SPACE ROBOTICS SOFTWARE TESTING WITH THE HYBRID LOCOMOTION ROVER SHERPATT AND A VALIDATION TOOLSET

Virtual Conference 19–23 October 2020

Raul Dominguez[1], Florian Cordes[1], Malte Wirkus[1], Thomas Vögele[1]

[1]1DFKI Robotics Innovation Center Bremen, Robert-Hooke-Str. 1, 28359 Bremen, Germany:

{name.surname}@dfki.de

ABSTRACT

The paper summarizes the experiences and recent developments in the context of robotics software and hardware testing for space missions gathered by our team at the DFKI Robotics Innovation Center in Bremen. In space robotics, two types of testing take traditionally place: offline and field testing. The robotic platform SherpaTT and testing software which has been used to evaluate the performance of components developed by multiple institutions is presented. The rover SherpaTT provides multiple sensors, advanced mobility features and payload carriage capability, features which meet the most common needs for these tests. The validation software for development, integration, and performance analysis that we have developed targeting this type of testing is also described. Finally, gathered experiences in field testing on representative environments are presented. Stable software system is not just a requirement for space robotics [1] but also gain massive importance in terrestrial fields like rehabilitation [2] and even Human-Robot Interaction [3].

1 INTRODUCTION

A cornerstone in the development of stable software is thorough testing. The case of space robotics software is an exemplary case: Complex software systems, often jointly developed by multiple institutions are integrated, and their final quality must be accurately estimated and demonstrated in order to defend the quality in terms of robustness, efficiency and safety of the produced work.

Software testing in the robotics context can be divided in two main parts: Offline (lab) testing and field testing. In the context of the Strategic Research Cluster (SRC) on Space Robotics Technologies funded by the European Commission and guided by ESA and several space agencies in the member states, more than 20 institutions were involved in software development and both aspects of testing were covered. To support extensive field testing of the technologies developed in the first suite of interlinked R&D projects, the DFKI Robotics Innovation Center in Bremen deployed the SherpaTT (Figure 1). This hybrid-locomotion rover was used as a platform to test and validate various software- and hardware modules needed for the autonomous operation of a robotic rover on a planetary surface. This included a general space-robot operating system (developed in the SRC project "ESROCOS"), software for autonomous trajectory- and mission planning (result of the SRC project "ERGO"), software for perception and navigation (outcome of SRC project "InFuse"), and also a standard interface for the connection of physical space robot modules (the "SI-ROM" interface).

After preliminary tests in the labs of DFKI in Bremen, SherpaTT was shipped to Morocco for extensive field tests in the Moroccan desert. To support the tests with SherpaTT, a Validation Toolset was developed.

In this paper, Section 2 introduces the Rover SherpaTT, involved in field test activities since 2016. Section 3 provides a description of the software involved in the preparation and the execution of the field test campaigns. Section 4 presents the latest field test activity in the Dessert of Morocco and Section 5 concludes the paper.

Figure 1 The SherpaTT Rover with the HCRU engineered by DLR Institute of Robotics and Mechatronics mounted during field tests in the desert of Morocco.

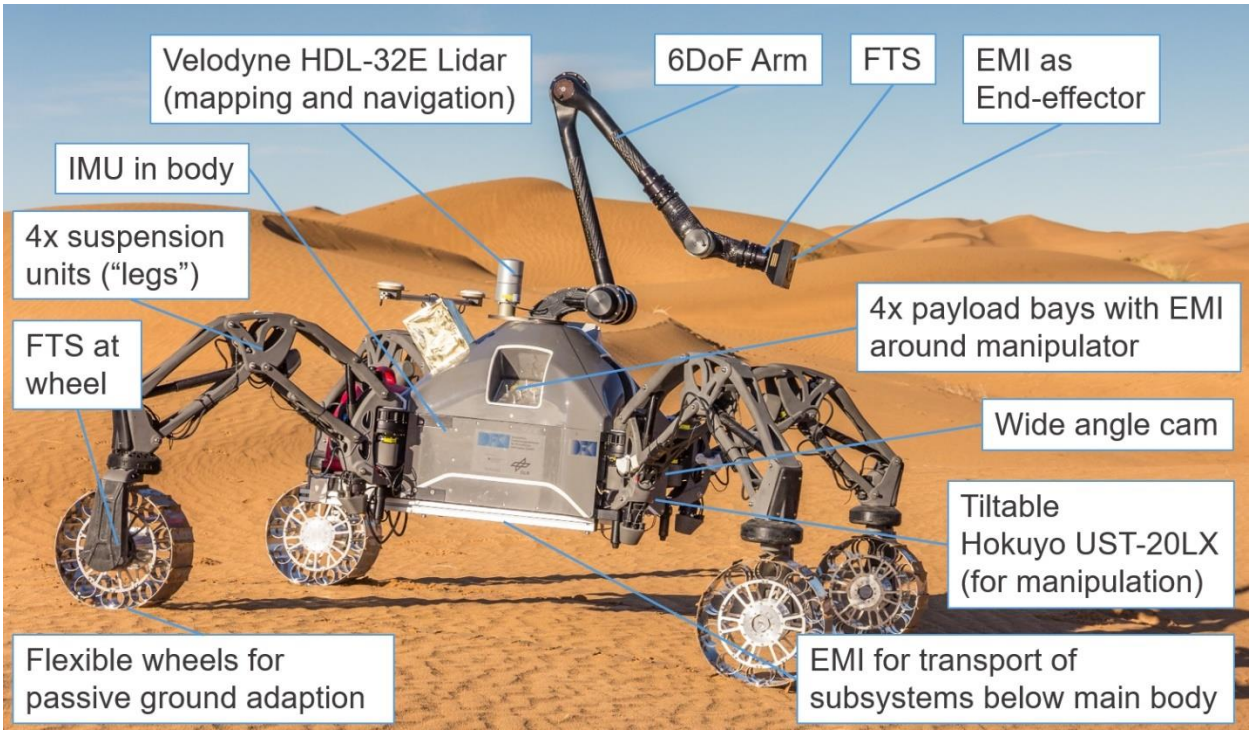

Figure 2: Rover system overview with indication of components.

2 SHERPATT

The robotic platform SherpaTT is a hybrid leg-wheeled rover with outstanding performance in locomotion capabilities [4]. SherpaTT has been designed to perform long traverses through unstructured, harsh environments (e.g. rocky surfaces). On the perception side, SherpaTT includes proprioceptive sensors (e.g. force torque at each wheel mounting) which can be used for state estimation and surface context identification as well as exteroceptive sensors (e.g. cameras) which can be used for environment reconstruction. Furthermore, the system allows for the mechanical integration of middle-size external modules that can increase computational, and sensing capabilities. In Figure 1SherpaTT is shown with a perception subsystem mounted for testing in a mars analog mission used in the project OG3-InFuse, in the follow-up project OG10-ADE a larger module, the Avionics Box, was mounted and is shown in Figure 4 (b).

The four-wheeled mobile robot is equipped with an actively articulated suspension system and a manipulation arm. The five limbs of the system add up to 26 active Degree of Freedom (*DoF)* in total, five in each of the four legs and six *DoF* in the manipulator arm. Apart from the active suspension system, a modular system approach with exchangeable Payload-Items (PLIs) is a key feature of the rover. Figure 2 presents a visual description of the hardware integrated in the rover.

Altogether SherpaTT offers what is needed to pose the required challenges for space robotics software and to collect the correspondent ground truth data for posterior accurate performance analysis.

3 VALIDATION SOFTWARE

Along with the rover, a comprehensive validation toolset aiming to assess the quality of the modules is under development. The Validation Toolset comprises on one hand a robotics simulator for planning and control tests, and on the other hand a perception framework, CDFF [5], which allows the replay of logged data for perception and state estimation tests. On top of these components, the Validation Toolset will perform automatic analysis of virtual missions. The simulation environments were generated from analog sites, which have been reconstructed with high level of accuracy (up to 1cm/px) using images from drone surveys. These environments are used also along with the DGPS data captured onboard of the rover as ground truth for the validation of the perception components.

3.1 SherpaTT Software

In the last test campaign, project results from different research projects were tested, each covering different areas of robotics and thus demanding different requirements of the validation system SherpaTT. In OG2-ERGO [6], for example, a robot autonomy control system was tested, which includes functionalities such as path and mission planning, guidance & control and decision making, but does not take into account the processing of sensor data for producing the map representations or localization capabilities that are needed to this purpose. The tests of OG3-InFuse, on the other hand, focused on the processing of specific sensor currents for mapping, object registration, recognition and tracking as well as data storage and analysis. Finally, OG1-ESROCOS tested a framework for the component-based integration of robotics software.

Based on the requirements of these individual tests, a set of software functionalities was provided for SherpaTT to complement the functionalities required for the respective tests. While OG3 only required access to various sensor data of the rover (D-GPS positioning, LiDAR, IMU and cameras), as well as capabilities for basic control of the robot (Motion Control System, MCS) [4, 7] and the possibility to tele-operate the robot from remote, OG2 required additional functionalities such as odometry, the creation of local and global maps and self-localization[1]. For this purpose, we mainly used existing software modules and integrated them with the framework Rock [8].

For the execution of the software from OG2, which was integrated by using ESROCOS [9], a separate PC, the OBC, was provided. The HCRU also provided a separate runtime environment for the software of OG3. The SherpaTT-API was developed for the communication between the hardware and software systems, which is explained in the following section.

3.2 SherpaTT API

Rover control is conducted by the Motion Control System (MCS) running in the Rock framework as described in Subsection 3.1. The MCS is interfaced by the modules to be tested through the *SherpaTT API.*

The API design consists of two parts, one being the regular *SherpaTT_API* and one additional, *SherpaTT_Sim_API*, being used to interface components mounted on the Rover System available initially only through the Simulator. The second API will not be used during the field deployment to control of the core hardware of rover, only its additions target of testing. Figure 3 illustrates the API design. The *SherpaTT_API* is used in both hardware as well as simulation usages. It provides the basic commands and telemetry channels for/from the system.

SherpaTT_Sim_API is used to interface in simulation with those components to be tested while the hardware parts are not completed and integrated. The parallel development of software and hardware has proven helpful for refining the design mainly of the software. The approach and parametrization that better suits the system can be identified and once the hardware is ready and integrated, the software requires only adapting the parametrization. Adaptations on the initial design of the hardware flow into the simulation and the software that uses it gets directly affected.

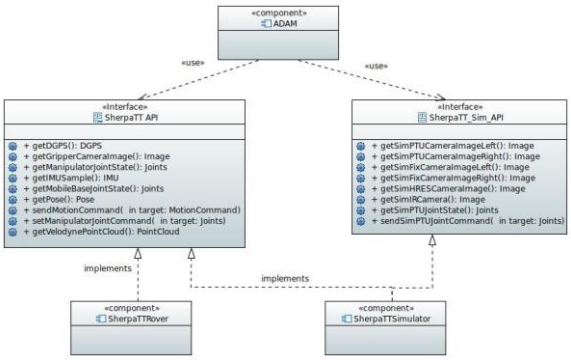

Figure 3 Layout of the SherpaTT API. The SherpaTT Sim API is used in combination with the SherpaTT API when using the simulation. For the final tests on the rover, the SherpaTT Sim API is not used.

3.3 Simulation

The software modules that are integrated in the simulator are grouped into three categories: 1) components which belong to core robotic simulation library MARS[2], 2) components that belong to the Rock framework, for wrapping the Mars library and for the Motion Control of SherpaTT, and 3) SherpaTT_API.

Mars, the simulation software uses in its core the Open Dynamics Engine (ODE) for the physics and the graphics. Mars includes an extended robot model loader, which takes a Universal Robot Definition Format (URDF) robot model along with additional information on the sensors and actuators to instantiate in the simulation the correspondent rover. The model has been developed using the open source plugin for

[1] SLAM3D Software: https://github.com/dfki-ric/slam3d/

[2] MARS (Machina Arte Robotum Simulans): https://github.com/rock-simulation/mars

(a)

(b)

Figure 4 (a) Starting from the CAD model design and the mechanical integration plan, the simulation model was generated and integrated with the robot model of SherpaTT.(b) After testing in simulation, the hardware module was produced and integrated.

Blender Phobos [10], which allows for the design and integration of the parts of the robot using a GUI.

All sensors and actuators are simulated. The sensors include 5 cameras, four force-torque sensors, LIDAR, IMU, GPS, and a thermal infrared camera. The avionics box, which is mounted on SherpaTT for testing and described in Section 2 has been integrated in the simulated model too. The resulting simulated rover is shown in Figure 4 (a).

The infrared camera component is currently in its initial version, the temperature values are generated from the real visual color values of the correspondent environment. Therefore, they are not realistic, but the produced values are in the range of values expected by the software that consumes them. So that a minimal level of integration is possible. In later iterations, more realistic temperature values are envisioned by either increasing the complexity of the simulated environment to include temperature or using Deep Learning for the generation of the simulated thermal images from normal images.

3.4 Simulation Modes

In the project OG10-ADE the Simulator is integrated in the Ground Control Station (GCS) for two reasons: Validation of software and on mission operation validation. These two use cases or modes share in its core, the same interaction routine between the operator and the simulation which is depicted in Figure 5. Basically, the GCS operator commands the simulated robot as the rover on a mission would be commanded, observes the behavior, and runs further operations or closes the simulator. The goal is to identify problems and to compare potential software improvements.

The Simulation Mode 1 (S1), is used for the integration and testing of the software from the different partners. S1 has the goal of identifying potential failures caused by errors in the code (bugs) as well for estimating the performance depending on different parameters values. In Figure 6 (a), the process of launching a simulation of type S1 is presented. For this feature to be complete the Avionics Box simulation model was produced and integrated with the SherpaTT rover model, *SherpaTT_API* was adapted, the simulated force torque sensors improved, the ground control adaptation checked with the improved force torque sensors and the new sensors (different cameras) and commands (PTU commands) were integrated.

The simulation mode S2 has the goal of identifying the potential outcome of an activity during a mission. This type of simulation is triggered by the operator on the

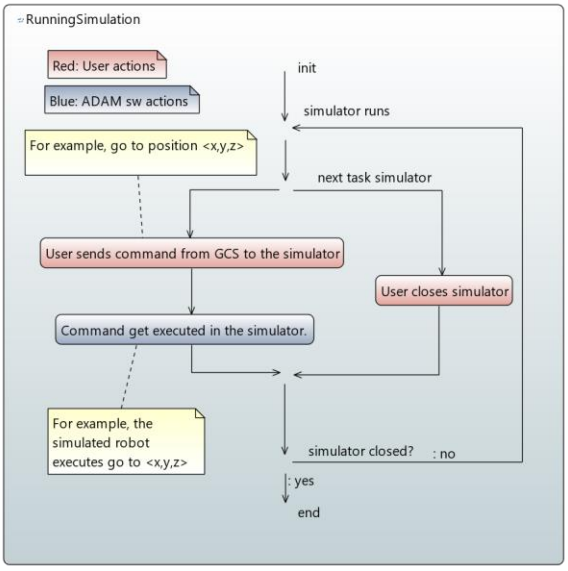

Figure 5 Workflow of interactions between the user and the Rover Simulator in OG10-ADE.

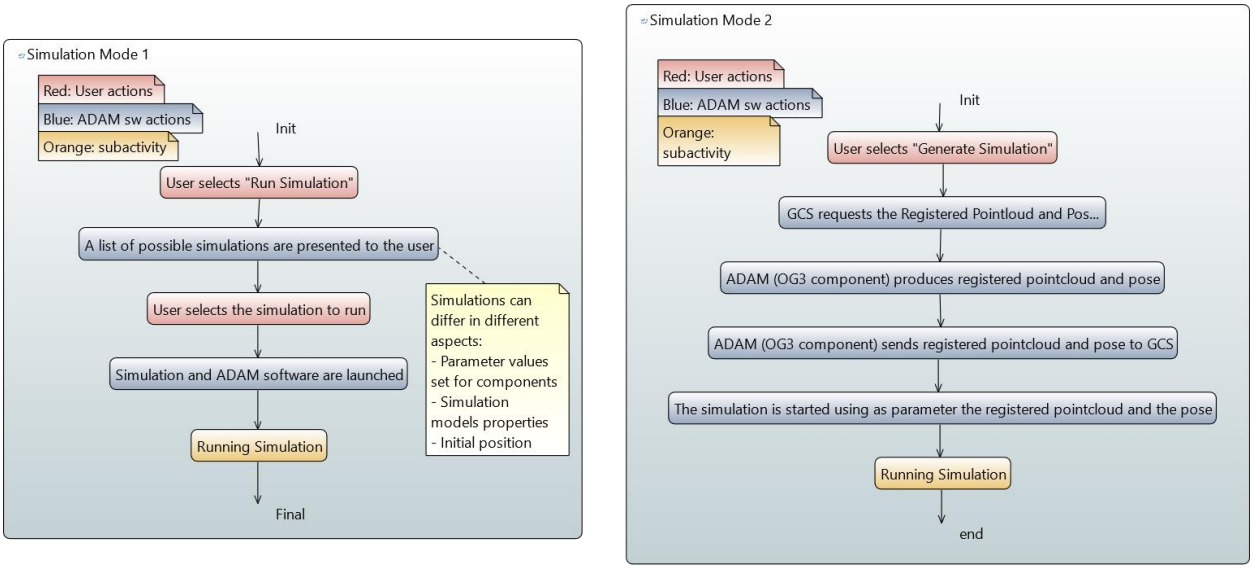

Figure 6 (a) Workflow of simulation mode S1. The user selects the specific simulation to be started. Each different simulation is characterized by an environment a robot model and several other parameters set to specific values. (b) Workflow of simulation mode 2. The user selects the option Generation Simulation which triggers the generation of a running simulation based on the information available to the robot about the environment and its location in it.

GCS when the particular outcome of an activity during the mission shall be validated before execution using the available fused data generated by the robot. In Figure 6 (b) the process of generating and running a simulation based on the data of the mission and the data collected by the robot is depicted as an activity diagram. For this feature to be available, the simulation core must be adapted to deal efficiently with the environment representation that the robot produces (registered pointcloud). Also, the interfaces to obtain from the rover in the GCS the map, position and pose must have to be implemented and tested.

In the GCS the status of all joints of the rover are retrieved through the *SherpaTT_API*, the position estimation and the DEM or registered pointcloud from the OG3 module. Once this data is received, a component merges the orbital map stored in the DataProductManager with the received DEM or registered pointcloud from OG3 and updates a robot state graph using the joint positions and the robot model. Finally, this class will provide these products (robot graph and fused environment representation) to the simulation.

3.5 Simulation of representative environments

Using the data collected by drone surveys (provided by ESA) on 4 different sites, correspondent simulation environments have been designed. The process involved the use of several open source libraries such as PDAL (for processing of the original point clouds), GDAL (for processing of the rasters that are used by the simulation) and other image processing tools and MARS plugins to ensure that the surfaces visual and physical dimensions of the simulation match and that simulated interaction between these and the rover are realistic enough to be useful for testing.

The four environments which have been developed are: 1) A first Morocco environment, where tests of OG2 and OG3 were performed, 2) A second Morocco environment, where data collection in OG3 was performed, 3) The environment of Colmenar where the first field test of the project is planned and 4) The environment of Fuerteventura, where the final field test will be performed. Figure 7 shows some of the mentioned environments, with the simulated rover in it.

3.6 Common Data Fusion Framework Development

For the validation of data fusion components, testing with data from field tests or even space missions are preferable to testing with synthetic data generated from the simulator. Since the data fusion components do not execute actions that affect the environment or the rover, there is no need to have a physical simulator engine supporting these interactions. For these reasons a replay mechanism, that can be connected to the data fusion modules is preferable. In the project OG3, such

Figure 7 Simulation environments produced from drone-survey generated point clouds. The simulated environments correspond to places were field tests were or will be performed in the context of the OGs.

mechanism was developed and tested and in OG10 it is as well been used.

The replay mode consists of the generation of data streams from existent logs so that software components can be connected to the streams and simulate a real execution. This process facilitates the debugging and evaluation of the components connected. Because the input data comes from field tests or past missions, it is highly realistic. It is worth noticing, that the replay mode is particularly suited for evaluating the performance of perception components and not for controlling or planning components. A useful feature to implement that makes use of logs from components is the automatic post processing of the logs for performance analysis.

The implementation of the connection of OG3 modules to replay sensor data streams is done using and extending the features implemented in the Development tools of the Common Data Fusion Framework software of OG3. The datasets produced are converted to an intermediate format (message packs) and then loaded in Python PANDAS data frames. These data structures can be then replayed and connected to the python wrapped OG3 components. The data products of the OG3 components, as well as the input data that these receive can be visualized. In addition, benchmarking software will provide insights on the computational requirements of the components.

3.7 Validation Toolset

To ease the process of evaluating the control and perception software capabilities and to help identifying errors in the earliest stage possible, a validation toolset will be implemented. The software will provide a database in which to register: 1) the logs captured, 2) the simulation environments, 3) the missions that can be performed in these environments, 4) the tests scripts which launch such simulated missions as well as test scripts to launch replay-based tests, and 5) the results of such tests.

An initial database Entity-Relationship model diagram is provided in Figure 8. The results will be generated automatically thanks to the supervision tools, but they can be extended by a human user.

The development of the toolset will be done independently of the other components of ADE to be able to demonstrate its functionality before the other software components are available. Its application will be limited to navigation functions and perception components. To provide a good evaluation of the simulated rover in terrain with different characteristics, separated tests with various inclinations will be included.

Logs from previous field test will be integrated for performance evaluation of the perception components and simulation environments based on ground truth maps too. Finally, the scripts to launch the different tests, produce and store the results will as well be included, so that the whole process of performing a full analysis of a system can be automatically launched.

In order to produce a valuable automatic performance analysis of the execution of a mission in simulation, supervision software will be developed to collect performance information during execution. These are some examples of the information which will be collected: duration of the execution mission, deviation between planned trajectories and executed trajectories, error in the localization, failure in the traverse of a path and collisions between arm and other parts of the robot. In addition to this, it will be pursued to detect

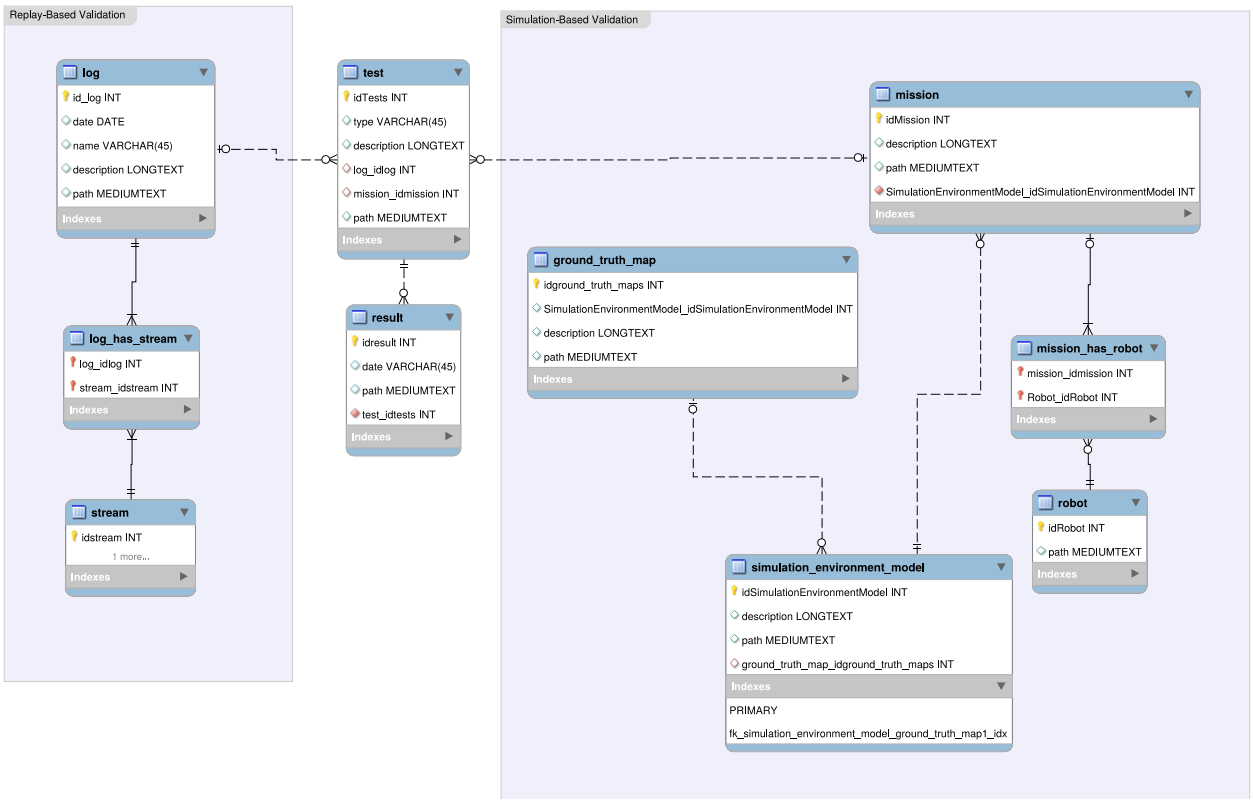

Figure 8 Design of the Validation Toolset database. A database and the correspondent management layer will be implemented to maintain and automatically launch the different tests based on simulated missions or replayed datasets.

cases of slippage, blockage, collisions, and reorientation events.

Validation against ground truth makes more sense indeed using the replay mechanism than the simulation. In that case, the ground truth is GPS, to validate position estimation. The simulation-based validation tests make more sense for analysis of the performance of a mission. There, a complete trajectory can be executed with different parametrizations and the comparison between parameter sets can be produced. Once the tool is developed it can be used in upcoming projects also to compare the performance of different versions of the navigation stack.

4 FIELD TESTS

From the beginning, the R&D project initiated as part of the Strategic Research Cluster on Space Robotics Technologies had a strong focus on field-testing of software and hardware. Although analogue missions on Earth will never be able to simulate the full spectrum of environmental conditions encountered on extraterrestrial planetary surfaces, they still provide a valuable – and probably the best currently available - tool to put both software and hardware to the test. The benefit of such missions is that they are able to evaluate how not only individual SW and HW modules, but also fully integrated systems can cope with the challenges imposed by realistic operating conditions. Only full-scale field testing can reveal conceptual flaws and potential problems of the robotic system interacting with a complex, unstructured and un-controlled natural environment.

To test the results of the OG2 and OG3 projects, extensive field tests were organized in November and December 2018 in a desert region in south-eastern Morocco. The site was chosen because the natural topography is similar to the topography of some regions on Mars and because the climatic conditions in Morocco are favorable of outdoor tests. In addition, the touristic infrastructure available in this part of Morocco proved to be of great value for the logistics of the mission and the Moroccan authorities exhibited a generally positive and supportive attitude towards the project.

The main objective of the Morocco analogue mission was to prove that the OG2 and OG3 software modules were able to support a fully autonomous long-distance traverse of a planetary rover in difficult terrain. The

SherpaTT rover was used as a hardware platform in the tests. The tests were successful as the SherpaTT, guided by OG2 and OG3 software, was ultimately able to cover in fully autonomous mode a distance of more than 1,6 km.

5 CONCLUSION

Development and testing of space robotics software involves the following: 1) Robust and reliable robotic systems to perform the physical tasks aimed to be controlled by the tested software, 2) Reliably working software to guarantee the requirements on which the software to be tested requires (e.g. locomotion is required for autonomous navigation), 3) Tools for accelerating the development and the optimization and 4) Planetary analog environments where to demonstrate the capabilities and identify limitations. This document has presented how these four elements have been successfully provided for testing in several projects.

The robotic system SherpaTT, its integrated software with all requirement for testing autonomy and perception software, the simulator -for developing and on mission testing- and the tools for development using logged datasets have been presented. Furthermore, the ongoing development of the Validation Toolset, an additional component to validate through automatic regular testing in virtual mission and with dataset from previous field missions, has been presented. Finally, the last test campaigns where a representative environment was selected and in which the rover SherpaTT satisfied all requirements for the final tests was presented.

Acknowledgement

We would like to thank the European Commission and the members of the PERASPERA program support activity for their support and guidance in the ADE activity. This project has received funding from the European Union's Horizon 2020 Research and Innovation program under Grant Agreement No 821988.

References

[1] *Locomotion modes for a hybrid wheeled-leg planetary rover.* Cordes F., Dettmann A., Kirchner F. In Proceedings of the IEEE International Conference on Robotics and Biomimetics, pages 2586-2592 (2011)

[2] *The capio active upper body exoskeleton and its application for teleoperation.* Mallwitz M., Will N., Teiwes J., Kirchner E.A. In Proceedings of the 13th Symposium on Advanced Space Technologies in Robotics and Automation. ESA/Estec Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA-2015), Noordwijk, European Space Agency (ESA) (2015).

[3] *Intrinsic interactive reinforcement learning - Using error-related potentials for real world human-robot interaction.* Kim, S.K., Kirchner, E.A., Stefes, A., *Kirchner F.* Sci Rep 7**,** 17562 (2017). https://doi.org/10.1038/s41598-017-17682-7

[4] *Design and Field Testing of a Rover with an Actively Articulated Suspension System in a Mars Analog Terrain.* Cordes F., Kirchner F, Babu A. In Journal of Field Robotics, Wiley, volume 35, number 7, pages 1149-1181 (2018).

[5] *Common Data Fusion Framework: An open-source Common Data Fusion Framework for space robotics.* Dominguez, R., Post, M., Fabisch, A., Michalec, R., Bissonnette, V., & Govindaraj, S. International Journal of Advanced Robotic Systems (2020).

[6] *The ERGO framework and its use in planetary/orbital scenarios*. Ocón, J., Colmenero, F., Estremera, J., Buckley, K., Alonso, M., Heredia, E., Schach, A. In Proceedings of the 69th International Astronautical Congress (IAC) (2018).

[7] *Static Force Distribution and Orientation Control for a Rover with an Actively Articulated Suspension System.* Florian Cordes, Ajish Babu, Frank Kirchner. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS-17), 24.9.-28.9.2017, Vancouver, BC, IEEE/RSJ (2017).

[8] *Modular Software for an Autonomous Space Rover.* Sylvain Joyeux, Jakob Schwendner, and Thomas M. Roehr. In Proceedings of the 12th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2014), Montreal, Québec, Canada (2014).

[9] *ESROCOS: Development and Validation of a Space Robotics Framework.* Miguel Munoz Aracon, Malte Wirkus, Kilian Hoeflinger, Nikolaos Tsiogkas, Saddek Bensalem, Olli Rantanen, Daniel Silveira, Jerome Hugues, Mark Shilton, Herman Bruyninckx. In Proceedings of the 15th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA 2019), (ASTRA-2019), 27.5.-28.5.2019, Noordwijk, European Space Agency (ESA) (2019).

[10] *Phobos: A tool for creating complex robot models.* von Szadkowski et al. Journal of Open Source Software, 5(45), 1326 (2020)