VERIFICATION AND VALIDATION OF AUTONOMOUS SYSTEMS WITH EMBEDDED AI: THE VIVAS APPROACH

Simone Fratini[1], Patrick Fleith[1], Nicola Policella[1], Alberto Griggio[2], Stefano Tonetta[2], Srajan Goyal[2], Thi Thieu Hoa Le[2], Jacob Kimblad[2], Chun Tian [2], Konstantinos Kapellos[3], Christos Tranoris[3], and Quirien Wijnands[4]

[1]*Solenix Engineering GmbH, Germany, {simone.fratini,patrick.fleith,nicola.policella}@solenix.de*
[2]*Fondazione Bruno Kessler, Italy, {griggio,tonettas,sgoyal,tle,jkimblad,ctian}@fbk.eu*
[3]*Trasys International, Belgium, {konstantinos.kapellos,christos.tranoris}@trasysinternational.com*
[4]*European Space Agency, ESA/ESTEC, The Netherlands, quirien.wijnands@esa.int*

ABSTRACT

This paper reports on the recent ESA-ESTEC activity "Verification & Validation of Autonomous System" (VIVAS), executed under the Technology Development Element (TDE) programme. Objective of VIVAS is to propose and demonstrate a generic Verification and Validation methodology based on the usage of the System-level Simulation Facilities, specifically targeted at autonomous systems using AI models. The activity had led to the design and implementation of a Verification & Validation (V&V) framework that leverages model checking technologies to automate a loop of test-cases generation, execution (on a system-level simulator encompassing AI models) and the validation of the executed traces against a given set of properties and coverage conditions. The implemented use-cases for the demonstration of the VIVAS framework is based on 3DROV, a Planetary Robot Design, Generic Visualization and Validation Tool to provide end-to-end robotic operations simulation capabilities in closed loop with the environment.

Key words: Verification&Validation, AI, Autonomy.

1. INTRODUCTION

The need for autonomy in space applications is well known. Stringent communications constraints (limited communication windows, long communication latencies and limited bandwidth, uncertainty in the operational scenarios), limited access and availability of operators, limited crew availability, system complexity, and many other factors preclude direct human oversight and control of many activities.

Growing usage of Artificial Intelligence (AI) and Machine Learning (ML) technologies in autonomous systems poses new challenges, because the approach is very effective in implementing specific functionalities, but it comes with inherent uncertainty, and thus may be not suitable "as-is" for mission-critical systems. The validation and verification of autonomous systems using AI/ML components, or integrating AI/ML components with control-based planning and scheduling systems, is therefore of paramount importance for future missions.

This paper reports on the recent ESA-ESTEC activity "Verification & Validation of Autonomous System" (VIVAS), executed under the Technology Development Element (TDE) programme. Objective of VIVAS is to propose and demonstrate a generic Verification and Validation methodology based on the usage of the System-level Simulation Facilities, specifically targeted at autonomous systems using AI models. The activity had led to the design and implementation of a V&V framework that, starting from a symbolic model describing system and environmental conditions, leverages model checking technologies to automate a loop of test-cases generation, execution (on a system-level simulator encompassing AI/ML models) and the validation of the executed traces against a given set of properties and coverage conditions, formally specified within the symbolic model. The outcome of the VIVAS framework consists of coverage statistics of the executed traces with respect to the symbolic models and quantitative and qualitative information for each executed test case. The proposed approach has been demonstrated by implementing a Proof of Concept based on a state-of-the-art simulator of a planetary robotic asset making use of on-board ML models.

More in concrete, the paper presents (1) a background on autonomy in space and on V&V for autonomous systems embedding AI/ML modules (in Section 2); (2) a V&V methodology, based on model checking and simulation, for autonomous systems leveraging AI/ML technology and the "VIVAS" Framework, a general, proof of concept, domain independent, software implementation of the proposed methodology (in Section 3) and (3) examples of application of VIVAS on two use cases simulating Martian operational scenarios (in Section 4).

The implemented use-cases for the demonstration of the VIVAS framework are based on 3DROV, a Planetary Robot Design, Generic Visualization and Validation Tool to provide end-to-end robotic operations simulation capabilities in closed loop with the environment. Using the 3DROV simulator, the VIVAS framework capabilities have been demonstrated on two rover based use-cases.

2. BACKGROUND

Existing autonomous systems in space applications have shown an increasing use of various kinds of AI based software technologies that contributed to the development of capabilities for autonomy, including Intelligent Sensing, Planning & Execution, Fault Protection and Health Management, Distributed Decision Making & Coordination. High levels of autonomy and automation enable a wider variety of more capable missions, and enable human operators to focus on higher level tasks for which they are better suited. Indeed, in many situations autonomy is far more than just a convenience; it is a need for the mission. Deep space and robotic exploration in particular requires more autonomy, as communications with ground mission operators are infrequent and delayed, so that the spacecraft must react to opportunities and hazards without immediate human control[8].

The most notable achievement of an autonomy in space is probably the Autonomous Sciencecraft Experiment (ASE) on Earth Observing 1 (EO-1) [4]. ASE demonstrated higher-level commanding, on-board scheduling and re-scheduling capabilities with robust execution by responding to events and anomalies at execution time. ASE flew for over 12 years maximizing EO-1's science return by processing on-board data to update and adjust the baseline observation schedule. With the ASE, ML has started to be used for on-board data analytics. The experiment has been continuously updated during the years of operation, with the progressive introduction of ML and on-board automated (re)-scheduling of activities that fostered the introduction of a new mission operation concept for the EO-1 mission. An integration of model based reasoning with advanced ML in flight has been shown at NASA also on the Intelligent Payload Experiment (IPEX) for a CubeSat that flew in 2013-2015[3]. IPEX used several artificial intelligence technologies: machine-learned random decision forests to classify images onboard and computer vision visual salience software to extract interesting regions for downlinking acquired imagery.

Besides that, autonomy has been used also in other space domains such as multi-satellite constellations management and robotics. Areas of autonomous tasking, resources and path planning, autonomous feature extraction and data analysis have been then deployed in part on the AEGIS software on the Mars Exploration Rover (MER) Opportunity and the Mars Science Laboratory(MSL) Curiosity, as well as in the more recent MARS2020 mission. NASA is turning increasingly to autonomy and machine learning to make the most out of Mars exploration missions. In fact Perseverance, like its predecessor Curiosity, relies primarily on two radiation-hardened processors called Rover Compute Elements that have roughly the same processing power as a state-of-the art desktop computer from the mid-1990s. That is not enough processing power to perform complex machine-learning operation, but still, Perseverance has more autonomy than Curiosity thanks to an additional flight computer programmed to help the rover to land safely. AI/ML are in fact involved in almost all the steps of the mission: Descent & Landing (AI-enabled Terrain Relative Navigation - TRN), AI Improved Autonomous Navigation on the planet's surface, Task & Operations Scheduling (Adaptive Control), AI for Targeting Instruments (Goal Oriented Behaviour and Opportunistic Science), AI-powered payloads (Autonomous Exploration for Gathering Increased Science system – AEGIS, Planetary Instrument for X-ray Lithochemistry, - PIXL, and more).

ESA is contributing to this progressive adoption of AI/ML with various initiatives. Among the others we can cite the OPS-SAT Autonomous Experiment [7] and the Robotic Digital Twin (RobDT). The OPS-SAT Autonomous Experiment consists of an on-board planning and execution architecture deployed at ESA to test autonomous operations on the OPS-SAT mission, a 3-Unit CubeSat structure. This experiment constitutes one of the first attempts to the application of AI in ESA for on-board model based autonomy on a flying mission. Focusing point of this architecture is the on-board integration of model based reasoning with advanced ML applications of data analytic, to shorten the loop of data collection, analysis, decision making and control.

The ROBDT activity proposes a new framework where engineering methods and AI techniques are integrated into a coherent Robotic Digital Twin Framework in order to allow on-line update of the system models, planning and what-if analysis and plan monitoring and fault diagnosis. The use cases described in Section 4 have been implemented using and updating the control structure inherited from the RobDT activity.

The progressive introduction of AI Technologies in autonomous system, especially learning-based, significantly boosted research in the area of V&V for AI-based components. In fact, the literature is vast and the approaches are multiple and diverse, ranging from fully formal techniques for the verification of AI/ML models in isolation (with particular focus on Neural Networks) with symbolic approaches [1, 13], to works exploiting quantitative verification techniques in probabilistic settings [2, 15], to testing-based methodologies [9] and approaches tailored to system-level V&V of autonomous systems containing AI/ML components [6]. In relation to the VIVAS framework presented here, the closest works are those proposing methodologies based on simulation-based testing and scenario generation. In particular, two frameworks closely related to VIVAS are VerifAI ([14]) and Pegasus ([11]). Similarly to VIVAS, both frameworks are based on the generation of test scenarios from an abstract model of the system, which are then executed on a system-level simulator, using a run-time monitor for determining the test outcomes. The main differences with VIVAS regard the application domains (with VIVAS be-

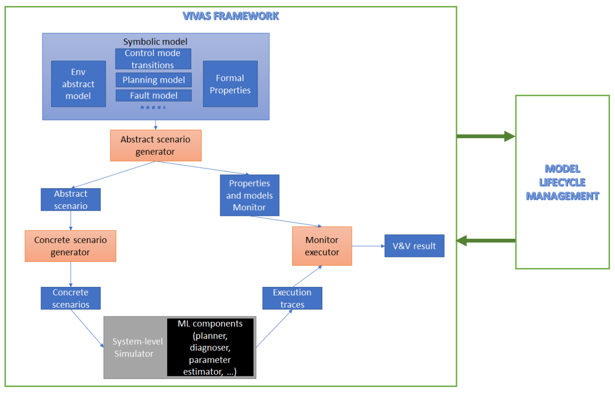

Figure 1. VIVAS V&V Approach

ing a more generic framework, whereas both VerifAI and Pegasus are specifically targeting autonomous driving) and the specific strategy for abstract scenario generation (where VIVAS uses automatic techniques based on symbolic model checking, whereas VerifAI and Pegasus use a combination of manual specifications and probabilistic sampling).

3. THE VIVAS FRAMEWORK

The VIVAS Framework is grounded on the V&V approach depicted in Fugure 1 . The V&V work flow is made of 4 main steps:

Abstract Scenario Generation. The scenario generation is the first step of the approach. The starting point is a formal, symbolic model of the system, which provides an abstract view of both the environment and the components under test (including AI/ML parts). Abstract test scenarios are generated from the formal system model using symbolic model checking techniques by the abstract scenario generator. Abstract scenarios are defined as combinations of values of predicates describing interesting behaviours of the abstract system. From the technical point of view, each abstract scenario is encoded as a formal property that is expected to be violated by the system (i.e. a property specifying that “the scenario cannot occur in the abstract system”). For each such property defined by the abstract scenario generator, a model checker will be executed on the system model, with the goal of finding a counterexample to the property. By construction, each such counterexample corresponds to an execution trace witnessing the realization of the abstract scenario of interest.

Concrete Scenario Generation. Each of the traces produced by the model checker is then refined into a (set of) concrete scenarios that can be used to drive the system-level (concrete) simulator. Ensuring an adequate level of coverage is one of the primary goals of a good set of tests. Although the specific criteria usually depend on the actual use-case application, VIVAS defines some general coverage criteria at abstract level, including coverage of the abstract scenarios with respect to the set of properties, coverage of the properties with respect to the abstract model and coverage with respect to a domain-specific notion of “interesting situations”.

Simulation. The objective of the system level simulator is to run a simulation of the target asset under the requested conditions and, at completion, to provide the execution trace. To this end, the models of all the subsystems are initialised at the provided state and the environment models are set at the given initial conditions from the concrete scenarios generated by VIVAS.

Execution Monitor Each concrete scenario produced is executed by the simulator, which generates a corresponding concrete execution trace. This trace is then used to determine whether (1) the concrete execution of the system satisfies the property of interest, and (2) the concrete execution of the system complies with the input abstract scenario (which defines the situation of interest for the current test). This is done by formally evaluating the trace with a run-time monitor that is automatically generated from the formal specification of the property and the abstract system model. The trace evaluation can have four possible outcomes:

- The trace complies with the abstract scenario (defining the situation under test), and it also satisfies the property: the test execution is relevant and the test passes.
- The trace complies with the abstract scenario, but it does not satisfy the property: this corresponds to a test failure on a relevant scenario, and it should be reported to the user.
- The trace satisfies the property, but it does not comply with the abstract scenario: this corresponds to a (good) execution in an unexpected situation, in which some of the assumptions defining the scenario might be violated. This might be due to imprecisions/abstractions in the symbolic model and in the concretizer, which might prevent the realization of the abstract scenario under analysis. This situation might be reported to the user, as it might suggest that a revision/refinement of the symbolic model might be needed.
- The trace violates the property and it does not comply with the abstract scenario: this corresponds to a test failure in an unexpected situation. Similarly to the above, it might be a warning that the symbolic model of the system is not precise enough to capture the situations of interest defined by the abstract scenario.

This V&V methodology has been implemented and tailored for a challenging scenario, a planetary robotic using 3DROV (see Section 4). This scenario, by its nature, lend itself well to the integration of different on-board ML models since it is characterized by various degrees of

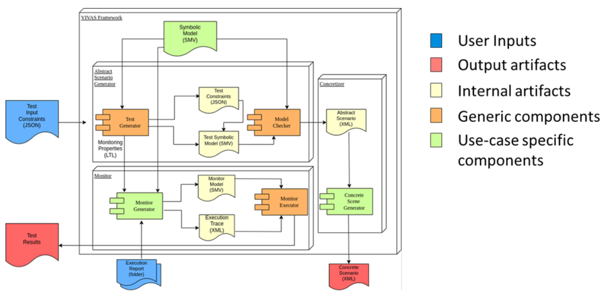

Figure 2. VIVAS Framework Architecture

uncertainty from its strong interaction with the environment in which it operates via impact, contact and sensing. We considered a typical scenario prepared for a 'sol' execution from the ExoMars planetary exploration mission: the 'Drilling site approach and surface sample acquisition'. In this scenario the rover moves over a grid to execute activities that shall be performed autonomously under the constraints of the available power and temporal constraints.

An overview of the framework implementation is provided in Figure 2 . It consists of the following main components:

Symbolic Model. The symbolic model defines the abstract representation of the system under test, i.e. the autonomous system and its environment. It is used for the generation of abstract test cases, covering (abstract) scenarios of interest. The model, which is necessarily use-case specific, is written as a symbolic transition system in the language of the nuXmv model checker [10]. For the selected use cases based on 3DROV, the symbolic model implemented consists of a combination of different modules:

- environment.smv models the operating environment of the autonomous system, including the type of terrain at the current position of the autonomous system in an abstract map, the time of the day and season of the year at the start of the mission, the evolution of air temperature and solar fluxes.
- planner.smv models the autonomous system at the logical level of mission activities, abstracting away the modelling of the electro-mechanical parts of the rover.
- estimator.smv models the logic for computing the resource consumption during the operation of the autonomous system. Approximations are used for more complex computations involving non-linear functions where needed, in order to keep the complexity of model checking under control.
- scenario.smv contains the definition of the monitor for the property used by the abstract test generation to obtain counterexample traces. This file is automatically generated from a high-level specification of the scenario under analysis, written in JSON, and described below.

Abstact Scenario Generator. The abstract scenario generator enumerates abstract test cases via symbolic model checking, based on the symbolic model and a test input file, specifying the space of the constraints for the abstract scenarios. More specifically, an abstract scenario is specified as a collection of N constraints, in which each constraint is defined by a domain (given as a list of predicates, evaluated in the symbolic model, that the abstract scenario must satisfy) and a type stating when the predicate must hold in the generated scenario (which can be initial, final, or intermediate). The generator then takes as input a file (in JSON format) with the definition of the constraints, and produces one abstract test per abstract scenario, enumerated by taking the NxM combinations of N constraints with M domain elements as defined in the JSON file. Each abstract scenario is then converted to a reachability property and injected in the symbolic model. An abstract execution trace is then computed via symbolic model checking.

Concrete Scene Generator. The concrete scene generator translates an abstract scenario to a concrete set of inputs for the system-level simulator. Due to the abstractions adopted in the symbolic model, such translation can be a 1:N mapping, in which a single abstract value might correspond to multiple concrete values (for example, in the 3DROV use cases implemented, locations of the rover are expressed as cells in a 3x3 grid, which might be mapped to multiple positions on the planet surface; another example is the orbital position in the symbolic model, which corresponds to a set of curves for temperatures and energy fluxes, with concrete values obtained via sampling). For the selected use cases based on 3DROV, the concrete scene generator produces a configuration file in XML format for the 3DROV simulators, consisting of the following sections:

- An activity plan request, containing the list of activities to be performed during the rover mission;
- The initial state of the environment and system, such as position in the map, time of the day, season of the year, initial battery charge;
- The evolution of temperatures and solar fluxes;
- The location of points of interest for opportunistic science.

Property Monitor. The property monitor checks the results of the executions to determine whether they pass the tests and maintains the database of test results and coverage information. Specifically, it performs two different checks on the execution traces of the system-level simulator: (1) Check of the satisfaction of some property of interest, to detect interesting test cases leading to property violations; and (2) Check that the execution traces comply with the abstract constraints used for scenario generation, to measure the actual coverage of the test suite in the concrete executions. From the implementation perspective, the monitor structure consists of a generic part and a use-case specific part. The generic part is based on the NuRV [5], which can generate runtime monitors from generic properties specified as formulas in Linear Temporal Logic (LTL) [12] over traces of symbolic transition systems. The use-case specific part has the goal of translating the specific simulator outputs into the generic input required by NuRV, consisting of a symbolic model, execution trace, and property to monitor. For the use-cases considered in VIVAS, the monitor verifies the following properties: (1) The requested mission plan can be successfully completed within the available power resources and (2) The novel objects present in the scene, which are within the vicinity of the rover during its normal operations for the mission, could be detected (See section 4). Finally, in order to check whether the simulated trace complies with the abstract scenario from which it originated; the monitor relies on a user-defined mapping that provides the meaning of the abstract constraints in the concrete simulation runs (i.e. it specifies what the symbolic model constraints mean in the real executions). With such mapping, defined in a JSON file, the monitor can automatically produce a "scenario compliance" property whose satisfaction is determined using NuRV.

4. TEST CASES

The use cases to show usage and advantages of VIVAS were designed targeting a challenging scenario in terms of autonomous capabilities and usage of AI/ML models.

4.1. 3DROV Simulator

To implement the use cases the 3DROV simulator has been adapted and interfaced with the VIVAS Framework. The simulator used consists of: (1) the rover models including platform and payload models (Platform models: GNC, ADE - Actuator Drive Electronics, Mast, DHS, Solar Panels, Power and Thermal; Payload models: Drill, RTB, CLUPI and Wisdom); (2) A component for the planning/scheduling of the on-board activities, which makes use of a high-level Rehearsal and Model service (RaaS/MaaS), integrating ML components, for estimating activity durations; (3) environment models (the orbital and timekeeping model, based on Naif/Spice; the Atmosphere model, based on the Mars Climate Database and the Terrain model visualising the planetary surface on which the rover operates). Additional objects of scientific interest can be included and placed in the scene in a configurable way.

The 3DROV simulator main components provide the following functions:

- The Rover model simulates the rover and payloads behaviours. It includes all rover subsystems, the payloads and the rover control logic reproduction. It aims at proving the feasibility of the commanded activity plans by reproducing the conditions expected to occur during its progress and by realistically identifying all the resources needed and available;
- The Environment model is in charge of the simulation of the environment which interacts with the Rover. This task encompasses real time reproduction of planetary and orbiter ephemerides as well as ground stations positions, real time provision of planetary atmospheric data, preparation of terrain related data, co-registration of test scientific data associated to local conditions to feed payload models;
- The Terrain model produces the input for the environment model creating a mesh of the terrain faced by the rover possibly enriched by test scientific data to be used by payload models. Terrain generation could be based on real data gathered on Mars or on fully/partially synthetic reconstruction. The complexity of the representation is tuneable with respect to the fidelity selected;
- The 3D visualisation environment shows the kinematics of the rover and its moving parts acting in the proper environment;
- The planning component is responsible for scheduling the rover activities necessary for achieving the mission goals. In order to estimate the activity durations and related resource consumptions, the planner interacts with the RaaS/MaaS components.

The simulator takes as input goal plans to be executed and the system initial state in XML format. It provides, in log files, traces of the evolution of each subsystem, the activities that have been executed as well as all the interactions with external components.

Using the 3DROV simulator, we considered a typical scenario prepared for a 'sol' execution from the ExoMars planetary exploration mission: the 'Drilling site approach and surface sample acquisition'. In this scenario the rover moves over a grid to execute activities that shall be performed autonomously under the constraints of the available power and temporal constraints.

4.2. ML Models

Two ML models have been integrated as shown in Figure 3: a "warm-up" model and a "novelty detector" model.

The "warm-up" ML model was deployed in within the RobDT project, to predict the duration of Warm-Up activities of the Actuator Drive Electronics (ADE). ADE shall be heated before activating the motors of the rover. As it is located at the external part of the rover, the duration of warm-up heavily depends on external atmospheric conditions and, as presented in the previous section, it is important from planning and what-if analysis. The model

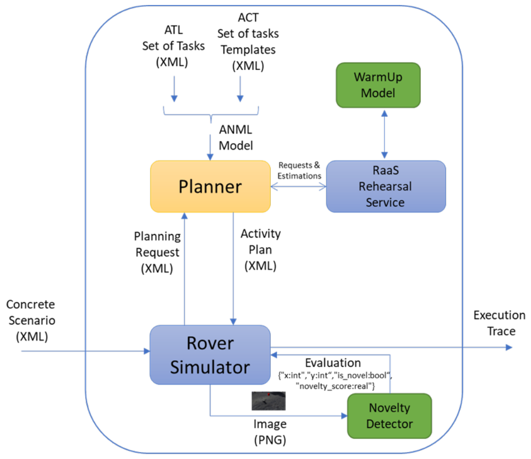

Figure 3. Autonomous System Architecture

inference is provided at planning time based on the location (longitude/latitude) the rover operates, the season in the year (Ls) and the time on the sol (Ltst) the given operation takes place.

The ”novelty detector” ML model provides functionalities to detect novel objects in images of the environment. The model has been designed and trained specifically for this activity using the 3DROV simulator. We used a Convolutional Autoencoder (CAE) architecture similar to previous activities performed at NASA for novelty detection on MAST Cam images for the curiosity rover. A structural similarity (SSIM) loss function has been used since according to NASA it can detect morphological novelties that are not detected by PCA, GANs, and mean squared error autoencoders. We carefully generated a trainset and validation set consisting of typical images only (no desired novelties) and a test set containing both typical and novel images to estimate the performances of the detector (at model level). The model inference pipeline run the novelty detector on each patch of an acquired image. The image is identified as novel, if any of the patch novelty score is greater or equal than the set threshold (0.25 to balance probability of false detection and probability of detection). Patching images not only enable to improve the local performances and increasing the amount of training data available, but it also enables to locate the most novel objects in the image. To locate it, the inference pipeline returns the central coordinate in the image of the patch which has the highest novelty score. Using the simulator, it is possible to project the image coordinate on the terrain map to identify its position and further plan the new science opportunity. Performances have been reported at image level for the ML-component tested in isolation. False Alarms: Given a normal image, the probability that the detector predict it to be novel is 0.16; Detection Probability: Given a novel obvious object, the probability that the detector predicts it to be novel is 0.8. The model inference endpoint is provided as RESTful API and invoked ar runtime during the simulation.

4.3. Use Cases

The 3DROV simulator, with integrtated ML models as shown in Figure 3, has been used to implement 2 use cases.

The first use case, namely the “Resource Allocation Estimation”, relates to the validation of execution of activities subject to uncertain duration and resource production/consumption estimated via the ”warm-up” model. From an ML V&V point of view, VIVAS has then been used to study the dependency and accuracy of the estimation of this model from environmental and operational conditions. The goal is estimate “how good” is the system using the ML prediction and its suitability for application. This estimation is done comparing the execution of plan optimized using predictions from the ML model with the result obtained by feeding the 3DROV simulator with a not optimized plan. At a system level V&V, System-Level properties considered for validation are:

- Mission achievement. Can the optimized plan be successfully executed in within the mission horizon and available resources under different environmental conditions? This is a pass/fail test.
- Resource allocation. Is the system properly predicting the time/resource necessary for the plan? Is the allocation congruent with an efficient use of the asset? This test compares the execution of the optimized plan with the execution of a nominal (not optimized) plan.

Tests are organized under different environmental conditions to study the adequacy of the system behaviour in different environmental conditions. Various combinations of environamneltal factors were considered, for instance time of the day and soil type.

The second use case, namely the “Novelty Detection in Images for Opportunistic Science” relates to the validation of a rover equipped with an ML model to detect interesting objects in the environment. The rover, while moving over a grid to perform science operations, acquires images and analyse them to identify novel objects, to possibly support opportunistic science. The purpose of this use case is twofold: (1) test that the system can properly detect interesting objects in the environment and (2) test that the system can properly handle this information.

To this purpose, the test cases are organized to verify that the V&V framework is able to generate scenario comprising of novel targets in unexplored terrain to assess the integrated novelty detector correctness (whether evaluated images are true positives, true negatives, false positive or false negative). More specifically, the system-level performances have been assessed with two metrics: Detection Probability (given a novel target in the vicinity of the autonomous system, what is the probability that the system will identify and report this region of interest as "novel"?) and False Alarm Rate (given that there is no novel target in the vicinity of the autonomous system, what is the probability that the system wrongly reports a new novel target of interest?).

To simulate a realistic scenario where a rover is visiting different places over its mission life, the zone being visited during the simulation is different from the zone used to train the novelty detector on typical terrain images, so it is considered unexplored. This leads to a distribution shift between the train and test dataset best reproducing a realistic scenario.

4.4. Results

Regarding the resource allocation estimation, tests were executed at different time of the day, with different illuminating conditions, with different sets of goals to achieve, involving moving the rover on the grid or only performing science in situ. The test results show a significant dependency of plan execution time and energy consumption from the time of the day. Comparing executions with and without ML optimization, we can observe that the costs of plans in poor illuminating conditions and/or not involving moving activities are evaluated correctly by the ML estimator, while the estimator appears overestimating warm-up times in some good illuminating conditions (not estimating correctly the power generated via solar panels) and for rover moving activities. In conclusion the test result showed that the estimation was adequate for sample collection activities but seems to overestimate time/resources when rover moving activities are involved, especially on good illuminating conditions. It is then suggested to re-train the model considering rover traverses. The time/resource estimation seems more accurate in poor illuminating conditions, probably underestimating the power generated by the solar panels in plain sun. It is suggested to check the estimation of the solar panels power generation.

Test are executed with different soil types show a dependency of drilling activities duration and energy consumption from the soil type. No differences have been observed between the execution of tests with or without the optimization with the ML estimator. We can then conclude that durations and power consumptions are correctly estimated for different soil types.

Regarding the novelty detection model, tests results show a system-level detection probability of about 0.55, lower than the model-level detection probability (calculated at 0.80 during model training and validation) and a system-level probability of false detection of approximately 0.04 (also in this case lower figure that what was reported at model-level, 0.16). Considering that these metrics should be taken with a word of caution given the limited size of the terrain used for demonstration purpose (the tests were organized more to show the capabilities of the VIVAS Framework in generating sets of diverse testing procedures than for actually providing significant results for the items under test), it was shown that uncertainties and inaccuracies that emerge at system level might have a significant impact on the performances of an ML model in operation with respect to the performances calculated in isolation.

A second testing campaign for this use cases has been implemented to test the capability of the autonomous system to support opportunistic science by: (1) identifying novel targets of interests, (2) stopping the activities and travel to the Region of Interest (RoI) identified, (3) carrying on analysis in the RoI and (4) going back on the original path and resume the original plan.

The testing campaign showed that the system seems able to support opportunistic science in within the boundaries tested (objects not too far and limited operational grid). Objects far from the rover trajectory are not detected, object detected are analysed and tagged. A problem was identified in terms of the actual trajectory followed by the rover when going to analyse the objects. For instance, an object "along the way but not on the way" of the rover is properly handled, while an object "on the way" of the rover is not efficiently handled. In that case, when the rover identifies the object in front on his trajectory, it moves there to analyse, but then it comes back to the position where the object was identified restarting the trajectory. Another problem identified was that the rover not always comes back to the original path after the analysis, recalculating a different path that might resulting in taking "shortcuts" that does not allow to visit all the planned locations.

This was only a reduced set of tests, once again aimed mostly at demonstrating the advantages in using VIVAS than at actually testing the system, hence the results cannot be considered conclusive nor exhaustive. But a recommendation that could be given from these tests is to check/optimize the trajectory of the rover to avoid erratic counterproductive moves when implementing opportunistic science.

5. CONCLUSIONS

In perspective, in the development of space system, the qualification of at component-level of subsystems encompassing AI/ML technolgies is a prerequisite for system-level qualification, hence AI/ML component should be handled like other space systems (hardware, software) components, and shall undergo a qualification campaign. The qualification shall be performed as part of the MLOps framework. For example: ML-robustness can be assessed on different slices of the dataset, and candidate models automatically trained, tested, qualified, and pushed to the qualified model registry. This first filter enables to serve to the next level of the V&V chain (subsystem or system-level) only with qualified models, i.e., models which comply with the ML-level requirements. The next level of V&V could be the subsystem

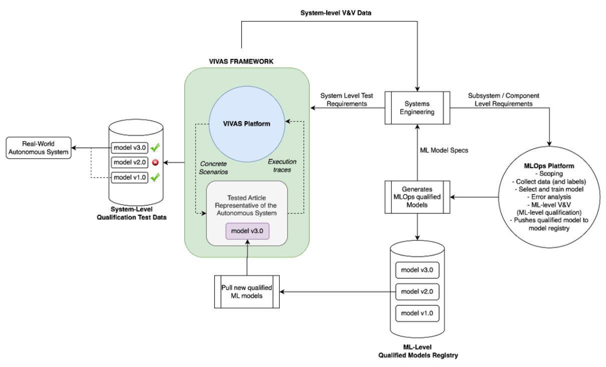
System-level V&V Data
VIVAS FRAMEWORK
VIVAS Platform
Concrete Scenarios
Execution traces
Tested Article Representative of the Autonomous System
model v3.0
System Level Test Requirements
Systems Engineering
Subsystem / Component Level Requirements
ML Model Specs
MLOps Platform
- Scoping
- Collect data (and labels)
- Select and train model
- Error analysis
- ML-level V&V
(ML-level qualification)
- Pushes qualified model to model registry
Generates MLOps qualified Models
Real-World Autonomous System
model v3.0
model v2.0
model v1.0
System-Level Qualification Test Data
Pull new qualified ML models
model v3.0
model v2.0
model v1.0
ML-Level Qualified Models Registry

Figure 4. VIVAS integration with Model Development Lifecycle

(for instance GNC) Integration, Verification and Validation (IV&V), followed by System-level IV&V.

The VIVAS project demonstrated the feasibility of a model-based approach to system-level validation and verification of autonomous systems integrating AI/ML components and proposed the VIVAS Framework, a general, domain independent V&V architecture that can be evolved and adapted to support qualification of AI/ML in operation, complementing the validation of the AI/ML model done when the model is designed/trained. In fact, with VIVAS the model can be tested considering interaction with other subsystems and the disturbances induced by such an integration, as well as the capability of the model to support system-level behaviours, leveraging the rigorousness provided by the model checking approach. The VIVAS framework would conceptually interface with the model development life cycle (see Figure 4), and with the space asset developed, in a similar way that Ground Support Test Equipment interact with articles under test in the AIT phase.

REFERENCES

[1] Aws Albarghouthi. Introduction to neural network verification. *Foundations and Trends in Programming Languages*, 7(1-2):1–157, 2021.

[2] Luca Cardelli, Marta Kwiatkowska, Luca Laurenti, Nicola Paoletti, Andrea Patane, and Matthew Wicker. Statistical guarantees for the robustness of bayesian neural networks. 08 2019.

[3] Steve Chien, Joshua Doubleday, David R Thompson, Kiri L Wagstaff, John Bellardo, Craig Francis, Eric Baumgarten, Austin Williams, Edmund Yee, Eric Stanton, et al. Onboard autonomy on the intelligent payload experiment cubesat mission. *Journal of Aerospace Information Systems*, pages 1–9, 2016.

[4] Steve Chien, Rob Sherwood, Daniel Tran, Benjamin Cichy, Gregg Rabideau, Rebecca Castano, Ashley Davis, Dan Mandl, Stuart Frye, Bruce Trout, and Seth Shulman. Using Autonomy Flight Software to Improve Science Return on Earth Observing One. *Journal of Aerospace Computing, Information and Communication*, 2(April):196–216, 2005.

[5] Alessandro Cimatti, Chun Tian, and Stefano Tonetta. Nurv: A nuxmv extension for runtime verification. In Bernd Finkbeiner and Leonardo Mariani, editors, *Runtime Verification*, pages 382–392, Cham, 2019. Springer International Publishing.

[6] Martin S. Feather and Alessandro Pinto. Assurance for autonomy – jpl's past research, lessons learned, and future directions, 2023.

[7] Simone Fratini, Nicola Policella, Ricardo Silva, and Joao Guerreiro. On-board autonomy operations for ops-sat experiment. *Applied Intelligence*, 52, 04 2022.

[8] ISECG. Autonomy Gap Assessment Report of the International Space Exploration Coordination Group. Available on the ISECG portal: `www.globalspaceexploration.org/wordpress/?page_id=811`, 2020.

[9] Chris Murphy, Gail Kaiser, and Marta Arias. An approach to software testing of machine learning applications. pages 167–, 01 2007.

[10] NUXMV. NUXMV Web Site. `https://nuxmv.fbk.eu/`, 2019.

[11] PEGASUS. Pegasus Symposium Web Site. `https://www.pegasusprojekt.de/en/`, 2016.

[12] A. Pnueli. The Temporal Logic of Programs. In *Proceedings of the 18th IEEE Annual Symposium on the Foundations of Computer Science*, pages 46–57. IEEE Computer Society Press, Providence, 1977.

[13] Caterina Urban and Antoine Miné. A review of formal methods applied to machine learning, 2021.

[14] VERIFAI. Verified Artificial Intelligence Web Site. `https://berkeleylearnverify.github.io/VerifiedAIWebsite/`, 2016.

[15] Matthew Wicker, Luca Laurenti, Andrea Patane, and Marta Kwiatkowska. Probabilistic safety for bayesian neural networks. In Jonas Peters and David Sontag, editors, *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, volume 124 of *Proceedings of Machine Learning Research*, pages 1198–1207. PMLR, 03–06 Aug 2020.