# Assurance Argument Patterns and Processes for Machine Learning in Safety-Related Systems

**Chiara Picardi, Colin Paterson, Richard Hawkins, Radu Calinescu, Ibrahim Habli**
Assuring Autonomy International Programme, The University of York, York, U.K.

## Abstract

Machine Learnt (ML) components are now widely accepted for use in a range of applications with results that are reported to exceed, under certain conditions, human performance. The adoption of ML components in safety-related domains is restricted, however, unless sufficient assurance can be demonstrated that the use of these components does not compromise safety. In this paper, we present patterns that can be used to develop assurance arguments for demonstrating the safety of the ML components. The argument patterns provide reusable templates for the types of claims that must be made in a compelling argument. On their own, the patterns neither detail the assurance artefacts that must be generated to support the safety claims for a particular system, nor provide guidance on the activities that are required to generate these artefacts. We have therefore also developed a process for the engineering of ML components in which the assurance evidence can be generated at each stage in the ML lifecycle in order to instantiate the argument patterns and create the assurance case for ML components. The patterns and the process could help provide a practical and clear basis for a justifiable deployment of ML components in safety-related systems.

## Introduction

Current state of the art for Machine Learning (ML) focuses on improving the performance and efficiency of the machine learnt models (MLM). This has led to an increasing interest in their use as part of systems performing safety-related tasks, particularly through application to problems in healthcare, e.g. for clinical diagnosis (De Fauw et al. 2018) and automotive, e.g. for autonomous driving (Burton, Gauerhof, and Heinzemann 2017). For such domains, it is imperative that MLMs provide not only excellent performance, but also that the MLMs can be shown, with sufficient confidence, to perform the tasks safely, i.e. risk of human harm is as low as reasonably practicable. Safety cases provide an established approach for justifying the safety of systems through the provision of an explicit argument supported by compelling evidence (Assurance Case Working Group 2018). In (Picardi et al. 2019) we discussed the need for a assurance argument for MLMs through the consideration of MLMs developed to support the diagnosis of retinal diseases. We presented an initial argument pattern that captured the form of the assurance arguments that would be required to provide the necessary assurance of safety. In particular, that work highlighted the need for confidence arguments to support any performance claim regarding the MLM. We provided only a brief discussion of the nature of these confidence arguments. Argument patterns provide reusable templates of the types of claims, evidence, justification and contextual information that must be covered in a compelling argument. However, on their own, they neither detail the assurance artefacts, e.g. evidence of data coverage and representativeness, that must be generated to support the safety claims for a particular system and MLM, nor provide guidance on the specific activities that are required to generate these artefacts.

In this paper we therefore extend the existing work in a number of ways:

- We update the assurance argument pattern to better reflect the relationship between the MLM and the safety of the overall system and provide patterns for the confidence arguments that form a crucial part of the MLM assurance argument.
- We develop an assurance process built upon existing best practice. This process facilitates the instantiation of the confidence argument patterns through consideration of the required activities that should be undertaken and the artefacts that should be generated.

The initial application of the assurance argument patterns focused on clinical diagnosis (Picardi et al. 2019) and autonomous driving (Burton et al. 2019). However, the underpinning principles and processes are domain independent.

### Arguments for the Safety Assurance of MLMs

Figure 1 shows the assurance argument pattern for a MLM, refined from that presented in (Picardi et al. 2019). This pattern is for a single component, the requirements of which have been identified through consideration of the wider system context, including defined safety requirements derived from the system safety process. For example, in an automotive context, this will include the safety requirements derived from vehicle- and architecture-level hazard and risk analysis, some of which will be allocated to ML components.

The argument pattern is expressed using the Goal Structuring Notation (GSN) (Assurance Case Working Group 2018) as represented in Figure 2. The ML assurance claim

is that the the model satisfies the defined ML safety requirements in the defined operating environment. It is important to note that the claim does not focus on the performance of the MLM directly, but instead on the specific ML safety requirements that have been defined to reflect the contribution of the MLM to safety of the overall system (we discuss the ML requirements in more detail later in the paper). Similarly, the verification evidence that supports this claim must not simply be evidence of the performance of the model, but specifically evidence that shows the ML safety requirements are satisfied (again we discuss model verification in more detail later in the paper). It is critical therefore that this assurance argument for the MLM must be recognised as one part of a larger assurance argument considering the safety of the entire system or platform. This is illustrated in Figure 3 where the ML Assurance Claim is supporting a system-level assurance claim regarding the contribution of the ML component to the safety of the overall system. The dotted arrow indicates that this will in turn support other claims about the safety of the system as a whole. Figure 3 highlights how as well as arguing the assurance of the MLM, it is also necessary to support a number of claims regarding the deployment of the MLM, including the assurance of the integration of the MLM into the wider system, the monitoring of the component during operation, and assurance of any updates to the component after deployment. We discuss deployment further in the description of the process.

The ML assurance claim in Figure 1 is made in the context of artefacts that provide information regarding:

- The operating environment of the system in which the MLM will be deployed
- The defined ML safety requirements
- The developed ML Model
- The data used for development and testing of the MLM

For each of these items of context, an Assurance Claim Point (ACP) (Hawkins et al. 2011) is defined. As highlighted in (Picardi et al. 2019), these ACPs are crucial to the safety assurance of the MLM as they provide the arguments that should demonstrate confidence in each of these items. Without providing these confidence arguments, the case for the use of the MLM in a safety-related task would lack the necessary clarity and completeness. In Figure 1 we have also indicated the need for an ACP for the verification evidence to demonstrate its sufficiency.

To provide guidance on how to develop the confidence arguments for each contextual artefact (e.g data set, ML model etc.), we have developed a pattern that can be instantiated for these arguments. Figure 4 shows this pattern which could be used to support each ACP in Figure 1. In the pattern it can be seen that the confidence claim for each artefact is made by considering a set of activities that are undertaken in order to ensure that some defined desiderata are met. It is important that a sufficient set of desiderata are identified for each artefact. In previous work (Ashmore, Calinescu, and Paterson 2019) the authors have proposed sets of desiderata for the artefacts generated by each lifecycle phase. The strategy adopted is to explicitly show how each desideratum is satisfied. This is achieved by describing the activities that have been undertaken during the ML lifecycle that ensure the desideratum is achieved, and providing evidence from those activities to support the claim. For example, Figure 5 illustrates how the pattern could be instantiated to provide a confidence argument for the MLM (ACP 3 in Figure 1). The identified desiderata are that the model is performant, robust and interpretable. The argument demonstrates how these desiderata are met through the various activities performed as part of the model learning phase of the lifecycle, such as the selection and optimisation of hyperparameters or the augmentation of the training data. In the next section we discuss the phases of the ML lifecycle, as illustrated in Figure 6, and the required development and assurance activities, in more detail.

## Overview of ML Assurance Process

Figure 6 illustrates the machine learning assurance process that we propose as a basis for instantiating the argument patterns described in the previous section. As observed by Marcus and Davis in (Marcus and Davis 2019), "Trustworthy AI has to start with good engineering practices, mandated by laws and industry standards, both of which are currently largely absent". To this end, the process as shown is derived from existing practice based upon discussions with experts in organisations that are actively engaged in the development of machine learning components for safety-related tasks. This process is augmented with the identification of activities and artefacts that are required for the creation of a compelling assurance argument. The process is designed to encourage a more systematic and justifiable approach to the development of ML components.

The process comprises five stages and four swim lanes. The top and bottom swim lanes contain the artefacts which are generated as part of the assurance and development activities respectively. These artefacts can be used as evidence to support the claims in the assurance argument. The phases in the lifecycle are conducted sequentially with the previous stage reducing the possibility of failure in the next stage of the process. The middle two swim lanes contain those activities associated with development and assurance and illustrate that these activities occur in parallel. Solid arrows within a stage indicate a flow between activities and dotted lines indicate the flow of information into and out of those activities. For reasons of clarity, we have not shown in Figure 6 all of the flows between activities in different phases. It is important to note however that the ML lifecycle is an iterative process that may require multiple cycles through the different activities before an acceptable model is produced. We use two separate swim lanes to indicate the preference for independence between the development of the ML components and the assurance of the development process.

In the following sections we describe the process of assuring the machine learning lifecycle and the activities and artefacts associated with each process stage.

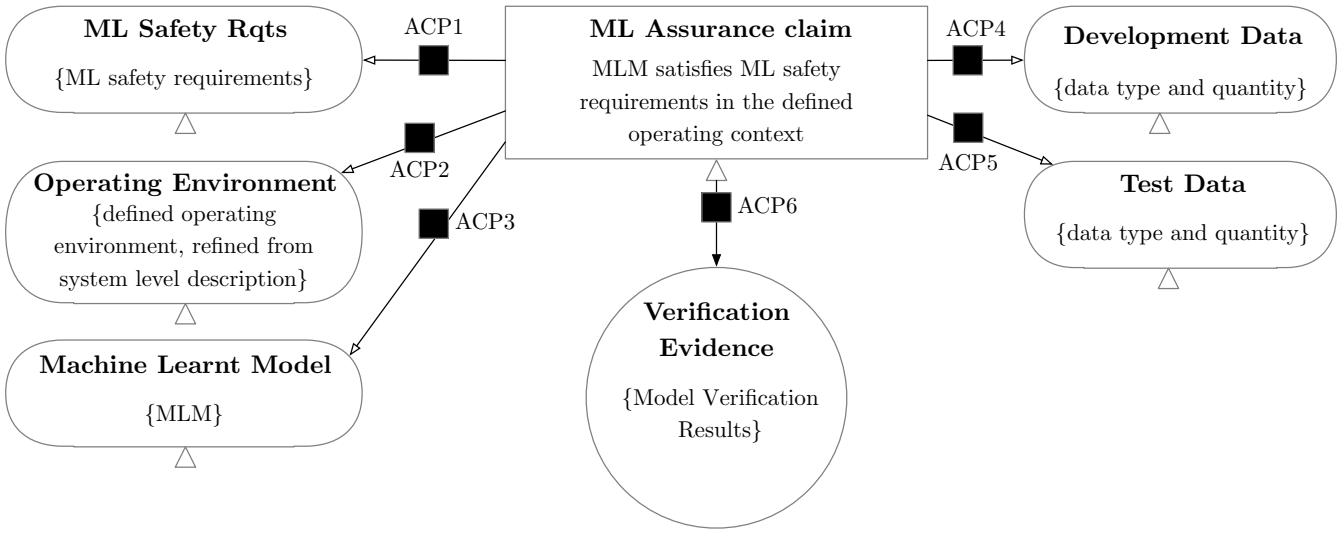


Figure 1: Assurance Argument Pattern for a Machine Learned Model

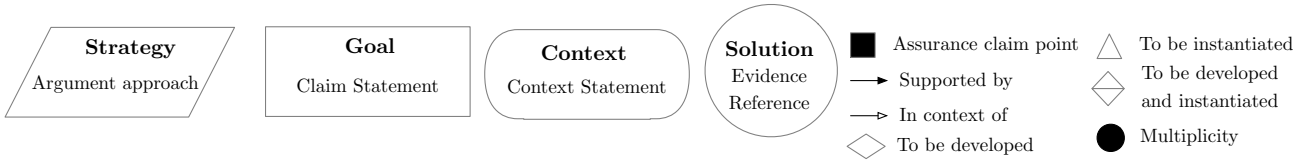


Figure 2: GSN Graphical notation

## Assurance of Requirements Elicitation (ACP1,2)

This phase of the ML lifecycle focuses on the specification of the ML safety requirements. The ML safety requirements must be developed to take account of the system safety requirements and the environmental and operating context of the system. The system safety requirements arise from the system safety activities (which are outside of the scope of this paper). The relevant system safety requirements will define the required behaviour of the ML component to ensure its contribution to system level hazards is acceptable within the defined operational context.

### Requirements Elicitation Activities

Activities in this phase determine the ML safety requirements that will be used throughout the rest of the ML lifecycle. It is important from an assurance perspective that the sufficiency of the defined ML safety requirements can be explicitly demonstrated. This is done through a confidence argument provided at ACP 1. In particular this argument must demonstrate that the ML safety requirements maintain the intent of the original system safety requirements (Hawkins, Habli, and Kelly 2013), are complete with respect to those requirements and to the defined operating conditions, and are unambiguous in their definition. The input to this process is the system safety requirements, a description of the intended operation of the system and the environment in which it will operate. The output is a set of ML safety requirements along with a requirements confidence argument instantiated from these assurance activities.

## Assurance of Data Management (ACP4,5)

Machine learning is a data centric development activity and any models produced are only as good as the data upon which they are based. The aim of the data management stage is to collect data that satisfies the defined data management desiderata, and to generate assurance evidence which demonstrates that they are satisfied. The data management desiderata we identified in (Ashmore, Calinescu, and Paterson 2019) are that the data is: relevant, complete, balanced, and accurate. The following activities are required to be undertaken in order to achieve this.

### Data Management Activities

During the data management stage the development team collects data sets that satisfy the data management desiderata. Table 1 shows examples of the data requirements relevant to the complete and balanced desiderata. Data may be collected from live or test systems and augmented through synthesis and data processing techniques. Typically two distinct sets are created. The development data is used to train the model and will be split into training and test data. The test data is used to assess the model‘s ability to generalise and reduce over-fitting. In parallel a third data set should be created by the assurance team, which will be used in model verification. This data set varies from the development test

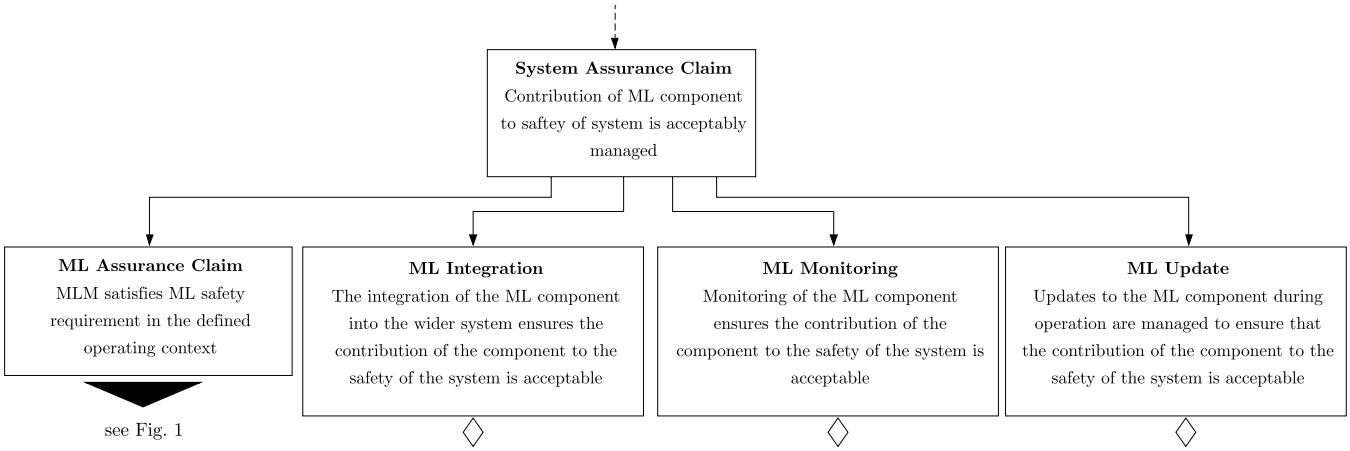


Figure 3: The MLM Safety Assurance Argument Within a Larger System Argument

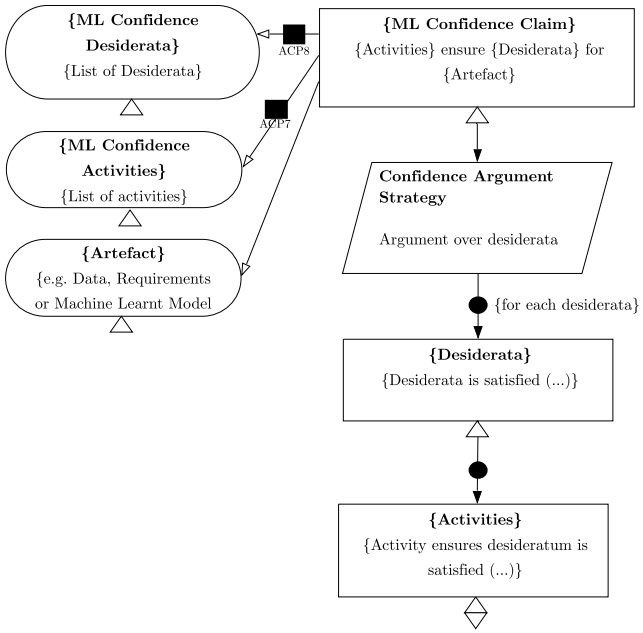


Figure 4: Confidence Argument Pattern for MLMs

set in two important ways. Firstly it will never be shared with the development team until a final published model has been created. In this way we avoid data leakage (Kaufman et al. 2012) and ensure that the generalisability claims of the model are strengthened. Secondly the assurance team take an adversarial approach, looking for difficult and rare edge cases. This type of verification testing is common in traditional software engineering. The assurance team also generates data assurance evidence through the examination of each data set.

The outcome of this lifecycle phase should produce the artefacts necessary for instantiating the confidence arguments for ACPs 4 and 5.

Table 1: Examples of requirements and associated assurance evidence for data management

| **Data Management** | |
|---|---|
| Original requirement | The component should be able to identify conditions from a scan obtained in UK hospitals |
| Desiderata | Completeness |
| ML requirements | Data should be gathered from all four of the models of MRI machines currently used in UK hospitals |
| Evidence | The training set contains scans from all four of these machines. |
| Original requirement | Pedestrians pushing bicycles have previously been found to lead to system failures. The system must be able to identify these cases. |
| Desiderata | Balanced |
| ML requirements | There should be sufficient samples in the training set showing pedestrians with bicycles. |
| Evidence | Exploratory data analysis shows that this subclass is not under-represented. |

## Assurance of Model Learning (ACP3)

Having provided assurance of the data to be used, the model learning phase aims to use that data to construct models which demonstrably meet the model learning desiderata, and to generate assurance evidence which demonstrates the sufficiency of those models. The model learning desiderata we identified in (Ashmore, Calinescu, and Paterson 2019) are that the model is: performant, robust, interpretable and reusable. The first three of these desiderata are the most relevant from a safety assurance perspective. The following activities are required to be undertaken in order to achieve this.

### Model Learning Activities

Model learning is a highly iterative process in which models are repeatedly created and then tested with the results of tests used to inform the next iteration of the development process. The model creation activity involves the selection of a model type and structure that the developer believes will satisfy the model learning desiderata. Table 2 shows examples of the model learning requirements relevant to the performant and

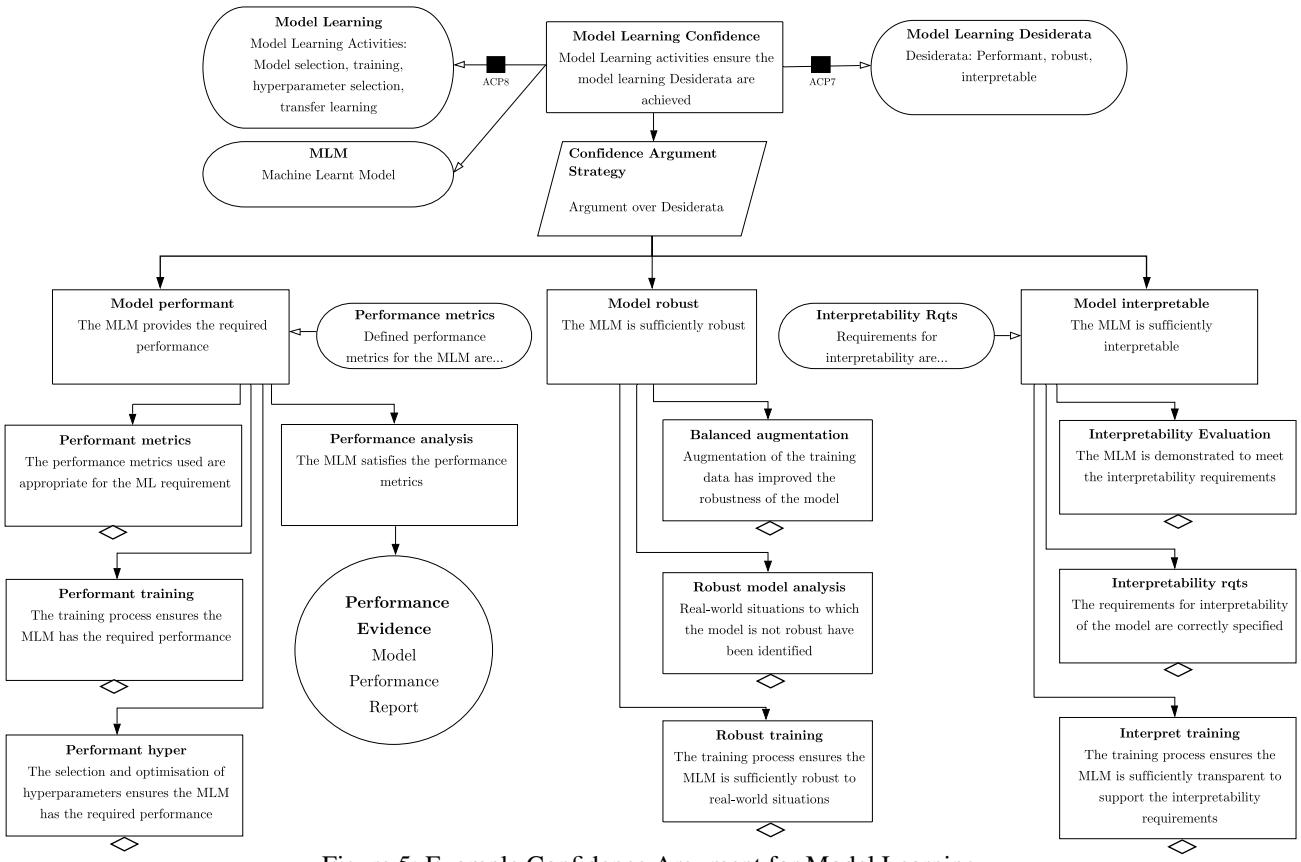


Figure 5: Example Confidence Argument for Model Learning

robust desiderata. The development data is repeatedly split into training and validation sets in order to learn model parameters such that errors in modelling are minimised. If a model fails to produce an acceptable level of performance against the development set then hyper-parameters (those parameters which control the learning process) are modified and the model creation process is repeated.

Once the model is able to meet the performance criteria with respect to the development data we move into the testing activity where the model is exercised with the internal test set. It is against this test set that the performance criteria are measured for comparison to the mandated acceptance criteria and a model performance report is generated. As well as data-centric testing the model may also be tested at this stage against acceptance criteria such as computational complexity and interpretability criteria.

The development of machine learnt components is commonly undertaken in an experimental manner with model structures and hyper-parameters varied by hand in an attempt to meet the, possibly numerous and conflicting, acceptance criteria. Such ad hoc processes are not acceptable for safety-related systems and as such we introduce an important development artefact, the development log. The development log serves a number of purposes. It encourages a systematic approach to development. Recording changes in software using version control repositories is common practice in traditional software development but is missing from most Machine Learning projects. The development log should document the changes that have been made to the training process at each iteration, with respect to the issues highlighted in the model performance report, as well as a rationale to support the choices made. A development process which is systematically documented in this way prevents unnecessary experimentation as well as allowing new team members to learn from failed experiments. From a safety perspective the development log provides evidence that activities have been undertaken to mitigate issues identified during the learning phase, e.g. over-fitting tackled by introducing dropout.

During the development and testing process we generate development assurance evidence through an examination of the development and testing process and the development log produced. The outcome of this lifecycle phase should produce the artefacts necessary for instantiating the confidence argument for ACP 3.

## Assurance of Model Verification (ACP6)

Once the development team have created a model which they believe meets the ML requirements the model verification stage is undertaken. In this stage the model is independently evaluated and a verification report generated which serves as an assessment that the the model will continue to

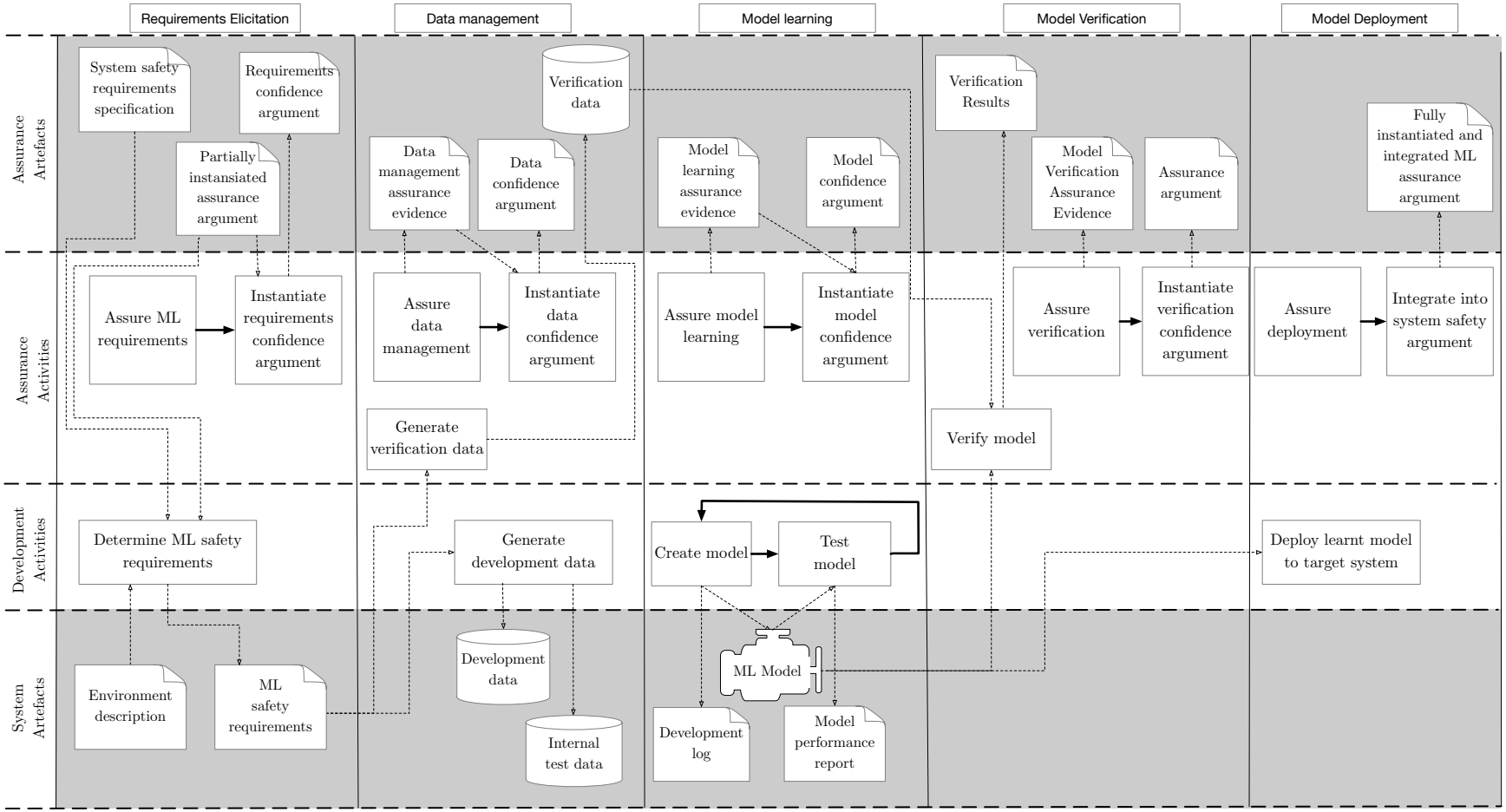


Figure 6: Stages and key artefacts of the Machine Learning Assurance Process

Table 2: Examples of requirements and associated assurance evidence for model learning

| **Model Learning** | |
|---|---|
| Original requirement | the impact of misclassification should be used to train and evaluate the ML perception function, e.g. misclassifying an adult as a child is better than misclassifying an adult as a tree. |
| Desiderata | Performant |
| ML requirements | The cost function used in training should reflect the safety implications of inter-class classification errors. |
| Evidence | Performance metrics calculated as a function of inter-class confusion. |
| Original requirement | Small scratches to the camera lens should not overly impact the accuracy of the classification. |
| Desiderata | Robust |
| ML requirements | When a scratch of no more than 1mm is present on the lens the accuracy of classification should $\geq 0.98$. |
| Evidence | The training data was augmented with synthetic scratches. Test data was generated in from lab based experiments in which scracthes where administered to a lens before data was gathered. |

Table 3: Examples of requirements and associated assurance evidence for model verification

| **Model Verification** | |
|---|---|
| Original requirement | The component should continue to operate when presented with adversarial inputs. |
| Desiderata | Comprehensive |
| ML requirements | For any given input, a small change to the inputs should not change the result returned by the component. |
| Evidence | Mathematical guarantees of robustness have been demonstrated using SMT. |
| Original requirement | Verification evidence should be understood by safety experts without an deep understanding of machine learning techniques. |
| Desiderata | Comprehensible |
| ML requirements | Any failures reported by formal verification techniques should facilitate remedial action. |
| Evidence | SMT analysis of the model reports bounds on robustness which have been mapped to limits in the input space. |

satisfy its requirements when exposed to inputs which were not available during training. The verification should be conducted in such a manner that the defined verification desiderata are shown to be met. The verification desiderata we identified in (Ashmore, Calinescu, and Paterson 2019) are that the verification is: comprehensive, contextually relevant, and comprehensible. In addition it must be demonstrated that the verification activities are performed independently of the development process. The verification activities are discussed below.

### Model Verification Activities

During the verification phase, the assurance team is responsible for performing verification activities that provide a sufficient independent evaluation of the MLM. Table 2 shows examples of the verification requirements relevant to the comprehensive and comprehensible desiderata.

Testing and verification consists of three sub-tasks. Firstly independent requirements encoding ensures that any assumptions made by the development team are validated and any ambiguity in the requirements are highlighted.

Secondly test-based verification is undertaken. Since the development test data has been used to inform interactive model development this can no longer be considered an independent data set. The verification stage must, therefore, use the verification data set gathered during the data management stage. Evaluation of the model with respect to this independent test set allows for confirmation that the results reported in the model learning stage generalise beyond the development data and demonstrate a form of robustness concerned with over-fitting. The assurance data set includes additional samples which have been collected from challenging, or rare, input spaces. Testing against this set will therefore provide a measure of robustness to known cases. Where collecting such data is impossible, contextually relevant data augmentation may be used to provide evidence of robustness to variations in the input space, for example adding synthetic samples of terminal failure.

Finally formal verification may be undertaken, using mathematical techniques to provide irrefutable evidence that a model satisfies formally-specified properties. Such a process could allow for unknown unknowns to be considered. Whilst mapping such cases to contextually relevant inputs is often challenging for high dimensional input spaces such techniques provide guarantees of robustness for a class of input which may be important due to environmental conditions.

Although testing the MLMs is the most common approach to verification, it is for the assurance team to decide what combination of testing and formal verification may be required. This will depend upon the nature of the model, the system and the properties that are to be verified.

The output of this phase is not just the results of the individual tests that are undertaken, but also a verification report that documents and justifies the verification activities that are undertaken. In this way we provide evidence that the results obtained can be trusted. The outcome of this lifecycle phase should produce the artefacts necessary for instantiating the confidence argument for ACPs 6.

## Model Deployment

Once the MLM has been developed and verified, the model deployment phase considers how that model can be justifiably deployed and utilised within a system. This phase of the lifecycle is crucial to the completeness of the overall assurance case for the MLM. This is an area requiring further research and will be the focus for much of our future work particularly from a safety-critical perspective. There are important assurance considerations during this phase. Firstly, the integration of the MLM must consider the architecture of the system into which it is being deployed. This will include linking the MLM with the system inputs, such as from sensor systems, as well as protecting the wider system from hazardous effects that may arise from any incorrect outputs

from the MLM. There are many architectural approaches that could be used for this, such as validity checkers or aggregators. We discuss integration strategies in more detail in (Ashmore, Calinescu, and Paterson 2019).

Secondly it is important to monitor the MLM during operation to ensure that it continues to perform as intended once deployed. In particular we must monitor for any deviations that could lead to violation of system safety requirements. It is important not just to monitor the outputs of the MLM, but also to monitor the environment (to check assumptions remain valid), to monitor the inputs provided to the MLM (e.g. to check these are within acceptable bounds), and to monitor the internals of the MLM (to identify failure events).

Thirdly it is expected that, like all software, MLMs will be updated during their lifetime. Although the frequency of those updates may vary between models and between systems, it is crucial that any updates are managed to ensure that potentially hazardous errors are not introduced, and that evidence to demonstrate this is provided. Some MLMs may be updated as part of an on-line learning process, for example with a reinforcement learning-based model. Such cases bring with them additional and substantial open assurance challenges, and we do not currently anticipate the use of online learning for safety-critical systems.

## Conclusions

This paper presented a machine learning process that covers both the development and assurance of the MLM and described the types of evidence and arguments that are necessary to generate a compelling and credible assurance case. Importantly, the process shows how such a case can be developed incrementally, where each lifecycle phase in the process contributes the relevant contextual and evidential artefacts, e.g. concerning the quality and quantity of the training and verification data.

Several areas for further work exist, three of which are as follows. Firstly, the process by which the ML requirements are generated remains an open challenge. Particularly for safety-related tasks, these requirements should be generated from rigorous hazard and risk analysis that considers the overall system risks and system safety requirements, including factors such as the criticality of the domain, the novelty of the application and the level of autonomy associated with the use of the system. This would dictate not only the types of performance requirements needed but also socio-technical factors such as the degree of explanation expected by the users and policy makers. Secondly, our process focuses on the generation and verification of the MLM. However, this has to be considered in the context of the wider systems architecture, e.g. considering the availability and performance of the sensing technology and the level of redundancy in the system. Take scanning technologies in the healthcare domain for example, the output of which would dramatically change the performance of the used MLM in clinical diagnosis. Thirdly, the evaluation of the process is on-going, largely in collaboration with our industrial partners that represent different critical domains (healthcare, automotive and manufacturing). This comprises both lab-based experiments, largely based on historic and synthetic data, and pilot applications, which expect the MLM outputs to shadow more conventional technologies and human-based decision making. In this respect, we expect the assurance argument and evidence for the MLM to evolve based on real-time feedback from the environment, the overall system and the actual output and performance of the MLM (Calinescu et al. 2017).

**Acknowledgements.** This work is funded by the Assuring Autonomy International Programme https://www.york.ac.uk/assuring-autonomy.

## References

Ashmore, R.; Calinescu, R.; and Paterson, C. 2019. Assuring the machine learning lifecycle: Desiderata, methods, and challenges. *arXiv preprint arXiv:1905.04223*.

Assurance Case Working Group. 2018. Goal Structing Notation Community Standard version 2. https://scsc.uk/r141B:1?t=1. Accessed on 11/13/2018.

Burton, S.; Gauerhof, L.; Sethy, B. B.; Habli, I.; and Hawkins, R. 2019. Confidence arguments for evidence of performance in machine learning for highly automated driving functions. In *International Conference on Computer Safety, Reliability, and Security*, 365–377. Springer.

Burton, S.; Gauerhof, L.; and Heinzemann, C. 2017. Making the Case for Safety of Machine Learning in Highly Automated Driving. In *International Conference on Computer Safety, Reliability, and Security*, 5–16. Springer.

Calinescu, R.; Weyns, D.; Gerasimou, S.; Iftikhar, M. U.; Habli, I.; and Kelly, T. 2017. Engineering trustworthy self-adaptive software with dynamic assurance cases. *IEEE Transactions on Software Engineering* 44(11):1039–1069.

De Fauw, J.; Ledsam, J. R.; Romera-Paredes, B.; Nikolov, S.; Tomasev, N.; Blackwell, S.; Askham, H.; Glorot, X.; O'Donoghue, B.; Visentin, D.; et al. 2018. Clinically applicable deep learning for diagnosis and referral in retinal disease. *Nature medicine* 24(9):1342.

Hawkins, R.; Kelly, T.; Knight, J.; and Graydon, P. 2011. A new approach to creating clear safety arguments. In *Advances in systems safety*. Springer. 3–23.

Hawkins, R.; Habli, I.; and Kelly, T. 2013. Principled construction of software safety cases.

Kaufman, S.; Rosset, S.; Perlich, C.; and Stitelman, O. 2012. Leakage in data mining: Formulation, detection, and avoidance. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 6(4):15.

Marcus, G., and Davis, E. 2019. *Rebooting AI: Building Artificial Intelligence We Can Trust*. Pantheon.

Picardi, C.; Hawkins, R.; Paterson, C.; and Habli, I. 2019. A pattern for arguing the assurance of machine learning in medical diagnosis systems. In Romanovsky, A.; Troubitsyna, E.; and Bitsch, F., eds., *Computer Safety, Reliability, and Security*, 165–179. Cham: Springer International Publishing.