ABZ 2026 Case Study: A Planetary Rover

Marie Farrell[1] and Tsutomu Kobayashi[2]

[1] The University of Manchester, Manchester, UK
marie.farrell@manchester.ac.uk
[2] Japan Aerospace Exploration Agency (JAXA), Tsukuba, Japan
kobayashi.tsutomu@jaxa.jp

Abstract. This document summarises the ABZ 2026 case study challenge. The system under consideration is that of an autonomous planetary rover. We summarise the architecture for the rover, its requirements and the tasks that participants in the case study track should focus on. We also outline several extensions that may be considered in future work or that offer other perspectives for formal modelling and verification.

1 Introduction

We propose the case study of a semi-autonomous planetary rover undertaking an inpsection mission. This use case is adapted from [2–4] and modified in several interesting ways. The software architecture for our rover is illustrated in Fig. 2. The requirements for the components are described throughout the remainder of this document, along with several system-level requirements. The examples discussed in [2–4] were inspired by real-world missions including the Curiosity [20, 21] and Perseverance [9] planetary exploration rovers illustrated in Fig. 1. These sophisticated robotic systems have more complex architectures and functionalities than we present here including sample collection [9] and autonomous laser targetting [12] capabilities. This ABZ 2026 case study focuses on simplified, but sufficiently complex, aspects of these rover missions including waypoint inspection, failure modes and communication.

It has been well observed that formal specification and verification of autonomous robotic systems remains a challenging topic, with heterogeneous and integrated formal methods recommended to tackle the intricate complexities of these systems [1, 11, 15]. This is further complicated by the unpredictable environment in the space domain alongside our inability to provide high-fidelity testing of these systems on planet Earth [5]. More specifically, a rover must handle interactions with the planet's ground while avoiding obstacles or valleys, relying on sensors and actuators that operate within strict physical constraints. The harsh environment and physical constraints also cause problems with the battery and unreliable communication. Therefore, functionalities for detecting failures and recovery from them are vital for safety. At the same time, the large budget and human resources invested in space development raise expectations for mission achievement, which often conflict with safety.

Fig. 1. NASA's Curiosity (left) and Perseverance (right) rovers. Source: NASA.

2 Requirements

We begin by defining the architecture for our autonomous robotic system. This architecture is shown in Fig. 2. We include several components and sub-components. Some of these are grouped to demonstrate the system decomposition. Specifically, we have a Vision system which is used for obstacle recognition and determining the current position of the rover. Then the MapValidator is used to validate the information coming from the Vision component along with the information that was communicated by the ground station which includes the list of prioritised goal locations and charging locations. The Goal Reasoning Agent (GRA) is charged with selecting the next goal that the rover should visit. This can either be the next one in the prioritised list or a charging location depending on the remaining battery available, as calculated by the BatteryMonitor which sits inside the Battery+Hardware component shown in Fig. 2. We then have two copies of the same functionality in the ComputePlan2Charging and ComputePlan2Destination components. Each of these comprises a Planner and a Plan Reasoning Agent (PRA). The idea here is that ComputePlan2Destination produces a plan (list of waypoints) that the rover should traverse to reach the goal location. In parallel, ComputePlan2Charging provides a plan from the goal location to the nearest charging location. This ensures that the rover will have enough battery power to reach the next goal and then recharge if necessary. This is an important point and is needed to meet the system-level requirement that the rover should never run out of battery (which will be dicussed later).

As mentioned already, the Battery+Hardware component keeps track of the battery level through the BatteryMonitor. However, it also interfaces with the hardware (wheels, motors, actuators, etc.) to traverse through the waypoints in the plans that are sent to it by ComputePlan2Charging and ComputePlan2Destination. We make the SolarPanelController explicit here. When the rover reaches a charging location the solar panels should be opened and subsequently closed once charing is complete. This is a sophisticated functionality that has not been present on older rovers but it will be useful in future missions. When the solar panels are left open during an entire mission, they can become very dusty

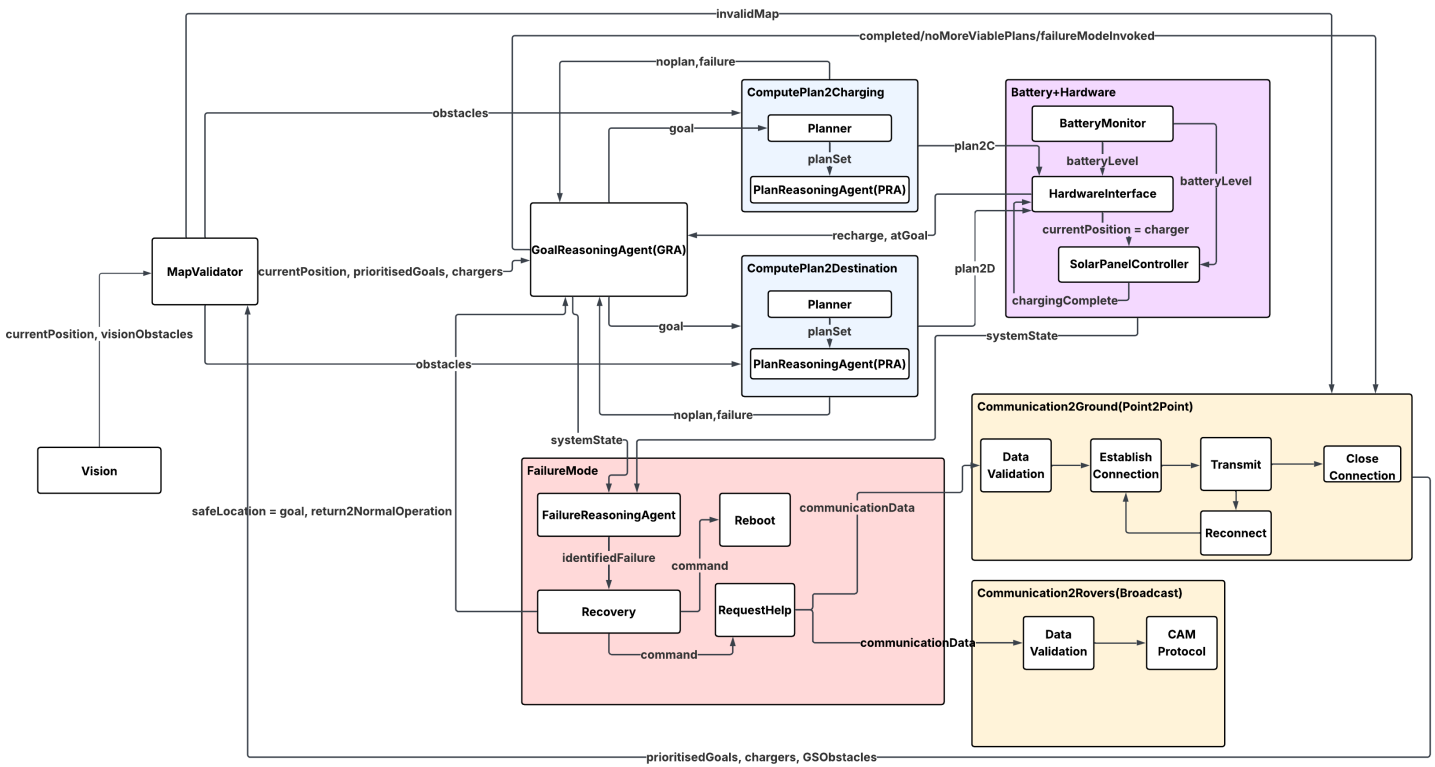

Fig. 2. System Architecture

making it difficult and sometimes impossible to charge. NASA's insight lander suffered from this exact issue[3].

Distant plantary missions in hostile and unpredictable environments can experience several causes of failure. Recovery from such failures is typically not trivial. Hence, we include the FailureMode component which comprises a FailureReasoningAgent who determines the most likely cause of the failure, a Recovery component who invokes various recovery operation. These include a simple Reboot or RequestHelp. Help can be requested from the ground station via Communication2Ground or from other nearby rovers using Communication2Rovers. Each of these communication options follow a different protocol. The communication to the ground station is a point-to-point communication whereas communicating with other rovers takes place in a broadcast style. We describe the kinds of communications in a bit more detail later but we note that we do not concern ourselves with security aspects of message passing in this case study. That said, cybersecurity is especially important in space applications [17] so an extension of this case study to focus on security would be a useful contribution. When assistance is requested through the RequestHelp component, it may take some time to receive a reply from the ground (40 minutes round trip communication time from Earth to Mars) and from other rovers so the WaitForHelp component periodically rechecks whether a response has been received.

2.1 System-Level Requirements

As we defined this use case, we moved through each of the components in Fig. 2 and elicited requirements for each. We also examined the system-level requirements. These were inspired by [2–4], with the addition of a requirment related to failure modes. We summarise the system-level requirements in Table 1.

ID	Description
SL1	The rover shall never run out of battery.
SL2	The rover shall eventually visit all of the goal locations.
SL3	The rover shall react appropriately to failures, requesting help from the ground station and other rovers if needed.
SL4	The rover shall not collide with an obstacle.
SL5	The rover shall have a low incidence of failures.

Table 1. System-level requirements.

2.2 The Vision Component

We assume that the rover is equipped with appropriate sensors and cameras to enable perception of obstacles and localisation. We do not focus on the low-level

[3] `https://www.jpl.nasa.gov/news/nasa-mars-orbiter-spots-retired-insight-lander-to-study-dust-movement/`

details of localisation algorithms that are commonly used in robotic systems such as Simultaneous Localisation and Mappling (SLAM) [7]. However, we welcome submissions that focus on these finer-grained details. We define several requirements for the Vision component in Table 2. We recognise that vision perception systems are not often completely exact in nature and this is reflected in the requirements.

ID	Description
V1	Vision shall take input from sensors including a camera and LiDAR.
V2	Vision shall return the current position of the rover (using SLAM algorithms [7]) and a set of obstacle positions.
V3	Vision shall identify obstacles with an accuracy of over 95%.
V4	The current position of the rover identified by the Vision component shall not collide with any of the identified obstacles.
V5	The current position identified by the Vision component shall be the actual current position of the rover within $\pm$ 2 metres.
V6	The Vision system shall be robust to small perturbations in input data (local robustness), e.g. [13].

Table 2. The Vision system requirements.

2.3 The MapValidator Component

Recognising that both the ground station and Vision system may incorrectly label `obstacles`, `GSObstacles`, `chargers`, `prioritisedGoals` and the rover's `currentPosition`. We include the MapValidator component to do some sanity checking that goals, chargers and the current position of the rover do not collide with obstacles that are identified by the Vision system or the ground station. In cases where there are issues, we sent an `invalidMap` notification to the ground station where we assume that human operators can assist by either (1) updating the data that they orignally sent or (2) overriding certain perceptions of the Vision system. We include the requirements for the MapValidator in Table 3.

2.4 The Goal Reasoning Agent (GRA)

We use a symbolic Goal Reasoning Agent to choose the next waypoint that the rover should navigate to. In previous work, a similar agent was implemented using Lustre [2, 3] and in Gwendolen [4]. This rules-based agent is responsible for ensuring that the rover can always reach a `charger` position from any `goal` that it selects and instructing the rover to navigate directly to a `charger` when needed. It also notifies the ground station when the current mission is completed and invokes the FailureMode as required. We summarise the requirements for the Goal Reasoning Agent in Table 4.

ID	Description
M1	The MapValidator shall validate the `obstacles` and `currentPosition` provided by the Vision component against the `prioritisedGoals`, `GSObstacles` and `chargers` provided by the ground station.
M2	The `obstacles` sent from the MapValidator to the Goal Reasoning Agent shall not collide with the `currentPosition` of the rover.
M3	The `obstacles` sent from the MapValidator to the Goal Reasoning Agent with the `charger` positions.
M4	If the map is invalid then the MapValidator shall send a message to the ground station to seek advice, operators can update and override the original map.
M5	The `obstacles` sent from the MapValidator to the Goal Reasoning Agent shall not collide with the goal positions of the rover that are given by the ground station.

Table 3. The MapValidator requirements.

ID	Description
G1	The GRA shall select the next `goal` for the rover to navigate to from the `prioritisedGoals` that were received from the ground station.
G2	Whenever the `recharge` flag is set to true then the GRA shall set the `goal` as the nearest charging position.
G3	If the `atGoal` flag is true then the GRA should remove the `goal` that was acheived from the prioritised list of goals before selecting the next one.
G4	The `goal` that is selected by the GRA shall be the one with the highest priority in the list, unless the `recharge` flag is set to true.
G5	If either planning component (ComputePlan2Charging or ComputePlan2Destination) returns `noplan` then the GRA shall inform the ground station and await instuctions before proceeding.
G6	If all goals have been acheived then the GRA shall notify the ground station that the current mission has been completed.
G7	If a failure is identified by either of the planners then the GRA shall enter FailureMode and notify the ground station that it is in failure mode
G8	If failure is detected then the GRA shall send the `systemState` to the FailureMode component.
G9	The ground station can instruct the GRA to set the goal as a designated `safeLocation`, in this case, the GRA shall put this `safeLocation` as the `goal` by updating the prioritised list of goals.

Table 4. The Goal Reasoning Agent (GRA) requirements.

2.5 The ComputePlan2Charging Component

The ComputePlan2Charging component is made up of two subcomponents: a Planner and a PlanReasoningAgent(PRA) as illustrated in Fig. 2. The requirements for this component are shown in Table 5. Various path planning algorithms might be used by the Planner that are appropriate for robotic systems. This can range from a simple depth-first search style algorithm to more sophisticated implementations. We do not seect a specific planning algorithm so participants in the case study track may choose any suitable and/or interesting algorithm.

ID	Description
CPC1	The Planner shall compute a set of plans (sequences of (x,y) coordinates) from the `goal` position to the nearest `charger` position.
CPC2	The plans produced by the Planner shall not contain any obstacles.
CPC3	The shortest path to the charger, `plan2C`, shall be selected by the PRA.
CPC4	If there are no viable plans then the ComputePlan2Charging component shall return `noplan`, otherwise the `plan2C` shall be sent to the Battery+Hardware component.
CPC5	If a failure in planning occurs (e.g. timeout) the ComputePlan2Charging shall notify the GRA.

Table 5. The ComputePlan2Charging requirements.

2.6 The ComputePlan2Destination Component

The ComputePlan2Destination component has the same structure as the ComputePlan2Charging component. The only difference is that is computes the plan to the destination goal, `plan2D`, rather than the charging location as default. Note that this `goal` destination might be a charging position in situations where the GRA sends the rover to recharge. In this case the `plan2C` produced would be empty because the rover will be at the charger. The requirements are listed in Table 6.

ID	Description
CPD1	The Planner shall compute a set of plans (sequences of (x,y) coordinates) from the `currentPosition` to the `goal` that has been selected by the GRA.
CPD2	The plans produced by the Planner shall not contain any obstacles.
CPD3	The shortest path to the goal location, `plan2D`, shall be selected by the PRA.
CPD4	If there are no viable plans then the ComputePlan2Destination component shall return `noplan`, otherwise the `plan2D` shall be sent to the Battery+Hardware component.
CPD5	If a failure in planning occurs (e.g. timeout) the ComputePlan2Charging shall notify the GRA.

Table 6. The ComputePlan2Destination requirements.

2.7 The Battery+Hardware Component

We group the BatteryMonitor, HardwareInterface and SolarPanelController together in Fig. 2 as these are concerned with low-level physical control and measurements. For ease of calculation, we assume that each step in any given plan for the rover consumes 1 unit of battery power. More sophisticated models could have knowledge of the terrain topography and use this to calculate battery used per plan step. For example, going up a hill may use more battery than traversing a level area. We encourage participants to consider more complicated models where possible. The HardwareInterface is responsible for sending low-level movement commands to various actuators that control motion. This rover system uses the Robot Operating System (ROS) [16] so pre-existing libraries are used here such as `move_base`. We include the requirements for the Battery+Hardware component in Table 7.

ID	Description
HI1	The HardwareInterface shall set the `recharge` flag to true if the `batteryLevel` is not high enough to reach the `goal` and subsequently reach a charging location.
HI2	Once at a charging position, the rover shall remain there until the battery has been recharged.
HI3	The `atGoal` flag shall be set to true only when the `goal` position has been reached.
HI4	The HardwareInterface shall send movement and velocity commands to the actuators that control robot movement.
HI5	The commands that the HardwareInterface sends to the actuators shall eventaully be received and acted upon i.e. an instruction to move left does indeed lead to the robot moving left.
HI6	When the `currentPosition` is a `charger` position then the solar panels shall be opened (and tilted towards the sun).
HI7	When charging is complete the solar panels shall be closed. This is to avoid dust accumulating while the rover is moving and protect the panels from adverse weather conditions.
BM1	The BatteryMonitor shall monitor the battery level of the rover.
BM2	The BatteryMonitor shall return the battery level as 5% less than the measured level. This is to ensure reliability under degradation.

Table 7. The Battery+Hardware component requirements.

2.8 The Communication2Ground (Point2Point) Component

The robot is capable of communicating with the ground station. It uses this communication link to send/receive data including updated goal lists, notify mission completed, and report erroneous behaviour. We view this as a point-to-point communication where one rover communicates with one ground station. While we do not dictate a specific protocol to use here, Fig. 2 outlines the appropriate steps that this communication should follow. We assume secure communications

and are not concerned with bit flips at this stage but a more realistic communications protocol should take these things into consideration [17]. We inlcude the requirements for this component in Table 8 and we welcome submissions exploring this aspect.

ID	Description
CG1	Communcation data that is received shall be validated by the Data Validation component to ensure it is in the correct format (`invalidMap, completed, noMoreViablePlans, failureModeInvoked, communicationData`).
CG2	Communication2Ground shall establish a connection with the ground station using a standard protocol (e.g. [6,14]). This shall operate with a greater than 99% success rate.
CG3	If there is a connection failure (broken connection, timeout, etc.) then Communication2Ground shall attempt to reconnect.
CG4	Messages/data shall be transmitted within a reasonable time frame (e.g. 10 ticks).
CG5	Once all of the data is sent then Communication2Ground shall wait until a response is received (e.g. up to 1 hour on Mars, up to 5 seconds on the Moon).
CG6	Once the response is received then Communication2Ground shall close the connection.
CG7	The response includes prioritised goals, charger and obstacle positions that shall be passed to the MapValidator component.
CG8	A failure reconnection notification shall be sent to the FailureReasoningAgent if the reconnection fails more than 3 times consecutively.

Table 8. The Communication2Ground requirements.

2.9 The Communication2Rovers(Broadcast) Component

In more futuristic space missions, we envisage teams of rovers carry out planetary tasks ranging from sample collection to habitat construction. In such scenarios, rovers work together in teams to achieve complex tasks, an aspect that we do not focus on in this case study. However, we do consider the situation where rovers must assist each other. For example, when one rover gets stuck on a rock or in sand then another rover can be used to (1) provide visual data to the fground station to help operators decide the best course of action or (2) to simply push the stuck rover away from the hazard. In these situations, we assume that a broadcast style of communication is used that alerts nearby rovers of the situation. One potential protocol for doing this is the Cooperative Awareness Message (CAM) protocol that is to be used in autonomous vehicles [10]. The CAM protocol is summarised in [8,18]. Table 9 summarises the requirements for the Communication2Rovers component.

2.10 The FailureMode Component

Since space robotic systems operate in an unpredicatable, hazardous and hostile environment that we do not have accurate simulations or means for physically

ID	Description
CR1	Communcation data that is received shall be validated by the Data Validation component to ensure it is in the correct format (`location, failure, etc.`).
CR2	The Communication2Rovers component shall communicate with nearby rovers using the Cooperative Awareness Message protocol [8].
CR3	Communication2Rovers shall send the `helperId` of the rover that is coming to assist so that the main rover can check it's ID when it gets there. This is a security check to ensure that the rover is not "assisted" by an adversary.

Table 9. The Communication2Rovers requirements.

testing, we assume that these systems will fail (repeatedly) [19]. This was already evident in the communication2Ground component that we discussed earlier (Section 2.8). As a result, we include the FailureMode shown in Fig. 2. The purpose of this component is to react appropriately to failures that occur in other parts of the system. (The FailureMode could also fail itself, but we ignore this for now.) We include a FailureReasoningAgent to diagnose the cause of failures. This could be a rather complex component. For example, since it is impossible to predict all failures in advance for space missions, this component may incorporate some kind of learning. Alternatively, and for simplicity, submissions to the case study track may assume that it has an accurate model of all failures. We define requirements for the FailureMode in Table 10.

ID	Description
FM1	In cases of failure, the FailureMode shall be invoked by the GRA and Battery+Hardware via the sending of the `systemState` or by the Communication2Ground by sending a `failed2Reconnect` message.
FM2	The FailureReasoningAgent shall consider the `systemState` and output the most likely cause of the failure. For example, the planner times out and fails to compute a plan. This could be a planning failure or a localisation failure.
FM3	Once the failure has been identified the recovery mode shall either reboot or request help from the ground station and/or other rovers that are nearby. For example, it could be stuck in a crater and need physical help and/or manual remote control procedures to be invoked.
FM4	If the rover is instructed to wait for help then it should wait for a reasonable amount of time before rechecking if help is on the way.
FM5	In Recovery mode, while deciding what to do the rover shall be sent to a `safeLocation` (if possible). This is achieved by updating the GRA to set the next goal to be the safe position. An example would be the SolarPanelController is failing or the communication continues to fail in its current physical position (poor signal).

Table 10. The FailureMode requirements.

3 Expectations: Case Study Track Submissions

We anticipate contributions to the ABZ 2026 case study track that tackle the rover system that is described in this document and outlined in Fig. 2. We recognise that the requirements that we have listed above have diverse characteristics, ranging from propositional logic style to incorporating probabilistic aspects that capture levels of uncertainty in movement and perception. We also include several requirements that are especially difficult to formally model and verify. We welcome submissions that focus on any combination of these aspects.

We note that the rover is built using a standard middleware such as the Robot Operating System (ROS) [16] which uses a range of in-built and contributed library functions including `move_base` for low-level movement. The majority of these library functions are not verified and reliability is usually argued based on sheer volume of use in-the-wild. While we do not instruct participants in the case study track to spend significant time worrying about modelling, specifying and verifying message passing in ROS, we would be glad to see submissions that consider this aspect. Providing formal guarantees at this level remains an open research challenge and contrubutions in this area would certainly be in scope for the ABZ 2026 case study track.

In an ideal world, participants should model, specify and verify the architecture shown in Fig. 2 and (at least a subset of) the requirements that we have outlined in this document. Due to the complexity of robotic systems, it is likely that a range of formal techniques as well as software testing and simulation would be needed to provide complete coverage of this system [15]. Submissions focusing on specific components, frameworks, modelling paradigms, verification tools and specification logics are thus encouraged.

Participants in the case study track are encouraged to make any simplifying assumptions about the environment, ground station functions, communication protocols, nearby rover functions, cybersecurity mitigations and precision of the onboard equipment. We expect that these assumptions are clearly outlined in the submissions.

Several components of our system embody Artificial Intelligence (AI) this includes the Vision and Goal Reasoning Agent components. The species of AI used in each is quite different with the Vision system relying on sub-symbolic AI and the Goal Reasoning Agent using symbolic AI. Verification for each of these kinds of AI is usually very different ranging from testing to formal proof and we encourage articipants to consider the AI aspects of this system using their chosen tools. They can also model/implement additional measures to provide assurance around the AI components such as the addition of safety shields and/or runtime monitoring.

References

1. A. Azaiez, D. A. Anisi, M. Farrell, and M. Luckcuck. Revisiting formal methods for autonomous robots: A structured survey. In *Annual Conference Towards Autonomous Robotic Systems*, pages 338–352. Springer, 2025.

2. H. Bourbouh, M. Farrell, A. Mavridou, and I. Sljivo. Integration and evaluation of the advocate, fret, cocosim, and event-b tools on the inspection rover case study. Technical report, NASA, 2020.
3. H. Bourbouh, M. Farrell, A. Mavridou, I. Sljivo, G. Brat, L. A. Dennis, and M. Fisher. Integrating formal verification and assurance: an inspection rover case study. In *NASA Formal Methods Symposium*, pages 53–71. Springer, 2021.
4. R. C. Cardoso, M. Farrell, M. Luckcuck, A. Ferrando, and M. Fisher. Heterogeneous verification of an autonomous curiosity rover. In *NASA Formal Methods Symposium*, pages 353–360. Springer, 2020.
5. R. C. Cardoso, G. Kourtis, L. A. Dennis, C. Dixon, M. Farrell, M. Fisher, and M. Webster. A review of verification and validation for space autonomous systems. *Current Robotics Reports*, 2(3):273–283, 2021.
6. CCSDS. Overview of space communications protocols. Technical report, CCSDS, 2007.
7. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 1403–1410. IEEE, 2003.
8. T. ETSI. Intelligent transport systems (its); vehicular communications; basic set of applications; part 2: Specification of cooperative awareness basic service. *Draft ETSI TS*, 20(2011):448–51, 2011.
9. K. A. Farley, K. H. Williford, K. M. Stack, R. Bhartia, A. Chen, M. de la Torre, K. Hand, Y. Goreva, C. D. Herd, R. Hueso, et al. Mars 2020 mission overview. *Space Science Reviews*, 216(8):142, 2020.
10. M. Farrell, M. Bradbury, R. C. Cardoso, M. Fisher, L. A. Dennis, C. Dixon, A. T. Sheik, H. Yuan, and C. Maple. Security-minded verification of cooperative awareness messages. *IEEE Transactions on Dependable and Secure Computing*, 21(4):4048–4065, 2023.
11. M. Farrell, M. Luckcuck, and M. Fisher. Robotics and integrated formal methods: Necessity meets opportunity. In *International Conference on Integrated Formal Methods*, pages 161–171. Springer, 2018.
12. R. Francis, T. Estlin, D. Gaines, B. Bornstein, S. Schaffer, V. Verma, R. Anderson, M. Burl, S. Chu, R. Castano, et al. Aegis autonomous targeting for the curiosity rover's chemcam instrument. In *2015 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, pages 1–5. IEEE, 2015.
13. D. Gopinath, G. Katz, C. S. Păsăreanu, and C. Barrett. Deepsafe: A data-driven approach for assessing robustness of neural networks. In *International symposium on automated technology for verification and analysis*, pages 3–19. Springer, 2018.
14. O. Kodheli, E. Lagunas, N. Maturo, S. K. Sharma, B. Shankar, J. F. M. Montoya, J. C. M. Duncan, D. Spano, S. Chatzinotas, S. Kisseleff, et al. Satellite communications in the new space era: A survey and future challenges. *IEEE Communications Surveys & Tutorials*, 23(1):70–109, 2020.
15. M. Luckcuck, M. Farrell, L. A. Dennis, C. Dixon, and M. Fisher. Formal specification and verification of autonomous robotic systems: A survey. *ACM Computing Surveys (CSUR)*, 52(5):1–41, 2019.
16. S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall. Robot operating system 2: Design, architecture, and uses in the wild. *Science robotics*, 7(66):eabm6074, 2022.
17. C. Maple, M. Bradbury, H. Yuan, M. Farrell, C. Dixon, M. Fisher, and U. I. Atmaca. Security-minded verification of space systems. In *2020 IEEE Aerospace Conference*, pages 1–13. IEEE, 2020.

18. J. Santa, F. Pereñíguez, A. Moragón, and A. F. Skarmeta. Vehicle-to-infrastructure messaging proposal based on cam/denm specifications. In *2013 IFIP Wireless Days (WD)*, pages 1–7. IEEE, 2013.
19. S. Stancliff, J. Dolan, and A. Trebi-Ollennu. Planning to fail: reliability as a design parameter for planetary rover missions. In *Proceedings of the 2007 Workshop on Performance Metrics for Intelligent Systems*, pages 204–208, 2007.
20. A. R. Vasavada. Mission overview and scientific contributions from the mars science laboratory curiosity rover after eight years of surface operations. *Space Science Reviews*, 218(3):14, 2022.
21. R. Welch, D. Limonadi, and R. Manning. Systems engineering the curiosity rover: A retrospective. In *2013 8th international conference on system of systems engineering*, pages 70–75. IEEE, 2013.