LADEE Flight Software: Verification Techniques

SPIN 2014

Karen Gundy-Burlet, Ph.D.

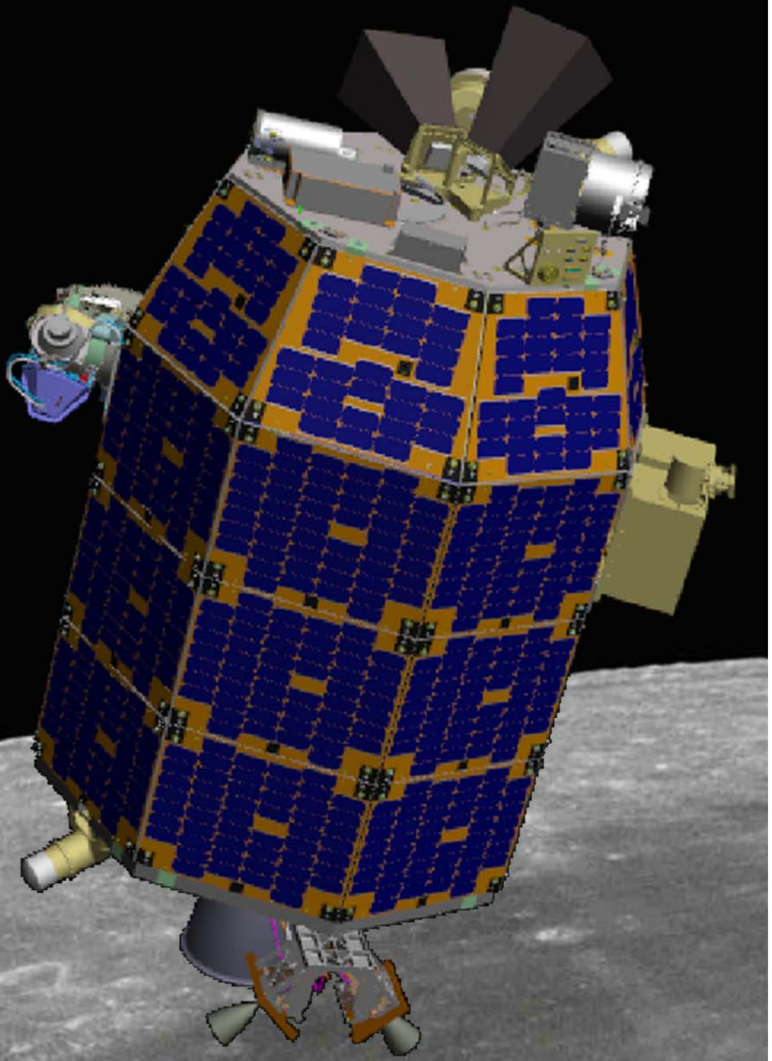
NASA-Ames Research Center

Moffett Field, CA

Roadmap

- Mission Description
- Software Engineering Practices
- Software Architecture
- Testing Infrastructure
- V&V practices
- Mission Operations

Lunar Atmosphere and Dust Environment Explorer



Objectives
- Measure Lunar Dust
- Examine the Lunar atmosphere
- 100 days in a low-equatorial lunar orbit

Key parameters
- Launched Sept 6, 2013
- Lunar Impact April 18, 2014

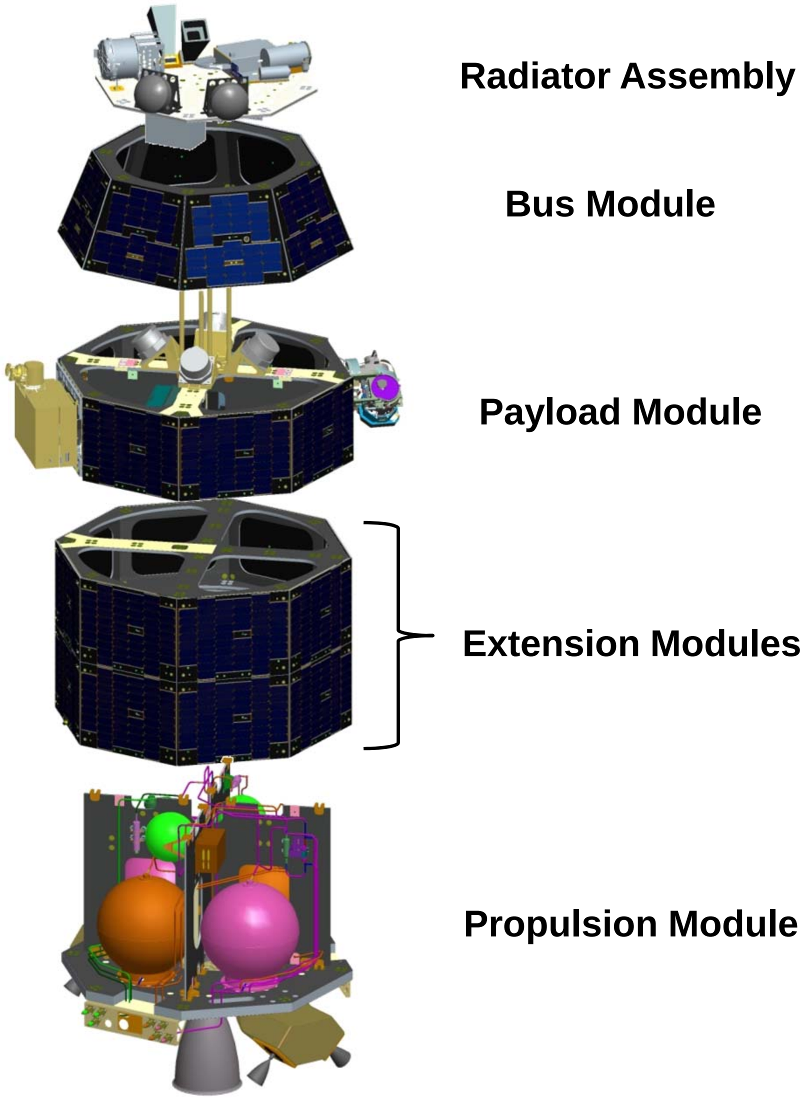
Spacecraft
- Type: Small Orbiter - Category II, Enhanced Class D
- Provider: NASA ARC and NASA GSFC

Instruments
- Science Instruments: NMS, UVS, and LDEX
- Technology Payload: Lunar Laser Communications Demo

Launch Vehicle: Minotaur V

Launch Site: Wallops Flight Facility

LADEE Observatory



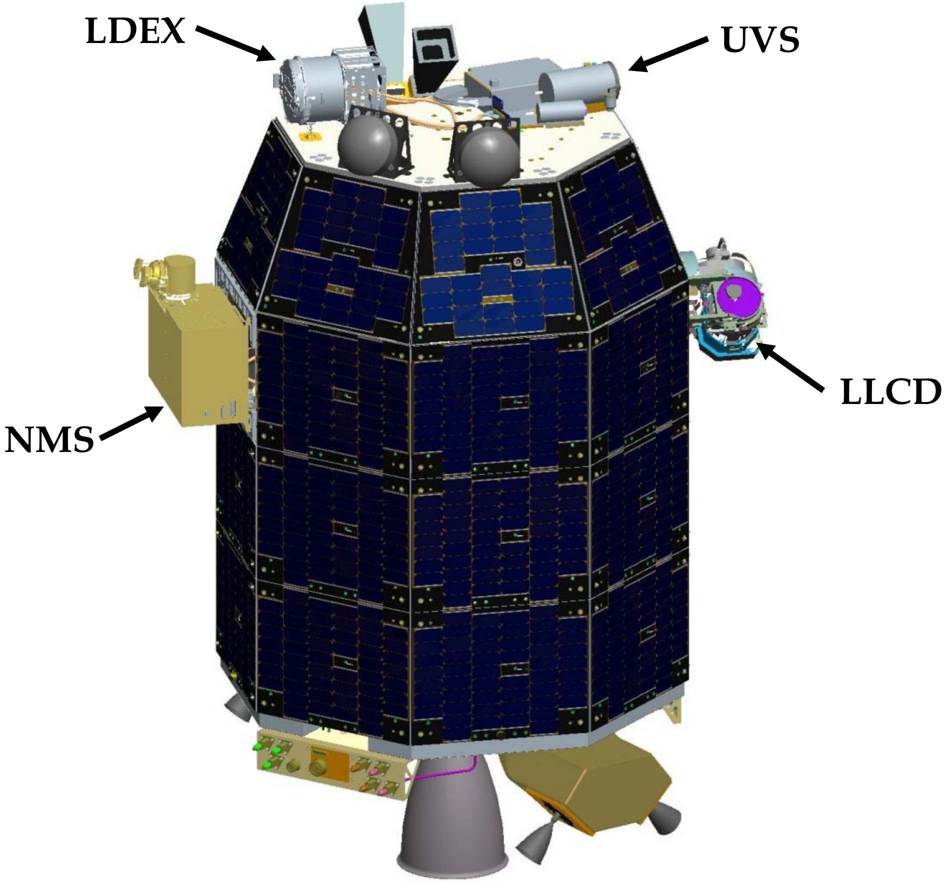
Radiator Assembly

Bus Module

Payload Module

Extension Modules

Propulsion Module



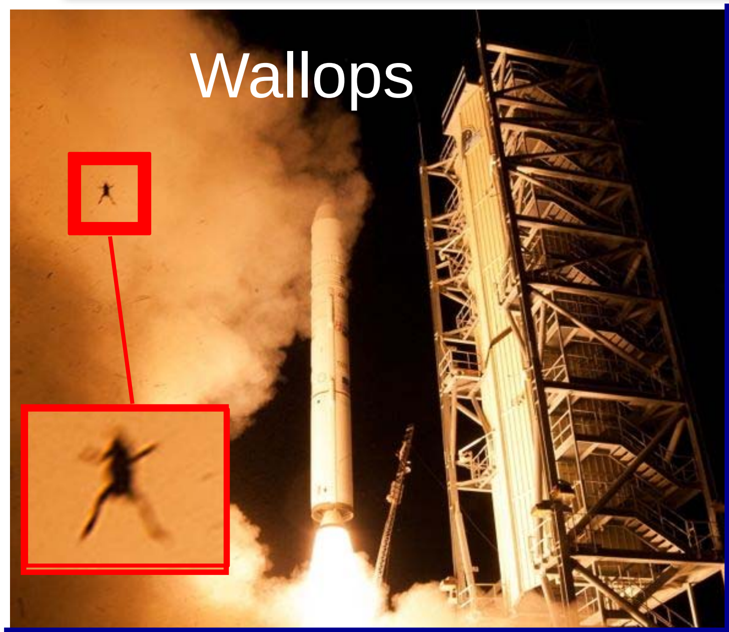
LDEX

UVS

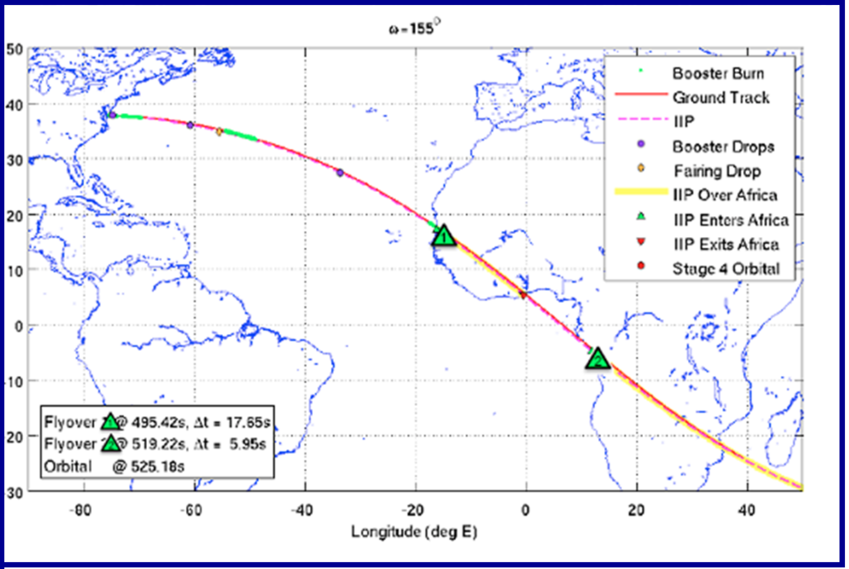
LLCD

NMS

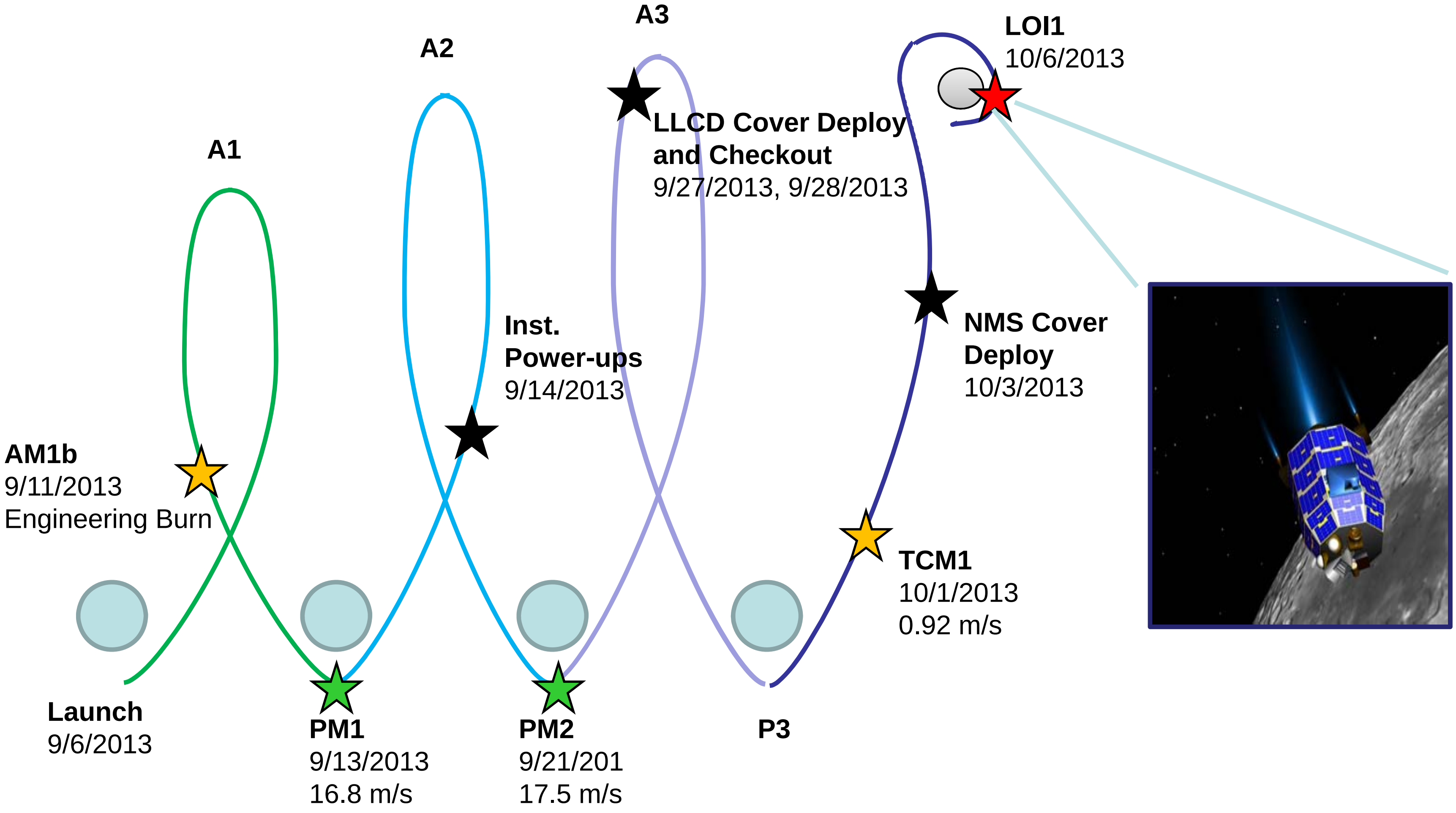
LADEE Launch: Sept 6, 2013



Ben Cooper / LaunchPhotography.com



LADEE's Journey to the Moon



A1

A2

A3

LOI1
10/6/2013

LLCD Cover Deploy
and Checkout
9/27/2013, 9/28/2013

Inst.
Power-ups
9/14/2013

NMS Cover
Deploy
10/3/2013

AM1b
9/11/2013
Engineering Burn

TCM1
10/1/2013
0.92 m/s

Launch
9/6/2013

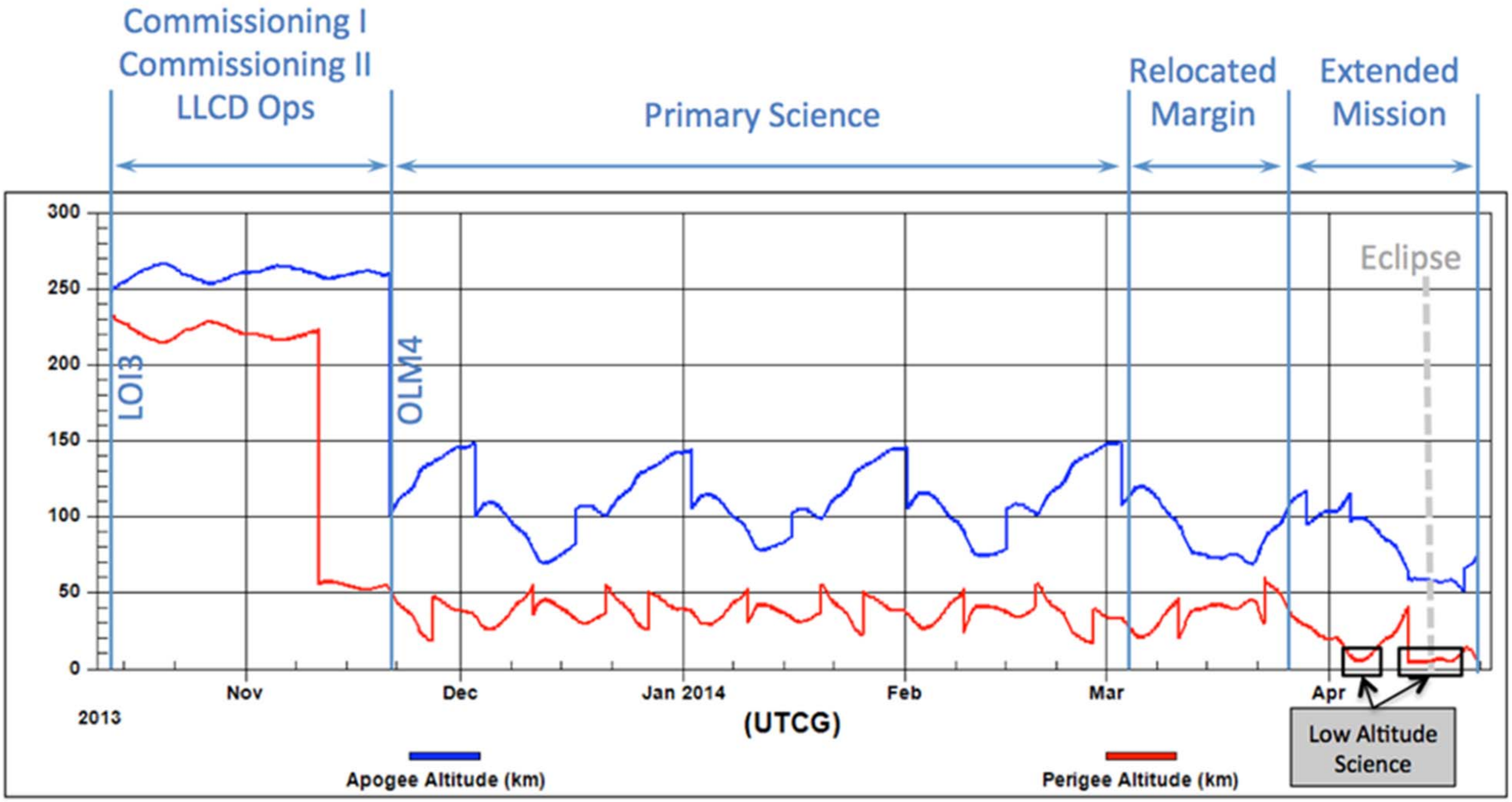
PM1
9/13/2013
16.8 m/s

PM2
9/21/201
17.5 m/s

P3

- Successful burns at Perigee 1 and Perigee 2 made burn at Perigee 3 unnecessary. Small single correction (TCM-1) all that was needed to set LADEE up for Lunar Orbit Insertion
- Checkout of all S/C components and Instruments performed

LADEE Lunar Trajectory Overview



Drivers for Software Development Practices

- Flight software development has an extensive history of cost overruns, schedule slips and failures in operation. Many reasons, among them:
  - Increasing complexity of missions
  - Lack of knowledge of best practices
  - Difficulties formulating requirements
  - Communication problems between stakeholders
- LADEE was driven by a fixed cost cap and firm launch schedule caused by extended periods of eclipse soon after our scheduled launch date
- LADEE looked to earlier small, successful rapidly prototyped missions for inspiration: XSS-10, XSS-11
  - Software designed from the start to be testable.
  - Early, rapid prototyping of algorithms/requirements/tests using a Model Based Development environment. Autocoding of models reduces transcription errors.
  - Implementation of best practices compliant with CMMI Level 2 and NPR 7150.2
  - Extensive automation in Verification and Validation test suite.

Flight Software Overview

- Scope

  -Onboard Flight Software (Class B)

  -Support Software and Simulators (Class C)

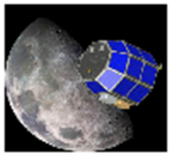
  -Integration of FSW with avionics

- Guiding Documents

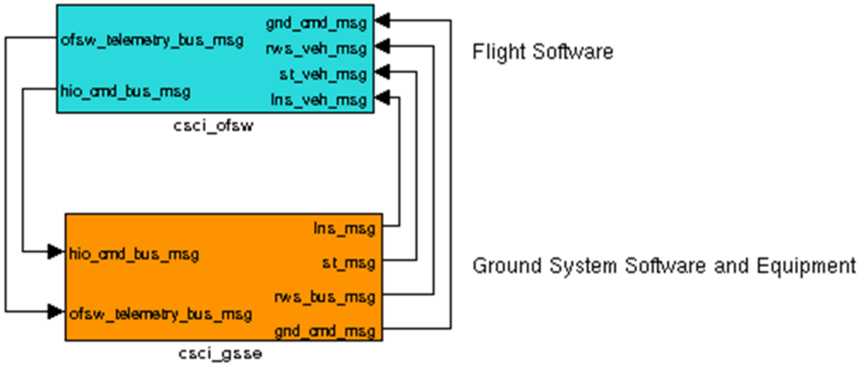
  -NPR7150.2 Software Engineering Requirements

  -CMMI Level 2 or Equivalent

  -NASA-STD-8739.8 NASA Software Assurance Standard

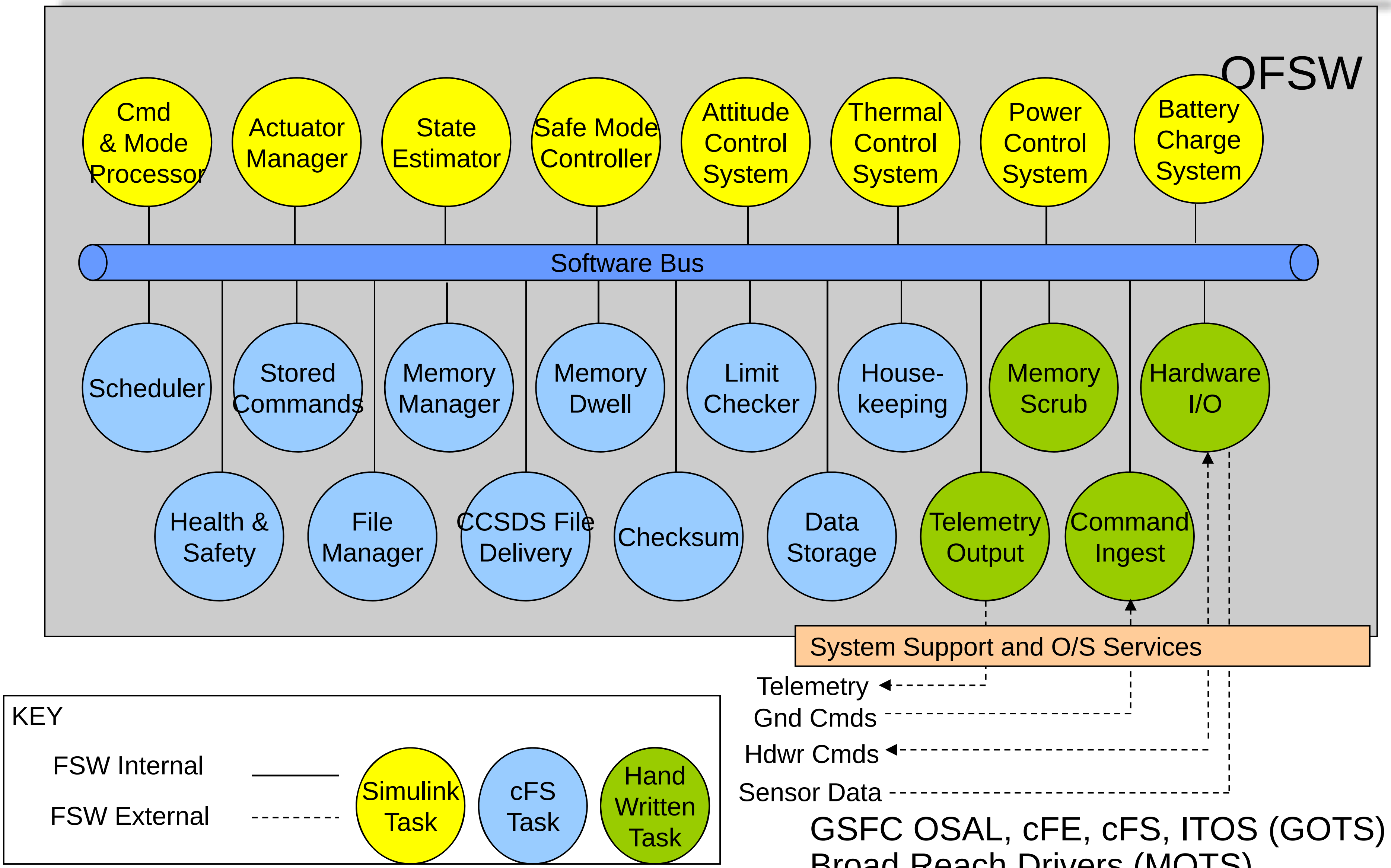


LADEE



- Development Approach
  - Model Based Development Paradigm (prototyped process using a “Hover Test Vehicle”)
  - 5 Incremental Software Builds, 2 Major Releases
- Leverage Heritage Software
  - GSFC OSAL, cFE, cFS, ITOS
  - Broad Reach Drivers, VxWorks
  - Mathworks Matlab/Simulink & associated toolboxes

FSW Architecture



OFSW

Cmd & Mode Processor

Actuator Manager

State Estimator

Safe Mode Controller

Attitude Control System

Thermal Control System

Power Control System

Battery Charge System

Software Bus

Scheduler

Stored Commands

Memory Manager

Memory Dwell

Limit Checker

House-keeping

Memory Scrub

Hardware I/O

Health & Safety

File Manager

CCSDS File Delivery

Checksum

Data Storage

Telemetry Output

Command Ingest

System Support and O/S Services

Telemetry

Gnd Cmds

Hdwr Cmds

Sensor Data

KEY

FSW Internal

FSW External

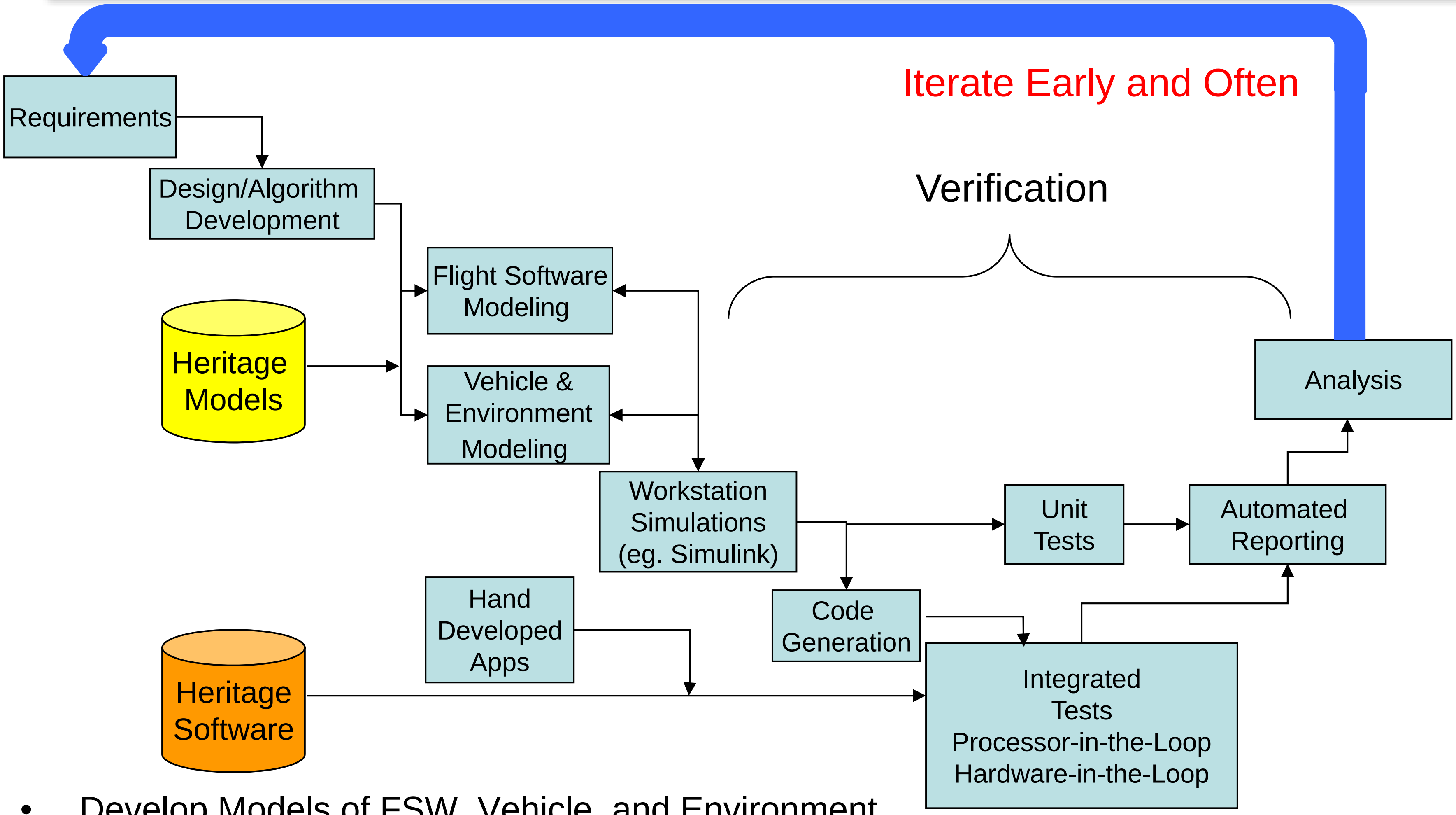
Simulink Task

cFS Task

Hand Written Task

GSFC OSAL, cFE, cFS, ITOS (GOTS)
Broad Reach Drivers (MOTS)
Simulink/Matlab, VxWorks (COTS)

Model Based Development



- Develop Models of FSW, Vehicle, and Environment
- Automatically generate High-Level Control Software
- Integrate with hand-written and heritage software.
- Iterate while increasing fidelity of tests – Workstation Sim (WSIM), Processor-In-The-Loop (PIL), Hardware-in-the-Loop (HIL)
- Automated self-documenting tests providing traceability to requirements

Hardware Test Systems

WSIM Workstation Simulations	Simulink on Windows, Mac, or Linux computers	•Models of GN&C, Prop, Power, & Thermal •Used by FSW to generate and test algorithms. •Provided to MOS for limited functionality maneuver simulations.
PIL Processor-in-the-Loop	PPC750 Processor(s) in Standalone chassis	•Includes all flight software functionality. Runs on 1 or 2 processors. •Multiple copies maintained by FSW as inexpensive system for real time software & fault management development. •Faster than real time (depending on selected fidelity of models and processor speed.
HIL Hardware-in-the-Loop	Avionics EDU with simulated vehicle hardware.	•Highest fidelity simulators includes hardware interfaces. •Run in real time. •Travelling Road Show used to test payload interfaces early in development cycle •Authoritative environment for verification of FSW requirements

Requirements

- Philosophy: Requirements inform you what to test.
- LADEE FSW Requirements: 144 Level 4 Requirements (The FSW shall…)
- Several Major Categories at the L4:
  - Mode Control
  - GN&C
  - Payloads
  - Command & Telemetry
  - Fault Management Response
  - Software Operations
  - Miscellaneous

  - Given the Requirements, the Software Design & the Command and Telemetry Dictionary, we were able to set up tests in advance of the functionality appearing in the code/models.
  - Metric used to focus development: Requirements not yet met.

Validation & Verification Approach

- NPR 7150.2 requires Software Test Plans, Procedures and Reports.
  - Test Plan developed per SWE-104. Specific testing includes:
    - Testing performed at multiple levels: WSIM, PIL, HIL
    - Unit test suite to verify low-level requirements.
    - Integrated test suite to verify system/subsystem compliance with associated requirements.
    - Custom Model Advisor checks to ensure compliance of Simulink models with Common Bus Modeling Guidelines.
    - Static Analysis to ensure correctness of auto-generated code, scripts & leveraged COTS/GOTS products.
    - Code Coverage analysis to assess missing Requirements/Testing
    - Validation of system nominal performance through scenario test scripts derived from Concept of Operations document.
    - Validation of system off-nominal performance through scenario test scripts derived from Fault Management document.
  - Test Procedures documented and maintained under configuration management.
  - Test Report system
    - Uses Matlab/Simulink Report Generator: Automatically produces documentation with bidirectional traceability between Requirements, Models, Unit/Integrated test suites and Test Evidence.
    - Gathers statistics on requirements, test suite and model complexity for both regression test and trending analyses
    - Artifacts under configuration management
  - Peer Review/Inspection program to assure quality of all artifacts

Comments on Model Checking (1)

- LADEE FSW was a hybrid system with multiple integrated layers.
  - High-level, very simple mode logic within Simulink Models
    - Mode Logic Implemented in Embedded Matlab rather than Stateflow
    - Some Simulink models in isolation could be formally checked, but this can miss critical problems with propagation of signals through all layers of software.
    - Model checking required expert intervention, and did not lend itself to the swift cadence and hard deadlines of a build verification cycle.
    - Nervous about effect of assertions in Simulink on the final autocode.
    - Not the authoritative flight software.
  - More complex mode logic in Fault Management.
    - Hand coded in "C" using Reverse Polish Notation based data structures.
    - Multiple action points could fire simultaneously, leading to potential of commands within Relative Time Sequence scripts interfering with each other.
    - Monte-Carlo techniques used to try to assess interactions, but not a formal proof.

Comments on Model Checking (2)

- Need to verify the integrated flight software, not just the models.
  - Significant interactions possible.
- Statement of the requirements is generally amenable to assessment of assertions
  - “The OFSW shall associate an ACS controller with each controller mode such that only that controller can send commands when the OFSW is in the associated mode.”
  - “In Delta V mode the OFSW shall be capable of controlling the spacecraft z-axis azimuth error to within XX mrad”
- Full flight software amenable to techniques like static analysis, but not practical to apply current generation of advanced formal methods.
- Test as we fly!
  - Telemetry is the normal indicator of the software health during flight so verify L4 requirements on the telemetry stream using same tool-chain as in flight.
  - Scenarios developed exercising each flight phase. Software response to identified fault conditions tested in Fault Management scenarios.
  - Assertions applied to telemetry stream and software artifacts to verify level 4s.

Patterns of Testing

- There were many common patterns identified for L4 testing
  - GN&C
  - Modes
  - Fault Management
  - Software Utilization
  - Functionality of all spacecraft commands
- Specific assertions were encoded as Matlab expressions in top level scripts
- Developed several core matlab scripts which dynamically interpreted these assertions of behavior, and applied them to the telemetry streams.

- One L4 requirement was to test each command in the C&T dictionary
  - Developed a test language for expected responses

Example Mode Function

```
function [status, msg, html_file] = fsw_243_test(scn_struct)
    % The OFSW shall shut off the VDU when transitioning to Safe Mode.

    scn = [8, 9, 10, 11];

    cmd = ['sqlite3 ', scn_struct(1).database, …
                ' " SELECT value FROM PCS_SW_ENUM WHERE name=''PCS_VDU_PWR''"'];
       [~, switch_no] = unix(cmd);
    var = ['sw_cmd_state', switch_no];
    enum_logic = [{'SAFE'},        {[var '==0']}]; ...
    enum_cmd = [{'acs_mode_cmd'},  {'cmp_ctl'}]; % Corresponds to first column in enum_logic
    test_var = [{var}, {'unp_hk_l4_high'}]; % Corresponds to second column in enum_logic
    and_cmd = [];
    tick_offset = 0.1;

    [status, msg, html_file] = telem_enum_and_logic(scn_struct, scn, enum_logic, enum_cmd,
and_cmd, test_var, tick_offset, 'first');

end
```

Challenges

- Regression test cycle should take less than a week for requirements that can be automated.
  - Scenarios themselves take a “long weekend” to compute.
  - Reduction of scenario data takes an additional day.
- Significant amounts of data in each regression test suite:
  - Down-selected to about 70Gb using housekeeping packets that contain a minimum variable set for verification.
  - Compress data to change-points and store as binaries.
  - With this amount of data, we are _not_ going to hand-inspect the requirements
- Application of formal methods techniques to telemetry streams.
  - 144 requirements to assess
    - GN&C
    - Modes
    - Fault Management
    - Software Utilization
    - Functionality of all spacecraft commands
  - Time correlations among different telemetry packets.
  - Bidirectional Traceability
    - Requirements ⇔ Models/Code ⇔Tests ⇔ Results
    - Simulink Report Generator scripts to associate artifacts

Interface Control Documentation

The primary cause of defect escape into Build 5.x was misunderstood or ambiguous ICDs.

Examples:

- Fuel Tanks (A,B) = FSW (1,2), Ox Tanks (A,B) = FSW (2,1)
- Star tracker: Interpreted 8Hz operation to mean heads alternately operating at 4 Hz. Instead, they operated synchronously.
- Reaction Wheels: “Current Limit Flag” was a warning flag, not an error flag.

Mitigation:

- EDICD in computer readable format and read into database, so could quickly reconfigure
- Ability to upload parameter tables & software patches

Best prevention for ICD problems was our “Travelling Roadshow”

- EDU in a mobile chassis loaded with the FSW
- Flown to payload sites and integrated with instruments.

Lessons Learned:

- Early integration with payloads/instruments essential to clarifying ICDs.
- Significant rework possible when one “saves money” by not purchasing instrument engineering development units

Emergent Behavior

Prior to one of the orbital phasing maneuvers, the spacecraft went into "safemode”

- Fault Management took action because it detected an excessive spacecraft rate from the state estimation system.
- It was determined that the star tracker caused a jump in the state estimator signal. Two primary factors:
    - The as-built alignment of the star tracker was slightly different than as-designed.
        - Fixed by a table upload. Reduced rate errors but did not eliminate further safemode transitions.
    - The star tracker was exhibiting delays in providing the spacecraft orientation and position when one of the cameras pointed at a "Big Bright Object" (BBO) such as the Sun, Moon or Earth. The behavior continued to worsen the closer we got to the moon.
        - Reworked state estimator model, reran scenarios, re-verified EST Unit Tests, GN&C L4 Requirements & uploaded patch to the spacecraft.

Lesson Learned:

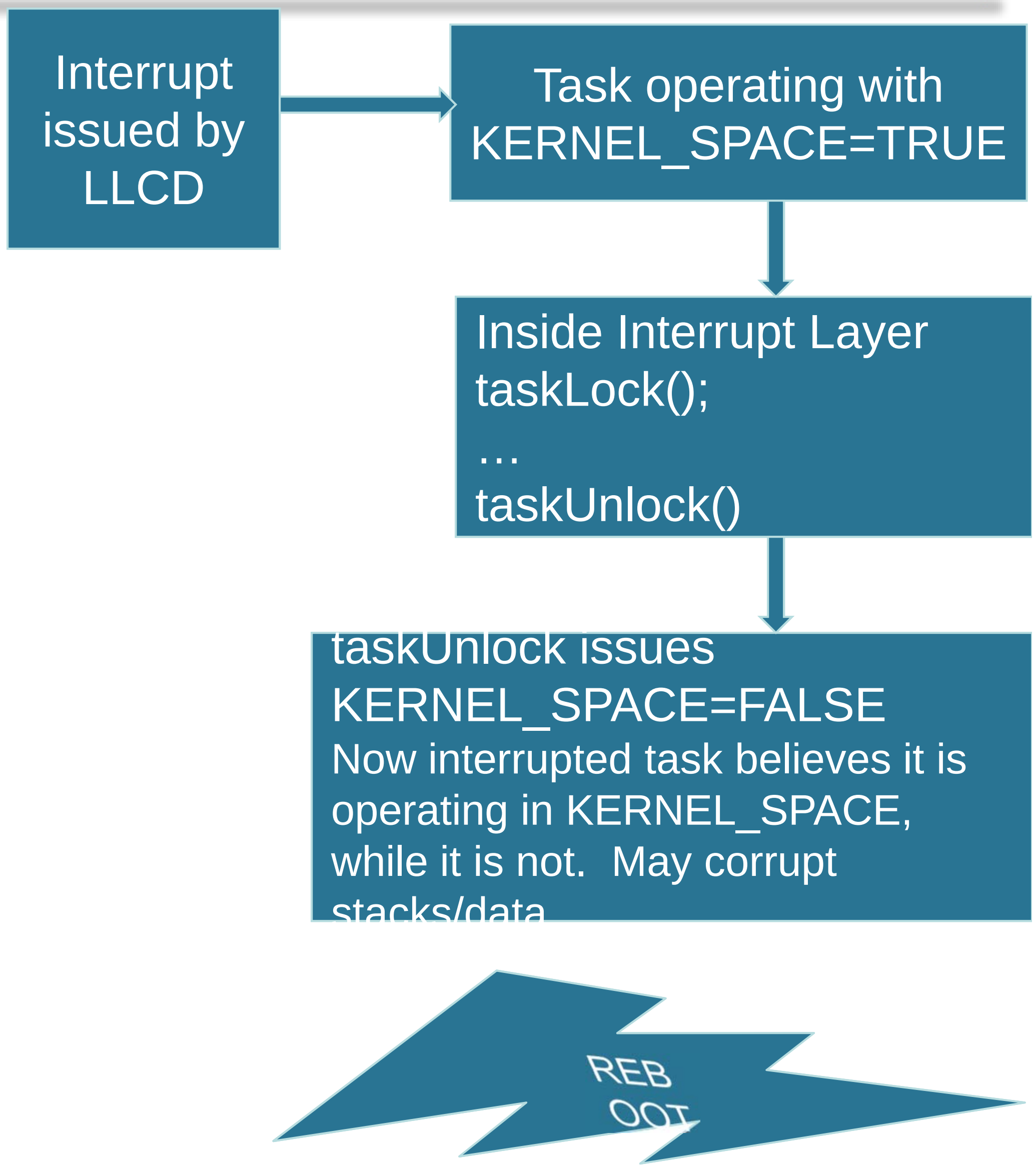
- The defects we had to correct under schedule pressure and duress in I&T and ORTs were excellent repeated practice for polishing our maintenance processes.

Reboot

At the end of one of the blocks of LLCD testing, the spacecraft unexpectedly rebooted:

- We cross-correlated data from onboard software history data, telemetered spacecraft data and DSN logs to determine that the event was highly correlated with the interrupt generated at the end of LLCD operations.

- “Next Time”, I know to have a formal inspection question about lock/unlock functions
- Static analyzers did not identify the code segments as questionable.
- Appears to be an “opportunity” for automated tools



Interrupt issued by LLCD

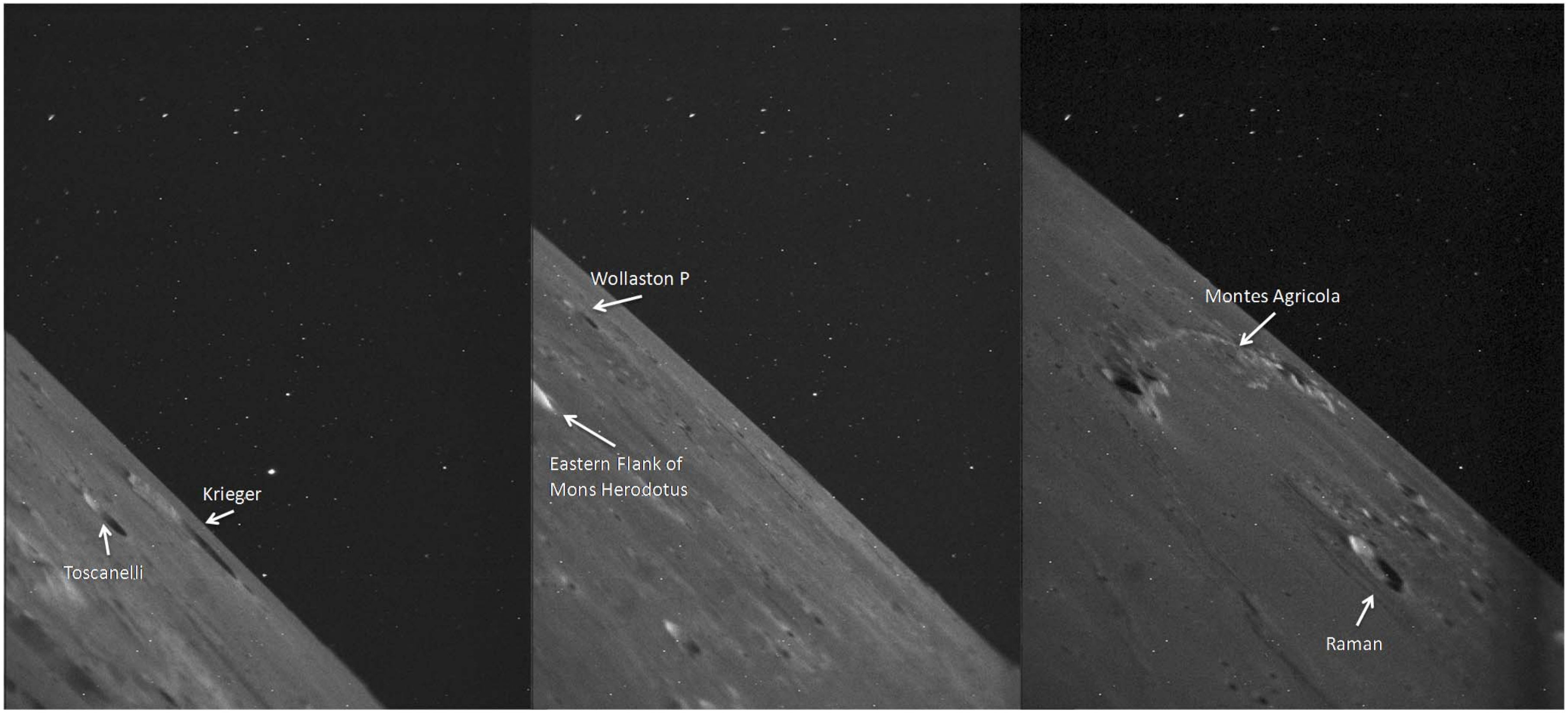
Task operating with KERNEL_SPACE=TRUE

Inside Interrupt Layer
taskLock();
…
taskUnlock()

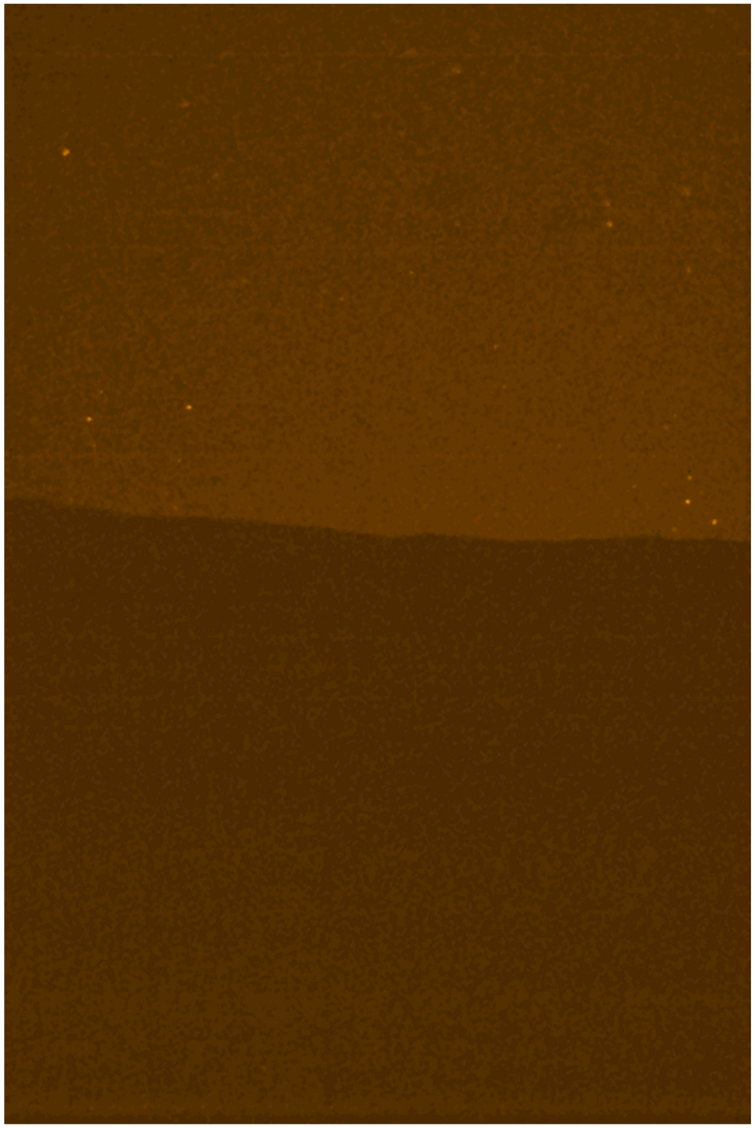
taskUnlock issues KERNEL_SPACE=FALSE
Now interrupted task believes it is operating in KERNEL_SPACE, while it is not. May corrupt stacks/data

REB
OOT

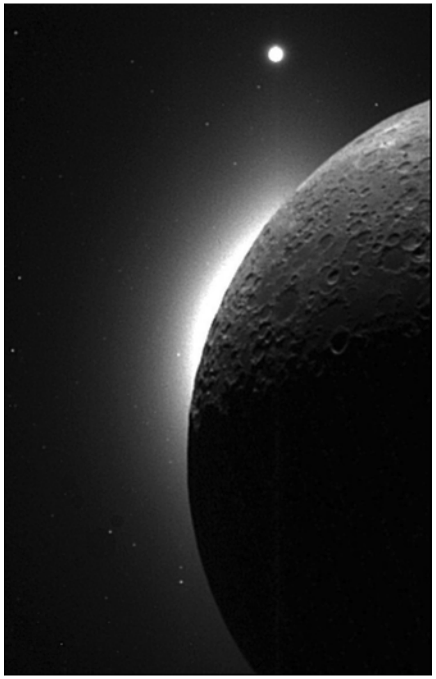
Star Tracker Images



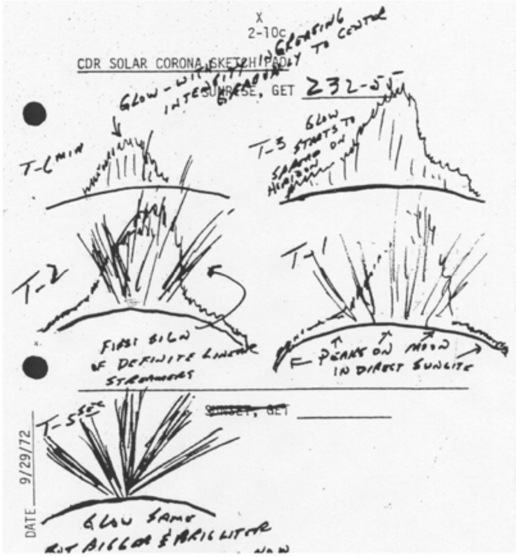
Star Tracker Images



- A series of star tracker images taken by LADEE Saturday, April 12. The lunar horizon is ahead, a few minutes before orbital sunrise.



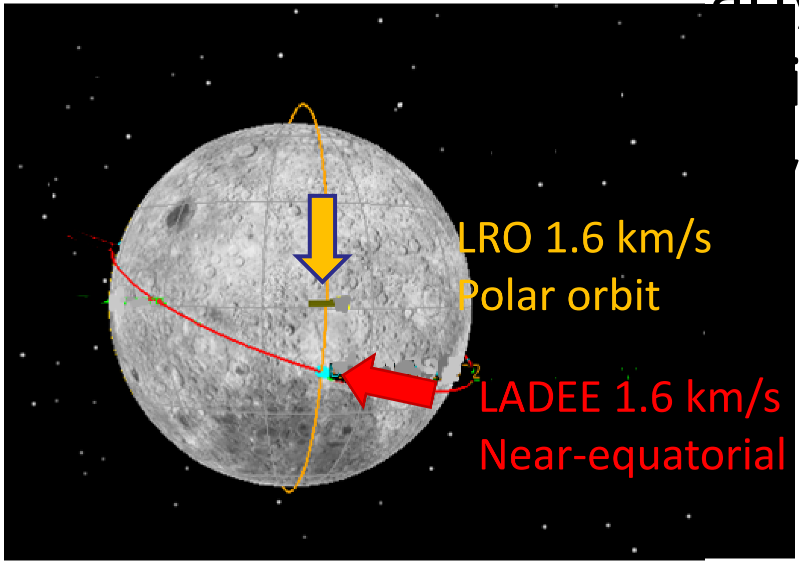
Clementine spacecraft image of moon dust corona

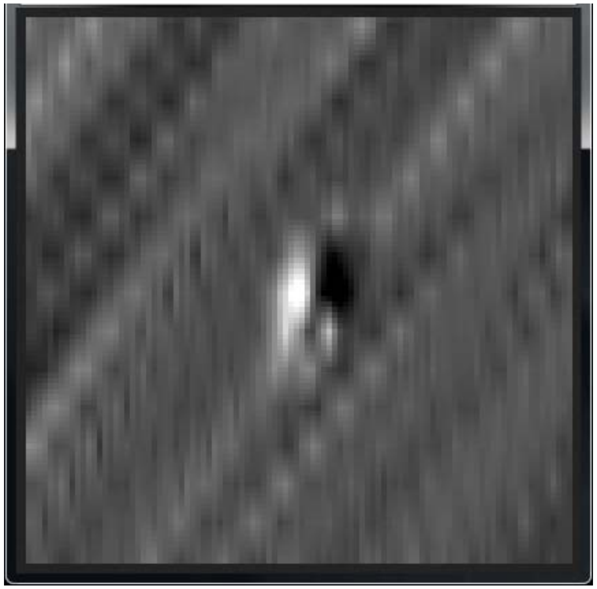

X
2-10c
CDR SOLAR CORONA SKETCH PAD
SUNRISE, GET 232-5
GLOW - WITH INTENSITY INCREASING GRADUALLY TO CENTER
T-6 MIN
T-3 GLOW STARTS TO SPREAD ON HORIZON
T-2
FIRST SIGN OF DEFINITE LINEAR STREAMERS
T-1
PEAKS ON MOON IN DIRECT SUNLITE
~~SUNSET, GET~~
T-5 SEC
GLOW SAME BUT BIGGER & BRIGHTER
DATE 9/29/72

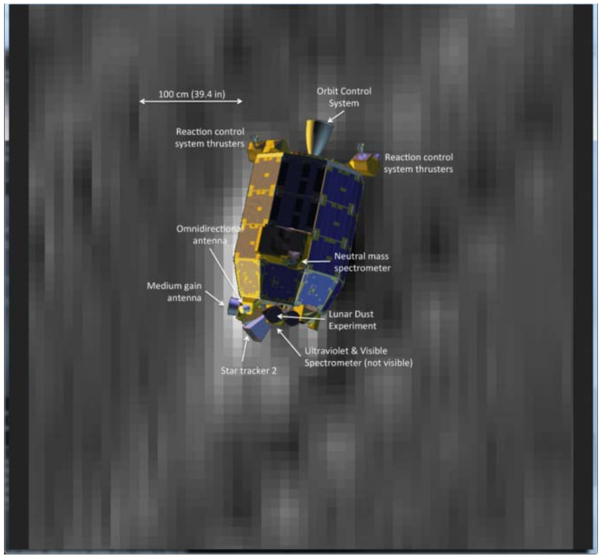
Gene Cernan’s drawings of the lunar sunrise

High Precision in Predicted Position

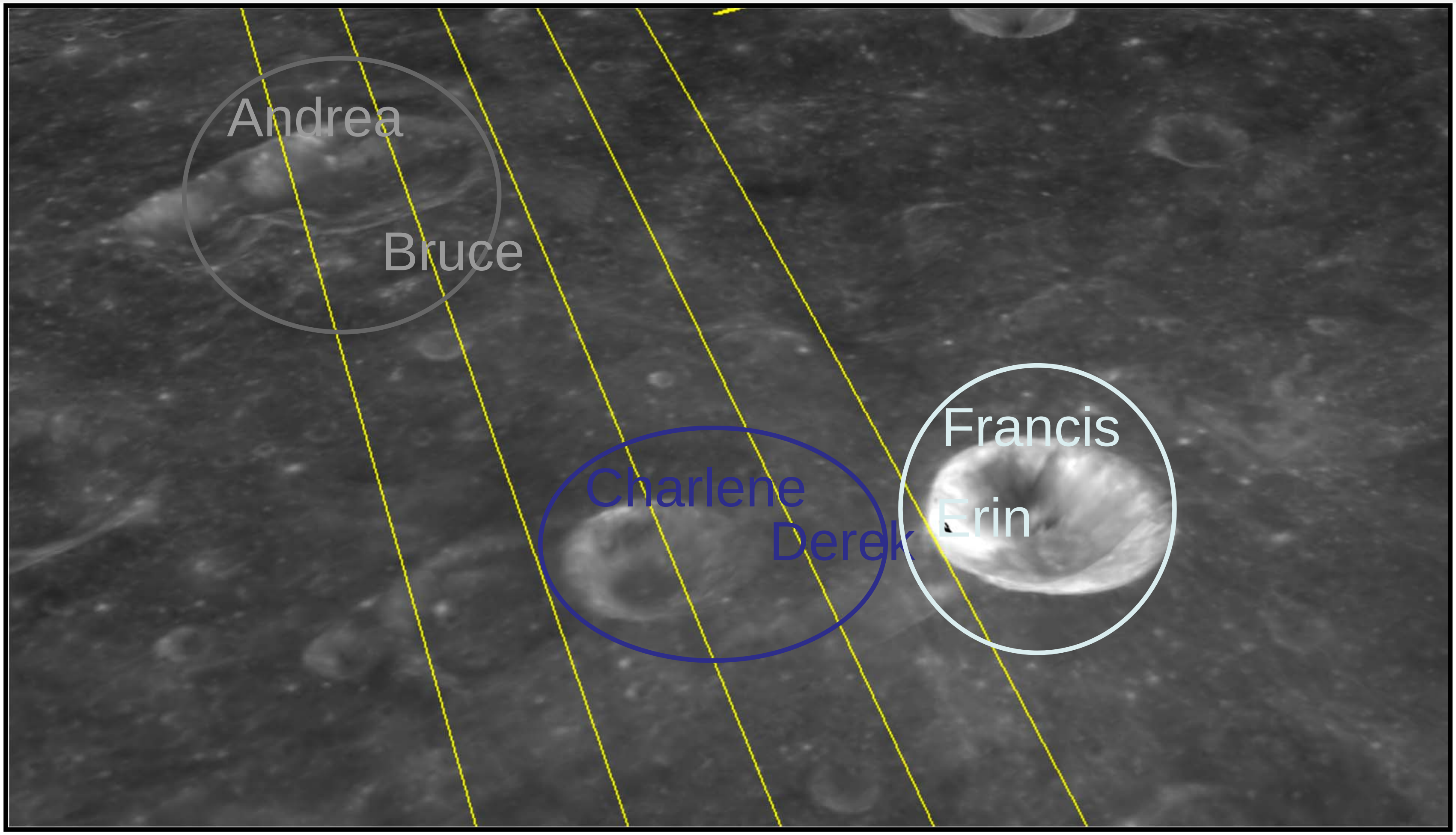
- LADEE Orbit Determination team predicted the location of the spacecraft precisely enough for an LROC photo at a high velocity fly-by
  - Two spacecraft at a nearly
    ing
  - /sec





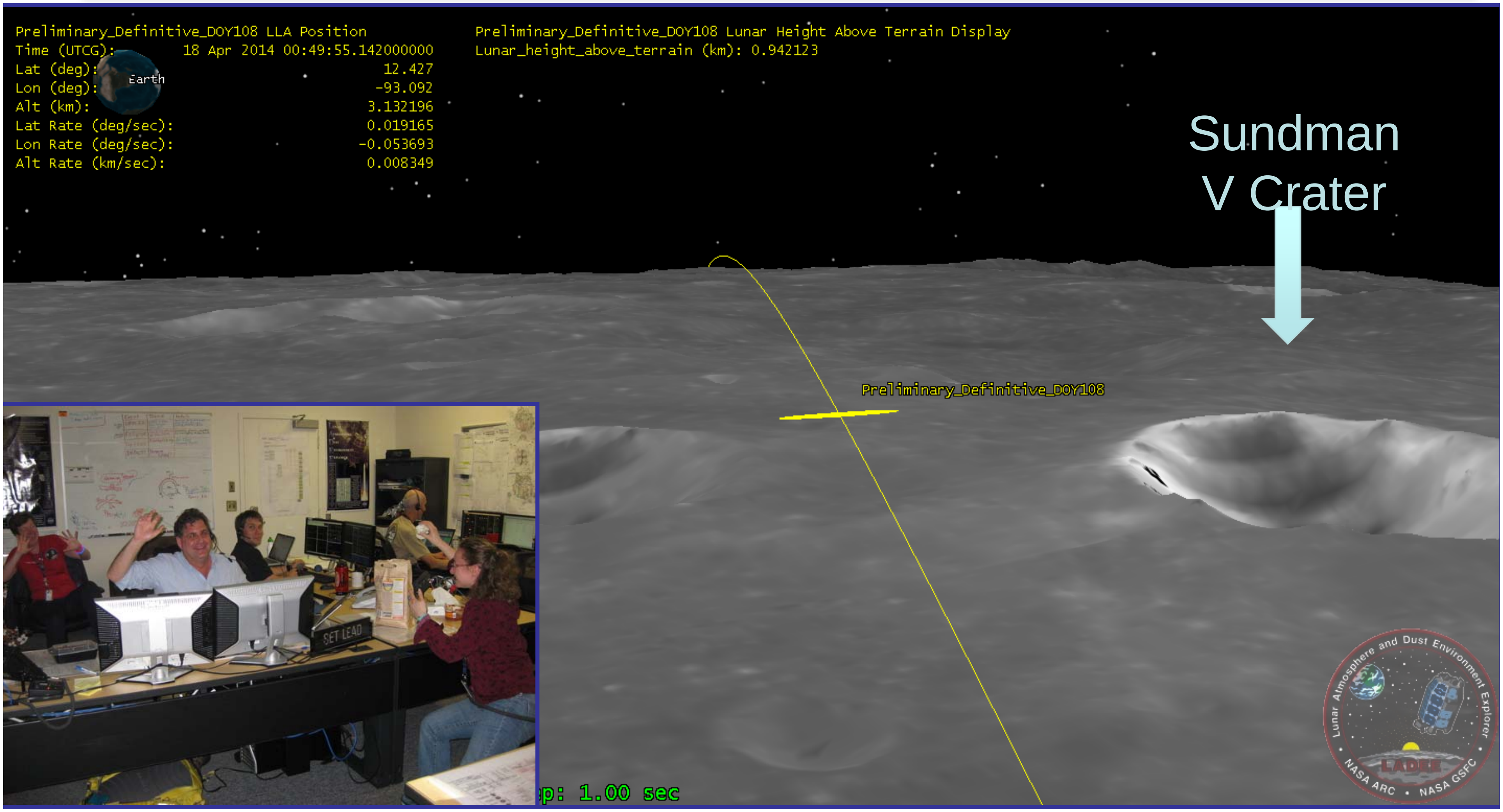

100 cm (39.4 in)
Orbit Control System
Reaction control system thrusters
Reaction control system thrusters
Omnidirectional antenna
Neutral mass spectrometer
Medium gain antenna
Lunar Dust Experiment
Ultraviolet & Visible Spectrometer (not visible)
Star tracker 2

Detail View of Region of Interest



- **In Florida they have a “Hurricane Watches”, on the Moon, we had an “Impact Watch”!**

Derek: 4/18/2014 00:49:55



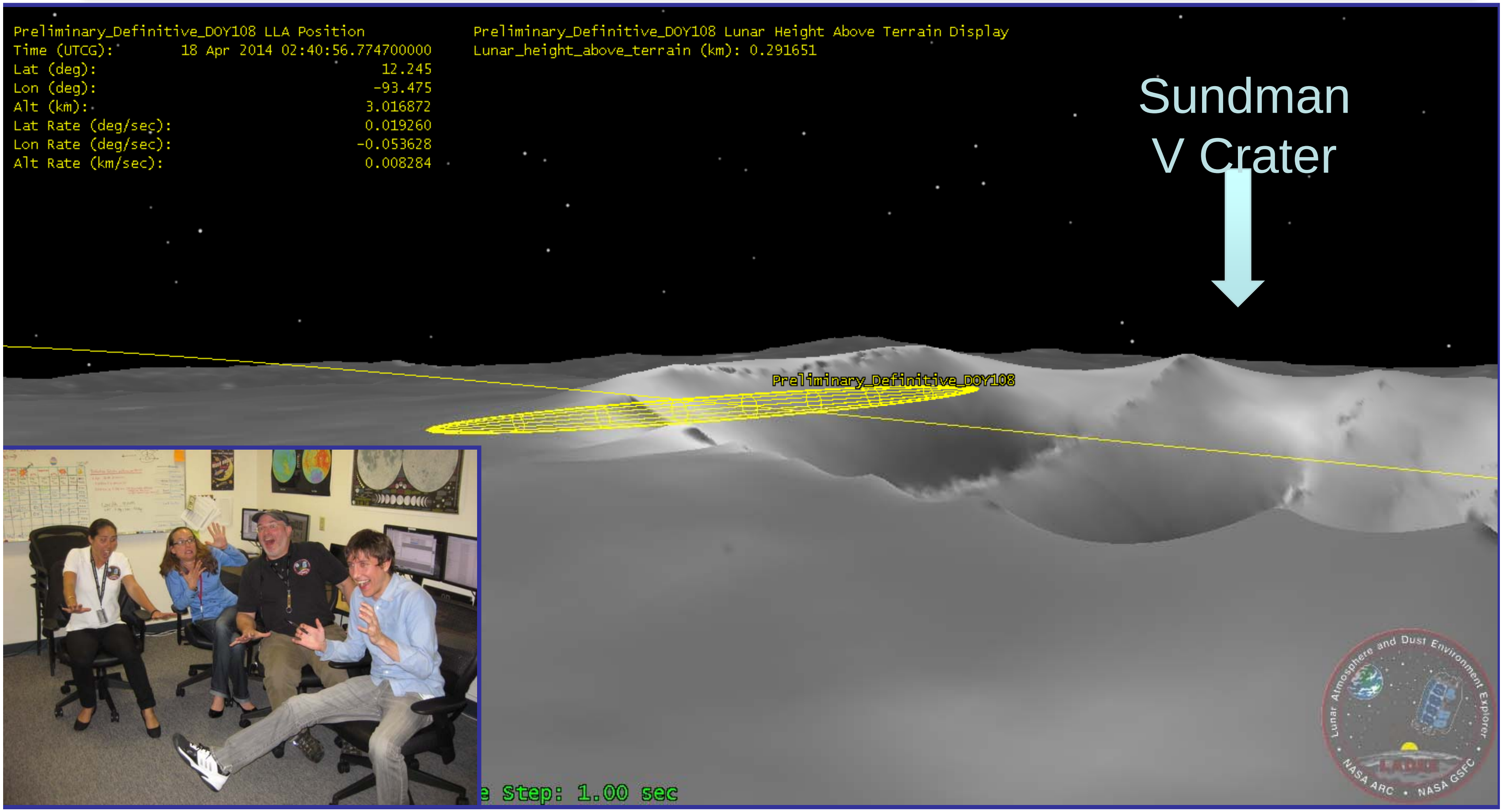
```
Preliminary_Definitive_DOY108 LLA Position
Time (UTCG):          18 Apr 2014 00:49:55.142000000
Lat (deg):                                  12.427
Lon (deg):                                 -93.092
Alt (km):                                 3.132196
Lat Rate (deg/sec):                       0.019165
Lon Rate (deg/sec):                      -0.053693
Alt Rate (km/sec):                        0.008349
```

```
Preliminary_Definitive_DOY108 Lunar Height Above Terrain Display
Lunar_height_above_terrain (km): 0.942123
```

Sundman V Crater

Actual Height Above Terrain @ Closest Distance: 0.942 km

Erin: 4/18/2014 02:40:56



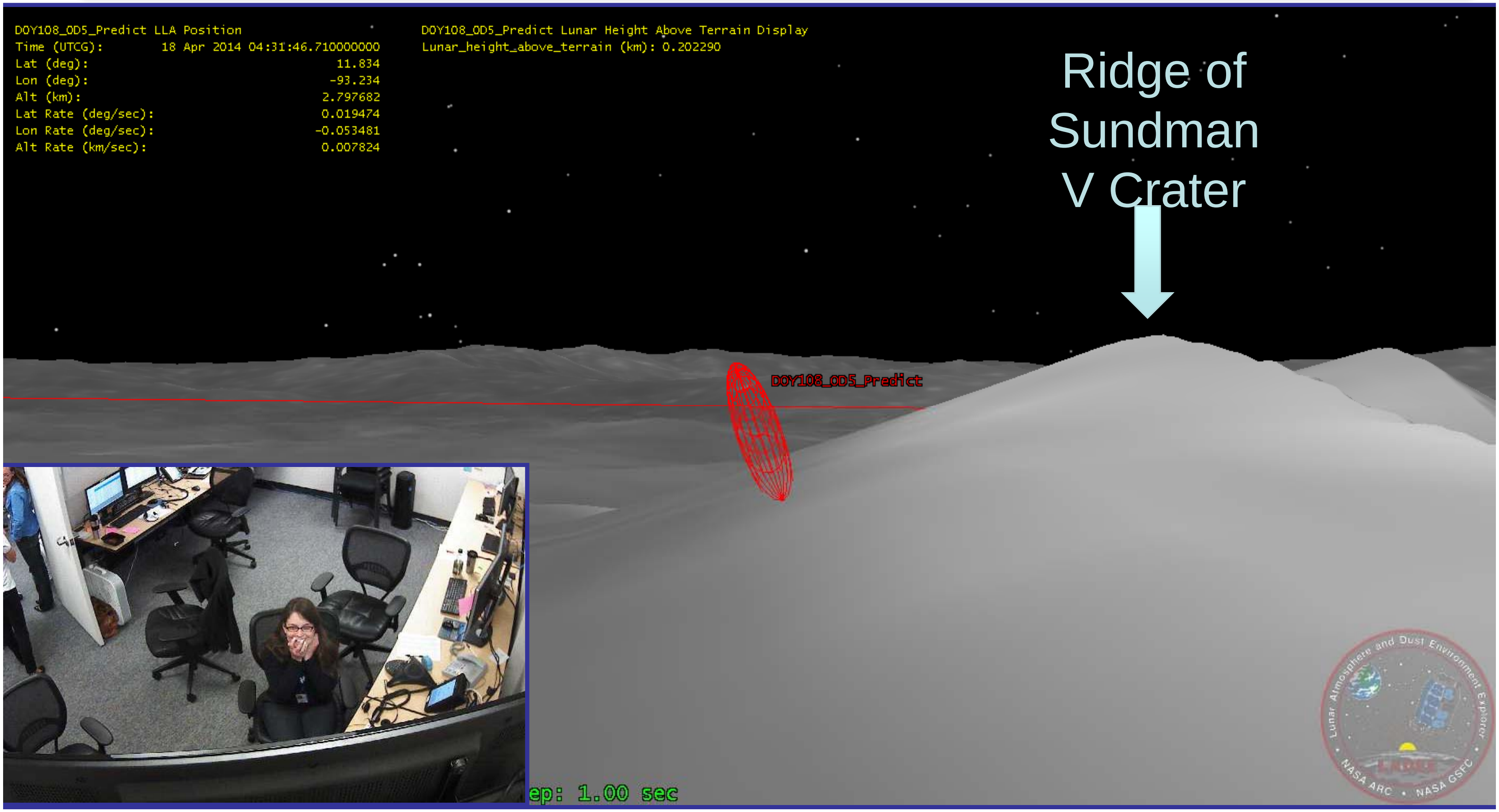
```
Preliminary_Definitive_DOY108 LLA Position
Time (UTCG):         18 Apr 2014 02:40:56.774700000
Lat (deg):                                  12.245
Lon (deg):                                 -93.475
Alt (km):                                 3.016872
Lat Rate (deg/sec):                       0.019260
Lon Rate (deg/sec):                      -0.053628
Alt Rate (km/sec):                        0.008284
```

```
Preliminary_Definitive_DOY108 Lunar Height Above Terrain Display
Lunar_height_above_terrain (km): 0.291651
```

Sundman
V Crater
Preliminary_Definitive_DOY108
e Step: 1.00 sec
Lunar Atmosphere and Dust Environment Explorer
LADEE
NASA ARC • NASA GSFC

Actual Height Above Terrain @ Closest Distance: 0.292 km

Francis: IMPACT!



```
DOY108_OD5_Predict LLA Position
Time (UTCG):         18 Apr 2014 04:31:46.710000000
Lat (deg):                                   11.834
Lon (deg):                                  -93.234
Alt (km):                                  2.797682
Lat Rate (deg/sec):                        0.019474
Lon Rate (deg/sec):                       -0.053481
Alt Rate (km/sec):                         0.007824
```

```
DOY108_OD5_Predict Lunar Height Above Terrain Display
Lunar_height_above_terrain (km): 0.202290
```

Ridge of Sundman V Crater

Official Reported Impact Time: 04:31:44 – 04:31:47

Impact Location: 11.8407º latitude, -93.2521º longitude

(estimate accounts for in-track, radial and cross-track position uncertainty and lunar terrain uncertainty)

Final Status

LADEE Mission

•Successful Laser Communications demonstration: 622Mbs downlink rate. Very useful to be able to download a SDRAM partition in less than 2 minutes.

•188 days of lunar orbit, with approximately 200% of planned science data returned to the earth.

•Lowest science operations conducted around 2 Km over the moon’s surface

LADEE Flight Software

•Table uploads performed (ATS, RTS, FM, Thermal updates & defect reduction)

•2 software patches to account for emergent star tracker behavior

•1 unanticipated reboot (Interrupt Handling)

•Upload and operation of new software load

•Team recertified for CMMI level 2 in May 2013

LADEE FSW: Enabling a successful mission!