RAIN

PRINCIPLES FOR THE DEVELOPMENT AND ASSURANCE OF AUTONOMOUS SYSTEMS FOR SAFE USE IN HAZARDOUS ENVIRONMENTS

>

White Paper, 14 June 2021

Contents

Executive Summary

Autonomous systems are increasingly being used (or proposed for use) in situations where they are near or interact (physically or otherwise) with humans. They can be useful for performing tasks that are dirty or dangerous, or jobs that are simply distant or dull. This white paper sets out principles to consider when designing, developing, and regulating autonomous systems that are required to operate in hazardous environments.

Autonomous systems use software to make decisions without the need for human control. They are often embedded in a robotic system, to enable interaction with the real world. This means that autonomous robotic systems are often safety-critical, where failures can cause human harm or death. For the sorts of autonomous robotic systems considered by this white paper, the risk of harm is likely to fall on human workers (the system's users or operators). Autonomous systems also raise issues of security and data privacy, both because of the sensitive data that the system might process and because a security failure can cause a safety failure.

Scope

This white paper is intended to be an add-on to the relevant existing standards and guidance for (for example) robotics, electronic systems, control systems, and safety-critical software. These existing standards provide good practice for their respective areas, but do not provide guidance for autonomous systems. This white paper adds to the emerging good practice for developing autonomous robotic systems that are amenable to strong Verification & Validation.

The intended audience of this white paper is developers of autonomous and robotic systems. It aims to provide a description of things that need to be demonstrable by or of their systems, and recommendations of ways to achieve this. This aims to enable strong Verification & Validation of the resulting autonomous system, and to mitigate some of the hazards already occurring in autonomous systems.

High-Level Recommendations

This white paper can be summarised in seven high-level recommendations for the development and deployment of autonomous robotic systems, which are each discussed in more detail in the main text.

1. Remember both the hardware and software components during system assurance.

It is important to remember that an autonomous robotic system contains both hardware and software; both parts should be developed to be as safe as possible. In addition to any sector-specific standard and guidelines for robotic systems, the choice of materials used in the robot should be suitable for the potential hazards in the environment. For example, a robotic system deployed in a radioactive environment should be assessed for the suitability of the materials used to build the robot's structure and casing, as well as the materials used in its internal, electrical, and mechanical components.

The software parts of the system should also be developed using appropriate, possibly sector-specific, standards and guidelines. In addition, careful attention should be paid to autonomous components, since they will be making executive decisions or interpreting sensor data, upon which decisions will be made. The strongest Verification & Validation should be used where this is possible, and Formal Methods should be considered where they are applicable.

2. Hazard assessments should include risks that have an ethical impact, as well as those that have safety and security impacts.

The introduction of robotic and autonomous systems will have an ethical impact on the workplaces and workforces where they are deployed. Risks involving safety and security are obviously ethical impacts, but here we mean risks that are not necessarily safety or security risks, but are risks of an only ethical nature.

The standard BS 8611 *ethical design and application of robots and robotic systems* (British Standards Institution 2016) provides a framework for ensuring that ethical impacts are understood and accounted for when developing an autonomous robotic system. BS 8611 uses the terminology of *ethical hazards*: a potential source of ethical harm, which is anything likely to compromise psychological and/or societal and environmental well-being. These ethical hazards describe a number of recognised ethical impacts that can arise from the development and deployment of robotic systems.

The identification of ethical hazards should be specifically included in the system's hazard assessment, and care should be taken to not introduce them where they can be avoided. Like other (for example safety) hazards it is important that ethical hazards are identified early and designed out of the final system. Ethical hazards that make it into the final system become more difficult to correct, they are 'baked in'. For example, hidden bias in a data set used to train a machine learning component can become an integral part of the system's decision-making process, producing systemically biased decision-making.

3. Take both a corroborative and a mixed-criticality approach to Verification & Validation.

Each of an autonomous robotic system's components (both hardware and software) may need different assurance methods. One methodology for achieving this could be the corroborative *Verification & Validation* described in (Webster et al. 2020); where Formal Methods, simulation-based testing, and physical testing are all used in assuring a single system. Formal Methods are very precise but the might require abstract models of the system (though there are formal approaches that work directly on programs or at runtime). Physical testing with robots provides realism, because it uses the robotic system in a real environment, but will struggle to be exhaustive. Simulations sit between these two extremes. The corroborative Verification & Validation approach links results from each of these methods to provide confidence in an overall result, and enables an iterative development and assurance workflow.

It is also likely that the different components of an autonomous robotic system will have different levels of criticality, where criticality means the level of assurance against failure that a given component needs. We recommend analysing the criticality of a system's components, so that the strongest Verification & Validation methods can be focussed on the most critical components, where they will have the biggest impact. This enables the use of, for example, Formal Methods for verifying a system's executive decisions without implying that the whole system must be formally verified. This idea pairs well with the concept of corroborative Verification & Validation described previously.

4. Autonomous components should be as transparent and verifiable as possible.

Where autonomy is used to make executive decisions about what the system should do, it is very important to be able to understand why a decision has been taken and to be able to verify that the correct decision will be made under all circumstances. This is particularly useful for mitigating the challenge of extensively testing robotic systems.

Where autonomy is used to, for example, interpret sensor data, it is important to minimise incorrect interpretations and to ensure that a incorrect interpretation does not lead to unsafe behaviour. If this cannot be achieved directly, then the system's architecture should be arranged so that there is a more analysable 'governor' between the autonomous component and the rest of the system.

In both cases, the choice of how to implement autonomy will impact the kinds of Verification & Validation techniques that are available. For example, an agent-based approach to autonomy is likely to be easier to formally verify than an approach based on machine-learning. How autonomous decisions are implemented should be considered analogously to picking a programming language for a safety-critical system: how easy it is to verify, understand, and demonstrate the safety of the autonomous component should be key factors in the decision of how to implement that component.

5. Tasks and missions that the system will perform should be clearly defined.

Task definitions could include expected inputs and outputs, a step-by-step description of the behaviour, potential failure modes, etc. Mission definitions could be seen as a collection of tasks, but should also include higher-lever concerns, like potential hazards in the deployment environment. These definitions makes it clear to the developers and end users what the system can and cannot do. It also describes the system's requirements, which are essential for meaningful Verification & Validation; and the system's deployment context, which is essential for things like assessing the suitability of the materials used to build the robotic system (as previously described).

6. Dynamic Verification & Validation should be used to complement static Verification & Validation.

Static checking can provide crucial confidence in a system's correctness. For example, verifying that an autonomous component that makes executive decisions will never choose an unsafe behaviour. However, any system that interacts with the real world is bound to face some uncertainty, so dynamic Verification & Validation techniques (techniques deployed at runtime) should be used as well. This helps to bridge the *reality gap*, between design-time assumptions about the environment and the run-time reality. In addition to just performing runtime monitoring, this approach could be extended to perform runtime enforcement.

Runtime monitoring (or enforcement) can be high-level, for example checking the robotic system doesn't exceed a safe speed; or it could be low-level and more focussed, for example governing a machine learning component to ensure that it doesn't make decisions outside of a safe operating envelope. Either way, dynamic Verification & Validation during deployment can help to provide extra confidence that the system is continuing to operate correctly.

7. System requirements should be clearly traceable through the design, the development processes, and into the deployed system.

Autonomous robotic systems that are deployed into hazardous environments often require approval from a regulatory body before they can be used. Even if they don't, then it is likely that the system will need to be acceptable to, and trusted by, human users or operators. Traceability of the system's requirements through the development process is key to showing that the verification is checking for properties or behaviours that support the requirements, giving overall confidence in the correctness of the system. Being able to trace the requirements through to the final system (either as artefacts of static Verification & Validation, dynamic Verification & Validation components, or both) helps provide confidence that the final system fulfils the original requirements. This can help to ease regulatory efforts at the same time as potentially reducing risk.

1 Introduction

These guidelines set out key principles to consider when designing, developing, and regulating autonomous systems that are specifically required to operate in hazardous environments (eg radiological, chemotoxic, etc). Particular questions that this white paper addresses include:

1. What do we consider to be an *autonomous* robotic system?
2. How do we demonstrate that the risks and benefits arising from the application of robotic or autonomous systems are acceptable?
3. What are the differences between safety engineering for a human-controlled robot and for an autonomously-controlled robot?
4. What are the factors to be considered in the implementation of an autonomous system?

An autonomous robotic system can be viewed as a combination of a physical (robotic) element and a logical (software) element. The robotic part of the system enables interaction with the physical world. The software part of the system enables autonomous decision-making, based on sensor input and (often pre-built) models of its environment. Autonomous robotic systems are highly complex, usually mission-critical, and often safety-critical. Autonomous systems are discussed in more detail later in this section.

Moving from a system being broadly human controlled (either remotely or via scripted automation) to autonomous control requires a new approach to Verification & Validation. Adding a software component or layer that make choices brings both new challenges and new opportunities. While it is challenging to ensure that the system will make safe choices, given the available sensor input; it also provides the opportunity to improve the confidence in the safety of the system by examining the decision-making process. As such, it should not be wasted.

Physical safety controls can be used to constrain the robotic system, but strong Verification & Validation of the software system is also highly recommended, not least to help establish user trust. Where physical controls are not possible, then the strongest Verification & Validation methods that are applicable to the system (or component) should be used. In either case, care should be taken to choose Verification & Validation methods that are suitable for describing both the system and the properties to be checked.

It is important to have detailed definitions of the tasks that an autonomous robotic system will perform. Task definitions are key design artefacts that ease Verification & Validation efforts, they also makes it clear to the developers and end users what the system can and cannot do. A task definition could include: expected inputs and outputs, a step-by-step description of the behaviour, potential failure modes, etc. It is also important to have a detail definition of the mission that an autonomous robotic system will perform. A mission definition could be seen as a collection of tasks that achieve the mission.

A good mission definition should also include higher-lever concerns, like the types of hazard expected in the deployment environment. Again, this sort of definition is useful for Verification & Validation methods and the mitigation of potential hazards, but also for useror operator trust.

A single system will often require multiple methods of assurance, because of the multiple different types of components that compose one system. One methodology for achieving this could be the corroborative *Verification & Validation* described in (Webster et al. 2020); where Formal Methods, simulation-based testing, and physical testing are combined in the assurance of a single system.

In addition to hazards that arise from the violation of traditional safety principles, there are hazards that can arise from a range of considerations that are often loosely grouped together under the term *ethics and artificial intelligence*. The global landscape of ethics guidelines for AI is fragmented, but a recent survey has found convergence on five principles: transparency, justice and fairness, non-maleficence, responsibility, and privacy, with a further six principles: beneficence, freedom and autonomy, trust, sustainability, dignity and solidarity mentioned in many guideline documents (Jobin, Ienca, and Vayena 2019).

Carelessly developed and deployed AI systems have already been seen to violate these principles both through unintended side-effects of their operation and through their interactions with users. This can bring reputational harm to the organisations responsible for the systems and, at worst, cause real damage to users, bystanders and the general public. As our understanding of the ethical risks posed by AI and autonomous systems is maturing, so too is the development of methodologies for ethical risk and impact assessments as well as standards for the design and development of systems that adhere to these principles. It is therefore both practical and desirable to perform such assessments when proposing the use of an autonomous system in order to identify potential hazards that can arise from violations of the principles. This is discussed in Section 2.1 and further guidance can be found in (British Standards Institution 2016)

Key Points

- The introduction of a robotic or autonomous system must be to the benefit of the operators or users of the system. This includes: giving priority to automating (with a robotic or autonomous system) tasks that are potentially harmful (physically, mentally, ethically, etc.) to operators or users, even if this would be more expensive (see (High-Level Expert Group on AI 2019 Recommendation 3.2); the consultation of workers during the design and development of robotic or autonomous systems (see (High-Level Expert Group on AI 2019 Recommendation 3.3)); and ensuring that data collection or monitoring by the system does not become surveillance of the operators, users, or any responsible person.

- Ethical issues are potential sources of hazards, so a hazard assessment should include the assessment of ethical issues. Such an assessment needs to be a specific part of the system's requirements and design. (See, for example, the BSI guide to *ethical design and application of robots and robotic systems* (British Standards Institution 2016).)

- Autonomous components that make high risk decisions should be designed and implemented in a way that enables strong Verification & Validation of their choices, for example through the use of Formal Methods. This is particularly necessary to mitigate the difficulty (and safety implications) of checking the autonomous decisions via extensively testing robotic system.

- The physical (robotic) part of the system should be developed and assured using appropriate standards or guidelines. Additionally, the choice of materials used in the robot need to be suitable for the potential hazards (e.g radiation) in the environment.

- The software architecture of the system should be arranged so that components that are unpredictable or difficult to analyse are not able to directly influence the system's decisions; their output should be checked by more reliable or analysable components. For example, a statistical data-driven machine learning classifier should not be directly connected to actuation, without having the classifications checked by a more analysable 'governor'.

- The overall design of a robotic or autonomous system should aim to maximise the system's effectiveness while also minimising safety risks, risks of failure, operation and maintenance risks, etc. This includes all aspects of the design, such as its mechanical arrangement, hardware and software selection, etc.
- Task and safety requirements need to be clearly traceable in the design, the development processes, and through to the system

This final point is considered to be of crucial importance to those organisations developing autonomous systems and justifying their use.

What is an Autonomous System?

At its most basic, an autonomous system is one that has the capability to make decisions free from human intervention. This is subtly different to automatic systems or adaptive systems, which can react to changes in their environment without human intervention but doesn't make decisions. For example, a thermostat can automatically toggle a switch or adapt the heating/cooling in reaction to sensed temperature changes, but we do not consider it to have made a decision. These three concepts can be described as follows.

- **Automatic** systems are pre-programmed to react to input and are unlikely to have internal models if their environment. They are useful for more predictable situations where the system is required to perform a small number of tasks repeatedly.
- **Adaptive** systems are tightly linked to (and often driven by) the system's operating environment using feedback controllers described using differential equations. Such approaches are useful where the system is performing monitoring tasks where the object is to maintain some state that can be described using calculus.
- **Autonomous** systems are able to make decisions that require intelligence and situational understanding. These decisions will will take the environment into account, but the system decides what to do based on its internal priorities and goals. They are useful in open environments where there is a large range of possible decisions to be made, often based on uncertain input.

Broadly, autonomous robotic systems can be described as:

- **semi-autonomous**, where the "robot and a human operator plan and conduct the task, requiring various levels of human interaction" (Standing Committee for Standards Activities of the IEEE Robotics and Automation Society 2015), or;
- **fully-autonomous**, where the robot performs a task "without human intervention, while adapting to operations and environmental conditions" (Standing Committee for Standards Activities of the IEEE Robotics and Automation Society 2015).

A newly deployed robotic system might be designed to be fully-autonomous, but often a human-controlled system will be adapted to become (semi or fully) autonomous. As a simple example of an autonomous system, consider a robot vacuum cleaner. An automatic cleaner will follow exactly the same path around a room, regardless of the different environmental conditions. An adaptive cleaner could be programmed to spend longer on the dirtiest area. Whereas, an autonomous cleaner could choose either of the above, but could also choose not to do *any* cleaning, perhaps because it knows that someone in the house is sleeping and it has decided that the cleaning noise will wake them or because waiting until tomorrow to clean (when no one is in the house) will provide better efficiency.

There are various frameworks describing the different levels of autonomy that a system might have, with no single framework being universally adopted. They can be useful for describing a system's capabilities, but it important to be clear *which* levels of autonomy you are referring to.

The original levels of autonomy were defined for undersea teleoperation systems (Sheridan and Verplank 1978), though they are worded rather neutrally so may have wider applicability, and were revised in (Endsley 1999). The levels of autonomy developed by SAE International for driverless cars (On-Road Automated Driving (ORAD) committee, SAE International 2018) are often adapted to suit the deployment context of the autonomous system, but this is by no means ideal. There have been other efforts to create frameworks that are specific to particular deployment contexts, for example spacecraft (Proud, Hart, and Mrozinski 2003); or to be generic and applicable to all autonomous systems, for example (Huang et al. 2005; Beer, Fisk, and Rogers 2014). A detailed review of different frameworks for levels of autonomy can be found in (Beer, Fisk, and Rogers 2014, Sect. 3).

Whichever 'level' of autonomy a system has, it is important to note that autonomy can be achieved in different ways and that the chosen approach has implications for the Verification & Validation techniques that can be applied to the system's decision-making. For safety-critical deployments, autonomy should be achieved by means that enable strong Verification & Validation methods. We discuss this in more detail in Section 2.3.

Document Structure

The rest of this white paper is arranged as follows. Section 2 discusses principles that we consider to be prerequisite to those discussed in later sections, but are are beyond the scope this document to describe in detail. Section 3 describes general principles for robotic systems intended for use in hazardous environments. Section 4 describes principles to consider when developing and deploying a *human-controlled* system, and Section 5 describes principles for *autonomous systems*.

Sections 3, 4, and 5 each build on the previous section. For example, when developing an autonomous system, the principles in Section 5 should be considered alongside those in Sections 3 and 4.

2 Prerequisite Concepts

The scope of this document is the development and assurance of autonomous systems. However there are principles that we consider to be prerequisite to the development and the deployment of an autonomous system. These principles are also very complex, so this section aims to introduce them and guide the reader to more detailed information.

2.1 Ethical Impact

All technologies have ethical impacts as they are introduced into society. The integration of complex computational and robotic systems into workplaces are raising a number of ethical issues which, even where these do not translate directly into hazards, have impacts on user trust and societal acceptability. The analysis and assessment of the ethical impact and risks posed by the introduction of autonomous systems is a fast moving area. While standards for the ethical development and deployment of AI and autonomous systems are in development by organisations such as the IEEE, most of these have yet to be published. However, BS 8611 *ethical design and application of robots and robotic systems* (British Standards Institution 2016), published in 2016, provides a standards-based framework for ensuring that ethical impacts are understood and accounted for when proposing the introduction of an autonomous robotic system. The development of similar standards is already underway.

BS 8611 uses the terminology of *ethical hazards*: a potential source of ethical harm, which is anything likely to compromise psychological and/or societal and environmental well-being. These ethical hazards describe a number of recognised ethical impacts that can arise from the development and deployment of robotic systems. We adopt this terminology of ethical hazard here but this should not be interpreted as implying an insistence that systems adhere to BS 8611 specifically.

Any hazard assessment should include the assessment of ethical hazards. Like other hazards (such as safety hazards) it is important that ethical hazards are identified early and designed out of the final system. Ethical hazards that make it into the final system become more difficult to correct, they are 'baked in'. For example, hidden bias in a data set used to train a machine learning component can become an integral part of the system's decision-making process, thereby producing systemically biased decision-making.

Ignoring ethical hazards, or leaving them to be discovered or dealt with after the system has been deployed is likely to be costly and less effective. The harms these hazards can cause can directly impact the system safety, particularly where they effect the way that the operators or users interact with the system. Ethical hazards that make it into the final system can also cause huge reputational damage to the system's developing and/or operating organisation. Finally, not addressing ethical hazards is, by definition, un-ethical.

The ethical hazards will change for different systems and different people, so a comprehensive assessment of the ethical hazards of the specific system with its operators and users should be performed. Defining the principles and process for this assessment is outside the scope of this document; this sub-section presents suggestions for both principles and process, but leaves the developer the flexibility to choose the most suitable route.

Before development, the impact of *introducing* a robotic or autonomous system needs to be analysed, to ensure that it would be to the benefit of the operators or users of the (current or proposed) system. For example, priority should be given to automating (with a robotic or autonomous system) tasks that are potentially harmful (physically, mentally, ethically, etc.) to operators or users, even if this would be more expensive (High-Level Expert Group on AI 2019 Recommendation 3.2). To ensure that the system would benefit the operators or users, they should be meaningfully consulted during the design and development of the system (High-Level Expert Group on AI 2019 Recommendation 3.3).

The standard BS 8611 *ethical design and application of robots and robotic systems* (British Standards Institution 2016) outlines a process of identifying the potential ethical hazard and then determining the ethical risks[1] (British Standards Institution 2016 Sect. 4).

A first step in understanding the potential ethical hazard posed by a system is identifying the relevant ethical principles to which the system should adhere. Listing the ethical principles to use in the assessment of every possible robotic or autonomous system would not be useful. BS 8611 recommends that the relevant ethical principles should be "identified and defined by engaging with end users, specific stakeholders and the public". Nevertheless numerous sets of suggested principles exist that can be used as starting points. A recent survey of standards and guidelines on ethics for artificial intelligence found that they had converged on the principles of transparency, justice and fairness, non-maleficence, responsibility, and privacy (Jobin, Ienca, and Vayena 2019). The same survey found a further six principles that were present in some of the existing ethics guidelines: beneficence, freedom and autonomy, trust, sustainability, dignity, and solidarity.

BS 8611 itself describes some general ethical principles that can be used in *addition* to a bespoke assessment (British Standards Institution 2016 Sect. 5). They are divided into four ethical issues, each comprising several ethical hazard. First are societal issues: loss of trust in the system, deception, anthropomorphisation, privacy and confidentiality, lack of respect for cultural diversity and pluralism, robot addiction, and employment.
Next are application issues: misuse, unsuitable use, dehumanisation of users and operators, inappropriate "trust" of a human by the robot, and self-learning systems exceeding their remit. Then, commercial or financial issues: approbation of legal responsibility and authority, employment, equality of access, learning by autonomous robots, informed consent, and informed command.
Finally, environmental issues: environmental awareness for both the robot and appliances, and the operations and applications.

BS 8611 highlights methods of mitigating, validating and verifying (the absence of) these types of hazards. Mitigation methods include things like including a particular principle in the system's design, or providing a particular type of information to an operators or users. Validation and verification includes well known techniques like user validation, compliance testing, and software verification; as well as techniques requiring wider expertise, such as structured assessments of economic, social, and legal impacts.

We reiterate that these are suggested starting points for the identification and mitigation of ethical hazards. A thorough analysis is needed for each system in its final context, which should be repeated if the context (mission, users, operators, software, hardware, etc) changes.

2.2 Materials Suitability

To affect the real world, an autonomous system is usually embedded in a robotic system. This gives it a physical presence and the ability to interact with its surroundings, but leaves the system vulnerable to hazards in its environment. So, the materials used in the robot need to be assessed for their suitability to the hazards presenting in the deployment environment. It is important to note that this covers both the materials used to build the

1. An ethical risk is the probability of ethical harm from occurring from the frequency and severity of exposure to a hazard.

robot's structure and casing, as well as the materials used in its internal, electrical, and mechanical components.

For example, deployment in a radioactive environment will influence choices of materials used in a robot. Different types of radiation will have different effects on different materials. This can impact a wide range of components, from the metals used to construct the robot, to the plastics used for electrical insulation, and even the lubricants used for moving parts. There is also the potential for radiation to affect the state of a robotic system's electronic components, so the likelihood and impact of this should also be analysed.

In this example deployment, the types of radiation that are expected in the robot's environment should be analysed to identify how they will impact the materials used in every part of the robotic system. This analysis should be used, alongside an agreed Mission Definition (detailing mission duration, forecast dose rate, operational requirements, etc.), to determine which materials are suitable for the deployment environment. A detailed description of how long each part of the robotic system can be expected to last under particular radioactive conditions should be compiled. This can be used by runtime monitors (see Sections 3.3, 4.3 and 5.3) to provide 'health' information to a user, operator, and/or autonomous system.

2.3 Verification and Validation of Autonomous and Robotic Software

There are a range of techniques for checking that a software system functions correctly. Much of an autonomous software system (in particular an autonomous robotic system) may use well-understood software and algorithms. As such, it is expected that relevant, current good practice is observed during the system's development. This may include current guidelines, standards, or regulator advice that is applicable to the system and its intended use. Examples of software verification techniques already in use for robotic systems include: standardised or restricted middleware architectures, software or physical testing and simulations, domain specific languages for specifying checkable constraints, graphical notations for designs, and generating code using model-driven engineering approaches (Luckcuck et al. 2019, Sect. 2). However, systematic testing of autonomous robotic systems can be challenging.

2.3.1 Verification & Validation Challenges from Autonomous and Robotic Systems

Physical tests with a robotic system can be dangerous in early phases of development, and are often difficult or time-consuming to set-up or run. As such, greater reliance on approaches based on simulation and code analysis may be necessary even though these may lack the fidelity of the actual operating environment. Even so, there is some evidence that even a low-fidelity simulation of a robot's environment can reproduce bugs that were found during field tests of the same robot; Sotiropoulos et al. (2017) found that of the 33 bugs that occurred during a field test of a robot, only one could not be reproduced in their low-fidelity simulation.

Autonomous systems are often designed to be used in (partially) unknown situations, where the autonomy enables the system to operate when plans cannot be made beforehand. Some techniques for implementing autonomy create artefacts that are opaque, difficult to analyse, which may find solutions that are surprising to humans. In many situations this very ability to behave flexibly in unexpected ways is a requirement of the software and the reason for its development, but it clearly presents challenges to the analysis of the risks of deploying the system.

For many autonomous systems it may be possible to limit the number of components that are difficult to analyse. This reduces the amount of mitigation, related to these components, required to argue that the system is adequately safe. It may also be possible to pair a component that is difficult to analyse with a monitor that checks that component's outputs against its expected behaviour or range of behaviours. If the component's behaviour differs from the expectation, then the monitor can take some mitigating action, for example: logging, alerting a useror operator, or some automatic remedial behaviour. In the case of a monitor triggering remedial behaviour, this could be considered runtime *enforcement*. The system's expected behaviour could be described in a formal specification or as a more traditional software artefact. The key features of a monitor is that is should be simpler and more analysable than the component it is monitoring. Also, it may be appropriate for a monitor to run on hardware that is separate from the component it is monitoring, to safeguard against a hardware error effecting both the component and the monitor. These software engineering and architecture techniques are to be encouraged where they can help improve analysability or overall safety.

Aside from the challenges that autonomous software presents, it also provides the opportunity to examine a system's decision-making mechanisms. Especially where an autonomous component is taking over from a previously human-made decision, this can play a key role in maintaining or improving system safety in a demonstrable way. Ignoring the chance to interrogate the decisions an autonomous system is making would be a waste of this opportunity.

The choice of technique to implement autonomy impacts the kinds of Verification & Validation techniques that can be applied to it – as mentioned in Section 1. Autonomous software is often implemented in a different programming paradigm to procedural or object-orientated programs. For example, agent-based autonomy is often a collection of guarded actions that the system chooses from at runtime, based on what is perceived in the environment. Machine learning techniques may enable a system to derive and exploit complex statistical relationships between inputs, sequences of actions, and results. Sometimes these relationships, once learned, can be expressed as clear rules which can then be analysed in traditional ways, but in some cases these relationships are too complex to be meaningfully expressed in this way. Clearly uses of machine learning that produce analysable results are preferable in terms of assurance, but where this is not possible consideration must be given to how incorrect derivations can be detected and mitigated.

Adaptive autonomous systems are often implemented using feedback control approaches, which usually use differential equations to model the system being controlled when the feedback controller is being tuned. Three key properties that should be checked for these approaches are: (1) that the model is validated against the real world, and that it is based on valid data; (2) that any abstractions in the implementation of the feedback controller are valid, and; (3) that the feedback controller is robust against variations in the systems performance, and can cope with any differences between its model and the real system.

Many autonomous robotic systems have their perception units and/or control units implemented with Deep Neural Networks (DNNs). Techniques are being developed for the verification and testing of deep learning (a survey of the current state of the art can be found in (Huang et al. 2020)). A key property that should be considered when making a case for the correctness of a DNN is whether two inputs to the classifier that appear identical to a human observer can be classified differently by the classifier - clear definitions that attempt to capture this property exist as do approaches to verifying (either formally or via testing) that they hold. Therefore any safety case for an autonomous system that includes such a classifier should include evidence that the classifier conforms to one of these definitions.

2.3.2 **Formal Methods**

The most rigorous Verification & Validation techniques are referred to as formal methods: mathematical and logical techniques for defining desired system properties, and the analysis of specifications, designs and even source code using mathematical proof-based techniques. Formal methods can also be used to generate provably correct source code from specifications and designs. These techniques can be strategically applied to minimise the likelihood of introducing faults during software development.

There are many examples of formal specification and verification applied to robotic and autonomous systems in the research literature (Luckcuck et al. 2019), but there are notable examples of their application in industry as well (Woodcock et al. 2009). One potential use for formal methods is to provide an unambiguous specification of the system's requirements. This enables the verification of a design against its requirements. Formal specifications are often built by examining a natural-language specification and corresponding with domain experts, to fill in the gaps. Formalising requirements was found to be the most often used technique in the industrial projects surveyed in (Woodcock et al. 2009). Clear and unambiguous specifications, linked to designs, can also help show where opaque and difficult to analyse parts of the system can be eliminated, and where they are truly necessary.

Formal methods can also be used to perform rigorous static and dynamic analysis. The static analysis technique most often used it the research literature is model checking (Luckcuck et al. 2019), which is an automatic process that exhaustively checks if a property holds in every state of a formal system specification. Some model-checkers accept timed or probabilistic specifications, and program model-checkers can check a program against a formal specification. There are also statistical model-checkers that, in similar way to statistical testing, take samples of the available paths through a specification. This can enable the checking of very large specifications.

Statistical AI approaches, such as deep neural networks, can also be analysed by statistical model-checking, which can be useful even where absolute guarantees of behaviour can't be given. Such techniques can help identify worst-case boundaries of the output, and analyse the stability of the system in the face of small changes in input (e.g., whether altering a few pixels in an image might cause a drastic change in the resulting analysis – for instance interpreting a red traffic light as a green one).

For dynamic analysis of a system, runtime verification can be used: this is the style of monitoring described above; where the system's behaviour is compared to a formal specification. If the system's behaviour differs from the specification, then the monitor can log the failure, alert the user, or trigger mitigating actions. An extension of this idea is predictive runtime verification, where a formal description of the system is used to predict the satisfaction or violation of a property. If the property will continue to be satisfied, then the monitor can be removed to save system resources; if the property will be violated, then action can be taken to prevent it.

In general, runtime verification bridges the reality gap (between a model and the real world) by checking formal properties and assumptions at runtime. runtime verification has the potential to mitigate the risks involved in incorporating unpredictable or statistical techniques into autonomous software, by providing guarantees that a component's behaviour will remain within some guaranteed safe envelope, while enabling the creativity of the statistical technique to find solutions within that envelope. As previously mentioned, a runtime monitor might be extended to trigger runtime enforcement of safety properties.

Using formal methods are not always possible or practicable, but they are a useful part of the toolkit for verifying autonomous systems. There are also a range of less formal but still rigorous approaches to the analysis and testing of software that help identify and eliminate bugs in order to provide high degrees of assurance of system behaviour.

2.3.3 **Verification & Validation Tools**

The development of software for autonomous systems relies upon effective use of appropriate Verification & Validation tools, such as debuggers, test frameworks, simulators, and formal methods tools, as part of a well-structured rigorous modern process that follows a defined software engineering lifecycle. These are software tools that do not make it into the final software but are used during development to support testing and assurance activities to demonstrate that the system's software is correct. A key concern is being able to establish the suitability of Verification & Validation tools that are used and to show that they cannot unintentionally undermine the assurance they are being used to provide.

In assessing the suitability of a Verification & Validation tool, consider how it might fail and the consequences of a failure. The types of failures are likely to be dependant on the type of tool. The following non-exhaustive examples demonstrate some of the types of failure that could occur:

- a software testing framework might be vulnerable to similar programming errors as the language it is testing, like confusing "=" and "==";
- a simulator might not accurately account for certain physical measurements, such as friction; or,
- the state space of a specification might be too big for the model checker (or rather the hardware running the model checker) to cope with.

If the consequences of Verification & Validation tool failing only have a small impact on the system's safety assurance, then a fairly untested tool might be acceptable. If, as is more likely, the consequences could have a high impact on the assurance for safety of the system or component, then prevention or mitigation measures should be considered. The categorisation of the impact of a Verification & Validation tool failing, and of the appropriate prevention or mitigation measures, will depend on things like the system's deployment environment, mission, and applicable regulatory regime. For example, the part of the tool that is known to fail to identify errors could be avoided, or if there are types of known error that a tool fails to identify these could be detected by an independent tool or technique, or for higher levels of assurance use of more than one third-party accredited Verification & Validation tool with comparison of types of errors detected.

2.4 Mixed-Criticality Systems

Real-Time and embedded systems are increasingly often composed of components with different levels of criticality on the same hardware platform, hence mixed-criticality systems. Here, criticality means the level of assurance against failure that a given system component needs. Different definitions of these criticality levels exist, and they may be named, for example, Automotive Safety and Integrity Levels, Design Assurance Levels (or Development Assurance Levels), or Safety Integrity Levels. The use of mixed-criticality systems is often driven by strict non-functional requirements, such as weight, heat generation, or power consumption. For a comprehensive review of mixed-criticality systems research see (Burns and Davis 2018).

In Sections 3, 4, and 5 we recommend analysis of the criticality of system components to enable to the strongest Verification & Validation methods (such as Formal Methods) to be used on the most critical components. This suggestion aims to ensure that the most critical components are assured against failure, without requiring the entire system to be (for example) formally verified.

To avoid confusion with nuclear criticality, in the rest of this document we refer to the criticality of system components as their 'importance to the system's safety'. This can still be thought of in terms of Automotive Safety and Integrity Levels, Design Assurance Levels (or Development Assurance Level), or Safety Integrity Levels.

3 General Principles for Safety-Critical Robotic Systems

This section introduces general principles to consider for robotic systems that are deployed in hazardous environments. These principles focus on the physical aspects and deployment context of the system. They should be considered alongside those in Sections 4 and 5, as appropriate to the system.

3.1 Design

1. Identify the importance to safety[2] of each software and hardware component, then use the most robust design methods for the most important.

   This can produce a design that can have its safety assured, while focussing robust methods on the components that are key to the system's safety. (See 3.2 (1), 3.3(1) and 4.2 (1)).

2. Design of the facility may need to be physically extended to accommodate the robotic system. This may include considering the storage and maintenance of the robot and how it is transported between these areas and its deployment environment.

3. If the design of the system requires that it, its components, or any debris may be removed from the deployment environment, this needs to be assessed for potential contamination.

4. The design needs to describe the decommissioning approach for the system. It may be that parts of the system can be decontaminated and removed, or the system may need to be disposed of 'whole'. This element of the design needs to take into account the environmental impacts of the decommissioning strategy.

2. See Section 2.4.

3.2 **Verification and Validation**

1. Use strong Verification & Validation techniques, such as Formal Methods, at early stages of the system's lifecycle. These can be used to prototype for a task or a software component that is highly important to the system's safety, while ensuring (if Formal Methods are used) that the specification of the task or component is rigorous.

2. Introduce simulation-based testing to check the system's integration and to find statistically unlikely states that cause a failure. It is crucial that the simulation's limitations are well understood and that the differences between it and the actual system are adequately managed. Simulations will, ideally, include high fidelity controls that enable the assessment of how a user's or operator's actions impact the system's behaviour.

3. Demonstrate by several methods – which should include physical testing – that the system operates as specified, in the absence of potential hazards. It is important to note that physical or software testing will struggle to be exhaustive, which is why it needs to be combined with or preceded by Verification & Validation techniques that can be exhaustive (such as Formal Methods). An example of using different Verification & Validation techniques to corroborate each other can be found in (Webster et al. 2020).

3.3 **Operation**

1. Verifiably correct runtime monitors could be used to perform 'online requirements checking', comparing the system's operational behaviour to its requirements. The behaviour of the whole system could be monitored, or the monitoring could be focussed on the components most important to the system's safety[3] as described in 3.1 (1). If the requirements are no longer being met, then the system could: move to a safe state until the problem is resolved (see. 3.3 (2)), report the failure to a responsible person, or log the failure.

2. For components that are important to the system's safety[3] runtime monitors could be extended to provide runtime enforcement. For example, if the monitor find that the system is no longer meeting its safety requirements, then it could trigger a safe state until the problem is resolved (see 4.1).This could include enforcing limits on the speed of movement or force applied by the system, even when a userattempts to exceed these limits.

3. In addition to any physical barriers used to contain the system, its software can be used to create logical barriers (this is related to 3.3 (2). For example, monitoring a robot's position and preventing it from moving forward once it enters a 'buffer' zone around the physical barrier. This has the advantage of reducing the likelihood of a robot becoming stuck at a physical barrier, which would require manual intervention to remedy. Validating the logical barriers is important and highly context dependent. One example of validating logical barriers could involve analysing the reaction and braking time of the robot at its maximum speed and ensuring that the logical barrier is far enough away that it wont hit the physical barrier.

3. See Section 2.4.

3.4 Maintainance, Recovery and Decommissioning

1. To enable for maintenance, recovery, and decommissioning; ensure either safe human access to, or safe removal of, the robotic system from its hazardous environment.

   This could include access to the system's deployment environment (while powered down) or the ability to remove the physical system from the deployment environment (if needed or possible). It needs to be demonstrable that this is possible even in the event of some or all of the system failing. This is likely to need multiple techniques, for example the analysis of the system and deployment environment, simulation, formal analysis of the control algorithm, physical testing, etc.

2. If the system, its components, or any debris may be removed from the deployment environment, this needs to be assessed for potential contamination. This requires that the potential for this to happen has been considered, as mentioned in 3.1(3) and 3.1(4).

3. The decommissioning strategy (which should be considered during the design phase, 3.1(4)) should take into account the environmental impacts of decommissioning the system.

4. Modifications to the software or hardware (which may include upgrade, reconfigurations, etc) need to be assessed against the system's original design and verification artefacts, to ensure that the safety properties (of both the modified component and the whole system) are preserved. Additional monitors, or modifications to existing monitors, may be required.

5. Components should be added to the system to allow it to monitor its own 'health'. This can be used to track its current capability and suitability for the task, and to predict future (hardware or software) failures. This is useful for issues like hardware wear and tear, augmenting the human ability to consider replacing a component when it 'feels like its about to break' with data- or model-based prognostics.

   It needs to be provable that the data collection and monitoring does not become surveillance of any human interacting with the system.

4 Principles for Human-Controlled Robotic Systems

In addition the principles in Section 3, this section outlines the principles to consider for robotic systems where there is significant human-in-the-loop control.

This can range from remote-control, where a "human operator controls the robot on a continuous basis, from a location off the robot, via only [their] direct observation" (Standing Committee for Standards Activities of the IEEE Robotics and Automation Society 2015); to telecontrol, where a "human operator, using sensory feedback, either directly controls the actuators or assigns incremental goals on a continuous basis, from a location off the robot" (Standing Committee for Standards Activities of the IEEE Robotics and Automation Society 2015)[4].

The principles below are organised into three sections: Design, Sect. 4.1; Verification & Validation, Sect, 4.2; and Operation, Sect.4.3. The principles are numbered for identification not to indicate ordering.

4.1 Design

1. Design of the robotic system's working environment should contain similar safety constraints to a conventional (non-robotic) deployment environment, to mitigate the foreseeable hazards caused by the task itself. This includes re-design of an existing working environment to accommodate a robotic system. Where an existing human working environment is being re-designed to accommodate human-controlled robotic systems, attention should be paid to the potential difference in speed, range of movement and strength of the robotic system in comparison to humans.

4. The key difference between remote-control and teleoperation is whether the operator is directly observing the robot, with their eyes; or indirectly observing the robot, via sensors.

2. Designing the task(s) that the robotic system will perform may involve simply replicating how a human worker does it, but the task may need re-designing to, for example: accommodate the differences between robots and humans, or to improve efficiency.

   The robotic system should perform a task demonstrably as safely a trained useroroperator. Additional safety considerations (e.g. failsafe states and/or passive measures) are likely to be necessary.

3. Multiple, parallel, safety systems could be designed, to provide the required level of assurance (e.g. through defence-in-depth). These could be a combination of physical, hardware systems, and software systems.

4.2 Verification & Validation

1. Use the most robust Verification & Validation methods for the most critical hardware and software components. This requires the identification of those components most important to the system's safety[5], as described in 3.1 (1). For example, if the system's design has identified the need for components that limit the robot's speed of movement or potential to apply force, then these components should be candidates for robust Verification & Validation.

   This approach enables system safety assurance, whilst prioritising the components that are key to the system's safety when chooisng where to apply robust Verification & Validation. (Also see 3.3 (1)).

2. Demonstrate (via human factors assessments and/or user evaluation studies) both user competence and user confidence in the new remote-controlled system.

3. If a robotic system that is replacing an existing system (robotic or otherwise), then the safety boundaries and constraints of the new system should be assessed to ensure that no additional safety concerns have been introduced and that existing safety mitigations remain valid. This should be done before deployment of the new system.

4. users and operators should only be in physical proximity to the robot after it has been demonstrated that the robot cannot harm them under any circumstance, including failures.

4.3 Operation

1. Information about the use of a robotic system could be logged to, for example, help model how a task is performed safely or highlight areas where new safety features could be useful. This information could also be used to validate how an autonomous system performs the task. However, his data collection needs to be transparent to the humans involved and only be collected with informed consent (see 4.3 (2)).

2. It should be demonstrable that any data collection or monitoring does not become surveillance of any operator, user, or any other human interacting with the system. Any monitoring by or of the system needs to be transparent to the humans involved and only be collected with informed consent.

5. See Section 2.4 for a discussion of this concept.

5 Principles for Autonomous Robotic Systems

In addition to the principles in Sections 3 and 4, this section outlines the principles to consider for systems ranging from semi-autonomous up to fully-autonomous.

In a semi-autonomous robotic system the "robot and a human operator plan and conduct the task, requiring various levels of human interaction" (Standing Committee for Standards Activities of the IEEE Robotics and Automation Society 2015). This works best where well-defined tasks are delegated by the operator to the autonomous system. In a fully-autonomous robotic system, the robot performs a task "without human intervention, while adapting to operations and environmental conditions" (Standing Committee for Standards Activities of the IEEE Robotics and Automation Society 2015).

A common route for the introduction of an autonomous robotic system is to begin with a remote-controlled system, where a human operator is still making the key decisions, and steadily delegate tasks or decisions to autonomous components of the system. This staged approach has the advantage of only automating small sets of tasks or decisions at a time, which can aid Verification & Validation and monitoring efforts, as well as slowly building the confidence of users and operators in the system's safety. As previously mentioned, priority should be given to automating tasks that are potentially harmful (physically, mentally, ethically, etc.) to operators or users, even if this would be more expensive (High-Level Expert Group on AI 2019 Recommendation 3.2). To ensure that the system would benefit the operators or users, they should be meaningfully consulted during the design and development of the system (High-Level Expert Group on AI 2019 Recommendation 3.3). Staged introduction of a robotic or autonomous system is likely to involve data collection and monitoring, to check the system is operating correctly and safely, but this should not become surveillance of the operators, users, or any responsible person.

The principles below are organised into three sections: Design, Sect. 5.1; Verification & Validation, Sect. 5.2; and Operation, Sect. 5.3. The principles are numbered for identification not to indicate ordering.

5.1 **Design**

1. The guiding principles of an autonomous system's design should be that it improves the well-being and safety of front-line workers, by complementing their skills; and be demonstrably trustworthy and adequately safe in the eyes of those workers. This will involve ongoing consultation with front-line workers about which task(s) would most usefully (for them) be delegated to an autonomous system.

   This is in addition to ensuring that the system's required level of safety can be demonstrated to the regulators.

2. If an autonomous system is being introduced to control a robotic system, then it should (initially) make use of existing physical barriers to contain the robot, to provide a final line of defence if the software and hardware safety systems fail.

3. The introduction and implementation of autonomy, or autonomous components, should focus on the system's transparency and amenability to strong verification (for example, Formal Methods, formally specified runtime monitors, etc). Opaque statistical AI techniques (e.g. data-driven machine learning) should only be used where they are necessary.

4. An autonomous system should maintain the remote-control option as a backup, so that the system can be operated or made safe if the autonomous software fails. This implies that the remote-control link is part of the system and should also be demonstrated to be adequately safe.

5. Clearly identify and fully define the task(s) that the autonomous system will perform. This should include, for example: its context and inputs (and any other preconditions), how it performs the task, the expected outputs (and any other postconditions), what might cause the task to fail and how the system should recover, etc. The intent of these task definitions is to make it clear to both the development team and the end users what the system can and cannot do.

6. If the autonomous system is being delegated a task from a human, then the task design process should begin with a clear analysis of how the human operator currently does the job safely. Safety enhancements, that are enabled by an autonomous robotic system with higher capabilities than a human operator, can be included after the system copying the human operator can be shown to be acceptably safe.

   This has several benefits. First, it involves current front-line workers in the design process, which improves the likelihood that actual safe practice is captured (and not just the documented safe practice). Second, it provides the system's requirements, which are essential for meaningful Verification & Validation (see Section 5.2). Third, the system's actions are more understandable, which can improve front-line worker's trust and the ability to identify failures in performing the task.

7. For each task, design high-level pre-and post-condition (or assume-guarantee) properties to describe what things the task assumes to be true before it starts and after it executes **and** a low-level procedural description of the task (e.g. a flowchart or state machine).

   The design should consider the potential for a task to fail during its operation, and define a fallback task (or tasks) to safely recover from the failure or transition to a failsafe state (for example, see Viard, Ciarletta, and Moreau 2019).

8. When designing an autonomous system to take-over tasks with a high importance to safety[6], a decision support system can be used as a prototype for the full system, to validate the task model and decision processes. This could also improve user trust in the autonomous system's decisions, in a way that does not give the software physical control of the task(s).

9. If an unpredictable component (such as a machine learning or evolutionary systems) is used, ensure that the goal of the system and its ethical and safety properties (things it must avoid doing or choosing) are well defined beforehand (see 5.3 (2)).

6. See Section 2.4 for a discussion of this concept.

10. If an AI Planning system is used to plan the movement (or other actions) from a user supplied world model, then experts in the operational context should be involved in the design of this world model because the correctness of the resulting plans depends upon the accuracy of the world model. Formal techniques can be used to analyse the model for consistency and detect ambiguities.

5.2 **Verification & Validation**

1. For each task, verify that its implementation respects both the high-level (pre- and post-condition) and low-level design (see 5.1 (7)).

2. users and operators should only be in physical proximity to the robot after it has been demonstrated that the robot cannot harm them under any circumstance, including failures.

3. If statistical techniques (such as deep neural networks) are used, then they should be analysed, to the extent possible, to identify worst-case mis-classification ratio and their robustness to small changes in input. For example, it should be demonstrated that if a human observer (or other oracle) would classify two inputs (that the neural network has been trained on) as the same (or the same type), then they are also classified as the same (or the same type) by the neural network.

4. Online learning: that is systems that will continue to use observations of the outcomes of their actions within the environment to modify their operation *after deployment* should be avoided unless absolutely necessary.

5. If feedback controllers are used, then it should be demonstrated that (1) the model of the system (which is likely to be a collection of differential equations) is validated against the real world, and that it is based on valid data; (2) that any abstractions in the implementation of the feedback controller are valid, and; (3) that the feedback controller is robust against variations in the systems performance, and can cope with any differences between its model and the real system.

5.3 **Operation**

1. The system should monitor its hardware for signs of degradation, this could include excess noise or vibration etc. This monitoring enables prediction of when components need replacing and could feed into the system's planning software. This is especially important where an autonomous system will work more intensively than a human, as degradation rates are likely to increase and may be less easy to predict.

   It needs to be provable that the data collection and monitoring does not become surveillance of any human interacting with the system.

2. If unpredictable software components (such as machine learning, or evolutionary systems) are used (see 5.1 (9)), then verifiably correct (for example, by using Formal Methods) runtime monitors should be used to enforce the goal, ethical, and safety properties that have been defined in the design. This ensures that the system remains in an acceptable behavioural envelope. This envelope should be transparent and understood by human operators, users, and the responsible person. This enables human detection of deviations from the behavioural envelope at runtime. (See (Mallozzi, Pelliccione, and Menghi 2018) for an example of this approach.)

3. Autonomous systems often use internal models of the real world to plan behaviour and interact with their environment, such as a pre-built map of the system's environment for navigation and planning. These models are abstractions of the real world, but if the abstractions are wrong or the real world changes, then the model becomes less correct. The system should monitor its environment to ensure that its internal models remain correct. If a change is detected, then the system could attempt to update its model or alert an operator or responsible person if it is unable to.

Authors, Glossary and References

Authors

- **Matt Luckcuck**, Department of Computer Science, Maynooth University, Ireland[7]
- **Michael Fisher**, Autonomy & Verification Group, The University of Manchester, UK[8]
- **Louise Dennis**, Autonomy & Verification Group, The University of Manchester, UK[9]
- **Steve Frost**, Office for Nuclear Regulation, Bootle, UK
- **Andy White**, Office for Nuclear Regulation, Bootle, UK
- **Doug Styles**, Office for Nuclear Regulation, Bootle, UK

Our thanks go to Vince Page, and Xiaowei Huang for contributing their expert advice; and to our early reviewers: Xingyu Zhao, Başak Saraç-Lesavre, and Nick Hawes for their invaluable discussion and comments.

Glossary

ASIL Automotive Safety and Integrity Level.

Assurance Justified confidence in a property.

Criticality The level of assurance against failure needed for a system component.

DAL Design Assurance Levels or Development Assurance Level.

Deployment Environment The environment in which the robotic or autonomous system has been designed to work.

Ethical Risk The probability of ethical harm occurring from the frequency and severity of exposure of a hazard. See BS 8611:2016 (British Standards Institution. 2016).

Ethical Hazard A potential source of ethical harm. See BS 8611:2016 (British Standards Institution. 2016).

Ethical Harm Anything likely to compromise psychological and/or societal and environmental well-being. See BS 8611:2016 (British Standards Institution. 2016).

Formal Methods Mathematical approaches to software and system development that support the rigorous specification, design, and verification of computer systems.

Fully-Autonomous Robotic System A system where the robot performs a task "without human intervention, while adapting to operations and environmental conditions" (Standing Committee for Standards Activities of the IEEE Robotics and Automation Society. 2015).

MCS Mixed-Criticality System.

Mission-Critical A mission-critical system is one where a failure may lead to large data- and financial-losses. Examples include deep-sea submersibles, automatic exploration vehicles (for example, the Mars rovers), and other scientific monitoring systems.

7. The majority of Matt Luckcuck's work on this white paper was done while he was employed by the Universities of Liverpool and Manchester. https://orcid.org/0000-0002-6444-9312
8. Michael Fisher https://web.cs.manchester.ac.uk/~michael
9. Louise Dennis: https://personalpages.manchester.ac.uk/staff/louise.dennis

Operator A human providing a robotic system with instructions or direction.

Predictive Runtime Verification An extension of Runtime Verification, where a formal specification of the system is used to try to predict satisfaction or violation of a property.

Responsible Person A human who has the legal responsibility for a robotic system, see (Principles of robotics – EPSRC website, Rule 5).

Runtime Verification An approach where the system is monitored, and its behaviour is compared to a formal specification. Action can be taken if the behaviour differs from the specification.

Safety-Critical A safety-critical system is one where a failure may lead to ecological or financial disaster, serious injury, or death. Examples include medical equipment, cars, aeroplanes, and power plants.

Semi-Autonomous Robotic System A system where the "robot and a human operator plan and conduct the task, requiring various levels of human interaction" (Standing Committee for Standards Activities of the IEEE Robotics and Automation Society. 2015).

SIL Safety Integrity Level.

User A human interacting with a robotic system, who may or may not be an employee.

V&V Verification and Validation.

References

Beer, Jenay M, Arthur D Fisk, and Wendy A Rogers. 2014. "Toward a Framework for Levels of Robot Autonomy in Human-Robot Interaction." *Journal of Human-Robot Interaction* 3 (2): 74. https://doi.org/10.5898/JHRI.3.2.Beer

British Standards Institution. 2016. *BS 8611:2016 - Robots and robotic devices. Guide to the ethical design and application of robots and robotic systems*. British Standards Institution. http://shop.bsigroup.com/ProductDetail?pid=000000000030320089

Burns, Alan, and Robert Davis. 2018. "Mixed Criticality Systems - A Review." August 2018. York: University of York. https://www-users.cs.york.ac.uk/ burns/review.pdf

Endsley, Mica R. 1999. "Level of Automation Effects on Performance, Situation Awareness and Workload in a Dynamic Control Task." *Ergonomics* 42 (3): 462–92. https://doi.org/10.1080/001401399185595

High-Level Expert Group on AI (AI HLEG). 2019. "Policy and investment recommendations for trustworthy Artificial Intelligence." European Commission. https://ec.europa.eu/digital-single-market/en/news/policy-and-investment-recommendations-trustworthy-artificial-intelligence

Huang, Hui-Min, Kerry Pavek, James Albus, and Elena Messina. 2005. "Autonomy Levels for Unmanned Systems (ALFUS) Framework: An Update." In, edited by Grant R. Gerhart, Charles M. Shoemaker, and Douglas W. Gage, 439. Orlando, Florida, USA. https://doi.org/10.1117/12.603725

Huang, Xiaowei, Daniel Kroening, Wenjie Ruan, James Sharp, Youcheng Sun, Emese Thamo, Min Wu, and Xinping Yi. 2020. "A Survey of Safety and Trustworthiness of Deep Neural Networks: Verification, Testing, Adversarial Attack and Defence, and Interpretability." Computer Science Review 37 (August): 100270. https://doi.org/10.1016/j.cosrev.2020.100270

Jobin, Anna, Marcello Ienca, and Effy Vayena. 2019. "The global landscape of AI ethics guidelines." *Nature Machine Intelligence* 1 (9): 389–99. https://doi.org/10.1038/s42256-019-0088-2

Luckcuck, Matt, Marie Farrell, Louise A Dennis, Clare Dixon, and Michael Fisher. 2019. "Formal Specification and Verification of Autonomous Robotic Systems: A Survey." *ACM Computing Surveys* 52 (5): 1–41. https://doi.org/10.1145/3342355

Mallozzi, Piergiuseppe, Patrizio Pelliccione, and Claudio Menghi. 2018. "Keeping intelligence under control." *2018 IEEE/ACM 1st International Workshop on Software Engineering for Cognitive Services* (SE4COG), 37–40. https://doi.org/10.1145/3195555.3195558

On-Road Automated Driving (ORAD) committee, SAE International. 2018. "Taxonomy and Definitions for Terms Related to Driving Automation Systems for on-Road Motor Vehicles." J3016_201806. SAE International. https://doi.org/10.4271/J3016_201806

Ryan Proud, Jeremy Hart, and Richard Mrozinski. 2003. "Methods for Determining the Level of Autonomy to Design into a Human Spaceflight Vehicle: A Function Specific Approach." In, 15. Gaithersburg, MD; United States. http://handle.dtic.mil/100.2/ADA515467

Sheridan, Thomas B, and William L Verplank. 1978. "Human and Computer Control of Undersea Teleoperators." Massachusetts Inst of Tech Cambridge Man-Machine Systems Lab. https://apps.dtic.mil/sti/citations/ADA057655

Sotiropoulos, Thierry, Hélène Waeselynck, Jérémie Guiochet, and Félix Ingrand. 2017. "Can robot navigation bugs be found in simulation? An exploratory study." *In Softw. Qual. Reliab. Secur.*, 150–59. IEEE. https://doi.org/10.1109/QRS.2017.25

Standing Committee for Standards Activities of the IEEE Robotics and Automation Society. 2015. "IEEE Standard Ontologies for Robotics and Automation." IEEE. https://standards.ieee.org/standard/1872-2015.html

Viard, Louis, Laurent Ciarletta, and Pierre-Etienne Moreau. 2019. "Monitor-centric mission definition with sophrosyne." *In 2019 Int. Conf. Unmanned Aircr. Syst. ICUAS 2019*, 111–19. Atlanta, United States. https://doi.org/10.1109/ICUAS.2019.8797898

Webster, Matt, David Western, Dejanira Araiza-Illan, Clare Dixon, Kerstin Eder, Michael Fisher, and Anthony G Pipe. 2020. "A Corroborative Approach to Verification and Validation of Human–Robot Teams." *The International Journal of Robotics Research* 39 (1): 73–99. https://doi.org/10.1177/0278364919883338

Woodcock, Jim, Peter Gorm Larsen, Juan Bicarregui, and John Fitzgerald. 2009. "Formal Methods: Practice and Experience." *ACM Computing Surveys* 41 (4): 1–36. https://doi.org/10.1145/1592434.1592436

The RAIN Hub: a radical change to existing nuclear programmes and initiatives.

The RAIN Hub's objectives are to overcome the challenges facing the nuclear industry. Through applying our scientific knowledge and understanding we can improve safety and efficiency in the nuclear sector, benefiting the UK economy and creating a safer working environment.

Department of Electrical and Electronic Engineering
The University of Manchester
Oxford Road
Manchester
M13 9PL
United Kingdom

Tel: (+44) 0161 306 2622
Email: info@rainhub.org.uk

www.rainhub.org.uk

@RAIN_Hub

3625.06.21