Current Robotics Reports (2021) 2:273–283
https://doi.org/10.1007/s43154-021-00058-1

SPACE ROBOTICS (Y GAO, SECTION EDITOR)

A Review of Verification and Validation for Space Autonomous Systems

Rafael C. Cardoso[1] · Georgios Kourtis[1] · Louise A. Dennis[1] · Clare Dixon[1] · Marie Farrell[2] · Michael Fisher[1] · Matt Webster[3]

Accepted: 13 May 2021 / Published online: 18 June 2021

Abstract

Purpose of Review The deployment of hardware (e.g., robots, satellites, etc.) to space is a costly and complex endeavor. It is of extreme importance that on-board systems are verified and validated through a variety of verification and validation techniques, especially in the case of autonomous systems. In this paper, we discuss a number of approaches from the literature that are relevant or directly applied to the verification and validation of systems in space, with an emphasis on autonomy.

Recent Findings Despite advances in individual verification and validation techniques, there is still a lack of approaches that aim to combine different forms of verification in order to obtain system-wide verification of modular autonomous systems.

Summary This systematic review of the literature includes the current advances in the latest approaches using formal methods for static verification (model checking and theorem proving) and runtime verification, the progress achieved so far in the verification of machine learning, an overview of the landscape in software testing, and the importance of performing compositional verification in modular systems. In particular, we focus on reporting the use of these techniques for the verification and validation of systems in space with an emphasis on autonomy, as well as more general techniques (such as in the aeronautical domain) that have been shown to have potential value in the verification and validation of autonomous systems in space.

Keywords Verification and validation · Formal methods · Space autonomous systems

Introduction

The launch of new hardware to space is costly. Consequently, the on-board software is usually limited by the power of the hardware that has been deployed in the past. Recently, the cost of launching new hardware to space has been decreasing, allowing more powerful hardware to be deployed. The addition of autonomy to a system can increase non-deterministic behavior, which can lead to unforeseen failures that can cause the loss of the deployed hardware. Moreover, stakeholders often find full autonomy undesirable in many domains due to the lack of trustworthiness in such systems [1]. Therefore, such systems are required to be properly verified and validated before their deployment [2].

As we move from directly controlled robotics towards more autonomous versions, the additional issue of 'autonomy' comes in to play. An autonomous system can (and, in the case of space, often must) make its own decisions and take its own actions without direct, real-time human control. It will clearly be important to verify that the decisions (and actions) taken will be the correct ones in a given scenario. However, unless the environment within which the system has to operate is especially simple, then we will not be

This article is part of the Topical Collection on *Topical Collection on Space Robotics*

Work supported by grant EP/R026092 (FAIR-SPACE) through UKRI and EPSRC Hubs for "Robotics and AI Hubs in Extreme and Hazardous Environments".

✉ Rafael C. Cardoso
rafael.cardoso@manchester.ac.uk

[1] Department of Computer Science, The University of Manchester, Manchester, UK

[2] Department of Computer Science, Maynooth University, Maynooth, Ireland

[3] School of Computer Science and Mathematics, Liverpool John Moores University, Liverpool, UK

able to enumerate all possible decisions and fix the 'correct' answers beforehand. In space, this will likely be impractical, and so it must be recognized that an autonomous system will have to make a decision that we have not pre-scripted. We consider a *space autonomous system* as a vehicle/machine deployed in orbit (such as an unmanned spacecraft) or located on a planet (such as a planetary rover) that is capable of operating safely with little or no human input.

A recent literature review on space robotics presents the history of past missions and discusses the challenges that are to be expected for future missions [3]. In their review, the authors highlight Verification and Validation (V&V) as one of the technological challenges for space robotics, emphasizing the need for providing assurances and guarantees towards reliable missions. Their review does not provide any details about techniques or approaches for V&V of space robotics.

Verification and Validation (V&V)

Verification is the process of establishing whether the system built conforms to the *requirements* and specifications of the system. *Formal* verification comprises a class of techniques that use formal, usually *logic-based*, techniques to provide strong evidence (usually in the form of a mathematical proof) that the system behaves correctly. There are many other, less formal, varieties of verification with the most popular being *testing*. *Validation* is the process of assessing whether the system produced actually matches the intended use or application. Here, formal techniques are less common with testing being prevalent.

A recent survey [4] provides an overview of formal verification approaches for autonomous robotic systems. The authors discuss the challenges, formalisms, and state-of-the-art formal approaches for the specification and verification of autonomous robotics. The results found in this survey indicate that model checking is the most prominent formal method used in the literature for the verification of autonomous robotic systems. It provides a comprehensive collection of research, but it is not focused on any particular application domain.

We note that, often verification methods are specifically focused on proving safety properties such as *"the rover shall not collide with an obstacle"*. However, with the advancement of technology in the space domain, it is becoming apparent that security-related properties are also crucial. Recent work in this vein seeks to integrate security properties into the formal verification effort by proposing a detailed security-minded verification methodology [5].

In this paper, we provide a systematic review of techniques and approaches for the V&V of space autonomous systems, as well as techniques and approaches that are considered to have potential value to the aforementioned domain. Some papers were identified via our work in the FAIR-SPACE Hub[1] and our interaction with the space research community. The majority of the papers were found by analyzing recent publications at relevant conferences/workshops including the NASA Formal Methods symposium (NFM), workshop on Formal Methods for Autonomous Systems (FMAS), etc. Additionally, some papers were discovered through a search limited to the past 5 years on indexing websites including Google Scholar.

Most of the research reported in this paper is directly applied to space autonomous systems. Some of the research relates to applications in the aeronautical domain which we included here for the following reasons:

- The aeronautical domain is strongly researched at NASA;
- There are a lot of works published at NFM that are focused on aeronautics but highlight their potential use in space;
- The recent launch of the Mars Perseverance Rover (landed on the 18th of February, 2021) included the Mars Ingenuity Helicopter.

We start with the available formal methods and discuss how these have been applied to verify autonomous systems in space applications. Then, a brief account of the current attempts to verify machine learning techniques is given, followed by some recent approaches to validation and simulation-based testing. To conclude, we discuss the complex challenge of providing compositional verification in modular autonomous systems.

Model Checking

In model checking [6, 7] we are given a formal model M for a system and a property ψ, and we wish to check whether M satisfies ψ. For example, M might be a model for the subsystem of a car automatically assisting acceleration and ψ might be that there is never unintended acceleration as has been the case in the past resulting in a number of fatalities [8]. The property is often given in a temporal logic [9, Chap. 11] such as *Linear-time Temporal Logic* (LTL) or a variant of *Computation Tree Logic* (CTL).

Broadly speaking, methods for verification can be divided into two categories: proof-based methods and state-exploration methods. *Proof-based methods* involve representing a given model M as a temporal formula and then attempting to prove that a given property ψ is a logical consequence of that formula [10]. *State-exploration methods* involve performing an exhaustive search of all

[1] https://www.fairspacehub.org/

possible behaviors of a given model M with the goal of finding a counter-example, i.e., a behavior that does not satisfy the property ψ that we wish to check. This can be done with a model checker such as SPIN [11] or NuSMV [12, 13]. State-exploration methods are also available in a probabilistic setting, in which case one uses probabilistic temporal logics [14] and most prominently the PRISM model checker [15].

Some examples of verification using state-exploration methods are [16, 17]. In [16], an autonomous agent developed using an agent programming language is used for simulated satellite control. In this scenario, the agent must ensure that the satellite can acquire and maintain a low Earth orbit. This approach is later expanded in [18] to a more complex scenario of four spacecraft controlled by autonomous agents with the goal of exploring asteroids. Verification in such scenarios was done through a program model checker [19], a variation of model checking that exhaustively runs over the actual code instead of a model or abstraction of the system.

In [17], the authors present a Brahms model of an astronaut–rover teamwork scenario in which the astronaut and the rover work together to improve and maintain a newly established outpost on another planet. Brahms [20] is a framework for modeling human–robot teamwork. The authors describe how the model's functionality can be validated using the Brahms IDE software, and how the model can be formally verified by translating it into the PROMELA formalism. The resulting PROMELA model is then used as input to the SPIN model checker in order to verify several safety- and mission-critical properties related to the teamwork between the astronaut and the rover.

The work in [21••] proposes a model-based approach to the specification of the requirements of the system in order to avoid ambiguity that is commonly found when using natural language. This initial model of the requirements is consistently incremented, deriving formal properties, identifying properties that can be enforced by design, instantiating the architecture, and trying to enforce the verification of the remaining properties by applying the architecture to the design model. If some property could not be enforced in the model built with the requirements then model checking is applied, and if the property is still not satisfied then the model has to be refined or the requirements revised. Experiments were performed with a CubETH nanosatellite [22] and low Earth orbit observation satellite show that this incrementally-built model can reduce the number of testing required in later stages of the design.

Some examples of probabilistic verification are [23–25]. In [23], a realistic use case of routing in a Walker satellite constellation (i.e., in circular orbits and with the same period and inclination) in low Earth orbit is verified using probabilistic model checking. The approach consists of using distributed schedulers that can only make use of local data. The routing strategies that were obtained are shown to have a high probability for message delivery.

In [24], the authors provide a framework for the decomposition of a system into a hierarchy of functionalities. The framework enables the assignment of probabilities to elementary functionalities. The probabilities of complex functionalities are then obtained from the probabilities of their (simpler) constituent parts and certain temporal connectives that the authors introduce. The authors present an application of this framework to a (hypothetical) deorbitation scenario for the (retired) satellite ENVISAT [26].

In [25], probabilistic model checking is used to accommodate inherent non-determinism in NASA's Small Aircraft Transportation System (SATS) [27]. SATS is a system created to increase safety and throughput in commercial airports. The authors introduce a synchronous discrete-time Markov chain model for SATS, whose intended properties they verify using the PRISM model checker [15].

In [28], the NuSMV model checker is used to verify the attitude and orbit control system for embedded satellite software. A key focus is on the abstractions used to alleviate the state explosion problem.

The COMPASS (COrrectness, Modeling and Performance of AeroSpace Systems) 3.0 tool for ensuring correctness of system-level properties in the aerospace domain is described in [29]. The tool was evaluated in several case studies involving a satellite's design and modelling a CubETH nanosatellite [22] which highlighted that models in COMPASS can cover discrete, real-time, hybrid, and probabilistic aspects of the system. An industrial-sized case study relating to a satellite platform that uses an earlier version of the COMPASS tool is discussed in [30]. A single model is used to which different types of verification are applied (including model checking).

Theorem Proving

Verification using theorem proving involves representing a system in higher-order logic/type theory [31] and then attempting to (manually) prove that the system obeys certain properties of interest, commonly using a proof assistant such as Isabelle/HOL [32] or Coq [33]. Some examples of proof-based verification applications follow below.

In [34], the authors propose a correction for an unwanted behavior that can potentially present when two unmanned aircraft systems (UAS) fly in close proximity. The authors formalize and prove the correctness of their proposal in the Prototype Verification System (PVS) [35].

In [36], the authors discuss interactive theorem proving with higher-order logic and its application in examples from

NASA's unmanned aircraft systems. The motivation behind their work is that existing formalisms are not expressive enough to be used to specify complex requirements that can usually be found in cyber-physical systems. The main benefit of this approach is the parametric nature of the models, since they are generic they can be instantiated and reused. The authors acknowledge that the main drawback of formalisms such as higher-order logic is that it they are undecidable.

In [37], the authors present a formalization (and their methodology for that formalization) in the Isabelle/HOL proof assistant [32] of a security-relevant part of PikeOS. (PikeOS is a commercial, real-time operating system used for safety and security critical systems in the automotive and aerospace industries, e.g. [38, 39] describe its use in space avionics.)

In [40], the authors report on their use of the Dafny formal method for the verification of a grasping algorithm which is to be used for Active Debris Removal (ADR) in space. The authors work from high-level natural language requirements and an existing Python implementation to derive a Dafny model and its associated formal specifications.

Runtime Verification

At runtime, unexpected events can happen that can cause the violation of the safety and reliability properties of the system. *Runtime Verification* (RV) [41] consists of attaching monitors to a system in order to observe events at runtime and match the occurrence of these events to a specified formal property. The monitor then outputs whether the property is consistent or inconsistent with the observed events. If an inconsistency is detected, an appropriate (and separate) mechanism for failure handling can be triggered. The properties to be verified in RV are usually specified in formal logic such as LTL [42] or in more user-friendly *Domain Specific Languages* (DSL) [43].

The main advantages of RV [41] are: (a) it is a formal method, therefore some formal assurances can be provided depending on the proofs of the tool being used; (b) monitors can scale well with the complexity of the system, especially when compared to other formal methods; (c) it is able to cope with dynamic environments. The main disadvantage of RV is that it is not exhaustive, since it is observing events at runtime it can not explore the entire search space such as in traditional model checking techniques.

In [44], the authors use RV to complement the static (and exhaustive) verification that was performed offline by a model checker. They have shown that by using these two formal techniques simultaneously it is possible to provide stronger assurances about the execution of a system in a dynamic environment. The approach consists of first modeling the environment, which usually results in an incomplete model that is an abstraction of the real environment, and then using this abstraction to validate the model at runtime using RV.

Another complement to the use of RV is proposed in [45] where, instead of model checking, the authors use simulation, specifically discrete-event stochastic simulation. Their work aims to use verification artifacts from simulation to augment RV and artifacts from RV to improve simulation. We discuss the combination of different V&V techniques further and in more detail in a later section.

The verification of properties at runtime of a radio onboard a spacecraft has been reported in [46]. In this scenario, the radio can communicate over different channels in order to send telemetry data to the ground unit. An example of a property, which was based on realistic log files from the Mars Curiosity rover [47], is that once a command is dispatched, it will not be dispatched again before that command is completed. The main RV approach used in this experiment was an extension of LTL to improve its expressiveness by including rules, extra propositions that prefix the execution and are expressed as past time temporal formulae. The authors have shown that their extension is as expressive as Büchi automata, which makes it an interesting alternative for RV of past temporal properties.

A similar scenario can be found in [48], once again focusing on the RV of properties related to the communication between the Mars Curiosity rover, a spacecraft, and the ground station. In this case, the technique used was based on the monitoring and verification of timed properties (i.e., with time constraints) through the extension of binary decision diagrams (data structure for representing Boolean functions) for RV. Results show that as the size of the trace increases, the time required to verify timed properties increases much faster than when compared to verification of formulae without any time constraints. Even though there is such an increase in time, the verification still finished in a few minutes for traces of length up to one million events.

Robonaut2 [49], shown in Fig. 1, is a humanoid robot running ROS (Robot Operating System) [50] onboard the International Space Station (ISS) to aid humans in complex tasks. Even though the robot was formally verified, when deployed to such a complex environment as the ISS a fault emerged where the sensors in the rotational joints of the robot were indistinguishable from high-torque data. This resulted in the control system freezing due to safety reasons and awaiting instruction from ground-control at Houston. To solve this problem, the work done in [51•] suggests

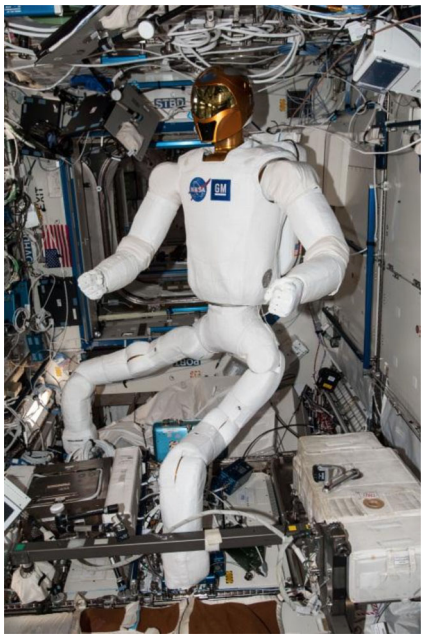

Fig. 1 The Robonaut2 robot (Source: NASA)

that the use of RV to disambiguate faults in Robonaut2. The challenge was finding a suitable tool that could be embedded in the robot, since the hardware had severe resource constraints. The authors presented new encodings for Mission-time LTL in order to cope with the resource limitation, and then implemented monitors using the new encodings in a Robonaut2 hardware on loan from NASA. The monitors were successfully able to disambiguate real-time faults in the system, even under limited resources.

ROSMonitoring [52] is an RV tool for ROS. The main difference between ROSMonitoring and past ROS-based approaches is that it does not change anything in the core of ROS, thus, it can easily be used across many different versions of ROS. It achieves this portability by instrumenting the nodes to be monitored so that they will publish their messages to a different topic, which will then be subscribed by the monitor. The monitor uses an external oracle (standard ones are provided but new oracles can be plugged in to support different formalisms) to verify the property based on the messages that are received. If the property is satisfied then the message is propagated to the original subscriber nodes, otherwise the monitor may choose to withhold the message. The tool is validated in a ROS simulation of the Mars Curiosity rover [53•] to verify that velocity commands being sent to the wheels of the rover would never surpass a safety threshold. Scalability experiments have also shown that minimal to no overhead is added by the presence of the monitors, as long as the frequency of messages being monitored does not exceed 5000 messages per second.

Software Testing

Software testing is an integral part of V&V of a system [54]. While formal verification can provide strong guarantees about the behavior of the system, there are many things that can still go wrong, such as properties that do not fully capture the requirements of the system, formal models that abstract away from the implementation too much and are no longer accurate representations, systems that are too complex to formally verify and consequently encounter state-space explosion problems, among others. Software testing can take many forms, such as incremental testing, simulation-based testing, physical testing, etc. Although software testing is not as exhaustive as formal verification techniques, it scales well and if correctly targeted it can be used to successfully test high-priority functionalities of the system against the most critical requirements.

The Curiosity rover [47], shown in Fig. 2, originally landed in Mars on August 6th, 2012. It is a complex rover with six wheels, a chassis with a suspension system, hardware that is radiation-hardened, many scientific instruments such as sample analysis and radiation detection, and 17 cameras. The high complexity of the system meant that it had to be verified and tested extensively and rigorously [55]. The Curiosity was verified using an incremental test program, starting from the verification of individual functions and building up to the validation of system capabilities in mission-like scenarios. The paper focuses on describing the latter, that is, system scenario tests. Due to the limited window of time for testing, the tests cases targeted high and medium priority objectives. An example of a test case used was the functionality of the rover in the first solar day of its execution. Tests included performing imaging, checking instrument liveness, the first ultra-high-frequency communication, and the first shutdown and restart of the system.

The AEGIS system [56] provides an autonomous target selection and data acquisition to augment the chemistry and camera instruments that are positioned in the mast of

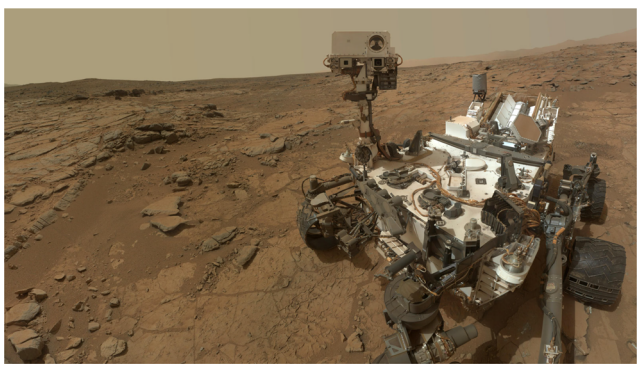

Fig. 2 The Mars Curiosity rover (Source: NASA JPL)

the Curiosity rover. It was uplinked and installed in the Curiosity rover in October 2015. The rover is capable of firing a powerful laser that is able to convert solid rocks into plasma, and therefore, the autonomous targeting system had to be properly validated to ensure that it would not affect the safety of the rover (e.g., inaccurate aiming). The system was tested thoroughly under simulation, and once deployed it was validated onboard the Curiosity before releasing it to be used by the science team.

Expressing the requirements of the system in a formal way is one of the first fundamental steps in V&V. One of the major challenges in using formal verification in real world applications is that formal notations often used to describe the system's requirements are not very intuitive. In [57], the authors use a tool (FRET) for the generation of formal requirements from a structured natural language (FRETISH). The approach was built using features from NASA's research and applications. The main goal of the language is to allow developers to intuitively express the requirements of the system in the early design stages. The tool currently supports the automatic generation of past-time, finite- and infinite-trace, and future-time metric temporal logic formulae. These formulae can then be formally verified for correctness. In particular, FRET requirements can be translated to CoCoSim contracts and then these properties can be model-checked using Kind2 [58].

Verification and Validation of Machine Learning

Machine learning (ML) [59] is one of the areas in artificial intelligence that has seen an exponential increase in research in recent years. Its main application in autonomous systems is in the use of ML for image classification [60], as evidenced by research made across several inter-disciplinary areas such as dentistry [61], climate [62], and education [63].

Despite the increase in the popularity of ML, its use in safety-critical scenarios remains limited due to the lack of formal guarantees about how ML approaches behave when using very large data sets for training. The literature review in [64] presents the state-of-the-art in V&V of ML applied to safety-critical systems and enumerates a set of challenges for providing assurances towards the use of ML in the automotive domain. The review concludes that the most pressing challenges for V&V of ML are state-explosion (combined with limited transparency and traceability) and unpredictable environments. Promising research directions are employing safety cages (monitoring of sensor input) and performing as much simulation-based testing as possible.

The survey in [65] discusses recent efforts in the verification and testing of techniques in ML. According to the survey, existing formal verification techniques for ML have been shown to work only in small-scale systems or by using approximation methods which provide an answer within a certain threshold. While mathematical approaches can, in some cases, be applied, a safety argument framework is required to address the partial knowledge and issues around the use of neural networks [66]. Clearly, the role for variations of testing is strong in machine learning, and a review of current and future activity in this area is given in [67].

In [68], the authors introduce their neural simplex architecture with the goal of providing safety guarantees for neural ML controllers. This novel architecture uses an adaptation module to retrain, at runtime, the controller in case of failure. Their technique is demonstrated in the scenario of rover navigation, where a rover has to move to a predetermined location while avoiding static obstacles. The retrained controller is shown to be safer than the controller that was generated with the initial training. Usually when an advanced controller (e.g., ML trained) fails, control is reverted to a baseline controller in order to recover from failure. The main advantage of the neural simplex architecture is that in case of failure the advance controller (ML trained) will be able to regain control after retraining is completed and the resulting controller should be safer than the previous one.

Two approaches [69, 70] attempted to verify deep neural networks in the ACAS Xu (Airborne Collision Avoidance System for unmanned aircraft) domain [71]. In the first [69], the DeepSafe technique is introduced to automatically infer safe regions of the input space where the network is robust against adversarial perturbations. In the second [70], the Marabou tool based on satisfiability modulo theories (SMT) that converts queries about a neural network's properties into constraint satisfaction problems. These deep neural network verification approaches can only verify simple properties about the input/output but not about *how* the network actually works.

Verification of Autonomous Behavior

In this section, we refer to autonomous behavior as symbolic autonomy, as opposed to sub-symbolic approaches such as machine learning from the previous section. Verifying systems with a bounded set of possible behaviors can be carried out through standard processes such as model-checking or testing. Once we get beyond this point we may have to verify the decision-making process to ensure that it will always 'try' to make the best decision [72]. As

highlighted in [4], there are relatively few viable verification approaches that consider the issue of increased autonomy.

Consequently, there are a wide range of international efforts aiming to address this. The IEEE Technical Committee on the *Verification of Autonomous Systems* brings together experts in this area, while IEEE standards such as *Failsafe Design of Autonomous Systems*[2] (standard P7009) are tackling the issue of increasingly autonomous systems. Academically, routes towards the reliable incorporation of autonomous components are beginning to appear [73] and mechanisms for specifying what we require of an autonomous system have appeared [74]. It is also important to note that there are different levels of autonomy, such as semi-autonomous or fully autonomous, and that autonomy can be implemented with either symbolic (e.g., cognitive agents) or sub-symbolic AI (e.g., neural networks) techniques. As we have shown in this paper, currently there are more examples of V&V in autonomous symbolic AI.

Compositional Verification

Although model checking can be applied successfully in the verification of many real-world systems, there are systems whose sheer size prohibits many of the techniques mentioned so far. Likewise, the complexity found in these systems makes it difficult to generate efficient test cases that can cover the whole system. Such systems often follow a modular design, that is, the system is decomposed into a number of individual components that encapsulate specific functionalities (e.g., sensors, actuators, etc.). V&V of such systems is usually done through the composition of the V&V of each component [75–78]. Thus, the properties of the original system can be obtained from the properties of its components and their composition. The difficulty in this approach lies in choosing the appropriate V&V technique for each individual component such that one has high confidence that the chosen technique correctly represents their assumed properties; and that from the properties of the components one can obtain useful properties for the whole system. Some examples of compositional verification follow below.

In [79], the authors present a modular framework for the verification of NASA's Quad-redundant Flight Control System (QFCS), a control system for NASA's Transport Class Model (TCM) aircraft. The framework enables the specification of components, together with assume/guarantee contracts for each component. Thus, the system of interest (QFCS) can be abstractly represented as the (synchronous) composition of various such components. This representation enables the authors to efficiently check whether the system satisfies certain requirements and detect possible errors.

Contract-based approaches [80] are an efficient way of performing compositional verification in modular systems. A contract for an individual component describes the assumptions (i.e., pre-conditions) that the component has and the guarantees (i.e., post-conditions) that it provides. These contracts can then be used to verify global properties of the system through compositional reasoning techniques. In [80], the CoCoSpec tool is introduced for specifying such contracts that can then be verified using mode-aware model checking. The effectiveness of their tool is demonstrated through its application to a flight-critical system case study. Monolithic analysis of the entire system was reported as not possible. Instead, a comparison between monolithic and compositional verification of the autopilot component determined that the monolithic approach was inconclusive after executing for 1 h; meanwhile the compositional finished in 80 s.

Recently, [81] analyzed a portion of the literature and identified common patterns appearing in robotic missions (e.g., patrolling, obstacle avoidance). They generated a catalogue with 22 mission specification patterns that are specified in structured English grammar and then translated into LTL or CTL formulae. These formulae are then verified for correctness using a model checker and used as input for a planner to generate plans that can satisfy the mission specification. While their approach is not capable of verifying global properties and does not provide any way of combining the verification of multiple patterns, these patterns do offer a practical way of specifying the requirements of modular systems.

The verification of a simulation of the Mars Curiosity rover is presented in [53•]. The simulation runs in Gazebo, a robot simulator with dynamics simulation and 3D graphics, using ROS. A rational agent autonomously performs high-level decision-making in the rover. Different components in the system have been verified with an appropriate V&V technique: the agent component is implemented using an agent programming language and verified with a program model checker; the communication between the ROS nodes containing low-level control of the wheels, mast, and arm of the Curiosity were specified in the Communicating Sequential Processes (CSP) process algebra and then verified using model checking; the environment interface between the agent and the ROS nodes has been verified with the Dafny program verifier, which uses automatic theorem proving to verify properties; finally, the properties verified in these different techniques were translated into runtime monitors and verified at runtime. This work presents a heterogeneous approach to verification but the integration of the results from each of the distinct techniques is left as future work which is explored in [78].

[2] https://www.ieee-ras.org/verification-of-autonomous-systems/

Table 1 V&V techniques grouped by type

Technique	References
Model checking	[16–18, 21••, 23–25, 28, 53•, 72, 78, 80, 81]
Theorem proving	[34, 36, 37, 40, 53•, 78]
Runtime verification	[44–46, 48, 51•, 53•, 68]
Software testing	[55–57, 81]
Other	[69, 70, 79]

Conclusions

This review paper presented recent advancements found in the literature towards verification and validation of autonomous systems with an emphasis in space applications. The literature contains a wide variety of different approaches and techniques that range from formal methods (model checking, theorem proving, runtime verification) to non-formal methods (simulation-based testing, physical testing). Table 1 includes the validation and verification techniques and approaches grouped by their type. Some entries belong to multiple groups due to their use of more than one V&V technique. As mentioned previously, the majority of the research is on model checking. Note that this table is limited to peer-reviewed research, and as such the low number of references in software testing are not an indication that it is not used or useful in the V&V of space autonomous systems but simply indicates the latest trends in state-of-the-art research.

The research on individual techniques for V&V has been progressing over the years and has been applied successfully to monolithic systems. However, complex space autonomous systems are often modular, comprised of multiple components that are specified at different abstraction levels. While some research has been done in compositional verification in order to try to bring together verification techniques and results from individual components into guarantees about the whole system, this remains an open and complex problem. Furthermore, an iterative V&V process using a corroborative approach [82] could further improve the reliability and trustworthiness in the system.

Declarations

Conflict of Interest The authors are all researchers on the FAIR-SPACE project together with Prof. Yang Gao, editor of the Space section of Current Robotic Reports. The authors also report grants from Industrial Strategy Challenge Fund (ISCF) (delivered by UKRI and managed by EPSRC) (Grant EP/R026092).

Human and Animal Rights and Informed Consent This article does not contain any studies with human or animal subjects performed by any of the authors.

References

Papers of particular interest, published recently, have been highlighted as:
- • Of importance
- •• Of major importance

1. Murphy RR. Trial by fire (rescue robots). IEEE Robot Autom Mag. 2004;11(3):50–61. https://doi.org/10.1109/MRA.2004.1337826.
2. Farrell M, Luckcuck M, Fisher M. Robotics and integrated formal methods: necessity meets opportunity. In: Integrated formal methods, LNCS. Springer; 2018. p. 161–71.
3. Gao Y, Chien S. Review on space robotics: toward top-level science through space exploration. Sci Robot. 2017;2(7). https://doi.org/10.1126/scirobotics.aan5074.
4. Luckcuck M, Farrell M, Dennis LA, Dixon C, Fisher M. Formal specification and verification of autonomous robotic systems: a survey. ACM Comput Surv (CSUR). 2019;52(5):100.
5. Maple C, Bradbury M, Yuan H, Farrell M, Dixon C, Fisher M, Atmaca UI. Security-minded verification of space systems. In: IEEE aerospace conference. IEEE; 2020. p. 1–13.
6. Clarke EM Jr, Grumberg O, Kroening D, Peled D, Veith H. Model checking. MIT Press; 2018.
7. Baier C, Katoen JP. Principles of model checking. MIT Press; 2008.
8. Kirchhoff SM. Unintended acceleration in passenger vehicles. DIANE Publishing; 2010.
9. Blackburn P, van Benthem JF, Wolter F. Handbook of modal logic. Elsevier; 2006.
10. Fisher M. An introduction to practical formal methods using temporal logic, vol. 82 Wiley Online Library; 2011.
11. Holzmann GJ. The model checker SPIN. IEEE Trans Softw Eng. 1997;23(5):279–95.
12. Cimatti A, Clarke E, Giunchiglia F, Roveri M. NuSMV: a new symbolic model verifier. In: International conference on computer aided verification. Springer; 1999. p. 495–9.
13. Cimatti A, Clarke E, Giunchiglia E, Giunchiglia F, Pistore M, Roveri M, Sebastiani R, Tacchella A. Nusmv 2: an opensource tool for symbolic model checking. In: International conference on computer aided verification. Springer; 2002. p. 359–64.
14. Kwiatkowska M, Norman G, Parker D. Stochastic model checking. In: International school on formal methods for the design of computer, communication and software systems. Springer; 2007. p. 220–70.
15. Kwiatkowska M, Norman G, Parker D. PRISM 4.0: verification of probabilistic real-time systems. In: International conference on computer aided verification. Springer; 2011. p. 585–91.

16. Dennis LA, Fisher M, Lisitsa A, Lincoln N, Veres SM. Satellite control using rational agent programming. IEEE Intell Syst. 2010;25(3):92–7. https://doi.org/10.1109/MIS.2010.88.
17. Webster M, Dennis LA, Dixon C, Fisher M, Stocker R, Sierhuis M. Formal verification of astronaut-rover teams for planetary surface operations. In: 2020 IEEE aerospace conference; 2020. p. 1–8.
18. Lincoln NK, Veres SM, Dennis LA, Fisher M, Lisitsa A. Autonomous asteroid exploration by rational agents. IEEE Comput Intell Mag. 2013;8(4):25–38. https://doi.org/10.1109/MCI.2013.2279559.
19. Dennis LA, Fisher M, Webster MP, Bordini RH. Model checking agent programming languages. Autom Softw Eng. 2012;19(1):5–63.
20. Sierhuis M, Clancey WJ. Modeling and simulating work practice: a method for work systems design. IEEE Intell Syst. 2002;17(5):32–41.
21.•• Stachtiari E, Mavridou A, Katsaros P, Bliudze S, Sifakis J. Early validation of system requirements and design through correctness-by-construction. J Syst Softw. 2018;145:52–78. **This study demonstrates in two different case studies involving satellites that the use of incremental design to build a model from simpler reusable designs that can enforce given properties can also improve the V&V cycle.**
22. Ivanov A, Masson L, Rossi S, Belloni F, Wiesendanger R, Gass V, Rothacher M, Hollenstein C, Männel B, Fleischmann P, Mathis H, Klaper M, Joss M, Styger E. CubETH: low cost GNSS space experiment for precise orbit determination. Tech. rep., EPFL. http://infoscience.epfl.ch/record/201520. 2014.
23. D'Argenio PR, Fraire JA, Hartmanns A. Sampling distributed schedulers for resilient space communication. In: Lee R, Jha S, and Mavridou A, editors. NASA Formal Methods. Cham: Springer International Publishing; 2020. p. 291–310.
24. Piel A, Bourrely J, Lala S, Bertrand S, Kervarc R. Temporal logic framework for performance analysis of architectures of systems. In: NASA Formal Methods Symposium. Springer; 2016. p. 3–18.
25. Sardar MU, Afaq N, Hoque KA, Johnson TT, Hasan O. Probabilistic formal verification of the SATS concept of operation. In: NASA formal methods symposium. Springer; 2016. p. 191–205.
26. Bonnal C, Ruault JM, Desjean MC. Active debris removal: recent progress and current trends. Acta Astronaut. 2013;85:51–60.
27. Dowek G, Munoz C, Carreno VA. Abstract model of the SATS concept of operations: initial results and recommendations. Tech. rep., NASA. 2004.
28. Gan X, Dubrovin J, Heljanko K. A symbolic model checking approach to verifying satellite onboard software. Sci Comput Program. 2014;82:44–55. Special Issue on Automated Verification of Critical Systems (AVoCS'11). https://doi.org/10.1016/j.scico.2013.03.005.
29. Bozzano M, Bruintjes H, Cimatti A, Katoen JP, Noll T, Tonetta S. Compass 3.0. In: Vojnar T and Zhang L, editors. Tools and algorithms for the construction and analysis of systems. Cham: Springer International Publishing; 2019. p. 379–85.
30. Esteve M, Katoen J, Nguyen VY, Postma B, Yushtein Y. Formal correctness, safety, dependability, and performance analysis of a satellite. In: 2012 34th International conference on software engineering (ICSE); 2012. p. 1022–31. https://doi.org/10.1109/ICSE.2012.6227118.
31. Andrews PB. An introduction to mathematical logic and type theory, vol. 27. Springer Science & Business Media. 2002.
32. Nipkow T, Paulson LC, Wenzel M. Isabelle/HOL: a proof assistant for higher-order logic, vol. 2283. Springer Science & Business Media. 2002.
33. Barras B, Boutin S, Cornes C, Courant J, Filliatre JC, Gimenez E, Herbelin H, Huet G, Munoz C, Murthy C, et al. The Coq proof assistant reference manual: version 6.1. 1997.
34. Munoz C, Narkawicz A. Formal analysis of extended well-clear boundaries for unmanned aircraft. In: NASA formal methods symposium. Springer; 2016. p. 221–6.
35. Owre S, Rushby JM, Shankar N. Pvs: a prototype verification system. In: International conference on automated deduction. Springer; 1992. p. 748–52.
36. Muñoz CA, Narkawicz A, Dutle A. From formal requirements to highly assured software for unmanned aircraft systems. In: Havelund K, Peleska J, Roscoe B, and de Vink EP, editors. Formal methods—22nd international symposium, FM 2018, Held as part of the federated logic conference, FloC 2018, Oxford, UK, July 15–17, 2018, Proceedings, Lecture notes in computer science, vol. 10951. Springer; 2018. p. 647–52. https://doi.org/10.1007/978-3-319-95582-7_38.
37. Verbeek F, Havle O, Schmaltz J, Tverdyshev S, Blasum H, Langenstein B, Stephan W, Wolff B, Nemouchi Y. Formal API specification of the PikeOS separation kernel. In: NASA formal methods symposium. Springer; 2015. p. 375–89.
38. Almeida J, Prochazka M. Safe and secure partitioning with PikeOS: towards integrated modular avionics in space. ESASP. 2009;669:27.
39. Windsor J, Hjortnaes K. Time and space partitioning in spacecraft avionics. In: 2009 Third IEEE international conference on space mission challenges for information technology. IEEE; 2009. p. 13–20.
40. Farrell M, Mavrakis N, Dixon C, Gao Y. Formal verification of an autonomous grasping algorithm. In: International symposium on artificial intelligence, robotics and automation in space. European Space Agency; 2020.
41. Leucker M, Schallhart C. A brief account of runtime verification. J Logic Algebr Program. 2009;78(5):293–303.
42. Bauer A, Leucker M, Schallhart C. Runtime verification for LTL and TLTL. ACM Trans Softw Eng Methodol. 2011;20(4):11:4–14:64.
43. Franceschini L. RML: runtime monitoring language: a system-agnostic DSL for runtime verification. In: Marr S and Cazzola W, editors. Conference companion of the 3rd international conference on art, science, and engineering of programming, Genova, Italy, April 1–4, 2019. ACM; 2019. p. 28:1–3. https://doi.org/10.1145/3328433.3328462.
44. Ferrando A, Dennis LA, Ancona D, Fisher M, Mascardi V. Verifying and validating autonomous systems: Towards an integrated approach. In: Colombo C and Leucker M, editors. Runtime verification—18th international conference, RV 2018, Limassol, Cyprus, November 10–13, 2018, Proceedings, Lecture Notes in Computer Science. Springer; 2018. p. 263–81. https://doi.org/10.1007/978-3-030-03769-7_15.
45. Rozier KY. From simulation to runtime verification and back: Connecting single-run verification techniques. In: Barrio AAD, Lynch CJ, Barros FJ, Hu X, and D'Ambrogio A, editors. 2019 Spring simulation conference, SpringSim 2019, Tucson, AZ, USA, April 29–May 2, 2019. IEEE; 2019. p. 1–10. https://doi.org/10.23919/SpringSim.2019.8732915.
46. Havelund K, Peled D. An extension of LTL with rules and its application to runtime verification. In: Finkbeiner B and Mariani L, editors. Runtime verification—19th international conference, RV 2019, Porto, Portugal, October 8–11, 2019, Proceedings, Lecture Notes in Computer Science. Springer; 2019. p. 239–55. https://doi.org/10.1007/978-3-030-32079-9_14.
47. Grotzinger JP, Crisp J, Vasavada AR, Anderson RC, Baker CJ, Barry R, Blake DF, Conrad P, Edgett KS, Ferdowski B, Gellert R, Gilbert JB, Golombek M, Gómez-Elvira J, Hassler

DM, Jandura L, Litvak M, Mahaffy P, Maki J, Meyer M, Malin MC, Mitrofanov I, Simmonds JJ, Vaniman D, Welch RV, Wiens RC. Mars Science Laboratory mission and science investigation. Space Sci Rev. 2012;170(1):5–56.

48. Havelund K, Peled D. First-order timed runtime verification using BDDs. In: Hung DV and Sokolsky O, editors. Automated technology for verification and analysis—18th international symposium, ATVA 2020, Hanoi, Vietnam, October 19–23, 2020, Proceedings, Lecture Notes in Computer Science. Springer; 2020. p. 3–24. https://doi.org/10.1007/978-3-030-59152-6_1.
49. Diftler MA, Mehling JS, Abdallah ME, Radford NA, Bridgwater LB, Sanders AM, Askew RS, Linn DM, Yamokoski JD, Permenter FA, Hargrave BK, Platt R, Savely RT, Ambrose RO. Robonaut 2—the first humanoid robot in space. In: 2011 IEEE International conference on robotics and automation; 2011. p. 2178–83. https://doi.org/10.1109/ICRA.2011.5979830.
50. Quigley M, Conley K, Gerkey B, Faust J, Foote T, Leibs J, Wheeler R, Ng A. ROS: an open-source robot operating system. In: Workshop on open source software at the international conference on robotics and automation. Japan: IEEE; 2009.
51.• Kempa B, Zhang P, Jones PH, Zambreno J, Rozier KY. Embedding online runtime verification for fault disambiguation on Robonaut2. In: Bertrand N and Jansen N, editors. Formal modeling and analysis of timed systems—18th international conference, FORMATS 2020, Vienna, Austria, September 1–3, 2020, Proceedings, Lecture Notes in Computer Science. Springer; 2020. p. 196–214. https://doi.org/10.1007/978-3-030-57628-8_12. **This work applies runtime verification using the hardware from a humanoid robot onboard ISS and demonstrates how difficult it can be to verify at runtime a resource constrained system.**
52. Ferrando A, Cardoso RC, Fisher M, Ancona D, Franceschini L, Mascardi V. ROSMonitoring: a runtime verification framework for ROS. In: Towards autonomous robotic systems conference (TAROS); 2020.
53.• Cardoso RC, Farrell M, Luckcuck M, Ferrando A, Fisher M. Heterogeneous verification of an autonomous curiosity rover. In: NASA formal methods symposium (NFM); 2020. **This paper successfully uses a range of V&V techniques in a space application by applying compositional verification to a simulation of the Mars Curiosity rover that is autonomously controlled by a cognitive agent.**
54. Myers GJ, Sandler C, Badgett T. The art of software testing, 3rd edn. New York: Wiley Publishing; 2011.
55. Kornfeld RP, Prakash R, Devereaux AS, Greco ME, Harmon CC, Kipp DM. Verification and validation of the Mars Science Laboratory/Curiosity Rover entry, descent, and landing system. J Spacecr Rockets. 2014;51(4):1251–69. https://doi.org/10.2514/1.A32680.
56. Francis R, Estlin T, Doran G, Johnstone S, Gaines D, Verma V, Burl M, Frydenvang J, Montaño S, Wiens RC, Schaffer S, Gasnault O, DeFlores L, Blaney D, Bornstein B. Aegis autonomous targeting for ChemCam on Mars Science Laboratory: deployment and results of initial science team use. Sci Robot. 2017;2(7). https://doi.org/10.1126/scirobotics.aan4582.
57. Araiza-Illan D, Western D, Pipe AG, Eder K. Systematic and realistic testing in simulation of control code for robots in collaborative human-robot interactions. In: Towards autonomous robotic systems, LNCS. Springer; 2016. p. 20–32.
58. Mavridou A, Bourbouh H, Garoche PL, Hejase M. Evaluation of the FRET and CoCoSim tools on the ten Lockheed Martin cyber-physical challenge problems. Tech. rep., Technical report, TM-2019-220374, NASA. 2019.
59. Alpaydin E. Introduction to machine learning. MIT Press. 2020.
60. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: 2016 IEEE conference on computer vision and pattern recognition (CVPR); 2016. p. 770–8. https://doi.org/10.1109/CVPR.2016.90.
61. Endres MG, Hillen F, Salloumis M, Sedaghat AR, Niehues SM, Quatela O, Hanken H, Smeets R, Beck-Broichsitter B, Rendenbach C, et al. Development of a deep learning algorithm for periapical disease detection in dental radiographs. Diagnostics. 2020;10(6):430. https://doi.org/10.3390/diagnostics10060430.
62. Nusrat A, Gabriel HF, Haider S, Ahmad S, Shahid M, Ahmed Jamal S. Application of machine learning techniques to delineate homogeneous climate zones in river basins of Pakistan for hydro-climatic change impact studies. Appl Sci. 2020;10(19):6878. https://doi.org/10.3390/app10196878.
63. Gomede E, Miranda de Barros R, de Souza Mendes L. Use of deep multi-target prediction to identify learning styles. Appl Sci. 2020;10(5):1756. https://doi.org/10.3390/app10051756.
64. Borg M, Englund C, Wnuk K, Duran B, Levandowski C, Gao S, Tan Y, Kaijser H, Lönn H, Törnqvist J. Safely entering the deep: a review of verification and validation for machine learning and a challenge elicitation in the automotive industry. J Automot Softw Eng. 2019;1:1–19. https://doi.org/10.2991/jase.d.190131.001.
65. Huang X, Kroening D, Ruan W, Sharp J, Sun Y, Thamo E, Wu M, Yi X. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. Comput Sci Rev. 2020;37:100270. https://doi.org/10.1016/j.cosrev.2020.100270.
66. Zhao X, Banks A, Sharp J, Robu V, Flynn D, Fisher M, Huang X. A safety framework for critical systems utilising deep neural networks. In: Proceedings of 39th international conference on computer safety, reliability, and security (SAFECOMP), Lecture Notes in Computer Science. Springer; 2020. p. 244–59. https://doi.org/10.1007/978-3-030-54549-9_16.
67. Zhang JM, Harman M, Ma L, Liu Y. Machine learning testing: survey, landscapes and horizons. IEEE Trans. Softw Eng 1–1. 2020.
68. Phan DT, Grosu R, Jansen N, Paoletti N, Smolka SA, Stoller SD. Neural simplex architecture. In: Lee R, Jha S, and Mavridou A, editors. NASA formal methods. Cham: Springer International Publishing; 2020. p. 97–114.
69. Gopinath D, Katz G, Păsăreanu CS, Barrett C. DeepSafe: a data-driven approach for assessing robustness of neural networks. In: International symposium on automated technology for verification and analysis. Springer; 2018. p. 3–19.
70. Katz G, Huang DA, Ibeling D, Julian K, Lazarus C, Lim R, Shah P, Thakoor S, Wu H, Zeljić A, Dill DL, Kochenderfer MJ, Barrett C. The Marabou framework for verification and analysis of deep neural networks. In: Dillig I and Tasiran S, editors. Computer aided verification. Cham: Springer International Publishing; 2019. p. 443–52.
71. Katz G, Barrett C, Dill DL, Julian K, Kochenderfer MJ. Reluplex: an efficient SMT solver for verifying deep neural networks. In: International conference on computer aided verification. Springer; 2017. p. 97–117.

72. Dennis LA, Fisher M, Lincoln NK, Lisitsa A, Veres SM. Practical verification of decision-making in agent-based autonomous systems. Autom Softw Eng. 2016;23(3):305–59. https://doi.org/10.1007/s10515-014-0168-9.
73. Fisher M, Mascardi V, Rozier KY, Schlingloff BH, Winikoff M, Yorke-Smith N. Towards a framework for certification of reliable autonomous systems. J Auton Agents Multiagent Syst. (2020). (to appear).
74. Vassev E, Hinchey M. Autonomy requirements engineering for space missions. NASA Monographs in Systems and Software Engineering. Springer; 2014. https://doi.org/10.1007/978-3-319-09816-6.
75. Bensalem S, Bozga M, Sifakis J, Nguyen TH. Compositional verification for component-based systems and application. In: Cha SS, Choi JY, Kim M, Lee I, and Viswanathan M, editors. Automated technology for verification and analysis. Berlin: Springer; 2008. p. 64–79.
76. Garavel H, Lang F, Mounier L. Compositional verification in action. In: Formal methods for industrial critical systems. Springer; 2018. p. 189–210.
77. Giannakopoulou D, Namjoshi KS, Păsăreanu CS. Compositional reasoning. Cham: Springer International Publishing; 2018, pp. 345–83. https://doi.org/10.1007/978-3-319-10575-8_12.
78. Cardoso RC, Dennis LA, Farrell M, Fisher M, Luckcuck M. Towards compositional verification for modular robotic systems. In: Proceedings second workshop on formal methods for autonomous systems, virtual, 7th of December 2020, electronic proceedings in theoretical computer science. Open Publishing Association; 2020. p. 15–22. https://doi.org/10.4204/EPTCS.329.2.
79. Backes J, Cofer D, Miller S, Whalen MW. Requirements analysis of a quad-redundant flight control system. In: NASA formal methods symposium. Springer; 2015. p. 82–96.
80. Champion A, Gurfinkel A, Kahsai T, Tinelli C. CoCoSpec: a mode-aware contract language for reactive systems. In: International conference on software engineering and formal methods, LNCS. Springer; 2016. p. 347–66.
81. Menghi C, Tsigkanos C, Pelliccione P, Ghezzi C, Berger T. Specification patterns for robotic missions. IEEE Trans Softw Eng. 2019.
82. Webster M, Western D, Araiza-Illan D, Dixon C, Eder K, Fisher M, Pipe AG. A corroborative approach to verification and validation of human–robot teams. Int J Robot Res, 2020; 39(1). https://doi.org/10.1177/0278364919883338.