

LADEE Multi-Domain Simulation

Danilo Viazzo

Cummings Aerospace

danilo.viazzo@cummingsaerospace.com



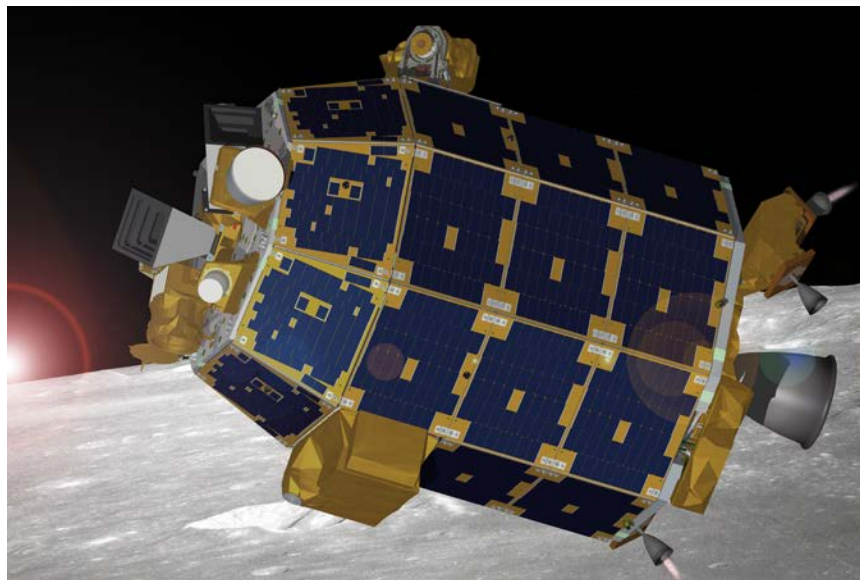
a Native American Woman-Owned Small Business

Nathan Benz

Millennium Engineering and Integration

nbenz@meicompany.com



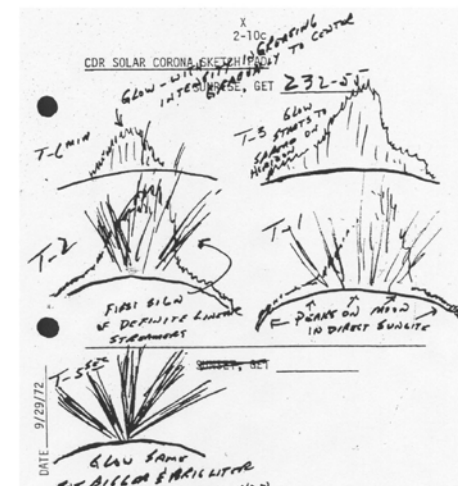


- Lunar Atmosphere and Dust Environment Explorer (LADEE) is a NASA mission that will orbit the Moon and its main objective is to characterize the atmosphere and lunar dust environment.
 - Low cost, minimal complexity and rapidly prototyped “common bus” design.
 - Model-Based Software Development

- Specific objectives are:
 - Determine the global density, composition, and time variability of the lunar atmosphere;
 - Confirm the Apollo astronaut sightings of dust jumps and diffuse emission
 - Laser Communications Demonstration: 622 Mbs Record download rate from the Moon!



Clementine spacecraft image of moon dust corona



Gene Cernan's drawings of the lunar sunrise



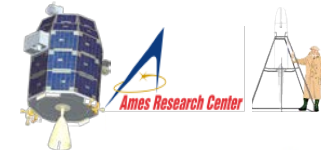
Outline



- Model Based Development
- Simulation Objectives
- Simulation Language and Structure
- Multi-Domain Elements
 - Simulation Interface
 - Electrical System Model
 - Thermal Model
- Lessons Learned



Model Based Development

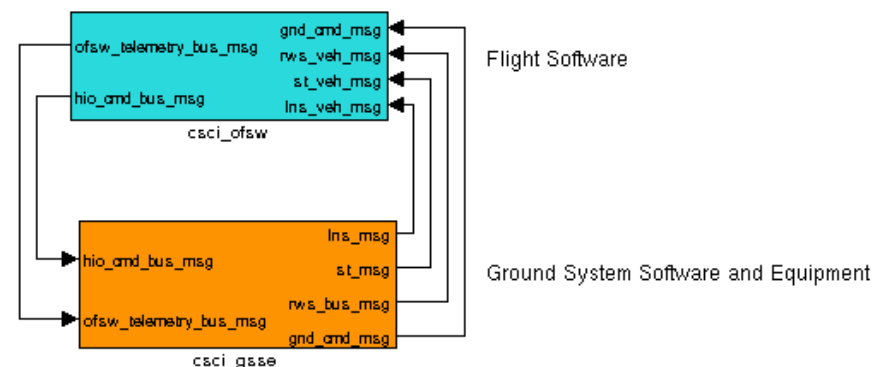


- Scope
 - Onboard Flight Software (Class B)
 - Support Software and Simulators (Class C)
 - Integration of FSW with avionics



LADEE

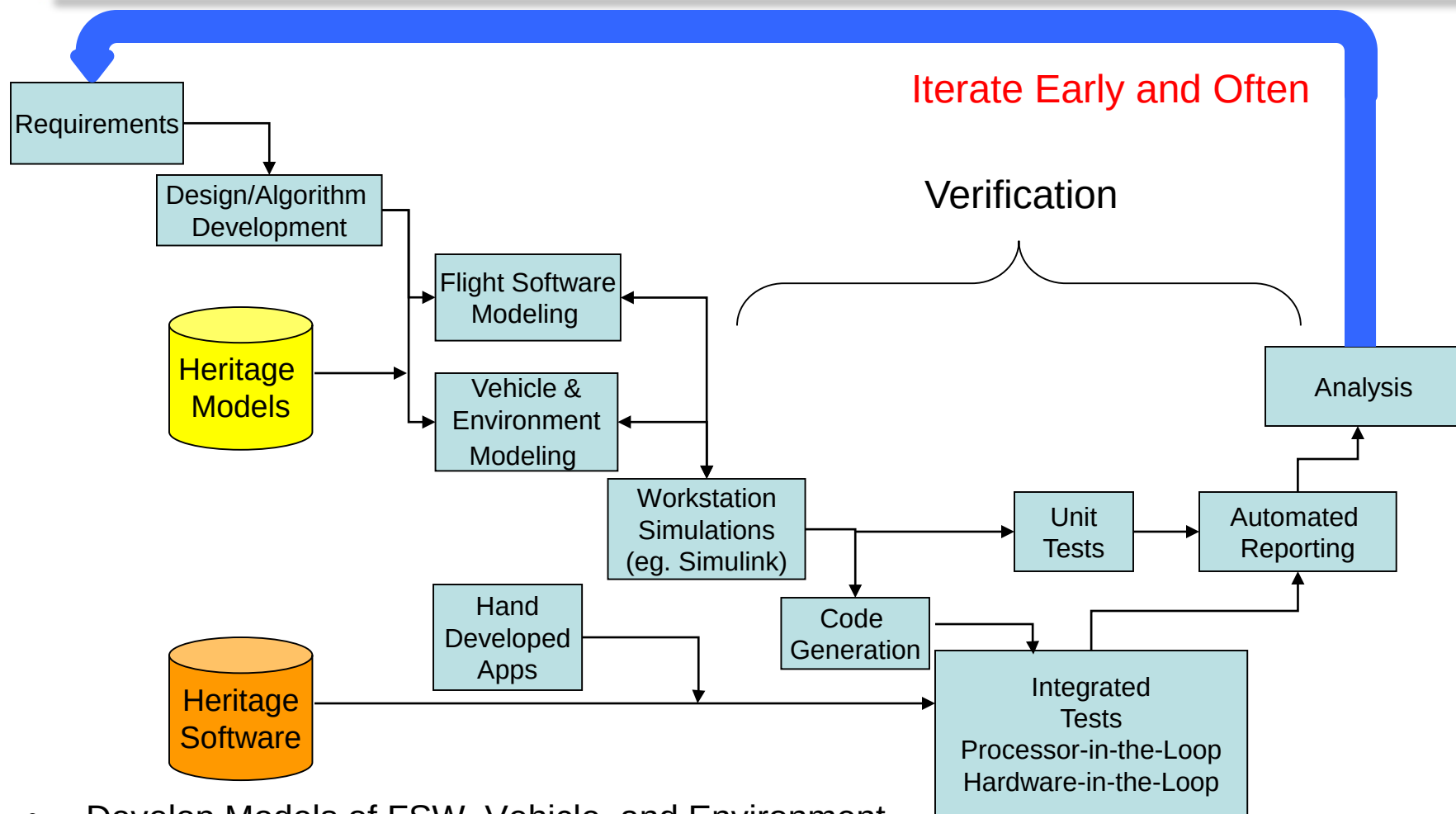
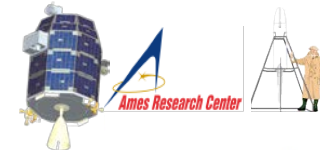
- Guiding Documents
 - NPR 7150.2 Software Engineering Requirements
 - CMMI Level 2
 - NASA-STD-8739.8 NASA Software Assurance Standard
- Development Approach



- Model Based Development Paradigm (prototyped process using a “Hover Test Vehicle”)
- 5 Incremental Software Builds, 2 Major Releases, 4 final sub-releases
 - 5.1: Defects found by I&T and 3DOF
 - 5.2: Defects found by Mission Operations Testing
 - 5.3: Final RTS set for Golden Load
 - 5.4: Platinum Load, uploaded during flight
- Leverage Heritage Software
 - GOTS: GSFC OSAL, cFE, cFS, ITOS
 - MOTS: Broad Reach Drivers
 - COTS: , VxWorks, Mathworks Matlab/Simulink & associated toolboxes



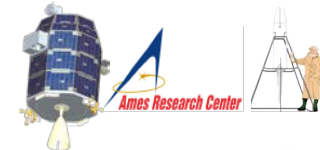
Model Based Development



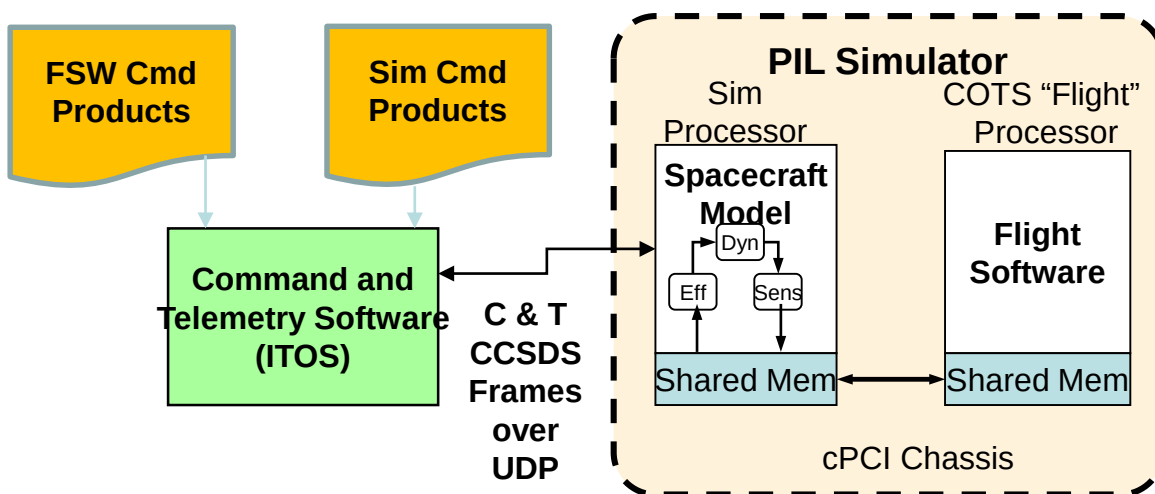
- Develop Models of FSW, Vehicle, and Environment
- Automatically generate High-Level Control Software
- Integrate with hand-written and heritage software.
- Iterate while increasing fidelity of tests – Workstation Sim (WSIM), Processor-In-The-Loop (PIL), Hardware-in-the-Loop (HIL)
- Automated self-documenting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



Simulation Objectives



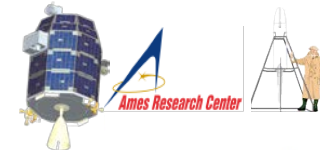
- Single Source Of Simulink Models
 - Superset of Models for Workstation Simulation
 - Onboard Clock Model
 - Onboard Stored Command Sequences
 - Spacecraft Commanding
 - Telemetry Collection
- Support Flight Software Development and Testing
 - Non Real-Time
 - Workstation Simulation
 - Monte Carlo
 - Real-Time
 - Processor In The Loop (PIL)
 - Hardware In The Loop (HIL)



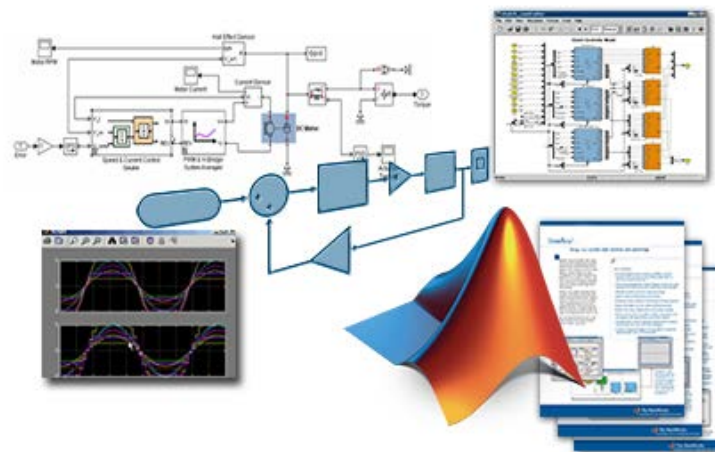
- Support Mission Operations
 - Training
 - Flight



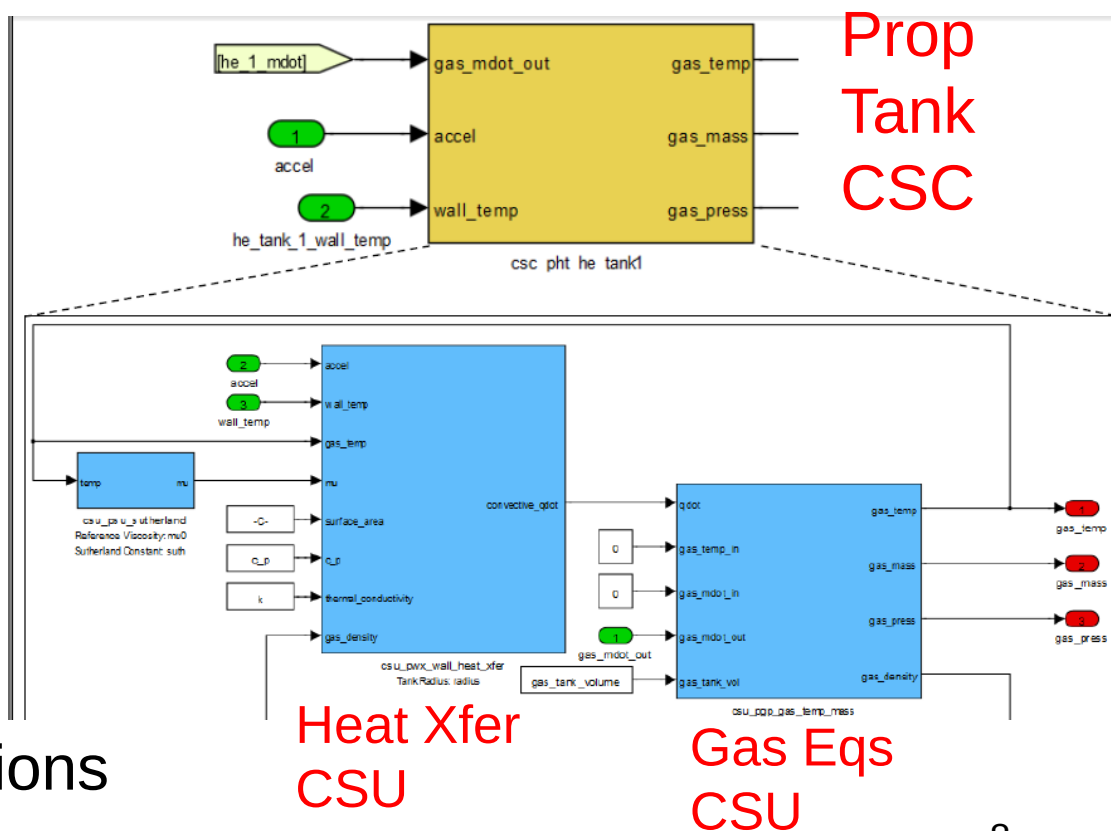
Simulation Language



- Simulation Tools
 - MATLAB/Simulink R2010b
 - Real-Time Workshop Embedded Coder
- Simulink
 - Native Blocks
 - Embedded MATLAB Blocks (MATLAB Function Blocks)
- MATLAB Scripts
- CSV Based Spreadsheet
 - Interface Definitions (Non-Virtual Bus Objects)
 - Subsystem Configuration Data
- External Data Files

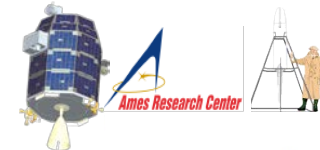


- CSCI (Configuration Item)
 - Flight Software
 - Simulated Vehicle and Environment
- CSC (Component)
 - Vehicle Dynamics
 - Sensors
 - Actuators
- CSU (Unit)
 - Time Model
 - Gravity Model
- Utility Libraries
 - Quaternion Operations





Simulation Interface



- Goal 1: Single Interface To Control Simulation
 - Workstation Simulation (WSIM)
 - PIL/HIL
- Goal 2: Simulated Spacecraft Command Interface Consistent With Ground Interface
 - Ground Commands
 - Onboard Command Sequences
- Implementation
 - MATLAB Based Parser for STOL Command Sequences
 - Spacecraft Command Sequences
 - Embedded Simulation Directives to Initialize Parameters
 - Both reduced to time based table for WSIM execution
 - Tunable Parameters
 - MATLAB Initialization Scripts to Define Default Values
 - MATLAB Override Scripts for WSIM
 - Memory Poke Mechanism to Override Parameters in PIL/HIL

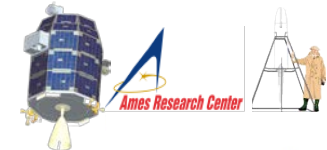
```
2013-223-02:00:00 /EVS_USRMSG MESSAGE="Preparing for midnight obser
; change the fine pointing gains to fast for science operations
2013-223-02:00:00 /acs_select_fpc_cont_gain_set gain_set=FPC_FAST

; NMS midnight observation starting quaternion
2013-223-02:05:27 /acs_set_waypt_rt cmd_end_time=300, omega_x=0.000

; Reset the nms and ldex error counters so we can keep track
2013-223-02:09:16 /nmsio_resetctr
2013-223-02:09:16 /ldexio_resetctr

; Enable RTS's that will be used for payloads
2013-223-02:09:22 /NMSIO_RESETCTRS
2013-223-02:09:22 /pcs_man_cmd_id switch_id=PCS_NMS_PWR, state=PCS_
2013-223-02:09:24 /nmsio_activate
2013-223-02:09:44 /nmsio_ena_timemsg

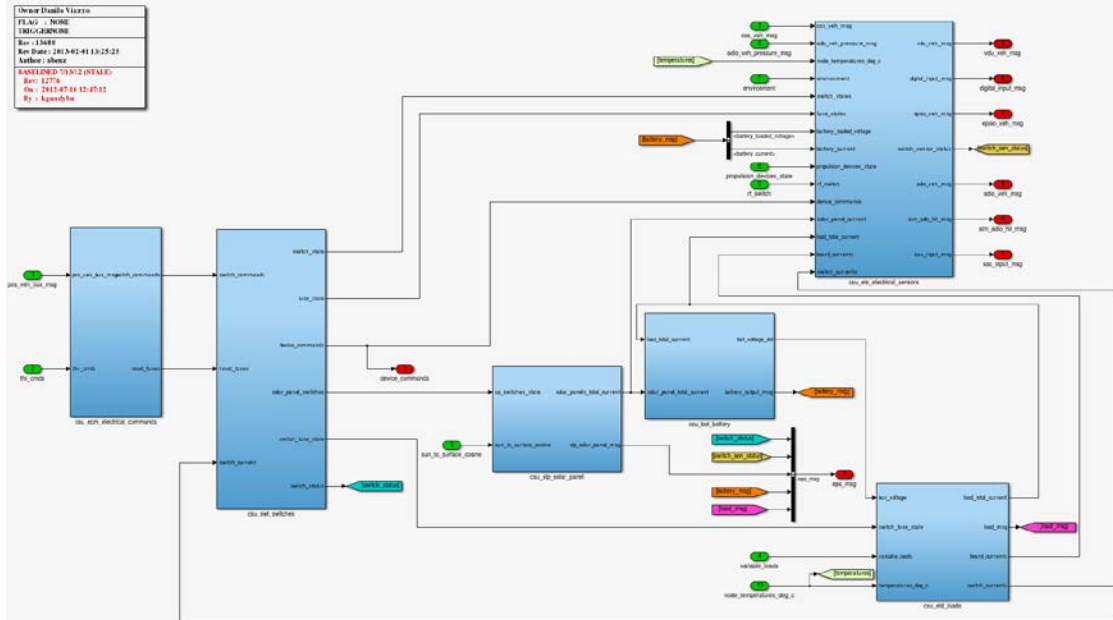
;Enable the NMS Fault Management
2013-223-02:09:44 /LC_SETAPSTATE APNumber=182, NewAPState=PASSIVE
2013-223-02:09:44 /LC_SETAPSTATE APNumber=183, NewAPState=PASSIVE
2013-223-02:09:44 /LC_SETAPSTATE APNumber=184, NewAPState=ACTIVE
2013-223-02:09:44 /LC_SETAPSTATE APNumber=185, NewAPState=ACTIVE
```



Electrical System Model

- Goal 1: Model the State of Charge of The Battery

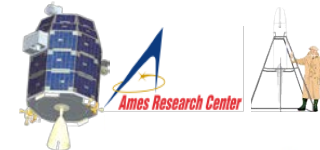
- Battery Model
- Solar Panel Model
- Switches Model
- Load Model



- Goal 2: Model the Switch Command Interface and Current/Voltage Sensor to Support Development and Testing of the Onboard Electrical Load Control Software
- Goal 3: Support Injection of Failures



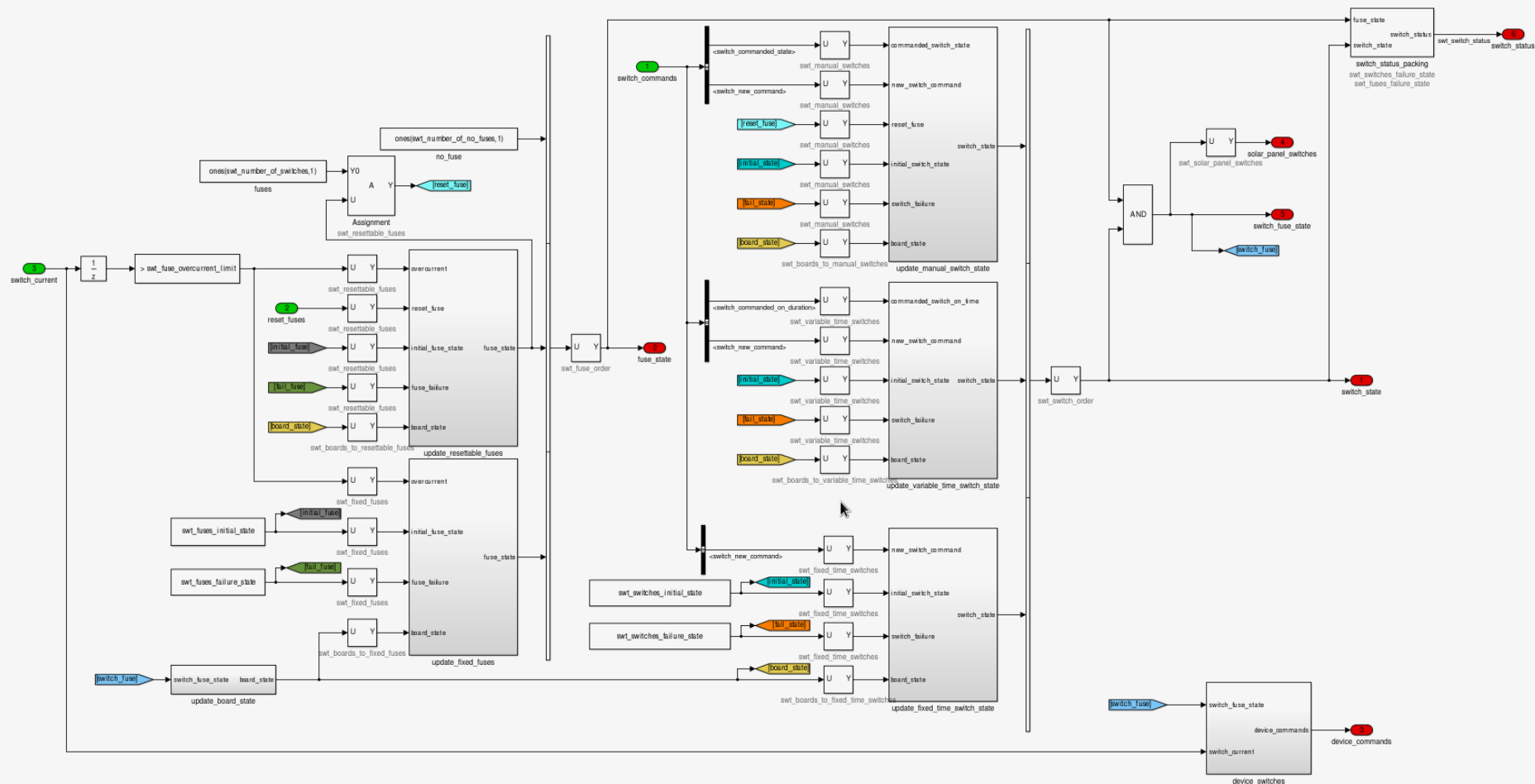
Electrical System Model

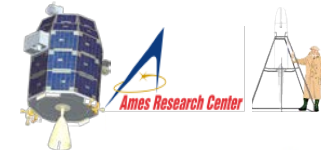


- Implementation

- Model was developed prior to completion of the design for the electrical system
- Battery model focused on integration of inflow and outflow of current
- Solar Panels modeled by section (30 section)
- Switches, Fuses, Loads model by type and vectorized
- Designed to automatically reconfigure based on external configuration file
 - Command signal routing to components and back reduced to tables
 - Vectorized component organized in stages
- Vectorized components built with failure states (on/off)

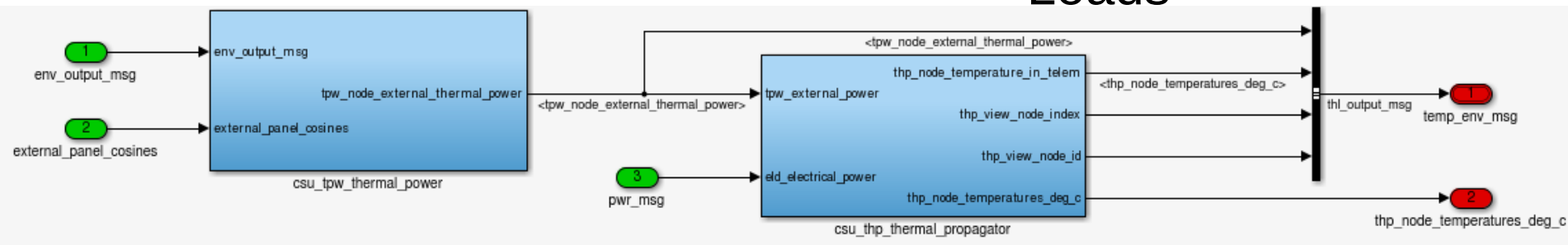
Electrical System Model





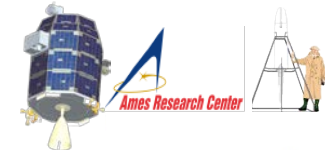
Thermal Model

- Goal: Model The Response Of The Thermal Sensors to External and Internal Heat Sources to Support Development and Testing of the Onboard Thermal Control Software
 - External Heat Sources
 - Sun
 - Moon Radiation
 - Moon Albedo
 - Thermal Propagation
 - Conduction
 - Radiation
 - Internal Heat Sources
 - Heater
 - Loads





Thermal Model



- Implementation

- Lumped Mass Thermal Model

- Node and transport properties defined by external file generated by thermal modeling tool
 - Resolution/Fidelity of model determined by input file selection
 - Automatic nodal mapping by node ID to external spacecraft surface, to internal heat sources, and to thermal sensors

- Thermal Propagation at 10Hz

- Thermal model input files tested for stability at 10Hz
 - Supported 400+ nodes model propagation in real-time

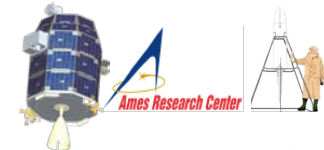
- Heat Sources

- External heat sources tied to vehicle orientation relative to Moon and Sun and eclipses
 - Internal heat sources tied to switch/load currents

```
1 function t_new = update_thermal(t_old, t_old_4, q_external, dt,
2 n_nodes, n_diffusion_nodes, n_linear_links, n_radiation_links,
3 n_arithmetic_nodes, a_coeff, n_arithmetic_linear_links, n_arithmetic_q_external, q_ext,
4 n_boundary_nodes)
5
6 %#eml
7 % Initialize new node temperatures
8 t_new = zeros(n_nodes,1);
9
10 %% Calculate the temperature for the diffusor nodes
11
12 % Initialize the diffusor node heat flux
13 q_temp = zeros(n_nodes,1);
14
15 % Handle the Diffusion Nodes
16
17 for idx = 1:n_linear_links
18     q_transport = (t_old(conductor_data(idx,1)) - t_old(conductor_data(idx,2))) / a_coeff;
19     q_temp(conductor_data(idx,1)) = q_temp(conductor_data(idx,1)) + q_transport;
20     q_temp(conductor_data(idx,2)) = q_temp(conductor_data(idx,2)) + q_transport;
21 end
22
23 for idx = (n_linear_links + 1) : (n_linear_links + n_radiation_links)
24     q_transport = (t_old_4(conductor_data(idx,1)) - t_old_4(conductor_data(idx,2))) / a_coeff;
25     q_temp(conductor_data(idx,1)) = q_temp(conductor_data(idx,1)) + q_transport;
26     q_temp(conductor_data(idx,2)) = q_temp(conductor_data(idx,2)) + q_transport;
27 end
28
29 % Process the external heat sources
30 for idx = 1:n_q_external
31     q_temp(q_external_map(idx,1)) = q_temp(q_external_map(idx,1)) + q_external(idx);
32 end
33
34 % Update the temperature for the diffusor nodes
35 for idx = 1:n_diffusion_nodes
36     t_new(idx) = t_old(idx) + dt/capacitance(idx)*q_temp(idx);
37 end
38
39 % Initialize the arithmetic node heat flux
40 q_temp = zeros(n_nodes,1);
41
```



Lessons Learned



- Command Interface
 - Development of a parser for STOL scripts for the simulation resulted in single source for test configuration
 - This also enabled the simulation to be used for mission ops training prior to the mission and command validation during mission operations
- Electrical System Model
 - Simplified electrical model was required to maintain real time performance
 - Design modularity and configurability minimized the time spent updating the model to match the actual configuration
 - Fault injection consideration in the initial design enable broad range of training scenario for mission operation personnel
- Thermal Model
 - Lumped mass thermal model proved sufficient for test and training purposes
 - Easy configurability of thermal model allowed user use smaller thermal databases for workstation simulation run that did not require consideration of thermal effects

Lessons Learned

- Overall
 - Multi-Domain simulations can provide broader application opportunities across a life-cycle, thus potentially reducing the cost of maintaining independent specialized tools
 - Multi-Domain simulations can be designed so as to minimize the performance hit by controlling the scope/fidelity of the models associated with each domain

