Safety of Artificial Intelligence: A Collaborative Model*

John McDermid[1] , **Yan Jia**[1]
[1]Department of Computer Science, University of York, York, UK
{john.mcdermid, yj914}@york.ac.uk

Abstract

Achieving and assuring the safety of systems that use artificial intelligence (AI), especially machine learning (ML), pose some specific challenges that require unique solutions. However, that does not mean that good safety and software engineering practices are no longer relevant. This paper shows how the issues associated with AI and ML can be tackled by integrating with established safety and software engineering practices. It sets out a three-layer model, going from top to bottom: system safety/functional safety; "AI/ML safety"; and safety-critical software engineering. This model gives both a basis for achieving and assuring safety and a structure for collaboration between safety engineers and AI/ML specialists. The model is illustrated with a healthcare use case which uses deep reinforcement learning for treating sepsis patients. It is argued that this model is general and that it should underpin future standards and guidelines for safety of this class of system which employ ML, particularly because the model can facilitate collaboration between the different communities.

1 Introduction

There is a growing recognition of the challenges posed by the use of artificial intelligence (AI), and more specifically machine learning (ML), in safety-critical systems. These challenges have been recognised by the AI community and there have been several influential publications, e.g. on "concrete problems in AI safety" [Amodei *et al.*, 2016], which have drawn the community's attention to the potential for harm arising from undesirable, and unanticipated, learnt behaviour of ML-based systems.

In parallel, the safety community has been considering the impact of AI and ML on system safety with an emphasis on autonomous systems (AS), e.g. including the "gaps" in engineering processes that arise from the use of AI and ML [Burton *et al.*, 2020]. In practice, development of safety-critical systems is strongly influenced by standards but they are slow to produce and often substantially lag technological developments. This situation is beginning to change, particularly related to autonomous vehicles (AVs) and, for example, the British Standards Institution (BSI) has an active programme developing Publicly Available Specifications (PAS) for AVs and in the USA a standard, UL 4600 [ANSI & UL, 2020], has recently been published.

Thus, both communities are very active but our experience, e.g. through the Assuring Autonomy International Programme (AAIP), suggests that the initiatives in the safety community are having limited impact on the AI community, and that the safety community is struggling to come to grips with the subtleties and complexities of ML. Part of the reason for this is the absence of common terminology and a framework that enables the two communities to collaborate. The intent here is to provide a model and some terminology which can facilitate greater collaboration.

Against this background, the paper is organised as follows. Section 2 reviews some of the rather "disconnected" activities in the AI and safety communities, and identifies foundations for our collaborative model. In section 3 the model is presented, showing how long-established safety-critical software development principles can support "AI safety" activities, which in turn support system/functional safety. This model is illustrated in section 4 using a case study from healthcare. Section 5 presents a discussion, particularly considering how the model might be developed and utilised to support collaboration between the AI and safety communities and this leads into the conclusions in Section 6.

2 Safety of AI: Two Communities

This section aims to identify both differences in views of the AI and safety communities and a basis on which to build the collaborative model. Both communities run regular "AI Safety" events. However, the attendance at such events is skewed, so they don't really act as a meeting of minds. Rather than viewing this as a sociological problem, it is more helpful to consider the differences in technical viewpoints between the communities to build the collaborative model.

The AI community viewpoint is considered first, then the safety perspective is introduced in two parts. First, we consider broad safety processes, then we consider "good practice" in safety-critical software engineering.

2.1 "AI Safety"

There is a growing interest in "AI safety" and many AI practitioners and researchers have sought to identify and address potential problems. We start with a seminal paper which sets out "concrete problems in AI safety" [Amodei *et al.*, 2016] arising from the use of reinforcement learning (RL). A brief introduction to RL is given in section 4.1; here all we need to know is that with RL the system learns by getting "rewards" from interaction with the environment and seeks to maximise its reward to achieve the desired behaviour, represented as a "policy". The paper uses a cleaning robot as an example and identifies undesirable (in safety terms, hazardous) behaviours for the robot. The description of these undesirable behaviours given here is intended to be general in terms of AI technologies and their applications:

1. Avoiding negative side effects – ensuring that the behaviour meets safety constraints; from a safety perspective this equates to avoiding hazards;
2. Avoiding reward hacking – ensuring that the behaviour doesn't get a high reward by "gaming" and producing solutions that are valid in some literal sense but don't meet the designer's intent;
3. Scalable oversight – this concerns the ability of the human to interact with the system both to monitor it and to respond to requests for confirmation of decisions prior to actions being taken;
4. Safe exploration – when the system needs to try new things to learn a better solution, how can negative outcomes (i.e. new hazards) be avoided;
5. Robustness to distributional shift – how the system adapts to changes in the operational environment, e.g. for a medical diagnosis system which is developed using an Asian population but deployed in Europe.

These problems can arise from a number of underlying causes, e.g. inappropriate reward functions and mis-specified feature spaces. Work at DeepMind [Leike *et al.*, 2018] identified similar problems to [Amodei *et al.*, 2016] but also highlighted the "reward-result gap", where the agent fails to converge on an optimal policy. Their proposed research direction focuses on "reward modelling" and points out a number of approaches including leveraging existing data, imposing side constraints and adversarial training, to achieve "agent alignment". The "desiderata" (desirable properties) for their work are that the results are: scalable, economic and pragmatic. It also rests on two assumptions (slightly rephrased):

- It is possible to learn user intentions to sufficiently high accuracy;
- For many learning tasks, evaluation of outcomes is easier than producing the correct behaviour.

Other work considers the issues of mis-specified feature spaces [Bobu *et al.*, 2018]. When training robots based on physical human-robot interaction the robot may not have a rich enough hypothesis space to capture everything the user cares about so the robot may "misinterpret" user input and learn inappropriate actions. The proposed solution includes using relevance of inputs (to the robot's set of hypotheses) and only learning from relevant inputs, thus reducing the impact of mis-specified feature or objective spaces.

Further, there is considerable interest in the problem of "adversarial" resilience for RL with many proposed solution approaches, e.g. [Behzadan and Hsu, 2019]. This is undoubtedly relevant in some application domains, and needs to be addressed in a complete approach to using AI in safety-critical contexts, but it is outside the scope of this paper.

More philosophically, the challenges of "AI Safety" arise because systems are developed using "narrow" or "specific" AI so they do not have an understanding (semantic model) of real-world objects that would be achieved by Artificial General Intelligence (AGI). Whilst AGI is a distant prospect, some work on AGI safety is producing interesting results, e.g. causal influence diagrams that bear a resemblance to the models used in safety engineering and thus might be a route to bridging between these communities [Everitt *et al.*, 2019].

2.2 System Safety/Functional Safety

System safety is a relatively long-established discipline, generally believed to have originated in US military projects in the 1940s. There are now well-established processes which include hazard identification and risk assessment to guide the design to produce acceptably safe systems. Although not universally accepted, in many industries, e.g. healthcare [NHS Digital, 2018], it is common for the results of the development and safety processes to result in the production of a safety case, a structured argument, supported by evidence, intended to justify that a system is acceptably safe for a specific application in a specific operating environment [Kelly and Weaver, 2004].

System safety has evolved and the term "functional safety" is often used for safety processes addressing computers and software. Many of the standards for functional safety provide requirements for product design and for the system, software and hardware development processes. They often use safety integrity levels (SILs) to rank the (potential) risk posed by the system and vary the requirements with SIL, for example requiring higher levels of redundancy and diversity in the system architecture at higher SILs (higher risk).

System and functional safety have continued to evolve but generally lagging behind technology developments [McDermid, 2017] – this lag is perhaps most apparent with the introduction of AI and ML into systems. Although there are some emerging standards such as UL4600 [ANSI & UL, 2020] generally they set requirements "around" the AI/ML elements but are not clear about what form of evidence would be needed to show that these requirements are met.

Instead, for advice on the safe use of AI and ML, one has to turn to the research community rather than standards. Initiatives such as the AAIP have pioneered work on safety of autonomous systems employing AI and ML. Two particularly important contributions are an ML lifecycle [Ashmore *et al.*, 2019] and an assurance process known as AMLAS [Picardi *et al.*, 2020] that builds on the ML lifecycle. The ML lifecycle has three stages, and identifies desiderata for each stage:

1. Data management – including collecting, pre-processing and analysing data to be used for model learning;

2. Model learning – including the model selection and choice of hyper-parameters used to control the learning process;
3. Model verification – including the use of mathematical analysis and testing.

As an example of the desiderata, robustness in model learning considers the model's ability to perform well in circumstances where the inputs encountered at run time are different to those present in the training data, e.g. due to environmental uncertainty, such as flooded roads, which is analogous to "distributional shift" in [Amodei *et al.*, 2016]. This shows the possibility of reconciling the viewpoints of the two communities to provide common terminology to underpin a collaborative model.

2.3 Safety-Critical Software Engineering

There are many good practices in developing safety-critical software, some of which are reflected in standards such as IEC 61508 Part 3 [IEC, 2010]. These include use of formal methods for specifying requirements, coverage criteria for testing, and traceability from specifications to programs to assist in verification. The required practices also vary with SIL, for example with more stringent requirements for test coverage at higher SILs. It is not possible to cover all of these practices here but we use two of them to illustrate our points here.

First, all programming languages have constructs which can lead programmers to make mistakes. This has led to the definitions of so-called "safe subsets" of programming languages, e.g. the C language [Hatton, 2004], which restrict the use of the language so as to avoid the most error-prone constructs. These subsets are well-defined and it is practicable to use tools to help police their use.

Second, static analysis assesses a program without executing it. This can identify "undesirable features" in programs, including indexing outside array bounds and dividing by zero, that could lead to undefined behaviour. Even if such "undesirable features" are not "unsafe", their presence may undermine the results of other analyses or test results, so static analysis is of broad utility in showing the integrity of programs. Language subsets and static analysis are not disjoint concepts and some of the most powerful tools combine the two, e.g. [McCormick and Chapin, 2015].

These practices address conventional programming languages but ML models are programs too and the collaborative model explores their relevance for ML.

3 The Collaborative Model: An "AI sandwich"

The collaborative model aims to provide a structure that links the different world views of the AI and safety communities. The AI elements are in the middle, with safety elements above and below – hence the "sandwich" analogy.

3.1 The Model

The collaborative model, see Fig. 1, has three layers which we number top-down. The intent of the layers is:

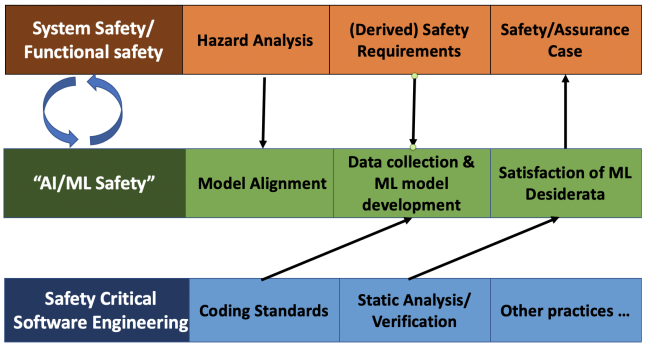


Figure 1: The Collaborative Model

1. System safety/functional safety – application of a classical safety process to understand hazards, produce derived safety requirements and to gather evidence from layers 2 and 3 into the safety/assurance case;
2. AI/ML safety – developing the ML systems to meet their performance objectives and satisfaction of the ML lifecycle desiderata as well as the derived safety requirements, and providing evidence to support the safety/assurance case;
3. Safety-critical software engineering – application of good software engineering practices to the development of the AI/ML software.

ML systems are generally developed in an agile manner, often with daily builds as new training data becomes available; this is reflected in the iterative loop between the top two layers. However, for simplicity, the description of the layers ignores the iteration.

3.2 Layer 1: System Safety/Functional Safety

The system safety/functional safety layer has three major elements:

1. Hazard analysis – use of hazard analysis techniques or domain knowledge to identify hazards and to estimate the associated risks;
2. Derived safety requirements (DSRs) – based on the identified hazards, establishing requirements for the ML and other elements of the system so that their contribution to hazards is controlled or their role in mitigating hazards is clearly defined;
3. Safety/assurance case – arguments for the safety of the system, supported where appropriate by evidence from the ML layer that the software meets the relevant desiderata and derived safety requirements.

Often requirements for AI/ML systems are articulated at a high level, e.g. to perform better than a human, and it can be difficult to map down to concrete requirements on the AI/ML components (the semantic gap [Burton *et al.*, 2020]). This is an open issue for reconciling conventional systems engineering with AI/ML, but it is not always so difficult when considering DSRs, see the case study in section 4.

3.3 Layer 2: "AI/ML Safety"

The "AI/ML safety" layer is intended to encompass the ML software development including verification and has three major elements:

1. Model alignment – meeting the design intent which is informed by the hazard analysis, as the intent, *inter alia*, is to avoid hazards;
2. Data collection and ML model development – these are the first two stages of the ML lifecycle [Ashmore *et al.*, 2019] and are informed by the derived safety requirements;
3. Satisfaction of the ML lifecycle desiderata – showing that the relevant desiderata have been met, including verifying satisfaction of the derived safety requirements and production of relevant evidence to support the safety/assurance case.

The term "model alignment" is used as a generalisation of "agent alignment" and is intended also to include the avoidance of problems such as distributional shift.

3.4 Layer 3: Safety-Critical Software Engineering

This layer incorporates good practices from safety-critical software engineering to ensure the integrity of the code; for brevity only two elements are illustrated:

1. Coding standards – the use of guidelines/rules to avoid the more error-prone constructs in programming languages, supporting ML model development;
2. Static Analysis/verification – the use of static analysis tools to help identify undesirable features in programs so they can be eliminated, supporting satisfaction of ML desiderata.

There are many different static analysis tools; an example is presented in section 4.

4 The Case study

The case study used to illustrate the collaborative model is from healthcare, particularly the use of RL to derive "optimal" policies for treatment of sepsis, which is a life-threatening condition and a major cause of fatalities in hospitals. The case study is from our previously published work [Jia *et al.*, 2020], but extended here to address the three layers in the collaborative model. Key aspects of the case study are introduced against each layer in the model, with the greatest emphasis placed on the middle layer as this is where the approaches of the AI and safety communities come together. Before presenting the case study, we first give a brief introduction to RL; this is not intended to be an exhaustive discussion of what is a very complex topic but enough to enable the case study to be understood. More details on the RL approach can be found in [Jia *et al.*, 2020].

4.1 Reinforcement Learning

RL is a very powerful ML technique which is widely used in complex decision making tasks to find an "optimal" policy. It consists of an agent interacting with its environment by performing actions and receiving feedback from the environment [Sutton and Barto, 2018]. Often the environment is represented using a Markov Decision Process (MDP). A policy defines the agent's behaviour and maps the perceived states of the environment to actions for the agent to take. There are many different RL algorithm and the case study uses a widely used modern RL algorithm known as double deep Q-Networks (double DQN) [Mnih *et al.*, 2015].

4.2 Sepsis Case

Sepsis is a life-threatening organ dysfunction caused by a dysregulated host response to infection. It is estimated that one in five deaths worldwide is caused by sepsis [Gallagher, 2020], but the optimal treatment strategy for sepsis remains unclear [Marik, 2015]. Evidence suggests that current practices in the administration of intravenous fluids and vasopressors are suboptimal [Waechter *et al.*, 2014]. Consequently, researchers have used RL to learn the optimal treatment strategy, e.g. [Raghu *et al.*, 2017].

First, we re-implemented the work from [Raghu *et al.*, 2017] using double DQN. The state space for the MDP included patients' demographics, Elixhauser premorbid status, vital signs, laboratory values, fluids and vasopressors received. The action space for the MDP is discretised into 25 possible actions with 5 possible choices for intravenous fluids and vasopressors respectively, as shown in Table I. Table I also shows the detailed dose ranges and dose medians for the five vasopressor choices; this is important as the case study focuses on the safety of vasopressor administration. Note that vasopressor dosage is shown in mcg/kg/min of Norepinephrine equivalent. The maximum dosage change occurs when the recommendation changes from action 0 to action 4, or *vice versa*, in the following step to treat the same patient. This change is 0 to 0.786 mcg/kg/min, as 0.786 mcg/kg/min is the median of the fourth quartile and is considered to be a dangerous dose change in one step in clinical practice. A sudden major change in vasopressor dosage can result in acute hypotension, hypertension or cardiac arrhythmias [Fadale *et al.*, 2014] [Hospira UK Ltd, 2018] [Allen, 2014] (hypotension can arise from rapidly decreasing doses, with hypertension or arrythmias arising from rapidly increasing doses).

Next, we evaluated the original learnt policy and discovered that it contains far more of these sudden major changes when recommending the vasopressor dosage than are found in the clinicians' treatments based on the real data – MIMIC III [Johnson *et al.*, 2016] (here we refer to these treatments as the clinician policy for ease of comparison). These results are shown in Fig. 3a. Thus the initial learnt policy raises some safety concerns and we used the collaborative model to guide us develop a safer learnt policy.

Layer 1: Hazard and Derived Safety Requirements

This layer is largely the province of the safety and domain specialists, i.e. clinicians in this case.

In an ideal world, the hazard analysis would be based on a clinical pathway (a model of the treatment process, including key decisions). For brevity, we illustrate the approach in terms of a single hazard identified using domain knowledge. The **hazard** is defined as: "sudden change in vasopres-

Table 1: Dosage Actions (from [Jia *et al.*, 2020])

		Dose of vasopressor (mcg/kg/min)				
		No.: 0 **Range:** 0 **Median:** 0	1 (0.002, 0.079) 0.04	2 (0.08, 0.2) 0.135	3 (0.201,0.449) 0.27	4 (0.45, 1.005) 0.786
Dose of IV fluid	0	0	1	2	3	4
	1	5	6	7	8	9
	2	10	11	12	13	14
	3	15	16	17	18	19
	4	20	21	22	23	24

sor dosage".

It can be seen from Fig 3a that the original learnt policy is "less safe" than the clinician policy, specifically 35% of the patients have a sudden major change in the learnt policy as opposed to 2.6% in the clinician policy. This can be viewed as an example of the hazard and it is visible when evaluating the learnt policy, which is consistent with the assumption in [Leike *et al.*, 2018] that "evaluation of outcomes is easier than producing the correct behaviour". It can also be seen as showing a failure to achieve "model alignment" and this triggers an iteration around the first two layers of the collaborative model, producing explicit DSRs.

In a normal safety analysis a systematic approach would be taken to identify causes of hazards, and there are many approaches to identifying and representing hazard causes, e.g. SHARD for software-intensive systems [Pumfrey, 1999]. However, what we are interested in here is understanding the potential causes of the hazard *across* the layers in the Collaborative Model, and we adapt the well-known "bow-tie" diagram for this purpose. This enables us to show causes and consequences of the hazard derived from an understanding of the potential limitations at each layer, see Fig. 2.

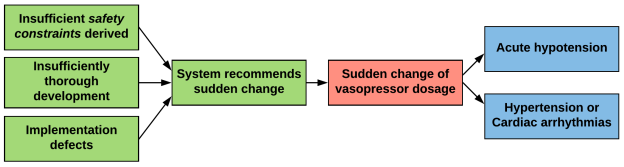


Figure 2: Bow-tie showing Hazard, Causes and Consequences

In Fig. 2 the consequences are the clinical outcomes described previously. The proximate cause of the hazard is "System recommends sudden change" (in excess of 0.75 mcg/kg/min) which has three potential causes relating to the three layers in the model in Fig. 1. The "Insufficient *safety constraints* derived" reflects inadequacy in hazard analysis; if a hazard is missed or misunderstood then the system developed might not be safe. The "Insufficiently thorough development" means failure to meet the safety constraints and the desiderata for ML including avoiding relevant "AI Safety" problems such as those identified in [Amodei *et al.*, 2016] and [Leike *et al.*, 2018]. This can also be seen as a failure to achieve "model alignment". The "Implementation defects" refers to code-level problems, such as "divide by zero" that can have unpredictable effects.

There are four DSRs arising from the bow-tie diagram:

DSR0: reduce changes in vasopressor dosage of more than 0.75 mcg/kg/min between treatment steps for an individual patient closer to clinician policy (maps to "System recommends sudden change");

DSR1: accurately identify hazards, hazards causes and *safety constraints* (maps to "Insufficient *safety constraints* derived");

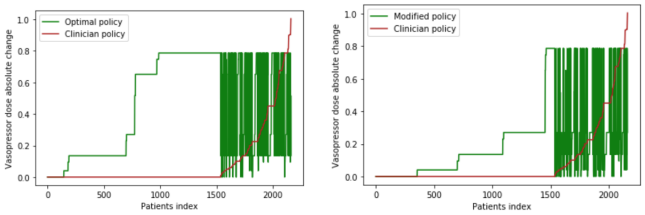
DSR2: (a) meet the desiderata for the ML lifecycle [Ashmore *et al.*, 2019] and (b) show satisfaction of *safety constraints* arising from **DSR1**, (maps to "insufficiently thorough development");

DSR3: avoid implementation deficiencies that could give rise to unintended behaviour (maps to "Implementation defects").

DSR1, **DSR2** and **DSR3** are intended to support **DSR0**, and should guide the development of the ML in the middle layer in Fig. 1. Our focus here is on **DSR1** and **DSR2**, in order to satisfy **DSR0**, as this shows most clearly the links between the concerns of the safety and AI communities.

Evidence that the DSRs have been met should form part of the safety case. Safety arguments are often produced graphically, e.g. using the Goal Structuring Notation (GSN) [Kelly and Weaver, 2004]. For brevity, we do not set out the safety argument here, but discuss the evidence needed to meet **DSR0** in the following two subsections. The evidence for **DSR1** comes from layer 1 in Fig. 1. In a full development it would rest on the rigour of the hazard analysis process and the suitability of domain knowledge; for the purpose of this paper we assume that an appropriate hazard has been identified, based on clinical knowledge, hence **DSR1** is satisfied.

Layer 2: "AI/ML Safety"



(a) Original learnt policy (b) Our modified policy with *safety constraint*

Figure 3: Performance of Learnt Policies (from [Jia *et al.*, 2020])

We start with **DSR1** and consider what can contribute to this hazard, from an "AI safety" perspective, which involves understanding some of the details of the RL process. First, the MDP only depends on the current state, that is, given the current state, the future state does not depend on the cumulative history of past states. If the current state in the MDP doesn't capture the dose *delta* or relative dose change compared with the previous dose, there is no guarantee that the agent will learn an optimal policy avoiding the sudden major dose change. This is an example of mis-specified feature spaces [Bobu *et al.*, 2018] and is corrected by extending the state space to enable the agent to take into account the difference between the current step and the previous step in terms of vasopressor dose while learning the policy.

Second, the process of learning an optimal policy is to minimise the cost function. If the cost function incorporate the *safety constraint*, then it can guide the agent to learn a safer policy. Thus, the cost function was modified to include an additional *safety constraint* "penalising" large changes in dose (specifically those over 0.75 mcg/kg/min between dose recommendations) which is one of the approaches suggested in [Leike *et al.*, 2018]. Therefore, two changes were made in order to guide the agent to learn a safer policy by altering the state space to include dose *delta* and by adding a regularization term in the cost function to "penalise" sudden changes, see [Jia *et al.*, 2020] for more details, including the explicit cost functions used.

After the implementation of these two alterations we have learnt a new modified policy and Fig.3b shows that the modified policy has fewer sudden major changes compared to the original learnt policy shown in Fig. 3a (the rate of such sudden major changes of vasopressor dose has been reduced by 77.5%). Particularly, we found that only 7.87% of patients have this sudden major change in the modified policy, which is much closer to the 2.6% in the clinician policy. Therefore, this is considered to be much safer and to support **DSR1** and hence **DSR0**, improving the "model alignment".

The evidence to support **DSR2** is multi-faceted. **DSR2** (a) can be supported through a development log, and other development artefacts to show how the three stages of the ML lifecycle (indicated in section 2.2) have been implemented appropriately. In general, judgement is needed on sufficiency of evidence. **DSR2** (b) is supported by the performance data, see Fig.3b, and the comparison with Fig. 3a which show the results of encoding the *safety constraint* in the learnt policy.

Layer 3: Safety-Critical Software Engineering

This layer is the province of (software) safety specialists. It "provides" coding standards to support the ML development and static analysis techniques to support demonstration of the integrity of the developed software. The evidence to support **DSR3** comes from the use of the static analysis techniques in this case.

The software we developed in this case study is written in Python and uses the TensorFlow library, thus we have chosen Pylint (see: https: www.pylint.org) which supports both coding standards and error detection to do the static analysis. Fig. 4 shows a fragment of the report (log file) from running Pylint on our code (the code module named "deeprl"). The labels are as follows:

C: coding convention violation;

R: for "refactoring" to improve the score against some quality metric;

E: for programming errors, likely a "bug";

W: for warnings, e.g. minor programming errors or stylistic issues.

Fig. 4 shows the log file for a stage in the development of the "deeprl" module. The progress in improving the code module can be seen via the overall rating in Fig. 5. Pylint is quite "pedantic" (the log files are usually very long), so it is very hard to get a score of 10 – but it is important to remove the type E problems (and F which are fatal and prevent the analysis from proceeding).

```
yj914@[redacted]/SepsisDeepRL$ pylint deeprl.py
************* Module deeprl
deeprl.py:16:0: C0301: Line too long (634/100) (line-too-long)
deeprl.py:44:0: C0115: Missing class docstring (missing-class-docstring)
deeprl.py:44:0: R0902: Too many instance attributes (38/7) (too-many-instance-attributes)
deeprl.py:44:0: R0903: Too few public methods (0/2) (too-few-public-methods)
deeprl.py:140:27: C0321: More than one statement on a single line (multiple-statements)
deeprl.py:185:4: R1720: Unnecessary "elif" after "raise" (no-else-raise)
deeprl.py:329:0: E1101: Instance of 'ConfigProto' has no 'gpu_options' member (no-member)
deeprl.py:371:0: R1711: Useless return at end of function or method (useless-return)
deeprl.py:7:0: W0611: Unused import math (unused-import)
```

Figure 4: Extract from Pylint Log File

```
--------------------------------------------------------------------
Your code has been rated at 3.78/10 (previous run: 1.03/10, +2.75)
```

Figure 5: Example of Code Rating

The evidence to meet **DSR3** is based on the log file, showing that the error count (E) is zero and, desirably, an assessment of the other entries to "sentence" them and to decide which need to be addressed, and which can be tolerated. For example, some of the refactoring comments (R) can indicate aspects of the program that are hard to test, and which may therefore weaken the value of the evidence in support of **DSR2** if not corrected. Others, e.g. C0321, are merely stylistic and would not undermine the evidence from layer 2 against **DSR2**.

5 Discussion

Our model provides a "big picture" which allows the viewpoints from the AI and safety communities to be reconciled. The illustration of our model in section 4 has shown that these viewpoints can be drawn together constructively to improve the safety of a system employing ML.

It is worth considering the relationship of "AI/ML safety" issues, e.g. [Amodei *et al.*, 2016] [Leike *et al.*, 2018] [Bobu *et al.*, 2018] and the ML lifecycle [Ashmore *et al.*, 2019] to support layer 2 in our collaborative model. As already mentioned some of these problems and desiderata in the ML lifecycle overlap, e.g. "distributional shift" and robustness. Further, issues such as the "reward-result gap" and mis-specified feature spaces can be seen to relate to system safety concerns through the case study. If a systematic relationship can be established between the problems identified in the "AI Safety" community and desiderata in the ML lifecycle, then this would further cement the links between the viewpoints of the AI and safety communities.

The illustration at layer 3, Safety-Critical Software Engineering, was limited due to space constraints. A decision was made to focus on code-level issues for the paper, but other techniques are relevant. For example, several standards promote the use of formal methods. Whilst full formal specification of ML systems is likely to be difficult due to the semantic gap [Burton *et al.*, 2020] it may be possible to use partial formal specifications for critical (safety) properties [Salay and Czarnecki, 2019] which also opens up the opportunities for more formal verification. Further, all safety-critical systems should be tested, and there is a need for systematic approaches to testing for systems employing AI or ML, e.g. considering how to guide testing based on estimates of residual risk [Wotawa, 2019]. In general, there is a need to con-

sider good practice in Safety-Critical Software Engineering, to identify what techniques can be drawn across into AI/ML development. As ML software tends to be developed in a dynamic and iterative fashion this suggests that it would be desirable to draw on work on agile approaches to safety-critical software development such as [Hanssen *et al.*, 2018] as well.

However, as noted above, most functional safety standards use SILs (or variants thereof) and alter the requirements for techniques to apply to each stage of the software development process based on SIL. The discussion of the ML lifecycle [Ashmore *et al.*, 2019] identifies multiple ways of addressing some of the desiderata, but it is not obvious that these can be "ranked" in terms of contribution to risk reduction and hence arranged in SIL-order. Whilst it might be possible to preserve the SIL concept at layer 3 in the collaborative model, it is much less obvious how to do this at layer 2. Either this means the two communities have a long way to go before they can fully support the notion of SILs for ML-based systems, or there needs to be an acknowledgement that the SIL concept is not readily applicable in the context of ML-based systems.

Additionally, standards for safety-critical systems often set stringent safety targets, e.g. unsafe failure rates of one in 10 million operating hours. In contrast, ML developers are often content with false positive/false negative rates of the order of a few percent – in some applications, e.g. autonomous driving, this might equate to many "failures" per hour! These figures seem incompatible. However, the safety targets set in standards are at system level, not algorithm level, so they can be reconciled, at least in some cases, with suitable system architectures including redundancy and diversity. Nonetheless, more work is needed to show how to bridge this "gap" and to find ways of demonstrating that AI/ML-based systems meet the stringent unsafe failure-rate targets that are widely used for safety-critical systems.

The case study uses RL and it was relatively easy to change the feature space and to introduce a *safety constraint* in the cost function to satisfy the derived safety requirements in a way that is traceable. With other ML techniques, e.g. unsupervised learning, it may be less easy to embed the *safety constraint* in the learning process and it may be necessary instead to design the system with a separate "monitor" that polices the system behaviour against the *safety constraint*, as suggested in [McDermid *et al.*, 2019], or to use other forms of diversity and redundancy as discussed above. Defining runtime monitors is not always straightforward, so the generality of our collaborative model across the wide and growing range of ML techniques remains an open issue.

6 Conclusions

The collaborative model has shown how to link the viewpoints of the AI and safety communities, with the DSRs and evidence flows providing the critical links in Fig. 1. Whilst the model is abstract, the case study has enabled us to show that the ideas can be made "concrete" although it has not been possible to provide full technical detail at all the layers. It is hoped that this model will help to facilitate broader engagement between the AI and safety communities by giving a structure in which they can recognise their own viewpoint and see how it relates to that of the other community.

Due to the speed of development of AI and ML and their safety-related applications, not least in autonomous vehicles, the standards are lagging behind the technology – and, arguably, the gap is growing. The collaborative model, or its future refinements building on more detailed models, e.g. the ML lifecycle [Ashmore *et al.*, 2019] and assurance processes [Picardi *et al.*, 2020], should provide a framework for producing future standards for safety critical systems using ML. It remains to be seen whether or not SILs form part of such standards – their introduction and use has been pragmatic (as much to manage cost as safety) and, at minimum, the rationale for their introduction should be reconsidered as standards for ML-based systems are developed.

However, as noted above, there are open issues, perhaps one of the most important is whether or not the "AI Safety" problems can be mapped to the ML lifecycle model and thus addressed in a unified way with the ML desiderata in the middle layer of our collaborative model. These open issues can best be addressed through greater collaboration between the two communities and they will be a focus in our future work.

Acknowledgements

This work funded by the Assuring Autonomy International Programme at the University of York and by Bradford Teaching Hospitals NHS Foundation Trust. The views expressed in this paper are those of the authors and not necessarily those of the NHS, or the Department of Health and Social Care.

References

[Allen, 2014] John M Allen. Understanding vasoactive medications: focus on pharmacology and effective titration. *Journal of Infusion Nursing*, 37(2):82–86, 2014.

[Amodei *et al.*, 2016] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016.

[ANSI & UL, 2020] ANSI & UL. Standard for Evaluation of Autonomous Products. https://www.shopulstandards.com/ProductDetail.aspx?productid=UL4600, 2020.

[Ashmore *et al.*, 2019] Rob Ashmore, Radu Calinescu, and Colin Paterson. Assuring the machine learning lifecycle: Desiderata, methods, and challenges. *arXiv preprint arXiv:1905.04223*, 2019.

[Behzadan and Hsu, 2019] Vahid Behzadan and William Hsu. Rl-based method for benchmarking the adversarial resilience and robustness of deep reinforcement learning policies. In *International Conference on Computer Safety, Reliability, and Security*, pages 314–325. Springer, 2019.

[Bobu *et al.*, 2018] Andreea Bobu, Andrea Bajcsy, Jaime F Fisac, and Anca D Dragan. Learning under misspecified objective spaces. *arXiv preprint arXiv:1810.05157*, 2018.

[Burton *et al.*, 2020] Simon Burton, Ibrahim Habli, Tom Lawton, John McDermid, Phillip Morgan, and Zoe Porter. Mind the gaps: Assuring the safety of autonomous systems from an engineering, ethical, and legal perspective. *Artificial Intelligence*, 279:103201, 2020.

[Everitt *et al.*, 2019] Tom Everitt, Ramana Kumar, Victoria Krakovna, and Shane Legg. Modeling agi safety frameworks with causal influence diagrams. *arXiv preprint arXiv:1906.08663*, 2019.

[Fadale *et al.*, 2014] Kristin Lavigne Fadale, Denise Tucker, Jennifer Dungan, and Valerie Sabol. Improving nurses' vasopressor titration skills and self-efficacy via simulation-based learning. *Clinical Simulation in Nursing*, 10(6):e291–e299, 2014.

[Gallagher, 2020] James Gallagher. 'Alarming' one in five deaths due to sepsis. https://www.bbc.co.uk/news/health-51138859, 2020. Accessed: 2020-03-01.

[Hanssen *et al.*, 2018] Geir Kjetil Hanssen, Tor Stålhane, and Thor Myklebust. *SafeScrum®-Agile Development of Safety-Critical Software*. Springer, 2018.

[Hatton, 2004] Les Hatton. Safer language subsets: an overview and a case history, misra c. *Information and Software Technology*, 46(7):465–472, 2004.

[Hospira UK Ltd, 2018] Hospira UK Ltd. Noradrenaline (Norepinephrine) 1 mg/ml Concentrate for Solution for Infusion. https://www.medicines.org.uk/emc/product/4115/smpc, 2018. Accessed: 2020-03-01.

[IEC, 2010] IEC. IEC 61508-3:2010, Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 3: Software requirements. https://webstore.iec.ch/publication/5517, 2010.

[Jia *et al.*, 2020] Yan Jia, John Burden, Tom Lawton, and Ibrahim Habli. Safe reinforcement learning for sepsis treatment. In *2020 IEEE International conference on healthcare informatics (ICHI)*, pages 1–7. IEEE, 2020.

[Johnson *et al.*, 2016] Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035, 2016.

[Kelly and Weaver, 2004] Tim Kelly and Rob Weaver. The goal structuring notation – a safety argument notation. In *Proceedings of the dependable systems and networks 2004 workshop on assurance cases*, page 6. Citeseer, 2004.

[Leike *et al.*, 2018] Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871*, 2018.

[Marik, 2015] PE Marik. The demise of early goal-directed therapy for severe sepsis and septic shock. *Acta Anaesthesiologica Scandinavica*, 59(5):561–567, 2015.

[McCormick and Chapin, 2015] John W McCormick and Peter C Chapin. *Building high integrity applications with SPARK*. Cambridge University Press, 2015.

[McDermid *et al.*, 2019] John McDermid, Yan Jia, and Ibrahim Habli. Towards a framework for safety assurance of autonomous systems. In *Artificial Intelligence Safety 2019*, pages 1–7. CEUR Workshop Proceedings, 2019.

[McDermid, 2017] John McDermid. Playing catch-up: The fate of safety engineering. In *Developments in System Safety Engineering, Proceedings of the Twenty-fifth Safety-Critical Systems Symposium, Bristol, UK, ISBN*, pages 978–1540796288, 2017.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[NHS Digital, 2018] NHS Digital. DCB0160: Clinical risk management: its Application in the Deployment and Use of health IT Systems. 2018.

[Picardi *et al.*, 2020] Chiara Picardi, Colin Paterson, Richard Hawkins, Radu Calinescu, and Ibrahim Habli. Assurance argument patterns and processes for machine learning in safety-related systems. In *Proceedings of the Workshop on Artificial Intelligence Safety (SafeAI 2020)*, 2020.

[Pumfrey, 1999] David John Pumfrey. *The principled design of computer system safety analyses.* PhD thesis, University of York, 1999.

[Raghu *et al.*, 2017] Aniruddh Raghu, Matthieu Komorowski, Imran Ahmed, Leo Celi, Peter Szolovits, and Marzyeh Ghassemi. Deep reinforcement learning for sepsis treatment. *arXiv preprint arXiv:1711.09602*, 2017.

[Salay and Czarnecki, 2019] Rick Salay and Krzysztof Czarnecki. Improving ml safety with partial specifications. In *International Conference on Computer Safety, Reliability, and Security*, pages 288–300. Springer, 2019.

[Sutton and Barto, 2018] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[Waechter *et al.*, 2014] Jason Waechter, Anand Kumar, Stephen E Lapinsky, John Marshall, Peter Dodek, Yaseen Arabi, Joseph E Parrillo, R Phillip Dellinger, and Allan Garland. Interaction between fluids and vasoactive agents on mortality in septic shock: a multicenter, observational study. *Critical care medicine*, 42(10):2158–2168, 2014.

[Wotawa, 2019] Franz Wotawa. On the importance of system testing for assuring safety of ai systems. In *AISafety@ IJCAI*, 2019.