. Inputs fall into one or more of the domains listed below. Considering each of these domains in turn can allow testing to be focused on the particular aspects that are most important for a specific algorithm in a specific system context.

Input domain: These are inputs that the algorithm can accept. Anything that can be passed into the software routine that implements the algorithm is in this domain. From the perspective of the problem that the algorithm solves, many of these inputs are nonsensical.

Operational domain: This is part of the input domain. These are inputs that the algorithm would expect to receive during operational use. The desired algorithm behaviour for these inputs should be understood, even if it cannot be precisely described without considering each input separately.

Failure domain: This is part of the operational domain: These are inputs that the algorithm may receive if there are failures in related sub-systems. Pictures that come from a faulty camera are a possible example.

Adversarial domain: This is also part of the operational domain. These are inputs that the algorithm may receive if it is being deliberately attacked. This concept is related to, but wider than, the 'school bus looks like an ostrich' notion of adversarial inputs.

Figure 5. Potential algorithm input data domains[7]

We often talk about how well inputs have 'covered' a domain. A greater number of evenly spaced inputs provides greater coverage, for example. Most coverage arguments are based on mathematical (or 'syntactic') considerations. Coverage that is contextually meaningful (or 'semantic') is also important. This may mean you need roughly the same number of images taken in different types of environmental conditions.

In general, it can be difficult to tell the difference between software that actually works and software that just looks like it's working. The nature of AI algorithms makes this even more difficult. We don't have an easy way of describing what the correct output should be, so we can't automatically run lots of tests!

This means we need to take special care when measuring an algorithm's performance. Knowing how many inputs from a test data set are classified correctly is important but only provides confidence in performance for that data. It might not provide useful insights into when and how failures might arise for other data, like that received during operational use.

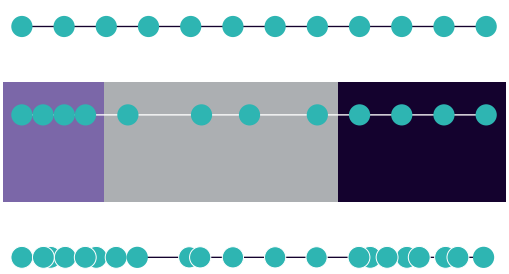


Figure 6. The difference between syntactic (mathematical) and semantic (contextual) coverage

Additionally, some wrong answers might be more important than others. Saying a diseased person is healthy could be worse than saying a healthy person is diseased. However, treatment can have significant costs, both financial and personal, so the latter error (or misclassification) is not problem free. Even though one misclassification might be more important, we need to understand each type of misclassification to properly measure algorithm performance.

Algorithm testing often relies on simulations, or other types of synthetic test environment. If we are to have confidence in the algorithm, we need to have confidence in the test results. So, we need to have confidence in the test environment. Otherwise, the algorithm might be getting great marks, but in the wrong subject!

A synthetic test environment needs to be a suitable representation of the algorithm's operational environment. Demonstrating this can be challenging. Also, just because a test environment is suitable for one algorithm, it is not necessarily suitable for others. That said, as with training data, test environments that are applicable to a number of algorithms (and systems) and that have their own Assurance Case will be very valuable assets.

Finally, a word of caution. It can be tempting to train an algorithm to achieve the highest performance possible against a specific set of test data. This can cause the algorithm to 'over fit' to specific data items and lead to a reduction in the operational performance where the input data is likely to be different to that experienced during development. This is called loss of generalisation. The balance between performance against test data and algorithmic generalisation should form part of the Assurance Case.

Assurance of integration

As noted before, we view an algorithm as mapping an input to an output. To be useful, the algorithm needs to be integrated, or 'wired into', a system. The way we do that integration can make it easier to use AI algorithms in autonomous (and other) systems.

Some features of the integration are about tolerating failures. These could occur in the sub-systems that provide inputs to the algorithm and a system still needs to operate safely, even if a camera breaks! Depending on the sub-system and the nature of the failure, the integration may be able to provide a replacement input. Even if this replacement cannot be maintained for a long time, it may be just enough to allow the algorithm to keep working so that system can reach a stationary safe state.

Another type of failure occurs when the algorithm receives an input that does not match its training data. The notion of 'match' here is deliberately loose. It is not always easy to decide whether an input is close enough to the training data for the algorithm to be safely used.

Failures can also be detected by looking at the algorithm's output. Some architectures include several algorithms that perform the same function. Comparing the results from each algorithm (or each channel) can spot cases where something's gone wrong. This type of approach is used for aircraft flight control systems, albeit with 'traditional' software rather than AI.

Other integration features include a simple monitor function, which can

iStock.com/enot-poloskun

replace a failed output with a default value, for example. As before, this might be just good enough for the system to reach a stationary safe state.

As well as increasing safety, integration features can also reduce the assurance demands placed on an individual AI-based component. If we can place significant confidence in a 'traditional' monitor then we might not need so much confidence in an AI-based algorithm. We'd still need some confidence, though. A system that spends most of its time in a stationary safe state isn't really that much use!

Since they provide the inputs to the algorithm and they receive the algorithm's outputs, integration features are also important for recording information. This can support maintenance and future development. It will also be important in investigating any incidents that may occur.

Assurance of systems

For our purposes, the system is the thing that is seen by the end user. It's the thing that actually does something useful. In many cases, the end user is a human being. In some cases, it may be another system.

Importantly, although it is easy to picture autonomous systems as physical assets (for example a car or medical robot), they can also be virtual systems that monitor and respond to non-physical events (for example raising security alerts in response to social media posts).

Understanding the system's intended operating environment is important. We wouldn't wear clothes designed for the North Pole on a tropical island beach. Similarly, we shouldn't expect an autonomous system designed for American Interstates to perform well on British country roads. As for 'traditional' systems, the intended operating environment should be captured in the Assurance Case.

kinwun/stock.adobe.com

Andrea Geiss/stock.adobe.com

Ways of detecting use outside this environment and responses that maintain safety should also be included. Of course, this may be difficult. By their nature, autonomous systems are designed to operate in uncertain, loosely defined environments. Keeping safe in all circumstances is a challenge!

People are an important part of the environment too. We need to consider how the system will interact with them. We also need to be confident that system safety does not rely on superhuman efforts from its users. Expecting instant reactions in a confusing situation is both unfair and unsafe.

The environment may also include bystanders. These people need to be protected against possible negative effects of an autonomous system. This may be especially important, as this group is less likely to directly benefit from the system's positive effects. More generally, benefits for one group of people should not come at a disproportionate cost to another group. More simply, my fun should not come at your expense!

Even if we have good understanding of the environment, it's not enough. We need to couple this with a description of what 'safe' means in this context. If the description is too narrow then problems can arise. A stationary vehicle may be safe if it is parked in a designated space. It may not be safe if it is stationary in the middle lane of a motorway or on a level crossing!

When thinking about what 'safe' means we need to take a wide perspective. We need to cover all aspects of the environment. This includes unexpected things that may reasonably be present, for example, livestock on roads. It also includes expected things not being present, for example, lane markings that are missing or significantly degraded.

Now that we understand the environment, and we understand what 'safe' means in this context, we can describe how the system should behave. Behaviours that can be phrased as 'the system should always' or 'the system should never' are particularly valuable from an assurance perspective.

System-level assurance should also include 'non-mission' behaviour. Ensuring that systems can be maintained safely is important. So, too, is protecting against 'The Bad Guys' when the system is at rest, as well as when it is in use.

This discussion shows that system level assurance is about two things:

- Firstly, providing confidence that the operational environment and the desired system behaviour are understood.
- Secondly, demonstrating that the system behaves as we would want.

The first part is difficult. So, too, is the second. Arguing that enough demonstration evidence has been generated is particularly challenging. As was the case for algorithm assurance, it can be difficult to show the suitability of any simulations used in system level testing.

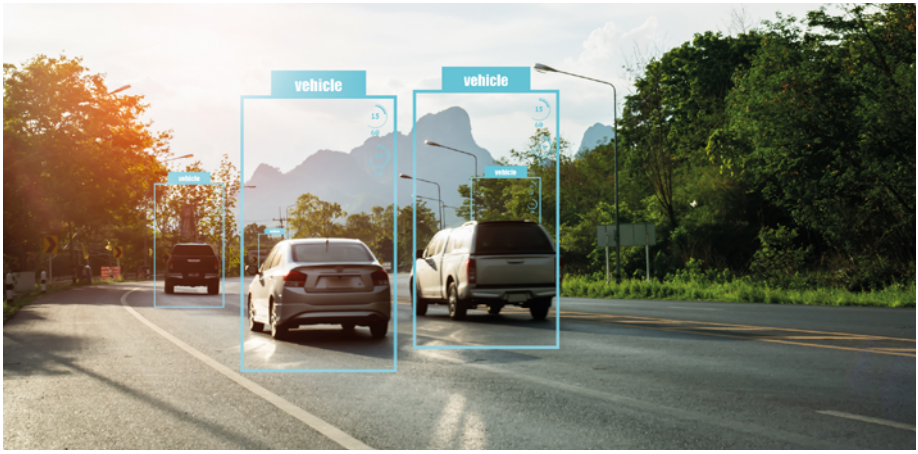
Assurance of systems-of-systems

Autonomous systems can react much faster than humans. They also tend to be much more predictable than humans and they are typically less aware of what's going on. This means interactions between autonomous systems and humans will not be the same as interactions between autonomous systems. The two types of interaction will exhibit different characteristics.

In some cases, autonomous systems may be designed to work alongside each other (the systems could be the same, or they could be different). In such cases, assurance relies on exploring the combined behaviour. This is not easy, as there is often a large input space that needs to be explored. Understanding the behaviour of one system is challenging enough. Adding more systems makes the problem exponentially harder.

In other cases, autonomous systems may share the same environment, but may not have been deliberately designed to work together. Here, unexpected results can emerge. Predicting these before they happen is challenging. Resolving them can be fraught with difficulties. If each system works fine in isolation, and the problem only occurs when they come together, whose fault is it? Which system should be changed to resolve the issue?

Here it may be tempting to suggest that all systems should behave 'benevolently' (for example, 'give way' to other systems) but this can itself cause problems as inaction can be as dangerous as interference. As yet, there are no good answers to these questions.



Yingyaipumi/stock.adobe.com

sdecoret/stock.adobe.com

Assurance in the presence of adversaries

There are a range of actors, with a range of motivations, who wish to do us harm. Rogue nations and terrorist organisations may wish to disrupt system performance to gain advantage, damage property, or even harm people. Criminals may wish to extort money. Inquisitive hobbyists may not intend to cause harm, but they may inadvertently do so.

If we are to have confidence in AI algorithms and autonomous systems then they need to be protected. Many of the vulnerabilities that apply to traditional software and systems also apply to AI algorithms and autonomous systems.

These types of algorithm and systems also introduce new types of vulnerability and hence opportunities for attack. For example, only a small amount of data needs to be changed to introduce a backdoor into a trained neural network (allowing the adversary to alter the network's behaviour in specific circumstances).

Additionally, small changes to inputs, imperceptible to humans, can generate incorrect outputs. These are often referred to as 'adversarial inputs'. They are just one example of many different things that an adversary might choose to try to do.

Adversaries may also exploit the specific way an autonomous system perceives its environment to influence certain types of behaviour. For example, a system design that is purely 'benevolent' could be easily 'gamed' into not providing its intended capability.

Several activities can increase our confidence in protection against adversarial actions. For example, 'red teaming', where teams of people deliberately try to break a system, can be beneficial and should form an integral part of security-informed TEVV activities. Other examples include:

- Only using training data from trustworthy sources.
- Attempting to identify backdoors in trained algorithms.
- Checking for common weaknesses in redundant systems (for example where common training data has been used).
- Monitoring behaviour during operational use.

Summing up

Yingyaipumi/stock.adobe.com

Without assurance, we will not be able to benefit from the potential of AI and autonomous systems. The strengths of these technologies are precisely the things that make assurance difficult.

Difficult does not mean impossible, though. Progress can be made by adopting a structured approach that considers requirements, computing hardware, data, algorithm, integration, system and system-of-systems issues, along with potential adversarial actions and legal and ethical concerns.

Such an approach can document, in a rigorous fashion, the level of assurance available for a particular system or algorithm. Informed decisions can then be made.

In most cases, more assurance can be achieved, but at greater cost. This raises the question of, what level of assurance is proportionate to a particular level of risk? Or, how much is enough? This is, as yet, an unanswered question.

Traditional assurance approaches rely on recognised good practice and consensus, which is not yet available for AI algorithms and autonomous systems. Currently, arguments must be made on a case-by-case basis. Where sufficient confidence cannot be confidently argued in a compelling Assurance Case, it may be necessary to 'architect' safety control into the system via non AI-based technologies.

Finally, assurance is not a one-time thing. It needs to be revisited as environments, systems and algorithms change. As new approaches to AI emerge, new assurance techniques will also be required. As long as people want to use AI and autonomous systems, we'll need to keep assuring them!

Further reading

- Safety Assurance Objectives for Autonomous Systems. https://scsc.uk/scsc-153A
- Assurance of Machine Learning for use in Autonomous Systems. https://www.york.ac.uk/assuring-autonomy/guidance/amlas/

References

[1] Dstl, 2019. **The Dstl Biscuit Book, Artificial Intelligence, Data Science and (mostly) Machine Learning**. 1st edition, revised v1_2. Available from: https://www.gov.uk/government/publications/the-dstl-biscuit-book (accessed 25 November 2021).

[2] Dstl, 2020. **Building Blocks for Artificial Intelligence and Autonomy, a Dstl Biscuit Book**. DSTL/PUB126301 v1.0. Available from: https://www.gov.uk/government/publications/building-blocks-for-ai-and-autonomy-a-biscuit-book (accessed 25 November 2021).

[3] HMG, 2020. **Data Ethics Framework**. Available from: https://www.gov.uk/government/publications/data-ethics-framework (accessed 25 November 2021).

[4] LESLIE, D. 2019. **Understanding artificial intelligence ethics and safety: A guide for the responsible design and implementation of AI systems in the public sector**. The Alan Turing Institute. Available from: https://doi.org/10.5281/zenodo.3240529# (accessed 25 November 2021).

[5] JOBIN, A., IENCA, M. & VAYENA, E. 2019. **The global landscape of AI ethics guidelines**. Nature Machine Intelligence. 1, pp. 389–399. Available from: https://doi.org/10.1038/s42256-019-0088-2 (accessed 25 November 2021).

[6] HMG, 2021. **Ethics, Transparency and Accountability Framework for Automated Decision-Making**. Available from: https://www.gov.uk/government/publications/ethics-transparency-and-accountability-framework-for-automated-decision-making Dated 13 May 2021 (accessed 25 November 2021).

[7] ASHMORE, R., CALINESCU, R. and PATERSON, C. **Assuring the Machine Learning Lifecycle: Desiderata, Methods, and Challenges**. ACM Computing Surveys (CSUR), 54(5), pp.1-39, 2021.

[dstl]
The Science Inside

T +44 (0) 1980 950000
E centralenquiries@dstl.gov.uk

For more information about Dstl's work,
visit **www.gov.uk/dstl**

DSTL/PUB136185