AID: Assuring the Safe Use of AI-enabled Autonomy in Nuclear Decommissioning

Calum Imrie Ioannis Stefanakos Sepeedeh Shahbeigi
Richard Hawkins Simon Burton

Department of Computer Science, University of York

Motivation

Integrating AI in robotic systems involves constructing approximations of the robot's perception and capabilities to operate effectively in complex environments. Reinforcement Learning (RL), a popular machine learning technique, enables a robot to learn how to interact with its environment through state-action pairs guided by a reward function. There are various ways to implement RL, each with unique advantages and challenges. While AI has the potential to improve control and adaptability, it raises the question: Are we solving problems or introducing new ones?

Use Case and Experimental Setup

Our use case involves a robotic arm for pick-and-place tasks in a glove box for nuclear decommissioning. While glove boxes are designed to safely contain radioactive materials, they still pose risks to operators. Automation can enhance efficiency and reduce hazards, but traditional methods often depend on perfect environmental information, limiting their use in unpredictable conditions. AI offers more adaptive control but requires rigorous assurance for safe deployment.

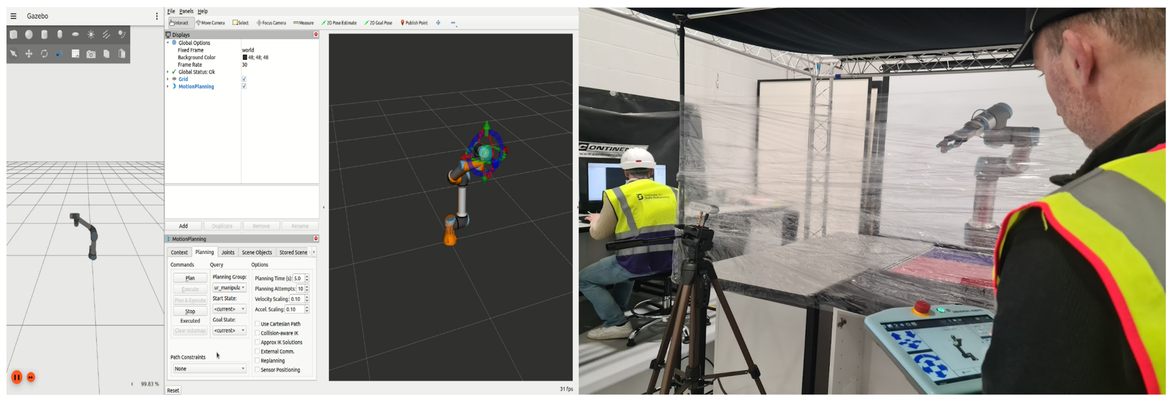

Figure 1. Our physical experimental setup at the Institute for Safe Autonomy, York (right), and the robot arm in the Gazebo simulator (gazebosim.org) (left)

Our experimental setup, shown in Figure 1, is designed to mimic a glove box where possible, featuring an encased environment with two Intel RealSense cameras for arm perception. Key features and considerations include:

- **Encased environment:** The enclosure, made of clingfilm, reduces the risk of damage during RL controller integration while introducing perception challenges due to its wrinkles;
- **Object detection:** A YOLO pipeline has been integrated, enabling object detection and providing 3D positions relative to the robotic arm;
- **Iterative methodology:** AMLAS is iterative, often requiring revisions to system design elements. To support this, the robot arm and camera software architecture is designed to accommodate iterative changes and evolving requirements;
- **Adaptable state space:** Initially, the RL model's state space may rely on YOLO-generated object detections. However, if an accurate dataset for the deployed environment cannot be obtained (e.g., due to window conditions), raw RGB-D data may be used instead.

Assurance of ML for Autonomous Systems (AMLAS)

We adopt and extend the AMLAS methodology (york.ac.uk/assuring-autonomy/guidance/amlas), which consists of six stages (Figure 2) and supports the development of a safety case for ML components through safety case patterns and a structured process to: (a) **systematically integrate safety assurance** into ML component development, and (b) **generate the evidence base** to justify the acceptable safety of these components in autonomous system applications.

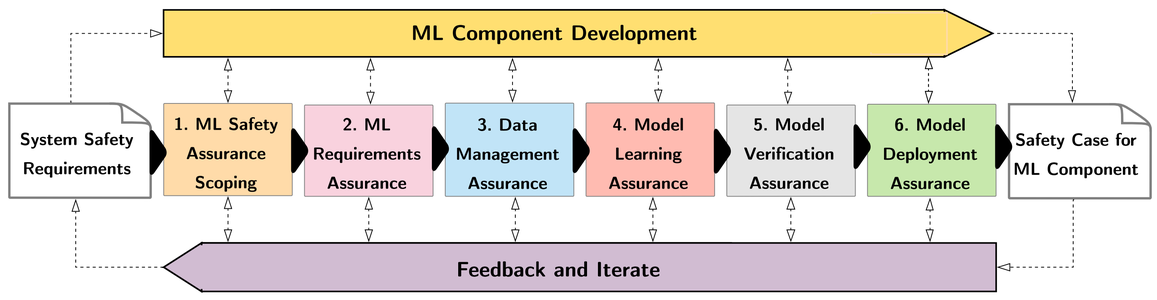

Figure 2. Overview of the AMLAS process

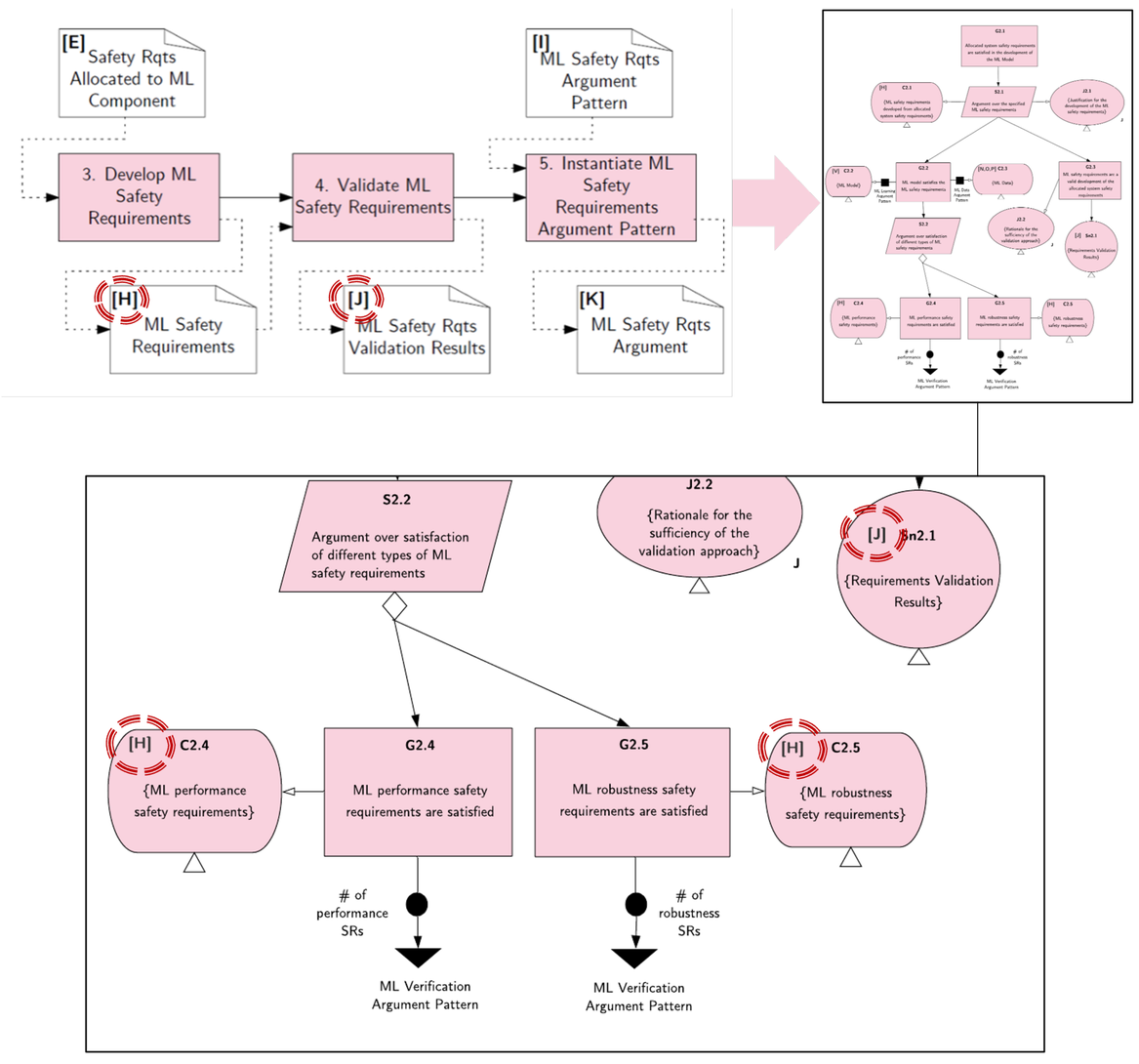

Figure 3. Stage 2 of the AMLAS process: ML safety requirements assurance process (left) and assurance argument pattern for ML safety requirements (right)

Preliminary Functional Failure Analysis (FFA) and Hazard and Operability Study (HAZOP)

Table 1. FFA analysis for object separation in autonomous mode

Failure condition	Mode of operation	Effect	Severity class	Top level safety requirements
1.1. At least 1 object of type A not in box X	Full autonomy	Some of type A objects remain in glovebox Type A objects do not get into correct waste route Delay to decommissioning and potential manual decommissioning operations	a. Delay in decommissioning b. Need for human intervention	All objects of type A are moved to box X – **TLSR1**
1.2. At least 1 object not of type A in box X	Full autonomy	Some non-type A objects placed in box X Non-type A objects get into incorrect waste route Contamination issue in waste stream	c. Contamination outside glovebox	Box X contains no objects not of type A – **TLSR2**
...	...	...	...	...
1.5. Incomplete movement	Full autonomy	Lead to 1.1 Release object at incorrect position	a, b, and c	The separation function performed by the arm shall be safely completed – **TLSR5**
...	...	...	...	...

Table 2. HAZOP study for RL component: move arm to target object

Guideword	Parameter	Deviation	Possible causes	Consequences	Recommendations
Subtle incorrect	Joint velocities	The velocity of any link is higher than intended but plausible	Out of learning distribution object Incorrectly learnt correct velocities Inaccurate reward function derived Generalisation error	Sequence of joint velocity vectors leads to a position on the end-effector, the arm or the box	Analyse object properties impacting grip at varying speeds and integrate them into the learning regime Out-of-distribution detection Limit velocity to a safe level for the worst-case object
...	...	...	...	...	...