2018 IEEE International Symposium on Software Reliability Engineering Workshops

Verifiable Self-Certifying Autonomous Systems

Michael Fisher, Emily C. Collins, Louise A. Dennis, Matt Luckcuck, Matt Webster
Computer Science Department, University of Liverpool, UK contact email: M.Webster@liverpool.ac.uk

Mike Jump, Vincent Page, Charles Patchett,
Engineering School, University of Liverpool, UK

Fateme Dinmohammadi, David Flynn, Valentin Robu, Xingyu Zhao
Engineering School, Heriot-Watt University, UK

***Abstract*—Autonomous systems are increasingly being used in safety- and mission-critical domains, including aviation, manufacturing, healthcare and the automotive industry. Systems for such domains are often verified with respect to essential requirements set by a regulator, as part of a process called certification. In principle, autonomous systems can be deployed if they can be certified for use. However, certification is especially challenging as the condition of both the system and its environment will surely change, limiting the effective use of the system. In this paper we discuss the technological and regulatory background for such systems, and introduce an architectural framework that supports verifiably-correct dynamic *self-certification* by the system, potentially allowing deployed systems to operate more safely and effectively.**

***Index Terms*—autonomy, verification, certification, software**

I. Introduction

Autonomous systems can be broadly defined as computer systems that decide for themselves what to do [51]. Such systems are increasingly being used, and proposed for use, in safety- and mission-critical domains, including aviation [45], automotive [22], manufacturing [29] and healthcare [9], [34]. Systems for such domains are often verified with respect to essential requirements set by a regulator, as part of a process called certification [15], [22]. Although routes to the certification of autonomous systems are being developed [13], [15], [22], [27], [28], this only covers part of the problem. Once we have certified our robot, driverless car, unmanned aircraft, etc., then we can deploy it [29]. However, we know that a key aspect of both these autonomous systems, and the environments in which they are deployed, is that the system/environment will undoubtedly change beyond the conditions at the time of certification. Consequently, assumptions and assessments made at the time of deployment will likely now be different.

So, elements of both the system itself and its assumptions about the environment will certainly change. As this is outside the behavioural envelope anticipated when certifying the system, what do we do? In some cases it might be necessary recall the system and re-certify for the new situation, but it might also be that the changes do not affect the safety/reliability of the system and so it could, if allowed, continue working.

But how should the system decide whether to continue or stop (because re-certification is needed)? It is here that an autonomous system can carry out *self-certification* in order to decide between these options. In using the term 'self-certification' we are neither referring to system security in self-certifying file systems nor to physical health in human self-assessment. However, our use of the term is much closer to the second of these.

Once we have some autonomous system, such as a vehicle or a robot, that has been *certified* for use then we have some human-controlled process/documentation (such as assurance arguments supported by *safety cases*) capturing *why* the system has been certified. Consequently, once the certification process has been completed (and approved by the regulator) we have some idea of the limits within which the system can reliably (and safely) operate. Thus, our use of 'self-certification' captures

> *techniques and procedures to assess whether the system remains within the bounds under which it was certified.*

Note that, at this stage, we are not considering how to move the system back in to a 'certified' mode if something has failed [16] but just to ensure we can assess and detect when the system either leaves, or is in danger of leaving, certified bounds. While we do not expect the system to be able to re-certify itself, it should be able to detect when it is in danger of leaving certified bounds and undertake some remedial activity to ensure that the problems do not get any worse.

Clearly, if a system is to be able to undertake this form of self-certification, it needs at least an up-to-date description of its own workings and a precise description of what constitutes the certified (or safe) boundaries of system behaviour. As we will see in this paper, these two aspects are provided by a 'self model' and a 'safety model', respectively. We here describe a software architecture for autonomous systems that supports *self-certification* whereby the system can decide whether it should/can keep going.

II. Background

A. Autonomous Systems

We define **Autonomy** here to be the ability of a system to

> *make its own decisions and to act on its own, and*
> *to do both without direct human intervention.*

However, even within this, there are many variations concerning where (and how) decisions are made and actions are invoked. We can identify some of these variations as follows.

Authors acknowledge funding for this work in the UK by EPSRC through the ORCA [EP/R026173] Robotics and Artificial Intelligence Hub.

978-1-5386-9443-5/18/$31.00 ©2018 IEEE
DOI 10.1109/ISSREW.2018.00028

1) Automatic: involves a number of fixed, and prescribed, activities; there may be options, but these are generally fixed in advance and follow a rigid cycle.

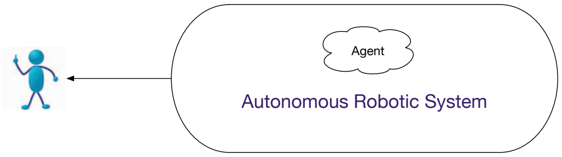
2) Adaptive: improves system performance/activity based on feedback from the environment — typically developed using tight continuous control and optimisation, e.g. a feedback control system. Usually very efficient, but driven fundamentally by environmental interactions.

3) Autonomous: decisions are made based on system's (view of its) current situation at the time of decision — the environment still taken into account, but internal motivations/beliefs are crucial. This moves much closer to human-level decisions as, based on its current views about the environment and itself, the system here decides whether to continue with adaptive processes, or to step away from the environment (perhaps because its sensors are deemed inaccurate) and take a different decision.

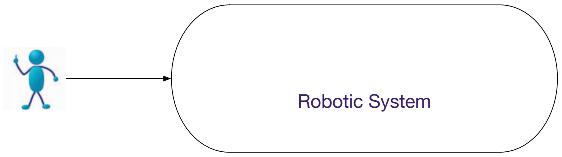
A simplistic example highlighting the differences might be as follows. If we keep seeing the same situation time and time again, an adaptive system will make the same decisions every time, perhaps just learning to recognise the situation much more efficiently. An autonomous system, however, could choose to take a different, divergent, decision after having seen the same situation a number of times.

B. Semi-Autonomous Systems

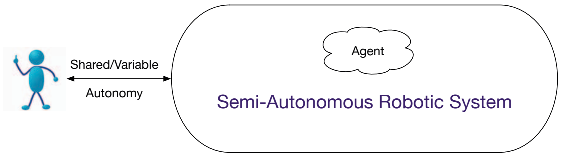
Within a truly *autonomous* system we can envisage a distinct software component, let us call it an 'agent' [51], that makes all the high-level decisions that a human operator/driver/pilot used to make:



(In this diagram and the two that follow, arrows $x \rightarrow y$ indicate that x is in overall control.) At the other extreme we have a remote-controlled system wherein a human operator/pilot/driver makes all the *key* decisions:



In practice, however, autonomous systems rarely fall clearly into these extremes. It is much more common for the human and the system to jointly be responsible for system behaviours. This leads us on to more nuanced and sophisticated issues concerning *shared autonomy* and, if these responsibilities can change over time, then further to issues of *variable autonomy*:



Not surprisingly there are many ways to categorise these, potentially dynamic, levels of autonomy. One, called 'PACT', is widely used in aerospace systems [7] and categorises levels of autonomy from 0 (no autonomy) to 5 (full autonomy):

0: 'No Autonomy'
→ *Whole task done by human except for actual operation*

1: 'Advice only if requested'
→ *Human asks system to suggest options and human selects*

2: 'Advice' → *System suggests options to human*

3: 'Advice, and if authorised, action'
→ *System suggests options and proposes one of them*

4: 'Action unless revoked'
- 4a: *System chooses action and performs it if human approves* ('consent')
- 4b: *System chooses action and performs it unless human disapproves* ('exception')

5: 'Full Autonomy'
- 5a: *System chooses action, performs it and informs human*
- 5b: *System does everything autonomously*

We are primarily concerned with the higher-levels of autonomy where there will likely be no human operator able to quickly and directly make decisions.

C. Verification and Validation

Increasingly, the key problem in autonomous systems is not just to construct an autonomous system, be it vehicle or robot, but to construct it in such a way that it is (verifiably) safe, reliable, and hence trustworthy.

1) Verification: The aim of *Verification* is to ensure that our system matches its requirements. These requirements may be informal, in which case it is hard to assess if, or how, our system does indeed correspond to them, or the requirements may be explicitly formal. The formal variety is often given in a clear, precise language with unambiguous semantics. *Formal Verification* takes this further, not only having precise formal requirements in a mathematical form, but carrying out a comprehensive mathematical analysis of the system to 'prove' whether it corresponds to the formal specification of these requirements [15]. Formal verification is particularly used for systems that are safety, business, or mission critical, and where errors can have severe consequences.

There are many varieties of formal verification, the most popular being *model checking* [4], [14], whereby the formal specification is checked (usually automatically) against *all* possible executions of the system. Verification, via model checking, is widely used especially for the analysis of critical systems. However, its use in autonomous software is relatively recent [8], [42], while application to the verification of practical autonomous systems is still at a very early stage [25]. Though these approaches are typically used *before* deployment, related techniques provide the basis for run-time monitoring and compliance testing. Such run-time verification [23], [43] is important in assessing how complex systems evolve, and ensuring that unacceptable behaviours are detected and mitigated.

2) Validation: Validation is the process of confirming that the final system has the intended behaviour once it is active in its target environment, and is often concerned with satisfying external stake-holders. For example, does our system match safety standards or legal rules set by regulators [10], [41]? Does our system perform acceptably from a customer point of view, and how well do users feel that it works [34]? The are many approaches to carrying out validation, incorporating diverse aspects, but typically involving the assessment of accuracy, repeatability, usability, resilience, etc.

In our context, *Verification and Validation* (V&V) necessitates a range of techniques [13], from *formal safety verification*, through *testing*, to in-situ *evaluation* and *monitoring*, and it is often difficult to delineate these phases. For example, since autonomous systems typically interact with the 'real world', we must ensure that verification is extended to this interaction. Yet it is impossible to accurately model the real-world, with its uncertain and continuous dynamics, in a finite way and so exploration of *all* possibilities via techniques such as model-checking is infeasible [21]. This leads to several options. We can try to *abstract* from the complexity of the real world and provide a finite description of this abstraction that we can then use in formal verification; this abstraction is likely to be incorrect in some way and will need refinement [1]. A practical alternative is to use sophisticated *testing* methods, appealing to Monte-Carlo techniques and dynamic test refinement in order to systematically 'cover' a wide range of practical situations [3], [27], [28]. Such *requirements-based* testing is regularly used in systems design. A further option is to *monitor* the system as it runs, detecting if it ever performs unacceptable behaviours. A key issue in V&V is maintaining consistency between these various models [49], which remains an open question.

D. Certification

We begin with ***Regulations***:

> *rules, policies and laws set out by some acknowledged authority to ensure the safe design and operation of systems.*

Once we have these, then ***Certification*** can be defined as

> *the determination by an independent body that checks whether the systems are in conformity or compliant with the above regulations.*

It is important to note that certification represents a legal, rather than scientific, assessment and usually involves *external review*, typically by some *Regulator*. Certification processes, and regulators, usually (though not exclusively) appeal to ***Standards***:

> *documents (usually produced by a panel of experts) providing guidance on the proving of compliance.*

There are very many standards for different types of systems, for example

- *ISO 61508: General safety standard*
- *ISO 10218: Safety requirements for industrial robots*
- *ISO 15066: Collaborative robots*
- *ISO 26262: Automotive*
- *RTCA DO-178B/C: Aerospace/UAVs . . . etc. . .*

However standards, and regulators, generally ignore the issue of *autonomy*. There are a few exceptions with some standards addressing autonomy, for example [11]

> *BS 8611 Guide to the Ethical Design and Application of Robots and Robotic Systems*

and new standards, such as the IEEE *ethically aligned design* standards (P7000, P7001, P7002, etc.), are being developed in this direction.

1) Problems with Autonomy: It is important to note that, when a system is certified, it does **not** guarantee it is safe — it just guarantees that, legally, it can be considered 'safe enough' and that the risk in deployment is acceptable.

A general problem is that current certification approaches often assume that

- there is a finite set of hazards/failures that the system will encounter,
- these can be identified beforehand,
- this finite set will not change over the life of the system,
- . . . and so a risk/mitigation based approach can be used,

many of which may not be true for complex autonomous systems. For example, [38] highlights the problems with certified designs:

- *The full consequences may be dependent on the situation the system finds itself in. These situations may be close to infinite and the consequent safety case analysis is invariably in-exhaustive.*
- *If the system is being applied in a manner that is unanticipated by the designer, the safety analysis is likely to break down completely.*
- *For safety critical systems, the probability of occurrence may have to be very low — for catastrophic failures typically less than 10^{-9} per flight hour in aviation. Probability of occurrence may be difficult to quantify, particularly over the lifetime of the system. So, how does one prove a level of assurance in such a case?*

Crucially, standards/regulations have little to say about intelligent software making complex decisions about safety, and even ethics [2], [11], [18], [30], [37], [50] — yet this is a fundamental part of (semi) autonomous robotic systems. This leads us to the view that greater autonomy requires much stronger V&V techniques, particularly formal verification.

E. Broader Issues with Autonomy

Once the key decision-making process is taken away from humans, how can we be sure what autonomous systems will do? Do we know that they are safe? Can we trust them? These questions are currently impossible to answer. Yet, without these answers, increased autonomy will not be accepted by engineers, allowed by regulators, or trusted by the public. This is especially the case as robots, autonomous vehicles, etc., are increasingly being deployed in safety-critical situations. The key aspect is not just that the system makes decisions, but that it makes the ***right*** decisions and has good ***reasons*** for making them [6], [44]. Once this decision-making is explicit, then we can use V&V techniques to try to ensure that our autonomous system will indeed make the decisions that the stake-holders consider to be safe, legal, and ethical.

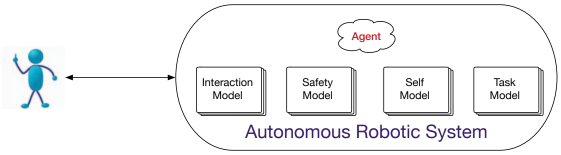


Fig. 1. Key Architectural Components within an Autonomous System

III. Architecture

Our approach is that these autonomous systems contain an **agent** for autonomous decision-making, together with a range of models capturing key aspects, see Fig. 1.

A. Agents

Identifying the core 'agent' in the system, wherein key decisions are made, is an important first step [25]. However, more complex issues concerning the dynamic nature of responsibility moving between human and machine (and back again) — ***variable autonomy*** — will later need to be addressed.

We are particularly keen to identify a **rational agent** [25], [52] as the focus of autonomous decision-making since this not only makes decisions and invokes actions but must be able to explain ***why*** it chooses one option over another. This not only supports *explainability* [53] and scrutability [12] but, by being able to formally verify these rational agents [19], we can prove that the correct, even safe and ethical [18], decisions will always be taken [17], [48].

B. The Role of Models

Within our system there are several models. Many of the agent's decisions will involve assessing these models, and many of the sub-processes will be driven by information from the models. These models can become quite complicated but, in their simplest form, contain elements described in Fig. 2. We now describe activities involving these models in more detail.

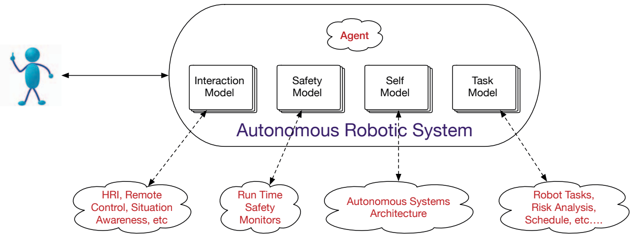

Agent
Interaction Model
Safety Model
Self Model
Task Model
Autonomous Robotic System
HRI, Remote Control, Situation Awareness, etc
Run Time Safety Monitors
Autonomous Systems Architecture
Robot Tasks, Risk Analysis, Schedule, etc....

Fig. 2. Key Processes Utilising Models within the Architecture

C. Interaction Model

The *Interaction Model* describes the form of, and requirements for, interaction with operators, humans, and other external entities (e.g., inspectors). This encompasses not only how to interact but what information to provide, for example in order to explain the robot's decisions or actions. There is a variety of mechanisms for this, ranging from a simple menu of straightforward interactions, through trust/expectation models, up to full mental models. So, our interaction model contains a model of Human-Robot Interaction (HRI) in order to explain decisions made by the system beyond the agent. A recognition of the impact that the human has on the system, which is not only person dependent, but crucially environment dependent.

Within this framework the human-interaction component can be understood as an agent–agent interaction paradigm. Where the autonomous system 'agent' is capable of making high-level decisions much as a human-agent is, and is in turn able to engage with the a human user of the system in order to adjust its decisions. The introduction of a human-agent into an autonomous agent loop poses many issues for the essential need for that loop to be verifiably safe, reliable and trustworthy.

A system does not work in isolation, and whether autonomous or semi-autonomous will be impacted by its human user. In the case of a semi-autonomous system the human decisions must be taken into account when considering what a system will do (e.g., non-human agents may be verifiable outside of machine learning algorithms, but human-agents are not). In autonomous systems it is important to consider the broader issues that go beyond a semi-autonomous system with a human in the loop, such as those addressed earlier. A fully autonomous system must be explicit with its decision making, and must be trusted.

D. Safety Model

The *Safety Model* is derived from the system certification process which, through techniques such as safety cases and fault identification, identifies safety aspects. For example, in aerospace, the certification process requires that a rigorous safety assessment is applied during the build process. There are several methods of safety analysis and these are documented in ARP 4761 [5]. This Safety Model describes issues which might affect the (certified) safety of the system. In particular, any safety case for the system will highlight the assumptions, expectations, and mitigations, relevant to ensuring acceptable system safety. We can extract these into a model that captures all these elements and that the autonomous system can interrogate to assess if a particular situation violates safety bounds.

The safety model in particular is required to cover how the system is operating, what are the safety requirements of the operational environment it is encountering and what responses is the system either proposing or conducting. To do this requires a considerable awareness of the operational environment (assuming it is complex) and an awareness of the consequences of its actions in how that affects the environment and other possible entities in that environment. Thus, it is required to understand when to take appropriate action if the environment (including other entities) has changed affecting safety, and to assess the impact of its actions regarding safety, both now and for the future to understand any impact of inaction.

E. *Self Model*

The *Self Model* describes the components, architecture, and effectiveness of the system itself. This will comprise the architecture for the system, together with a clear description (ideally formal) of the expected behaviour of the components within [17], [20]. This provides a strong form of self-awareness useful for health management and reconfigurability but, here, is especially useful for self-certification (in combination with the Safety Model). Here, prognostic techniques [32], [39] will be important in updating and maintaining this Self Model.

For example, a Self Model for a robotic system might include the central agent, robot arms, sensors, control systems, actuators, process tooling, power supplies, or planning systems. For each one of these subsystems there would be a formal description of the expected behaviour that the agent can use to monitor the various subsystems. For example, the agent might know that a particular sensor should monitor the environment and provide an update 20 times each second. If this rate drops to 10 times per second, for example, the agent could determine that this sensor has malfunctioned. The agent might then take the necessary steps to remedy this, including notifying the human operator, adjusting the mission parameters, disregarding the sensor's output or relying on other sensor systems in future. A related, but often harder, problem is when the updates are provided at the required rate but the data is incorrect (for example because the sensor has decalibrated). If the agent has a representation of the form of data expected within its Self Model, then it might again recognise the problem and undertake remedial action.

Principles from prognostics [20] will be essential for the agent in determining the ongoing health status of the various robot subsystems. For example, to successfully perform a task, the robot system needs to deliver the position and orientation accuracy of the robot's tools, the trajectory of the arm, the correct speed, force, and torque. Robot subsystems will degrade over time (due to wear-and-tear and use in harsh conditions [40]) and this can lead to a decrease in inspection quality and efficiency. One of the objectives of prognostics systems in robotic inspection is to predict the remaining useful life (RUL) of the robot system or its individual components as they degrade from an initial state to a failure state.

F. *Task Model*

The *Task Model* describes the goals, tasks, and schedules for the robotic system's activities. It is useful for planning, typically in combination with the Safety and Self models. For example, within the ORCA project[1] which is concerned with the autonomous inspection of offshore assets (oil rigs, wind turbines, pipelines, etc.) the Task Model will incorporate inspections that must be carried out, prognostic procedures that can be employed, etc., and will be used as the basis for planning the asset inspection tasks while maintaining the safety of the vehicle (and, indeed, asset). Typically, such planning/scheduling activities take into account the Task Model (what must be done), the Safety Model (what boundaries there are), and the Self Model (how capable the autonomous system is in carry out these tasks).

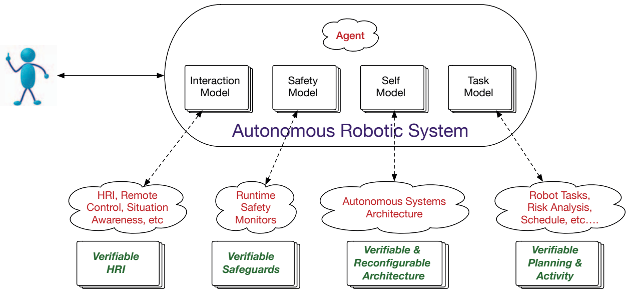

Agent
Interaction Model
Safety Model
Self Model
Task Model
Autonomous Robotic System
HRI, Remote Control, Situation Awareness, etc
Runtime Safety Monitors
Autonomous Systems Architecture
Robot Tasks, Risk Analysis, Schedule, etc....
Verifiable HRI
Verifiable Safeguards
Verifiable & Reconfigurable Architecture
Verifiable Planning & Activity

Fig. 3. Verifiability of Processes based on Internal Models

IV. Self-Certification

From a regulatory point of view, self-certification refers to the ability of a manufacturer to certify that their product conforms with regulations [47]. We extend this term naturally to cover autonomous systems, which can monitor themselves to ensure their continuing adherence to regulatory standards. Note that 'self-' here refers to both the autonomous system *and* the manufacturer, as such systems will effectively perform the act of self-certification on behalf of the manufacturer.

Within our architecture, we bring together

1) the Safety Model, providing a description of when safe behaviour might be threatened; and
2) the Self Model, providing a description of current system competencies.
3) the Task Model, providing a description of the tasks to be completed by the system.
4) the Interaction Model, providing a description of the way in which the system can interact with its user(s).

[1] https://orcahub.org

All of these are under the control of the 'agent' at the heart of the autonomous system. This provides mechanisms for self-certification as systems will have:

1) a clear view of current system capabilities/health;
2) a well-defined set of tasks for the system to achieve;
3) a set of requirements for the way the system must interact with the user;
4) a set of system monitors recognising if/when original certification may be violated; and
5) an agent able to assess (1) and (2) and decide on appropriate actions.

A. Example

Let us consider the case of a simple autonomous unmanned aircraft system (UAS). The UAS's *Interaction Model* demands that it should relay its current flight status to the user (via a ground control station) at least once per second, formulated as the following formula in a bounded linear temporal logic [46]:

$$\Box\Diamond^{\leq 1}\mathsf{send}(\mathsf{flightStatus}, \mathsf{user}) \tag{1}$$

This formula states that it is *always the case* ($\Box$) that *within one second* ($\Diamond^{\leq 1}$) the UAS should send its flight status to the user ('$\mathsf{send}(\mathsf{flightStatus}, \mathsf{user})$'). Within the *Self Model* there is a logical description of the abilities of each subsystem, including the antenna. For example, the Self model might contain the following logical formula:

$$\mathsf{working}(\mathsf{antenna}) \iff (\mathsf{send}(x, \mathsf{user}) \land \mathsf{receive}(y, \mathsf{user}))$$

This simply states that the antenna is working *if, and only if* ($\iff$) the system can send information to, *and* ($\land$) receive information from, the user. Suppose that the UAS's health monitoring systems indicate that the radio antenna used to transmit information has been damaged. This information is used to update the *Self Model* so that the rational agent in charge of the UAS becomes aware of the malfunction. The rational agent updates its belief base with the belief $\mathsf{B}\,\neg\mathsf{working}(\mathsf{antenna})$. A runtime monitor associated with the *Safety Model* then determines that $\neg\mathsf{working}(\mathsf{antenna})$ in fact implies that $\neg\mathsf{send}(x, \mathsf{user})$ which in turn means that requirement 1 can never hold and has therefore been violated. The runtime monitor responds by sending a message to the rational agent to inform it that there has been a safety-critical communication failure.

The rational agent can then assess the severity of the failure, in terms of safety violations. For example, it may be that the antenna's failure is actually only in terms of receiving information and that by only using the send functionality, the safety constraint can actually be satisfied. Alternatively the rational agent, through its *Self Model*, might be able to examine its other capabilities and so find another way to safely continue the mission by compensating for the malfunction. This might involve some software reconfiguration to rectify these failures [16]. In the worst case, for example in the case of an irreparable and unrecoverable failure, the only available course of action might be to conduct an emergency landing. In all these cases, the agent will use the *Interaction Model* to designate the levels and modes of interaction with human agents.

B. Generating Certification Monitors

A key aspect of this self-certification is having a set of runtime monitors that capture safety requirements and assumptions [36]. These are generated from the initial system certification and the constraints captured in the *Safety Model*. Our initial approach will be to generate such monitors 'by hand', since automatically extracting these from large, informal documents such as safety cases will likely be extremely complex. In extraction these monitors the Engineering challenge is to faithfully capture all the key safety constraints whilst ensuring that the monitors do not place a large computational burden on the system.

V. Verification

Verification, of various forms, is important for all the different types of model shown in Fig. 3.

A. Validating the Interaction Model

For the *Interaction Model* we cannot feasibly carry out any formal verification activities, but might carry out some testing and, more importantly, user validation activities. To validate the interaction model an understanding of the user and the specific environment in which the semi/autonomous system is to be deployed needs to be provided. Depending on the complexity and detail of the Interaction Model, a wide range of potential HCI/HRI validation techniques might be applied, e.g [9], [31].

B. Verifying Safety Monitors

A runtime safety monitor is a software component that consumes events from the system, compares them to the system's expected behaviour, and then takes some action if the events differ. The action can be to log the deviating event, flag the event to the user, or initiate mitigating behaviour. Monitoring the system's runtime behaviour can mitigate the problems of the *reality gap* — the difference between the behaviour of a simulated system and how it behaves when deployed on a real robot.

The monitored properties can be extracted from the *Safety Model* (which may include safety concerns from safety cases) and formalised. A runtime monitor for these properties can then be built that is amenable to formal verification [35]. This process provides confidence that the monitor checks for the correct properties and behaves as expected. Using verifiable monitors like this can often be easier to verify than the entire system, because they are simpler. Despite this, they can be effective in enforcing claims in a safety case, checking assumptions and context of safety cases or certification at runtime, and improving traceability of the system's safety requirements to the monitor. This approach can also be used for runtime 'health' monitoring of the system's physical components, which can then suggest or initiate mitigating action.

There are several challenges involved in using this approach. Firstly, it must be amenable to whatever certification processes are required by regulators. Secondly, the organisation of the Safety Model (or safety cases, etc.) must be clear enough so that the properties can be easily extracted. The key questions being: will there be enough detail, and will the information always be located in the same place? Finally, for automation of this approach, there are questions about the safety case or Safety Model that contains the properties that we want to model. Key questions here include: what notation, what (electronic) format, and what tools are being used?

C. Formal Description of Self Model

In order to assess a robot's *Self Model* we need to be able to formally describe its architecture. The specification of each software component can then be verified with respect to the system's requirements. When a component is changed, due to repairs or reconfiguration for example, the system can be reverified including the specification of the new component. A challenge here is linking the specification and verification of the system in the abstract with the contents of the components.

A related challenge is ensuring that the deployment of the system's architecture is correct. Robotic systems often use a middleware framework to interface with the hardware (for example, ROS or GenoM) but these are usually only assumed to be correct. These middlewares (including ROS) often use a node-based, publisher–subscriber model; which we can take advantage of to develop a high-level (meta) model that captures a wide range of middlewares. This can be used to verify the deployment of the robotic system onto a middleware framework. Further, specifically with ROS, this work can start to provide a formal description of how ROS works, independently of either of the two supported implementations.

D. Verifying the Task Model

For the *Task Model* we can formally verify the planning process such that any plan produced balances the risk/capabilities (from Safety and Self models) with its goals, such as asset inspection. This may be based on standard plan description languages, such as PDDL [26], and mechanisms for verifying (sometimes formally) planning processes [33].

E. Verifying the Whole System

In Section IV-A we showed how an autonomous system for a UAS might handle an antenna failure using runtime monitors and the Interaction, Self, Task and Safety models. In this particular case, our system identified the problem and identified a safe course of action. However, we may formally verify that the autonomous system will always handle subsystem failures safely by creating an environment model in which different subsystems may fail at any given time. This environment model stimulates a model of the autonomous system to explore various ways in which the system might fail. Combining these models with a model checker [24] allows us to analyse every state in this system to determine that the autonomous system will always maintain safe operations.

F. Simulation and Testing

As the system must inhabit the real-world, we need to extensively test it's behaviour, in all the above aspects, in more realistic environments. Real physical testing is often difficult and expensive, and therefore comprehensive simulation-based testing is vital and will be needed as evidence within certification processes [48].

VI. Concluding Remarks

We have proposed an architecture for autonomous systems comprising a core agent 'decision-maker' together with a range of models. Key processes invoked by the agent will utilise and update these models, assessing the safety and reliability of the system as it proceeds. The continuous updating of the models, for example through prognostic techniques, will allow for flexible deployment in complex and evolving environments. Furthermore, the insistence of transparency and, crucially, verifiability for all/most of these processes will provide strong evidence for the trustworthiness of the self-certification aspects.

Of course, as regulations concerning autonomous systems are currently under-developed, there still remains the question of exactly what constitutes a 'certified' autonomous system/vehicle. However, our framework is general enough that this should not adversely impact either the form of our models or the verifiability of the requirements. This remains separate work and we continue to engage with standards bodies and regulators to develop effective routes to certification for systems with this new aspect of 'autonomy'.

In future work we aim to develop and deploy such systems, particularly where robots/vehicles replace humans in hazardous environments. This is surely not completed work, but sets out a framework for the development of systems, the evolution of regulations, and the increased reliability of autonomous robotic systems.

References

[1] R. Alur, T. Dang, and F. Ivančić. Counterexample-Guided Predicate Abstraction of Hybrid Systems. *Theoretical Computer Science*, 354(2):250–271, 2006.

[2] M. Anderson and S. L. Anderson. *Machine Ethics*. Cambridge University Press, 2011.

[3] D. Araiza-Illan, D. Western, A. G. Pipe, and K. Eder. Coverage-Driven Verification — An Approach to Verify Code for Robots that Directly Interact with Humans. In *Proc. 11th International Haifa Verification Conference (HVC)*, volume 9434 of *LNCS*, pages 69–84. Springer, 2015.

[4] P. J. Armstrong, M. Goldsmith, G. Lowe, J. Ouaknine, H. Palikareva, A. W. Roscoe, and J. Worrell. Recent Developments in FDR. In *Proc. International Conference on Computer-Aided Verification(CAV)*, volume 7358 of *LNCS*, pages 699–704, 2012.

[5] ARP 4761, *Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment*.

[6] M. Boden *et al.* EPSRC Policy Statement: Principles of Robotics. http://www.epsrc.ac.uk/research/ourportfolio/themes/engineering/activities/principlesofrobotics, 2011.

[7] M. C. Bonner, R. M. Taylor, and C. A. Miller. Tasking Interface Manager: Affording Pilot Control of Adaptive Automation and Aiding. In *Contemporary Ergonomics 2000*, pages 70–74. CRC Press, 2004.

[8] R. H. Bordini, M. Fisher, W. Visser, and M. Wooldridge. Verifying Multi-Agent Programs by Model Checking. *J. Autonomous Agents and Multi-Agent Systems*, 12(2):239–256, 2006.

[9] C. Breazeal, K. Dautenhahn, and T. Kanda. Social Robotics. In *Springer Handbook of Robotics*, pages 1935–1972. 2016.

[10] British Standards Institution (BSI). BS EN ISO 10218-2: Robots and Robotic Devices — Safety Requirements for Industrial Robots. http://www.bsigroup.com, 2014.

[11] British Standards Institution (BSI). BS 8611 Robots and Robotic Devices — Guide to the ethical design and application. http://www.bsigroup.com, 2016.

[12] M. W. Caminada, R. Kutlak, N. Oren, and W. W. Vasconcelos. Scrutable Plan Enactment via Argumentation and Natural Language Generation. In *Proc. International Conference on Autonomous Agents and Multi-agent Systems*, pages 1625–1626. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[13] C. Cârlan, B. Gallina, S. Kacianka, and R. Breu. Arguing on software-level verification techniques appropriateness. In S. Tonetta, E. Schoitsch, and F. Bitsch, editors, *Computer Safety, Reliability, and Security*, pages 39–54, Cham, 2017. Springer International Publishing.

[14] E. M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

[15] D. Cofer and S. Miller. DO-333 certification case studies. In J. M. Badger and K. Y. Rozier, editors, *NASA Formal Methods*, pages 1–15, Cham, 2014. Springer International Publishing.

[16] L. A. Dennis, M. Fisher, J. M. Aitken, S. M. Veres, Y. Gao, A. Shaukat, and G. Burroughes. Reconfigurable Autonomy. *KI - Künstliche Intelligenz*, 28(3):199–207, 2014.

[17] L. A. Dennis, M. Fisher, N. K. Lincoln, A. Lisitsa, and S. M. Veres. Practical Verification of Decision-Making in Agent-Based Autonomous Systems. *Automated Software Engineering*, 23(3):305–359, 2016.

[18] L. A. Dennis, M. Fisher, M. Slavkovik, and M. P. Webster. Formal Verification of Ethical Choices in Autonomous Systems. *Robotics and Autonomous Systems*, 77:1–14, 2016.

[19] L. A. Dennis, M. Fisher, M. Webster, and R. H. Bordini. Model Checking Agent Programming Languages. *Automated Software Engineering*, 19(1):5–63, 2012.

[20] F. Dinmohammadi, M. Fisher, D. Flynn, M. Jump, V. Page, C. Patchett, V. Robu, W. Tang, and M. Webster. Certification of Safe and Trusted Robotic Inspection of Assets. In *Proceedings of the 2018 Prognostics and System Health Management Conference*, 2018.

[21] B. Edmonds and J. Bryson. The Insufficiency of Formal Design Methods: Necessity of an Experimental Approach for the Understanding and Control of Complex MAS. In *Proc. AAMAS*, pages 938–945, 2004.

[22] F. Falcini and G. Lami. Challenges in certification of autonomous driving systems. In *2017 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pages 286–293, Oct 2017.

[23] Y. Falcone, K. Havelund, and G. Reger. A Tutorial on Runtime Verification. In *Engineering Dependable Software Systems*, pages 141–175. IOS Press, 2013.

[24] M. Fisher. *An Introduction to Practical Formal Methods Using Temporal Logic*. John Wiley & Sons, 2011.

[25] M. Fisher, L. A. Dennis, and M. Webster. Verifying Autonomous Systems. *CACM*, 56(9):84–93, 2013.

[26] M. Fox and D. Long. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *J. Artif. Intell. Res.*, 20:61–124, 2003.

[27] B. Gallina and A. Andrews. Deriving verification-related means of compliance for a model-based testing process. In *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*, pages 1–6, Sept 2016.

[28] A. Gannous, A. Andrews, and B. Gallina. Bridging the gap between testing and safety certification. In *2018 IEEE Aerospace Conference*, pages 1–18, March 2018.

[29] J. Guiochet, M. Machin, and H. Waeselynck. Safety-critical advanced robots: A survey. *Robotics and Autonomous Systems*, 94:43–52, 2017.

[30] IEEE Ethically Aligned Design. https://ethicsinaction.ieee.org.

[31] B. E. John and D. E. Kieras. The GOMS Family of User Interface Analysis Techniques: Comparison and Contrast. *ACM Trans. Comput.-Hum. Interact.*, 3(4):320–351, 1996.

[32] K. Kapur and M. Pecht. *Reliability engineering*. John Wiley & Sons, 2014.

[33] B. Lacerda, D. Parker, and N. Hawes. Optimal Policy Generation for Partially Satisfiable Co-Safe LTL Specifications. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI*, pages 1587–1593. AAAI Press, 2015.

[34] H. Lehmann, D. S. Syrdal, K. Dautenhahn, G. Gelderblom, S. Bedaf, and F. Amirabdollahian. What Should a Robot do for you? Evaluating the Needs of the Elderly in the UK. In *Proc. 6th Int. Conf. on Advances in Computer-Human Interactions*, pages 83–88, 2013.

[35] M. Machin, F. Dufossé, J. Blanquart, J. Guiochet, D. Powell, and H. Waeselynck. Specifying Safety Monitors for Autonomous Systems Using Model-Checking. In *Proc. 33rd International Conference on Computer Safety, Reliability, and Security (SAFECOMP)*, volume 8666 of *Lecture Notes in Computer Science*, pages 262–277. Springer, 2014.

[36] M. Machin, J. Guiochet, H. Waeselynck, J. Blanquart, M. Roy, and L. Masson. SMOF: A Safety Monitoring Framework for Autonomous Systems. *IEEE Trans. Systems, Man, and Cybernetics: Systems*, 48(5):702–715, 2018.

[37] J. Moor. The Nature, Importance, and Difficulty of Machine Ethics. *IEEE Int. Sys.*, 21(4):18–21, 2006.

[38] C. H. Patchett. *Encyclopedia of Aerospace Engineering*, chapter Requirements: Levels of Safety, pages 1–10. Wiley, 2015.

[39] G. Qiao, C. Schlenoff, and B. Weiss. Quick Positional Health Assessment for Industrial Robot Prognostics and Health Management (PHM). In *Proc. IEEE International Conference on Robotics and Automation*, 2017.

[40] G. Qiao and B. Weiss. Accuracy degradation analysis for industrial robot systems. In *Proceedings of ASME International Manufacturing Science and Engineering Conference*, pages 1–9, 2017.

[41] Radio Technical Commission for Aeronautics (RTCA). DO-178B: Software Considerations in Airborne Systems and Equipment Certification, 1992.

[42] F. Raimondi and A. Lomuscio. Automatic Verification of Multi-agent Systems by Model Checking via Ordered Binary Decision Diagrams. *Journal of Applied Logic*, 5(2):235–251, 2007.

[43] G. Rosu and K. Havelund. Rewriting-Based Techniques for Runtime Verification. *Automated Software Engineering*, 12(2):151–197, 2005.

[44] Royal Academy of Engineering. Autonomous Systems: Social, Legal & Ethical Issues. http://www.raeng.org.uk/publications/reports/autonomous-systems-report, 2009.

[45] UK Civil Aviation Authority. *CAP 722 — Unmanned Aircraft System Operations in UK Airspace: Guidance*. The Stationery Office, 4th edition, 2010.

[46] C.-I. Vasile, D. Aksaray, and C. Belta. Time Window Temporal Logic. *Theoretical Computer Science*, 691:27 – 54, 2017.

[47] C. Visvikis. Evolution in electric vehicle safety legislation and global harmonisation activities. In *2013 World Electric Vehicle Symposium and Exhibition (EVS27)*, pages 1–9, Nov 2013.

[48] M. Webster, N. Cameron, M. Fisher, and M. Jump. Generating Certification Evidence for Autonomous Unmanned Aircraft Using Model Checking and Simulation. *Journal of Aerospace Information Systems*, 11(5):258–279, 2014.

[49] M. Webster, D. Western, D. Araiza-Illan, C. Dixon, K. Eder, M. Fisher, and A. G. Pipe. A Corroborative Approach to Verification and Validation of Human–Robot Teams. *ArXiv e-prints*, Aug. 2016. https://arxiv.org/abs/1608.07403.

[50] A. F. T. Winfield, C. Blum, and W. Liu. Towards and Ethical Robot: Internal Models, Consequences and Ethical Action Selection. In *Advances in Autonomous Robotics Systems*, volume 8717 of *LNCS*, pages 85–96. Springer, 2014.

[51] M. Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley and Sons, Ltd, 2002.

[52] M. Wooldridge and A. Rao, editors. *Foundations of Rational Agency*. Applied Logic Series. Kluwer Academic Publishers, 1999.

[53] Explainable AI. https://www.darpa.mil/program/explainable-artificial-intelligence.