

Rover Flight Software for the VIPER Mission. H. Utz, L. Fluckiger, NASA Ames Research Center, Moffett Field, CA. (Contact: hans.utz@nasa.gov)

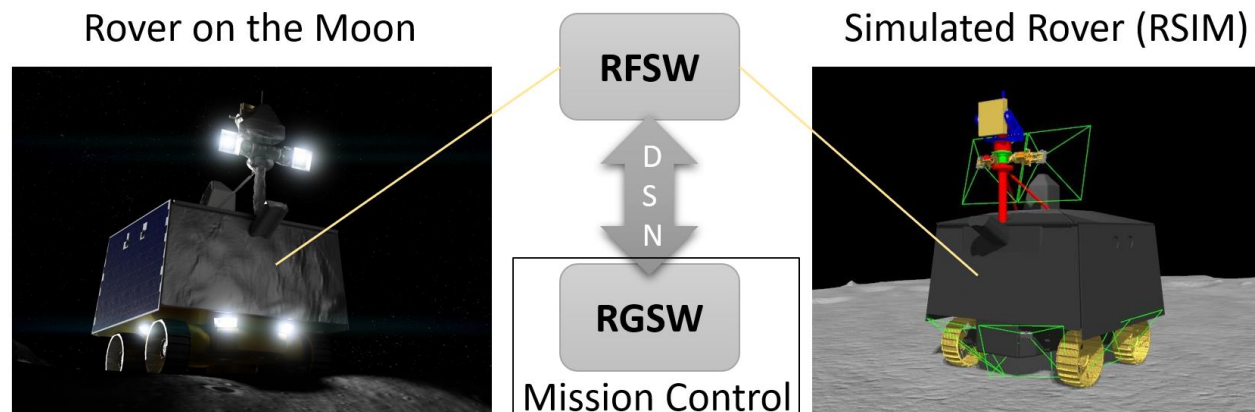
Introduction: The Volatiles Investigation Polar Exploration Rover (VIPER) mission objective is to characterize the surface and subsurface water ice at the lunar south pole. For that purpose the VIPER mission will send a solar powered mobile robot to the Moon in late 2023 as a payload on the Commercial Lunar Payload Services (CLPS) flight provided by Astrobotic's Griffin lander. The rover is designed to traverse up to 20 km during a mission lasting up to four lunar days while carrying a suite of instruments (three spectrometers and a drill).

The VIPER mission exhibits a unique set of operational and environmental constraints, which provide opportunities as well as impose unique challenges for a robotic surface mission. VIPER Rover Software architecture and design targets these constraints in innovative ways to meet the mission's technical and operational requirements. In this presentation we describe the design of VIPER's rover software and present early results from its realization.

RSW Architecture Split: The rover is designed to operate in continuous communication via a "Direct-To-Earth" low latency link (below 10s one way). The VIPER Rover Software (RSW) architecture takes advantage of this unique configuration by splitting the rover's software between the onboard Rover Flight Software (RFSW) and off-board Rover Ground Software (RGSW). RFSW runs onboard the rover on radiation hardened (RAD 750) and radiation

tolerant (AiTech SP0-S) avionics. RFSW is based on NASA's Core Flight System (cFE/cFS architecture). It implements classic space craft functions such as low-level hardware interfacing, basic error checking, and device/payload services. It's mobile robotics functions are limited to basic mobility control, relative pose estimation and waypoint driving. RGSW in contrast runs at mission control on Earth using commodity desktop computing. RGSW is realized as a set of Robot Operating System 2 (ROS2) nodes and implements typical mobile robotics functions such as global pose estimation, mapping and navigation, realized as advanced driver assist functions (ADAS) for human robot operators.

VIPER Simulation Capability: The 1/6 gravity target environment makes fully integrated testing of the actual flight rover in Earth gravity prohibitively complex and costly. Furthermore, the aggressive mission schedule (i.e., just over 4 years from project start to launch) severely limits the availability of HW for software development and testing purposes. To address these constraints, VIPER RSW has placed significant emphasis on creating and using virtual end-to-end high fidelity tests on developer machines and continuous integration agile development. As part of this approach, VIPER has developed lunar Rover Simulations (RSIM) as a set of plugins to the Gazebo simu-



lation environment. Gazebo is a robotics simulator featuring a physics engine as well as OpenGL based rendering. RSIM models all the rovers sensors, actuators, cameras and science instruments at the protocol level, connected via virtualized serial ports to RFSW. This provides a fully-featured proxy (i.e., stand-in) for the rover hardware and will also be utilized for full-scale mission test and training prior to lunar landing. The flight computers can be furthermore emulated using the processor simulator Qemu.

RSW Development Environment: The mission timeline required a rapid, proactive software development approach, with rework expected due to requirement updates from hardware deliveries, as well as implementation activities started before finalization of the interface control documents (ICD's).

The RSW team targeted these challenge through an iterative, agile development approach as well as via infusion of modern software technologies into NASA's Core Flight System (cFS/cFE). For instance, RFSW apps are implemented in C++ using a small C++ framework to encapsulate the core cFE services and provides an event-based programming model. This approach allowed each RFSW app to have generic functionality that could be adapted centrally to meet system-level requirements. In addition, app interfaces with the ground, as well as with hardware devices, are defined using the XCTE interface definition language. These definitions provide a single source from which user documentation, ground interfaces and header files for RFSW and RSIM and other representations are generated.