Development of UCLA ELFIN CubeSat Mission Operations Software

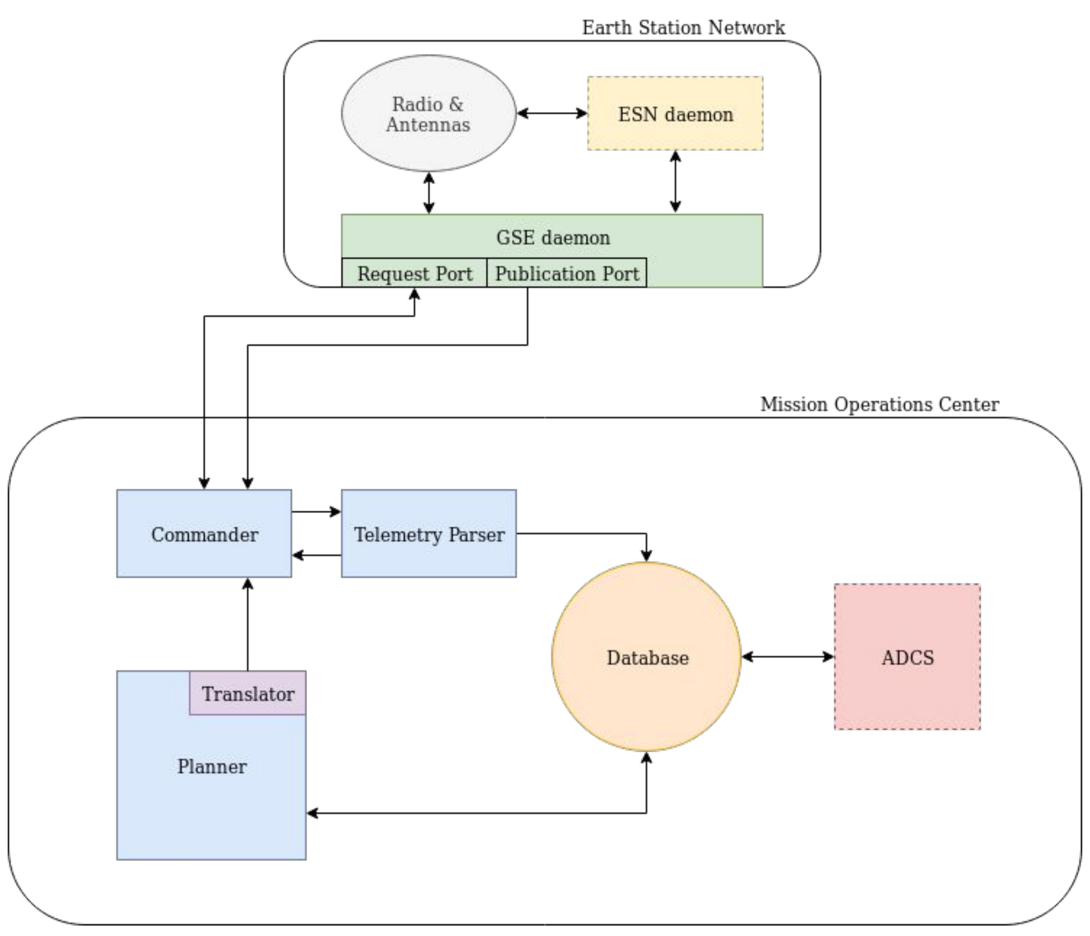
Jason Mao (jmmp8@ucla.edu), Vassilis Angelopoulos, Ethan Tsai, Akhil Palla,
Daniel Schwartz, Chanel Young, Sharvani Jha, Austin Norris, Andrew Evans, Koji Kusumi

Abstract

The Electron Losses and Fields Investigation (ELFIN) is a student run cubesat project at UCLA with a pair of spin stabilized 3U+ cubesat currently in LEO, each using three custom scientific instruments to study precipitating electrons and correlating them with EMIC waves. Due to heavy constraints on instrument commanding and power consumption, most of the flight software is necessarily custom made and low in computational power; prebuilt control systems are incapable or ineffective at interfacing with the low level procedures of the flight computer, so most of the ground software is analogously custom.

ELFIN ground control software is a suite of python modules working in tandem to assist satellite operators by turning logical commands into radio messages for the flight computer as well as receive, parse, and store satellite responses. A high degree of automation is also included to accommodate for the varying availability of student operators. We present the design, methodology, and lessons learned for future CubeSat applications.

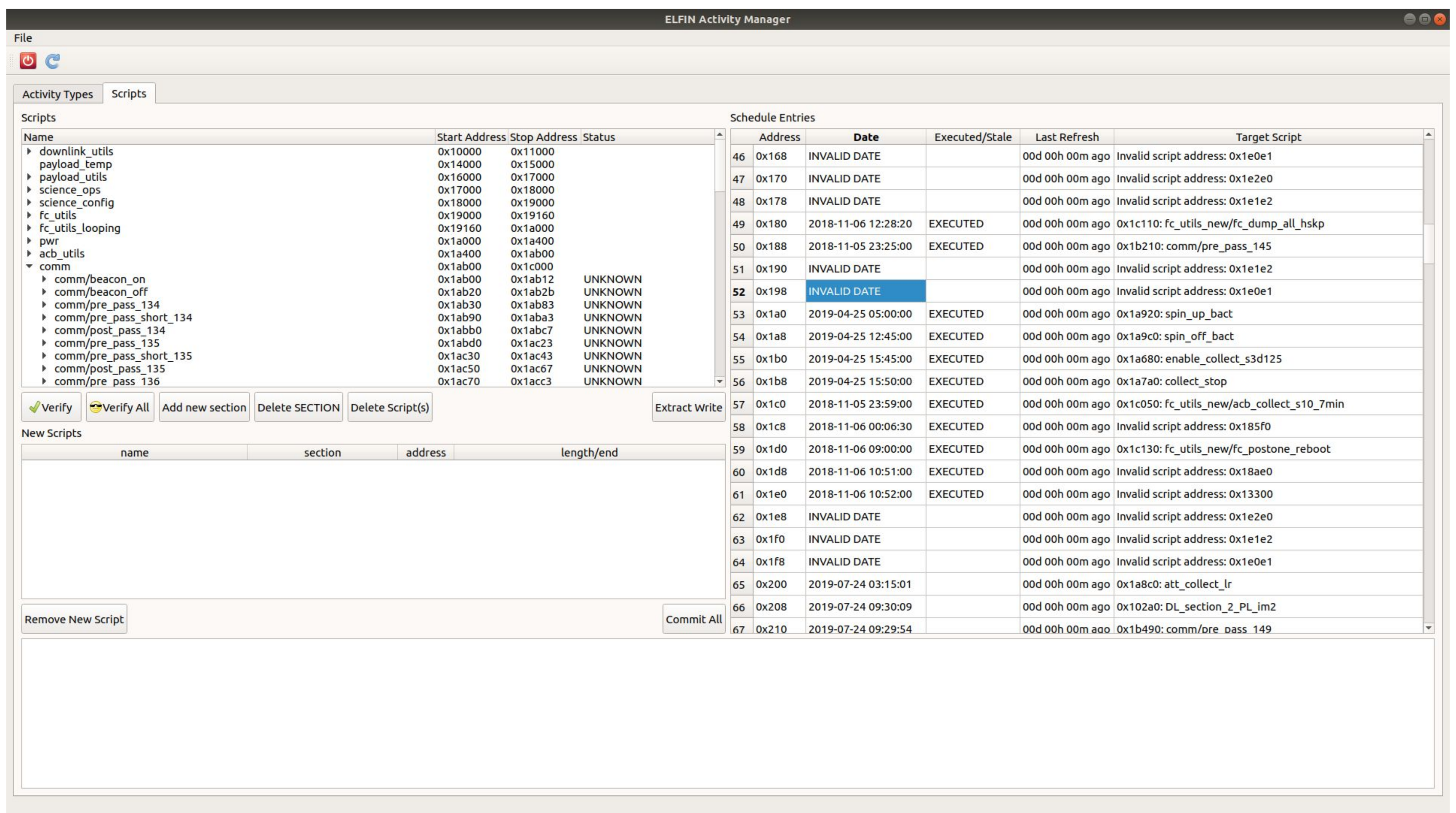
At a Glance

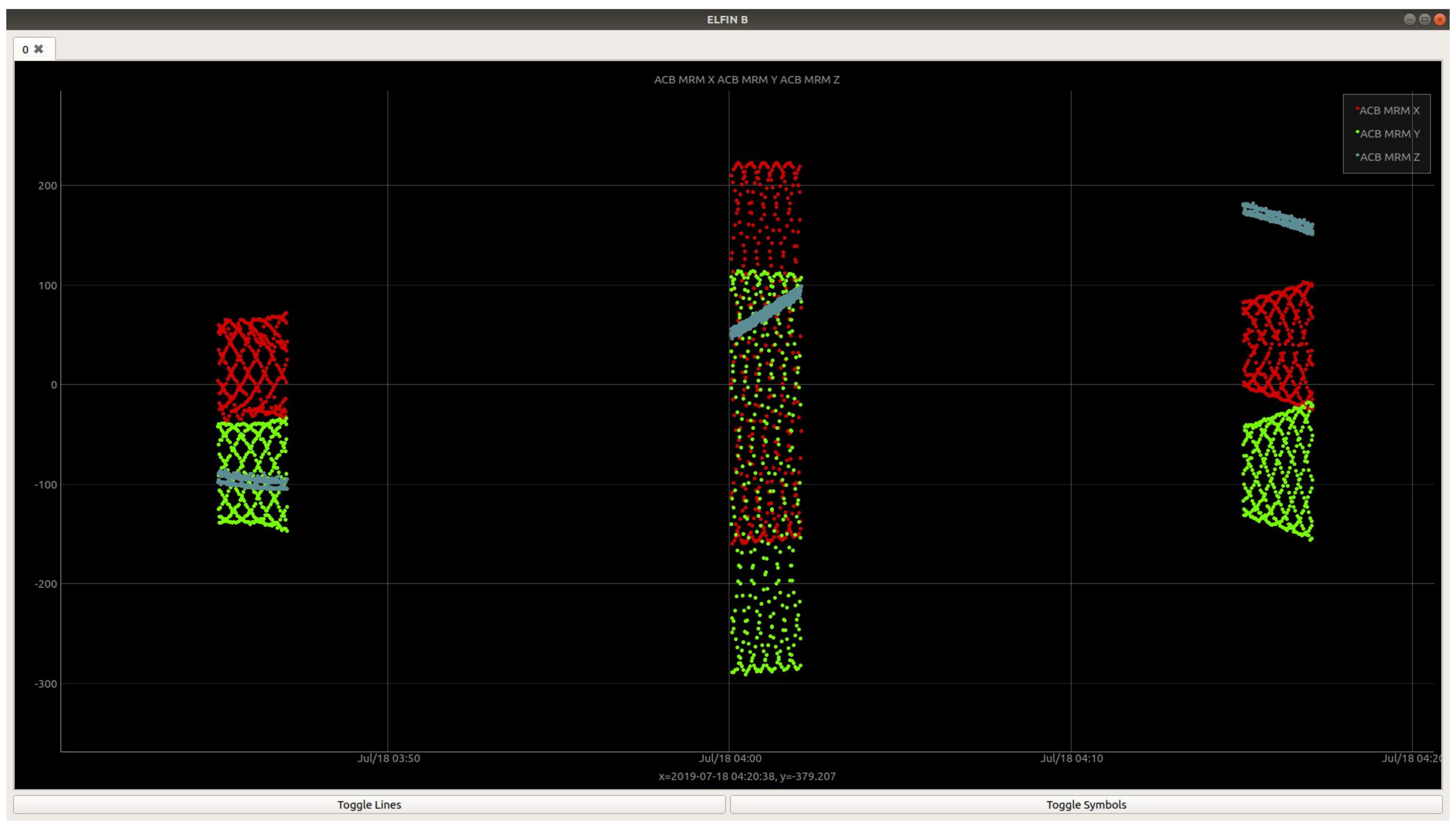


ELFIN software diagram. ADCS and ESN segments are condensed to blackbox representations. Two instances of the Commander, Telemetry Parser and Planner are used concurrently - one for each of the two ELFIN satellites.

- User Interface for commanding and planning
- Server-side command automation
- Real-time telemetry parser and graphing
- Central database for storing state and planning information
- Interconnection with ADCS and ESN segments for automation and maneuvers
- Server published state information to ensure consistent operations across multiple modules

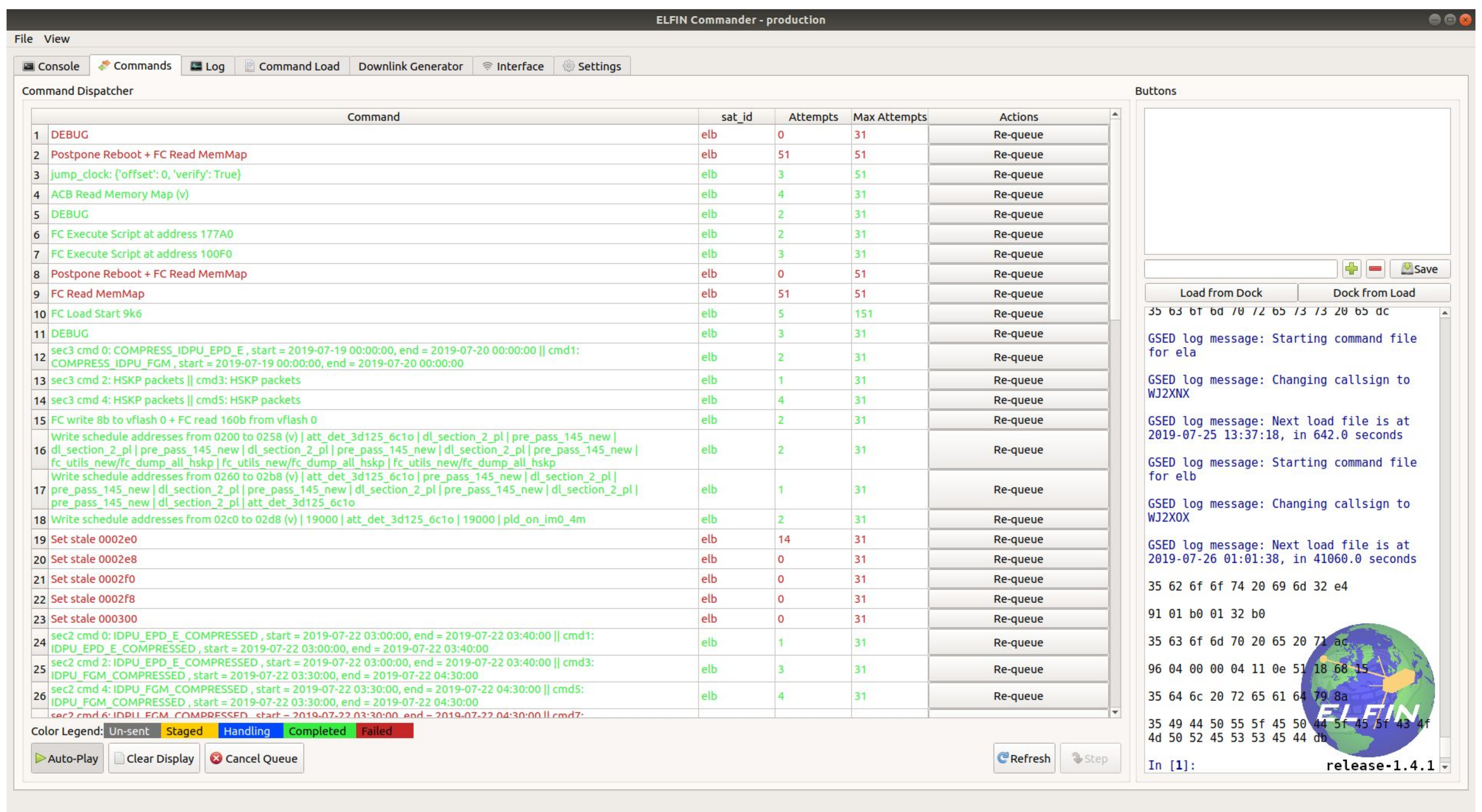
Telemetry Information





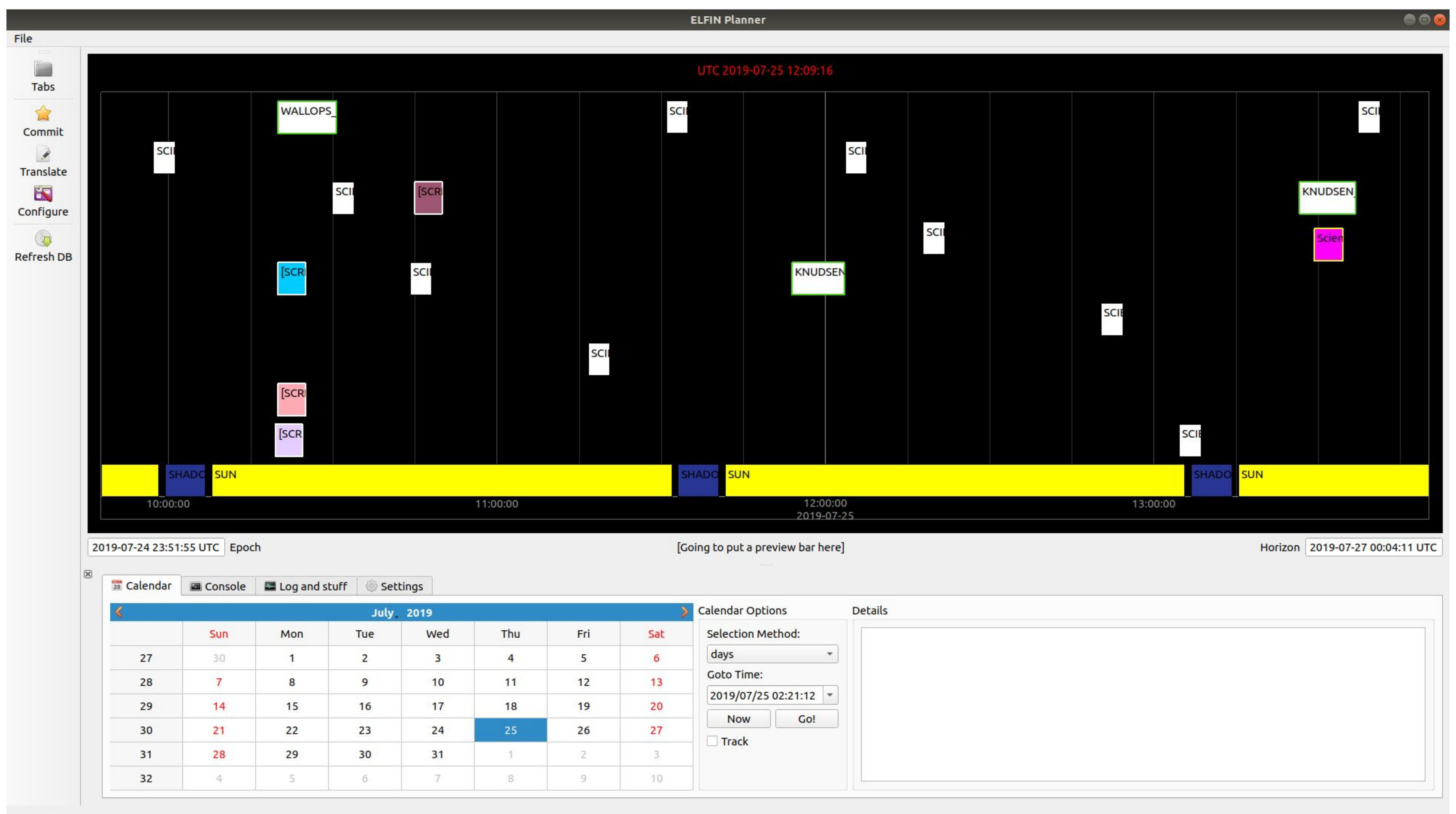
Above: State information displaying state on on-board scripts and scheduled executions. Updated in real-time.
Below: Plotted magnetometer data from the Attitude Control Board (ACB) MagnetoResistiveMagnetometers (MRM).

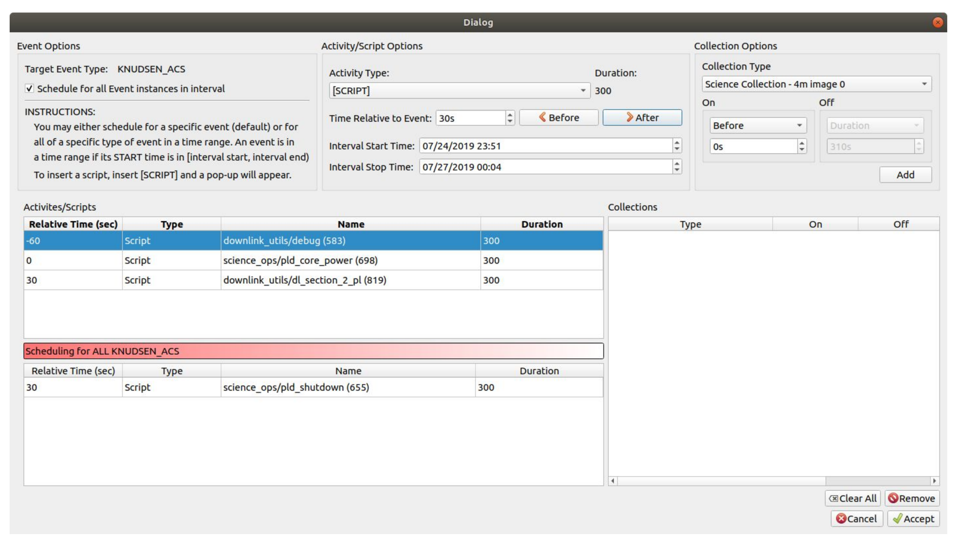
Commanding Interface



List of currently and previously sent commands. Commands may be constructed and queued for sending via the embedded python terminal. The commanding interface also contains elements to modify ESN/GSE daemon server settings

Planning Interface



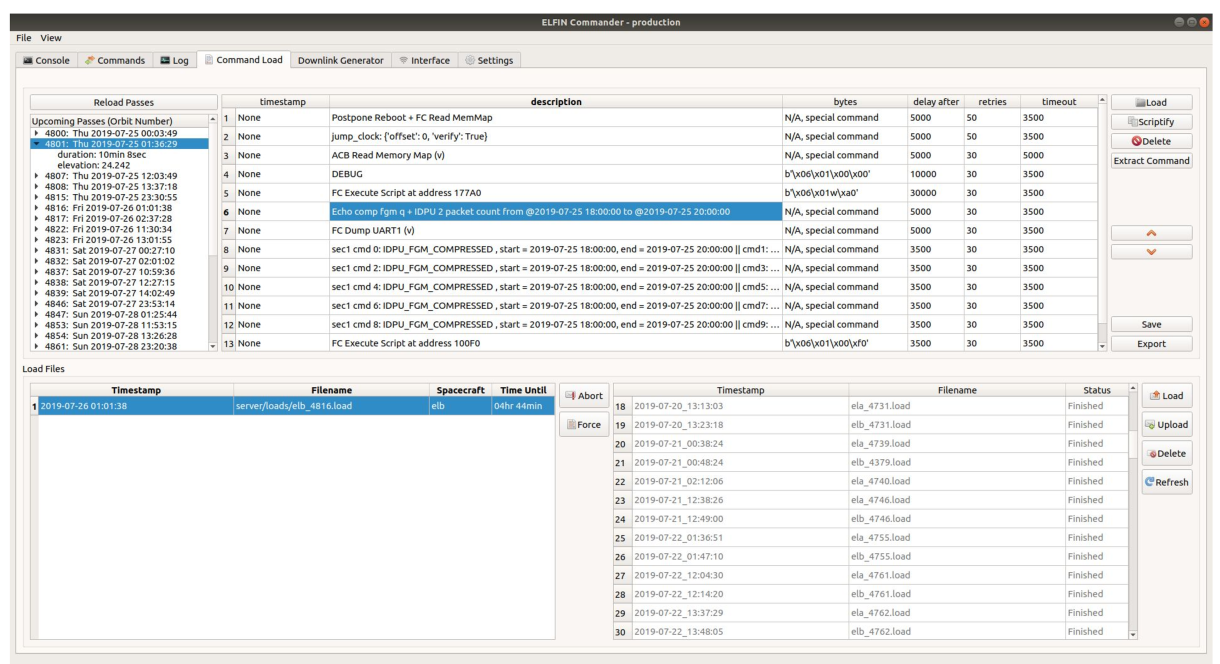


Long term planning via inserting downlinks, data collection, script executions, or ADCS maneuvers into the interactive timeline displayed above. Multiple planned activities may be added relative to specific orbital events.

A translation module turns these logical activities into direct satellite commands.

Automation

Command automation is achieved by pre-uploading a timestamped list of commands to the GSE daemon server. The commands will be parsed and executed reaching the timestamp, corresponding the start of a communication pass.



Future Considerations

Further improvement to the software include more advanced and automated planning capabilities. Specifically, constraint checks on filesystem space, collection processing to downlink pipelines, and de-conflicting collection times.We hope that developers of mission operations software for future missions may take into considerations the methods we used for automation, planning, and command handling and apply them to their own challenges.

Contact
ELFIN: https://elfin.igpp.ucla.edu/
Author (Jason Mao): jmmp8@ucla.edu