UNIVERSITY of York

ASSURING AUTONOMY INTERNATIONAL PROGRAMME

Safety of Artificial Intelligence: A Collaborative Model

Professor John McDermid OBE FREng
Ms Yan Jia
University of York, UK

Overview

Key Topics

- Motivation
- The Collaborative Model
  - System/functional safety
  - AI/ML safety
  - Safety-Critical Software Engineering
- A Case Study – Sepsis treatment
- Refining the Collaborative Model
- Building a Community

Motivation

Different Communities

- Growing understanding of the potential for AI/ML-based systems to produce undesirable results
  - For example, the COMPAS system recommending prison sentences showed systematic bias
- AI/ML community
  - Identified generic issues
  - Solutions by adapting ML methods, i.e. in paradigm
- Safety community
  - Concern with the challenges to established approaches
  - Solutions by adapting approaches, i.e. in paradigm

Motivation

AI Community

- Various “formalisations”
  - The “concrete problems” of AI [Amodei et al 2016]
  - The “reward-result gap” [Leike et al 2018]
- Various approaches to resolution
  - Reward modelling
  - Adversarial resilience
  - Explainability
- In the long-run artificial general intelligence (AGI)
  - Providing *context* and *semantics* missing in “narrow” AI

Motivation

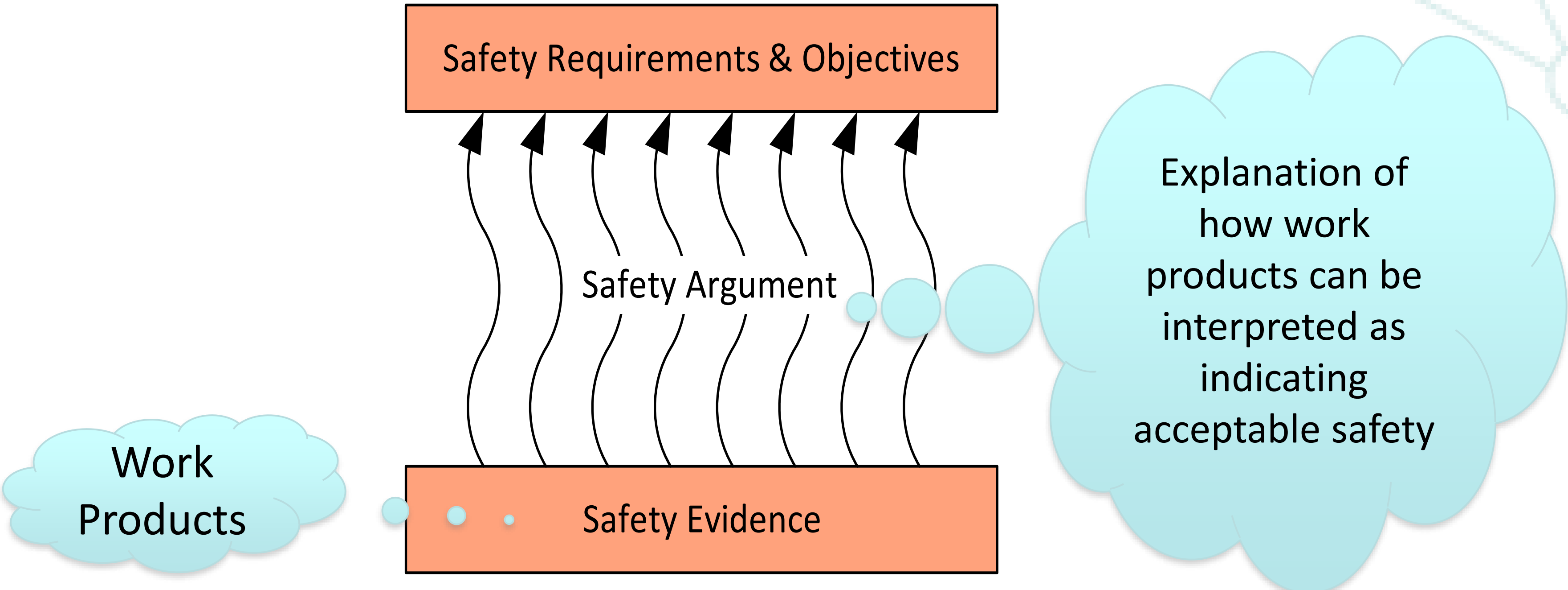
Safety Community

- Some attempts to prohibit/limit use
- Some work on standards
  - UL 4600 for autonomous systems includes requirements on ML-elements of systems
- Some work on adapting safety principles
  - Assurance of ML in Autonomous Systems (AMLAS) [Picardi et al 2020] based on ML lifecycle model [Ashmore et al 2019] give *desiderata* for lifecycle stages
- Requirements typically for safety/assurance cases
  - Less clarity on how to meet the requirements (evidence)

Assuring Safety

Safety/Assurance Cases

- Concept – argument supported by evidence



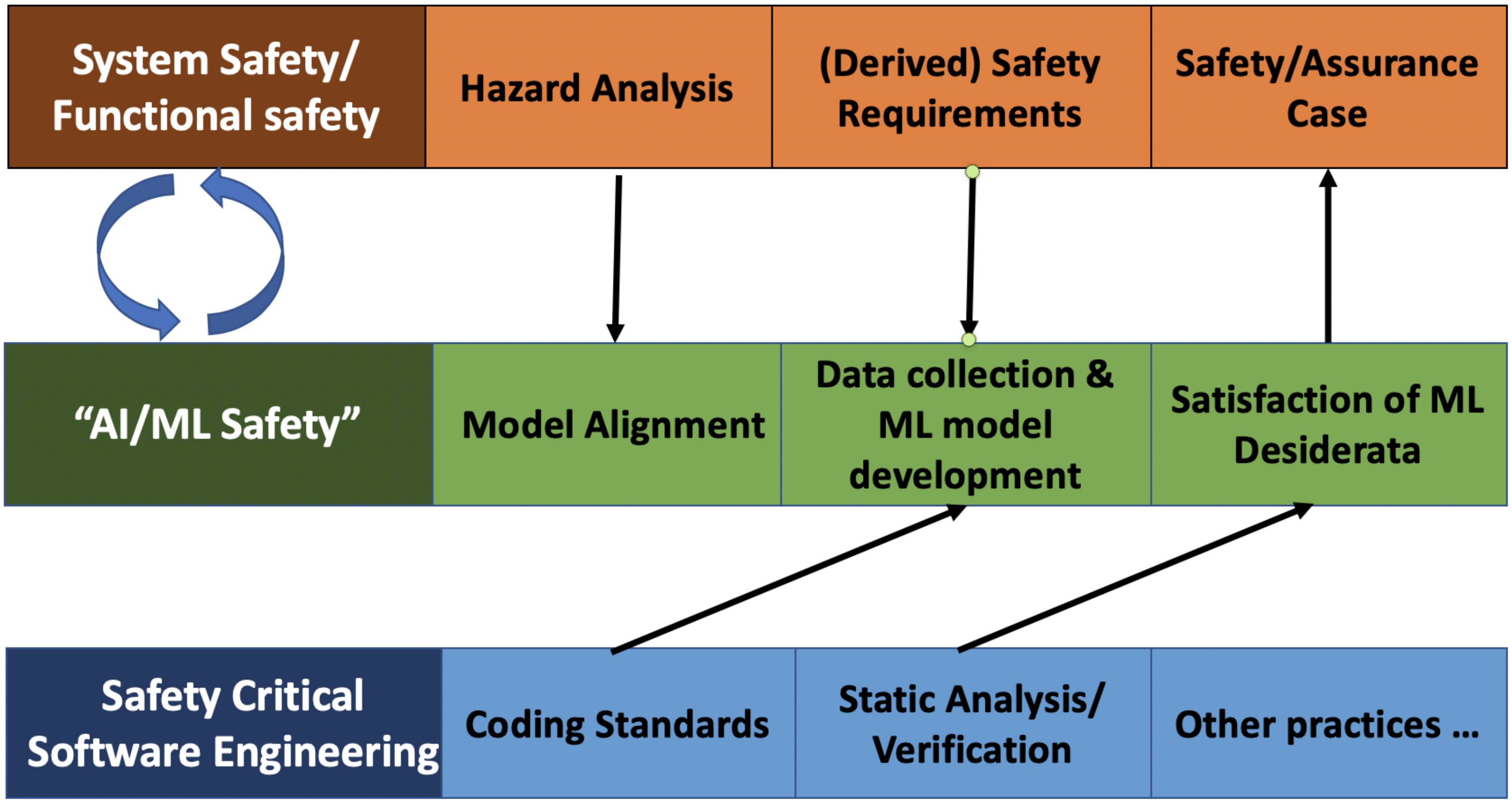
Motivation

Collaboration

- ML produces the deployed system
- If safety processes “merely look on” & document
  - Probably can’t make an adequate judgment of safety
  - Likely to be ignored (as irrelevant)
- To provide value
  - Safety must *influence the design*
  - Help produce a better system (*ensure* safety)
  - Provide evidence done so (*assure* safety)
- Need collaboration – a shared paradigm

The Collaborative Model

Overview



The Collaborative Model

Layer 1: Functional/System Safety

- Hazard analysis
  - Identify hazards and estimate associated risks
- Derived safety requirements (DSRs)
  - On ML and other elements of the system so contribution to hazards is controlled (or role in mitigation is defined)
  - May be on the product and/or on the development process
- Safety/assurance case
  - Arguments for safety of the system supported by ML layer evidence that the desiderata and DSRs are met

The Collaborative Model

Layer 2: AI/ML Safety

- Model alignment
  - Meeting the design intent, including avoidance of hazards
- Data collection and model development
  - The first two stages of the ML lifecycle [Ashmore et al 2019] (third is verification) informed by the DSRs
- Satisfaction of DSRs and desiderata
  - Verification, producing appropriate evidence for the safety/assurance case

  NB doesn’t resolve the ”how much evidence” question

Safety Standards

EN50128 (Software, Rail Sector)

TECHNIQUE/MEASURE	Ref	SWS ILO	SWS IL1	SWS IL2	SWS IL3	SWS IL4
1. Formal Proof	B.31	-	R	R	HR	HR
2. Probabilistic Testing	B.47	-	R	R	HR	HR
3. Static Analysis	D.8	-	HR	HR	HR	HR
4. Dynamic Analysis and Testing	D.2	-	HR	HR	HR	HR
5. Metrics	B.42	-	R	R	R	R
6. Traceability Matrix	B.69	-	R	R	HR	HR
7. Software Error Effects Analysis	B26	-	R	R	HR	HR

Requirements

1. For Software Safety Integrity Level 3 or 4, the approved combinations of techniques shall be:-
   - a) 1 and 4
   - b) 3 and 4
   - or c) 4, 6 and 7
2. For Software Safety Integrity Level 1 or 2, the approved technique shall be 1 or 4.

The Collaborative Model

Layer 3: Software Engineering

- Many relevant software engineering techniques
  - Paper and example focus on coding standards and static analysis
    - Coding standards – rules for programming that avoid common classes of error, e.g. divide by 0, buffer overflow
    - Static analysis – checks on code without executing it
- In practice, ML development very agile
  - Are some good techniques
  - However, safety standards mainly based on V life-cycle
  - So need to draw on principles not specifics of standards

Overview

Key Topics

- Motivation
- The Collaborative Model
  - System/functional safety
  - AI/ML safety
  - Safety-Critical Software Engineering
- A Case Study – Sepsis treatment
- Refining the Collaborative Model
- Building a Community

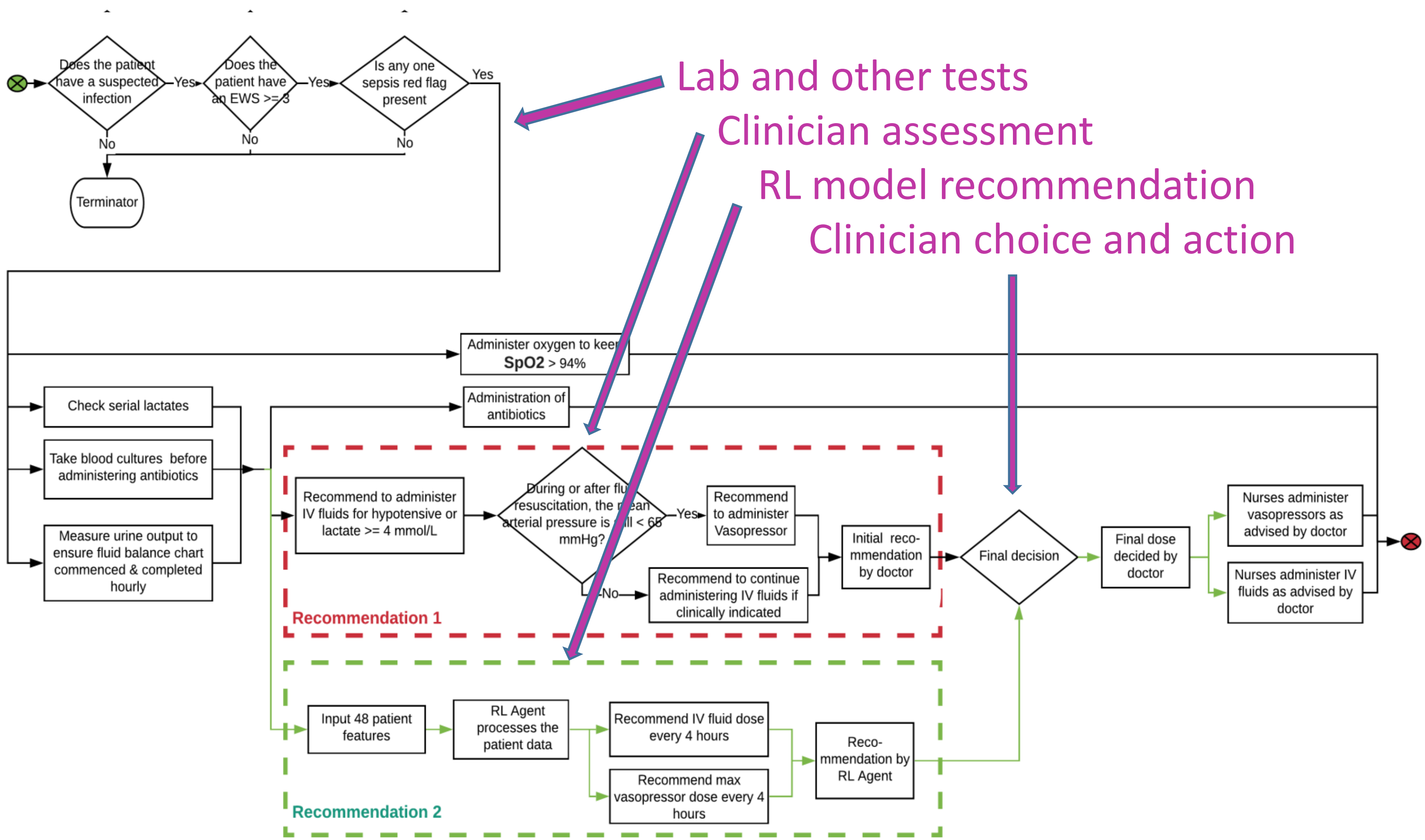
A Case Study

Sepsis and its Treatment

- Sepsis is a complex life-threatening situation known to be difficult to diagnose and to treat
  - One of the highest causes of deaths in hospital
  - Generally, leads to organ dysfunction
- Treatment includes
  - Delivery of vasopressor
  - Delivery of intravenous (IV) fluids
- Case study uses reinforcement learning (RL) to derive an optimal treatment policy for vasopressor and IV
  - Analysis and refinement of already published RL model

Case Study

Clinical Pathway Incorporating RL



CASE Study

RL Model Basics

- Feature space
  - 48 features, mainly clinical, showing patient state
- Action space
  - 25 discrete actions, codifying a combination of IV and vasopressor doses

		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

Case Study

Layer 1: Hazard and Risk Analysis

- SHARD analysis method
  - Initially developed for analysing software-intensive systems
  - Applies guidewords to flows in design
    - Omission
    - Commission
    - Incorrect
    - Early
    - Late
  - Adapted to the clinical pathway in the case study

Table 1. Fragment of SHARD analysis showing a single hazard

Guide word	Deviation (Hazards)	Possible Causes	Effects	Severity
Incorrect	Sudden change of vasopressor dose is administered (concerns two consecutive doses)	1 Kink of line 2 The pump fails, e.g. due to electrical problem or bag/syringe not installed correctly 3 The delivery line might not be connected to patient's central line, e.g. due to the patient pulling out the central line 4 The drug might not be added to the diluent, so the syringe/bag just contains saline (a problem when bags/syringes are being changed over) 5 Initial recommendation by doctor has a sharp change in dose and doctor carried through the recommendation (not considered in this paper) 6 RL agent recommends a sharp change in dose and doctor accepts the advice, e.g. due to automation bias 7 Inappropriate titration of dose by nurse 8 Doctor fails to check current dose 9 Features in state space of the RL model are not sufficient to represent the patient conditions for sepsis decision making 10 Reward function used for RL model is coarse 11 Cost function used for RL model development is not appropriate 12 Hyperparameters used for RL model development are not optimised 13 Training data for RL model development is not appropriate 14 Nurse prepared wrong dose (e.g. due to calculation error) 15 Data corruption (e.g. invalid or wrong data produced by over-writing patient's features) 16 Features for wrong patient entered 17 Wrong patient feature values entered (e.g. due to unit difference) 18 Test results for wrong patient received 19 Incorrect test results received	Acute Hypotension, Strokes, Renal failure, Heart attack could occur from a sharp drop in the dose Hypertension, Cardiac Arrhythmia, Strokes, Raised intracranial pressure, Pulmonary oedema could occur from a sharp rise in the dose	Major/ considerable

Case Study

Layer 1: Derived Safety Requirements

Table 2. Safety Requirements for RL model derived from Hazard analysis

ID	Description	Type	Allocation
R0	Sudden changes in recommended dose shall be close to clinician practice	Performance & Safety	RL model development
R1	Feature representation in the state space shall be sufficient to allow the control of sudden changes in recommended dose	Performance & Safety	RL model design
R2	An appropriate reward function shall be defined to allow the recognition of desired clinical outcome	Performance & Safety	RL model design
R3	An appropriate cost function shall be defined to penalise hazardous behaviours	Performance & Safety	RL model development
R4	Hyperparameters shall be optimised based on the validation dataset	Performance & Safety	RL model development
R5	Patient cohort shall be defined using recognised criteria, i.e. sepsis-3	Performance & Safety	RL model design

Case Study

Layer 2: ML Model Development

- DSRs met by modifying the model state space (R1) and the cost function (R3)
  - NB approach uses double DQNs

Table 4. Major changes in the modified RL model

	Features in state space (R1)	Cost Function(R3)
RL model in [32]	48	$L(\theta) = E[(Q_{double-target} - Q(s,a;\theta))^2] + \lambda_1 max(\lvert Q(s,a;\theta)\rvert - Q_{thresh}, 0)$
Modified RL model	48 (Removed one feature – timestep, added an extra one – relative dose change)	$L(\theta) = E[(Q_{double-target} - Q(s,a;\theta))^2] + \lambda_1 max(\lvert Q(s,a;\theta)\rvert - Q_{thresh}, 0) + \lambda_2 max(\lvert V_{change}\rvert - 0.75, 0)$ V_{change} is the agent recommended dose (argmax of $Q(s,a;\theta)$) minus the vasopressor dose in the previous step; λ_1 and λ_2 are the tuning parameters that decide how much to penalise the flexibility of the model.

[32] Raghu, Aniruddh, et al. "Deep reinforcement learning for sepsis treatment."
arXiv preprint arXiv:1711.09602 (2017).

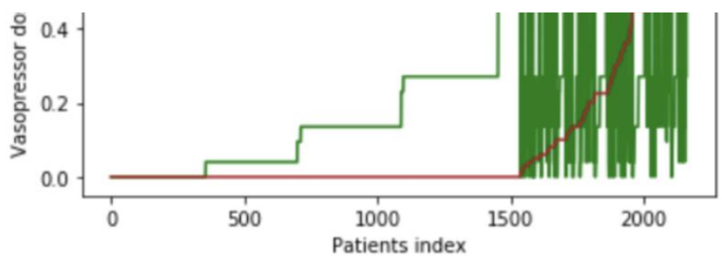
Case Study

Initial Approach

- High rate of sudden changes

Table 4. Summary of max dose change between consecutive doses for the three policies

	Dose of vasopressor (mcg/kg/min)	
	Small-Medium Dose Change (0-0.75)	Large Dose Change (>0.75)
Clinician Policy	97% (2,100)	3% (60)
Original Policy	65% (1,404)	35% (756)
Modified Policy	92% (1,990)	8% (170)



- Meets R0

Figure 5. Modified Policy: Comparison of max absolute vasopressor dose change in one step for each patient in the test data set between the clinician and the learnt modified policy

Case Study

Layer 3: Static Analysis

- Software developed in Python
  - Analysis done using PyLint
- PyLint “tags” issues
  - C: coding convention violation;
  - E: for programming errors, likely a “bug”;
  - F: for fatal;
  - R: for “refactoring” to improve the score against some quality metric;
  - W: for warnings, e.g. minor programming errors or style

Case Study

Layer 3: Static Analysis

```
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)
```

- Fragment above shows errors and style issues
- Code improvement recorded over time (10 is hard)

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

Refining the Model

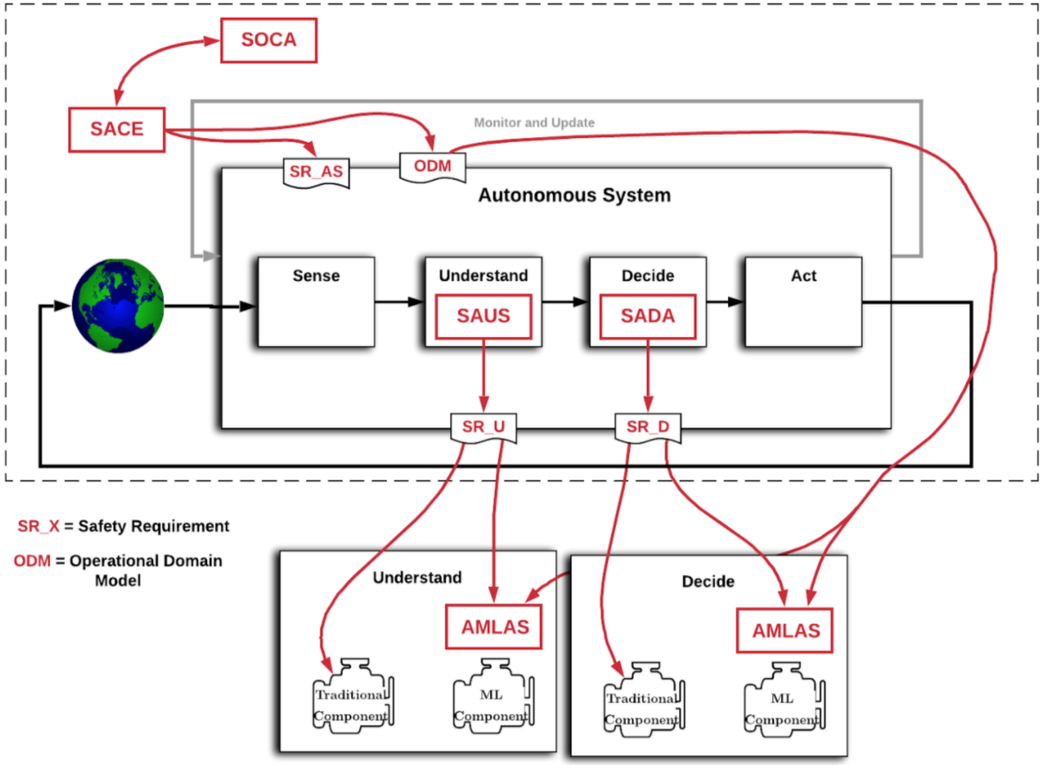
Some Necessary Steps

- Approach works directly in the case study
  - Can “implement” DSRs by changing the feature space and the cost function
  - But need to adapt to other ML models – e.g. for unsupervised learning may need to encode DSRs in monitors
- Need to consider wider issues, e.g.
  - A more “ML aware” safety process
  - Role of explainability in assurance
  - Sufficiency of evidence
  - Moving more to a “continuous assurance” model

Refining the Safety Process

Refinement for Autonomous Systems

- Safety processes
  - SOCA: acceptability
  - SACE: whole system, including shared control
  - SAUS: understanding
  - SADA: decision-making
  - AMLAS: assurance of ML



SR – Safety Requirement

Building a Community

ML and Safety and More

- ML and safety communities use different languages
  - Perhaps even mean different things by “AI Safety”!
  - Need to establish better means to communicate and collaborate to achieve safe AI/ML/autonomy
- But the onus is with the safety engineers
  - ML developers produce the systems
    - They will make them safe (or not)
  - Safety engineers must add value, e.g. derived safety requirements to use in ML performance evaluation
- Also involve safety-critical software engineering

UNIVERSITY of York

ASSURING AUTONOMY
INTERNATIONAL PROGRAMME

Funded by

Lloyd's Register Foundation

UNIVERSITY of York