Development of UCLA ELFIN CubeSat Mission Operations Software

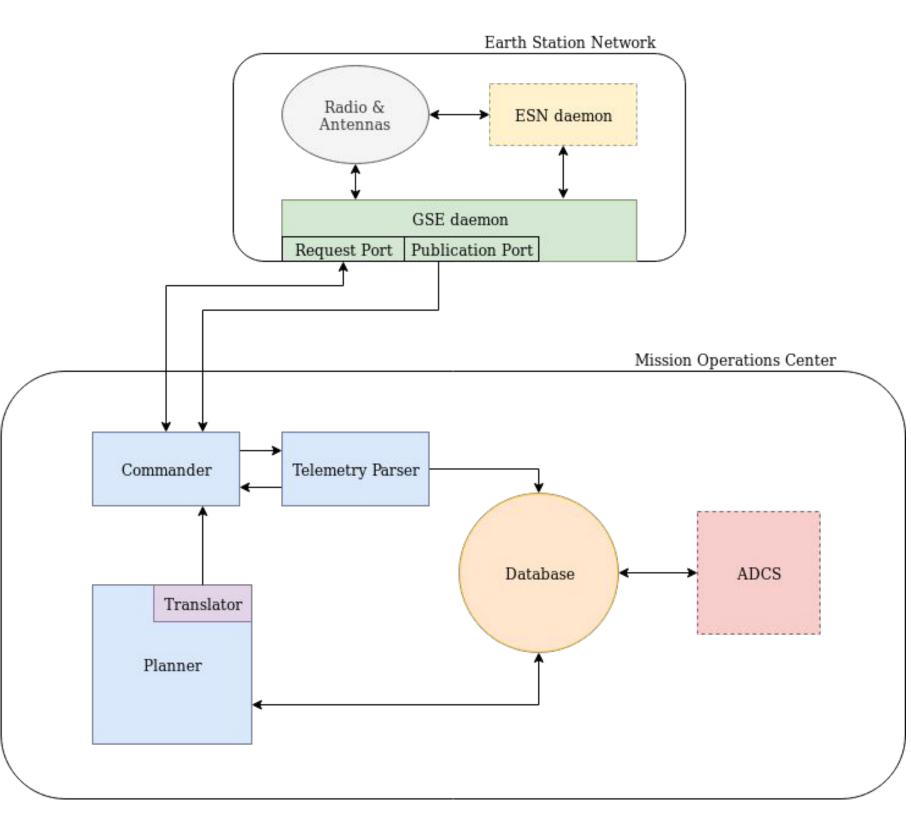
Jason Mao (jmmp8@ucla.edu), Vassilis Angelopoulos, Ethan Tsai, Akhil Palla, Daniel Schwartz, Chanel Young, Sharvani Jha, Austin Norris, Andrew Evans, Koji Kusumi

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.

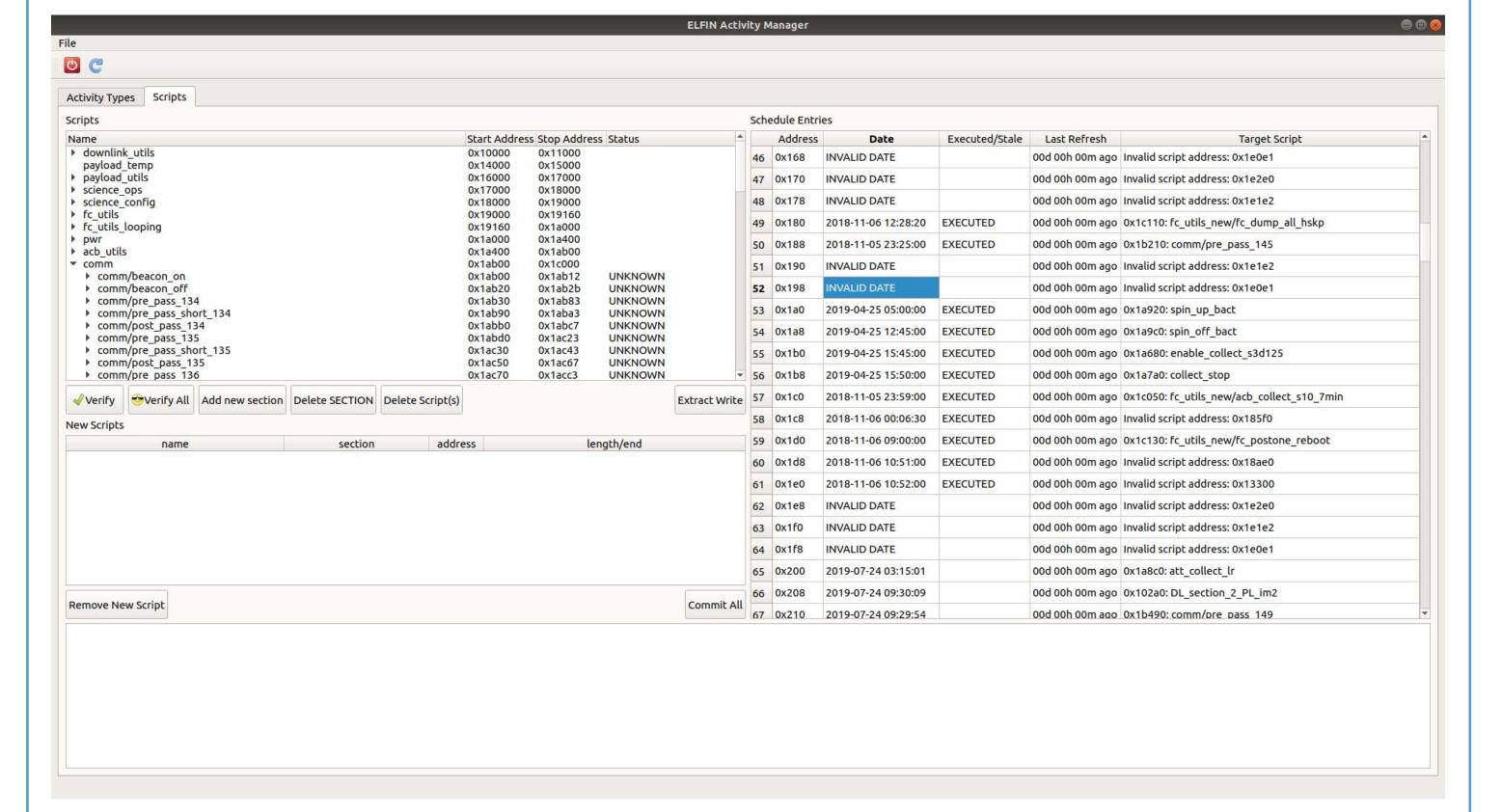
At a Glance

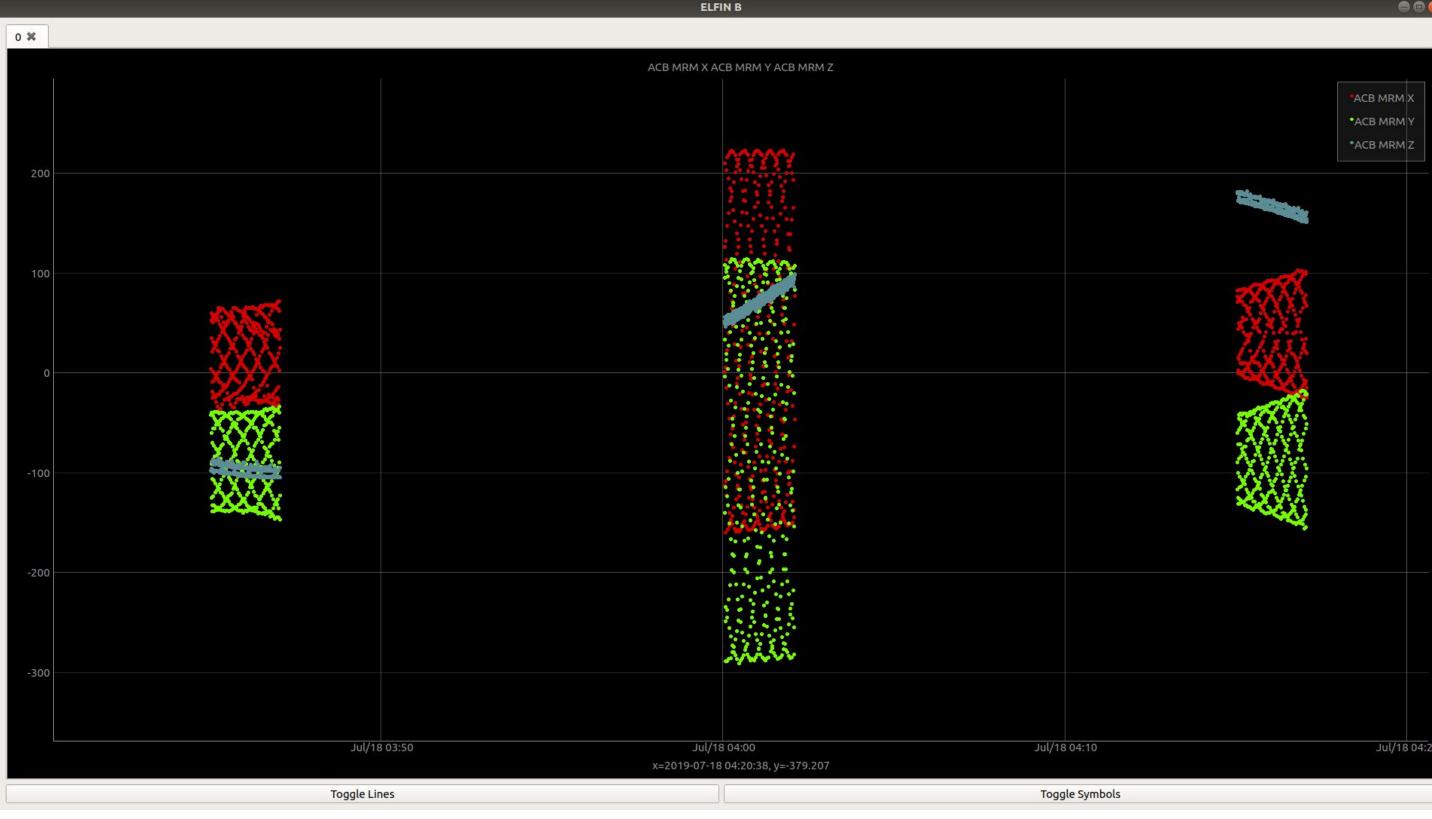


ELFIN software diagram. ADCS and ESN segments are condensed to blackbox representations. Two instances of the Commander, Telemetry Parser and Planner are used concurrently - one for each of the two ELFIN satellites.

- 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

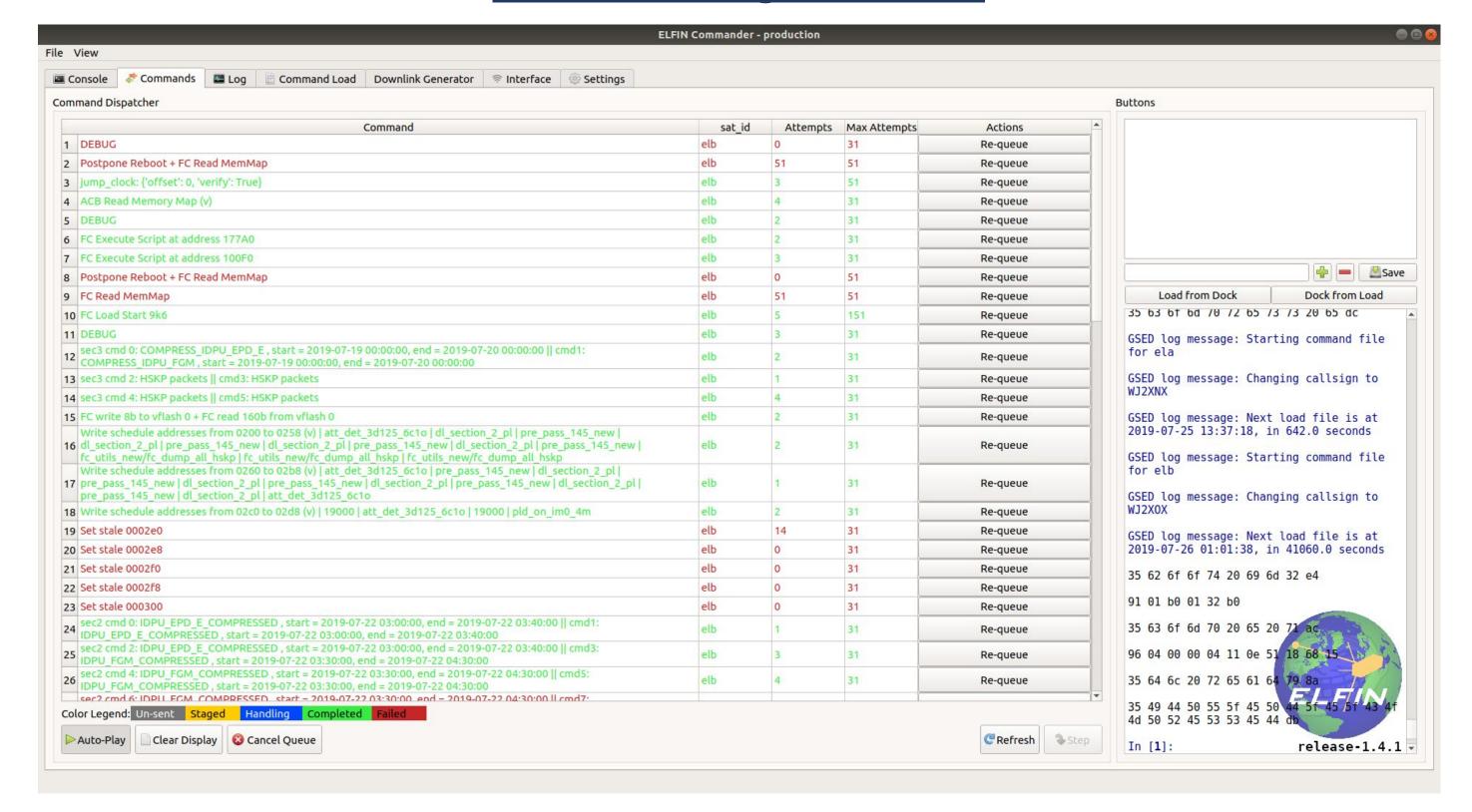
Telemetry Information





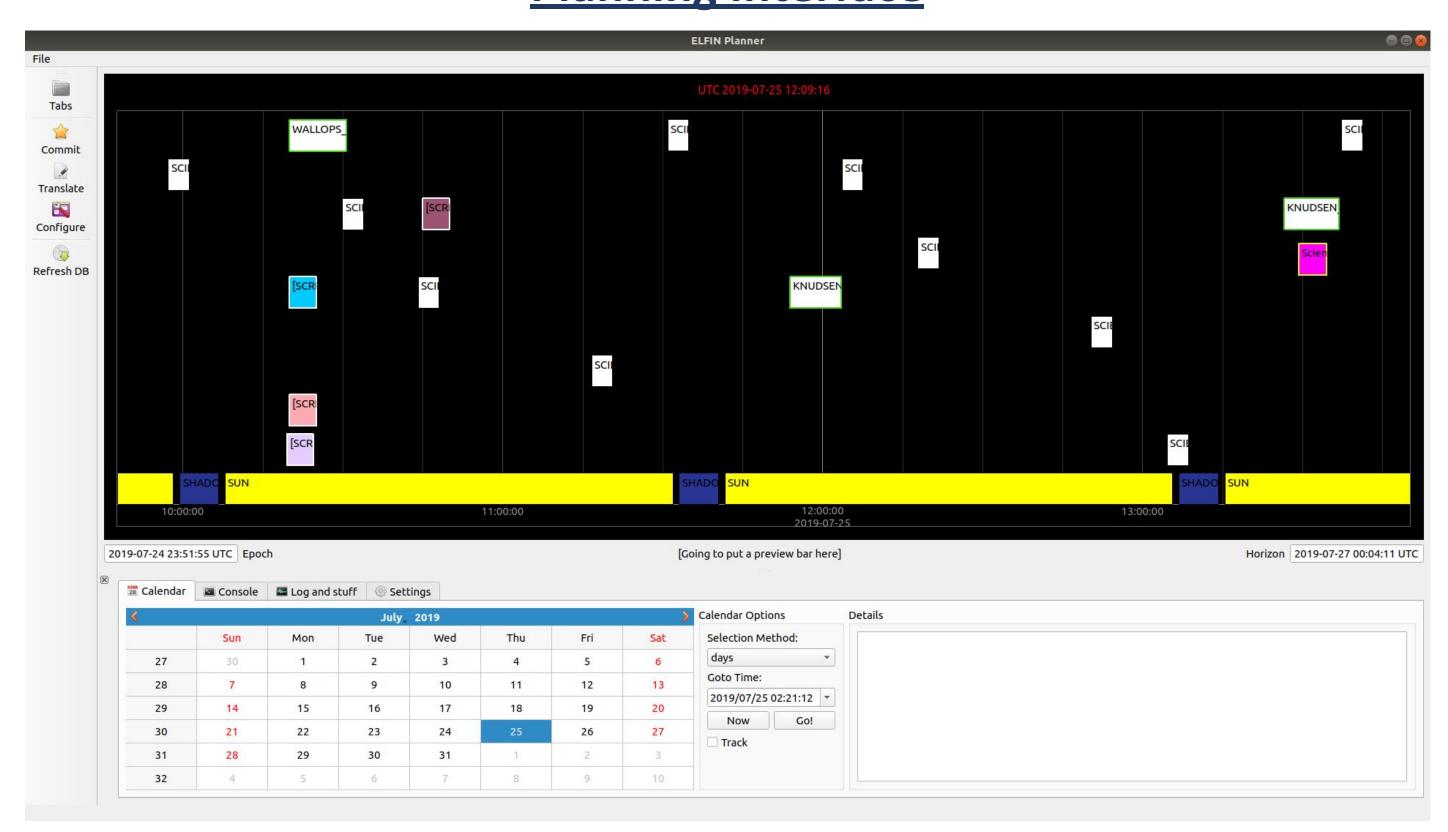
Above: State information displaying state on on-board scripts and scheduled executions. Updated in real-time. Below: Plotted magnetometer data from the Attitude Control Board (ACB) MagnetoResistiveMagnetometers (MRM).

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.

				Dialog							
vent Options			Activity/Script Options		Collection Options						
Target Event Type: KNUDSEN_ACS Schedule for all Event instances in interval INSTRUCTIONS: You may either schedule for a specific event (default) or for all of a specific type of event in a time range. An event is in a time range if its START time is in [interval start, interval end) To insert a script, insert [SCRIPT] and a pop-up will appear.			Activity Type:	Collection Type							
			Activity Type: Duration: [SCRIPT] * 300			Science Colle	ction - 4m ima	ion - 4m image 0 ▼			
			Time Relative to Event: 3 Interval Start Time: 07/24 Interval Stop Time: 07/2	> After	On Before Os	Off Duration 310s		Add			
Activites/Scripts					Collections						
Relative Time (sec)	Туре		Name	Duration	Ту	pe	On		Off		
-60	Script	downlink_utils/debu	g (583)	300							
0	Script	science_ops/pld_core	e_power (698)	300							
30	Script	downlink_utils/dl_se	ction_2_pl (819)	300							
Scheduling for ALL K			No.	0000							
Relative Time (sec)	Type Script	science_ops/pld_shu	Name	Duration 300							
					4						
								⊠Clear All	⊗ Rem		
									 ✓ Acc		
									- V		

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.

Reload Passes	timestamp			descri	iption			bytes	delay after	retries	timeout	A	Load
Upcoming Passes (Orbit Number)	1 None	Postpone Reboot + FC Read MemMap						N/A, special command	5000	50	3500		Scriptify
▶ 4800: Thu 2019-07-25 00:03:49	2 None	jump_clock: {'offset': 0, 'verify': True}					N/A, special command	5000	50	3500		◎ Delete	
▼ 4801: Thu 2019-07-25 01:36:29 duration: 10min 8sec	3 None	ACB Read Memory Map (v)						N/A, special command	5000 30 50		5000		xtract Comm
elevation: 24.242 • 4807: Thu 2019-07-25 12:03:49	4 None						b'\x06\x01\x00\x00'	10000	30	3500		Ktract Comin	
▶ 4808: Thu 2019-07-25 13:37:18	5 None						b'\x06\x01w\xa0' 30000		30	3500			
 4815: Thu 2019-07-25 23:30:55 4816: Fri 2019-07-26 01:01:38 	6 None	Echo comp fgm q + IDPU 2 packet count from @2019-07-25 18:00:00 to @2019-07-25 20:00:00						N/A, special command	5000	30	3500		
 4817: Fri 2019-07-26 02:37:28 4822: Fri 2019-07-26 11:30:34 	7 None	FC Dump UART1 (v)						N/A, special command	5000	30	3500	- -	
▶ 4823: Fri 2019-07-26 13:01:55									3500	30	3500		^
 4831: Sat 2019-07-27 00:27:10 4832: Sat 2019-07-27 02:01:02 	8 None	sec1 cmd 0: IDPU_FGM_COMPRESSED , start = 2019-07-25 18:00:00, end = 2019-07-25 20:00:00 cmd1: .						100 000 000	100000000		C Breezeway (1970)		~
 4837: Sat 2019-07-27 10:59:36 4838: Sat 2019-07-27 12:27:15 	9 None	sec1 cmd 2: IDPU_FGM_COMPRESSED , start = 2019-07-25 18:00:00, end = 2019-07-25 20:00:00 cmd3: .						Section 15 and 15	3500	30	3500		
▶ 4839: Sat 2019-07-27 14:02:49	(A)	10 None sec1 cmd 4: IDPU_FGM_COMPRESSED , start = 2019-07-25 18:00:00, end = 2019-07-25 20:00:00 cmd5:							3500	30	3500		
 4846: Sat 2019-07-27 23:53:14 4847: Sun 2019-07-28 01:25:44 	11 None sec1 cmd 6: IDPU_FGM_COMPRESSED , start = 2019-07-25 18:00:00, end = 2019-07-25 20:00:00 cmd7: .							120000	3500	30	3500		
▶ 4853: Sun 2019-07-28 11:53:15 ▶ 4854: Sun 2019-07-28 13:26:28	12 None								3500	30	3500		Save
▶ 4861: Sun 2019-07-28 23:20:38	→ 13 None	13 None FC Execute Script at address 100F0						b'\x06\x01\x00\xf0'	3500	30	3500	▼	Export
oad Files													
Timestamp		Filename		Time Until	Abort	Timestamp		Filename			Status	≜ Lo	
1 2019-07-26 01:01:38	server/loads/elb_4	4816.load	elb	04hr 44min	17.0011	18	2019-07-20_13:13:03	ela_4731.ld	oad		Fir	ished	
					Force	19	2019-07-20_13:23:18	elb_4731.ld	oad		Fir	ished	□ □ Up
						20	2019-07-21_00:38:24	ela_4739.ld	oad		Fir	ished	
						21	2019-07-21_00:48:24	elb_4379.ld	oad		Fir	ished	■ De
						22	2019-07-21_02:12:06	ela_4740.ld	oad		Fir	ished	C Ref
						23	2019-07-21_12:38:26	ela_4746.ld	oad		Fir	ished	
						24	2019-07-21_12:49:00	elb_4746.ld	oad		Fir	ished	
							2019-07-22 01:36:51	ela 4755.ld			Fir	ished	
						100	2019-07-22 01:47:10	elb 4755.ld				ished	
							2019-07-22 12:04:30	ela_4761.ld				ished	
							2019-07-22 12:14:20	elb 4761.ld			-	ished	
						20	2012 01 22 12.17.20	CID_4701.10	000		EII	isiled	
						20	2019-07-22 13:37:29	ela 4762.ld	her		Ei,	ished	

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/





