V&V for Model-Based Software Development

Karen Gundy-Burlet, Ph.D.
NASA-Ames Research Center
Moffett Field, CA

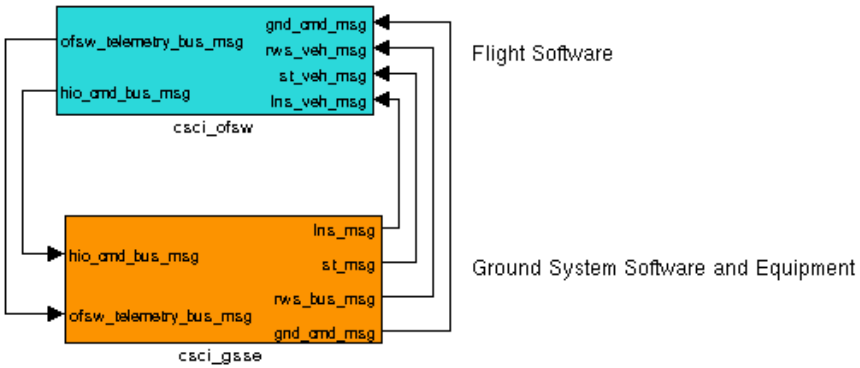
Project Overview

- 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. Specific objectives are:
  - Determine the global density, composition, and time variability of the lunar atmosphere;
  - Confirm the Apollo astronaut sightings of diffuse emission at 10s of km above the surface were Na glow or dust and;
  - Document the dust environment (size-frequency).
- The mission will be also be testing a new spacecraft architecture called the 'Modular Common Bus' with plug-and-play components and model-based software development process.

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

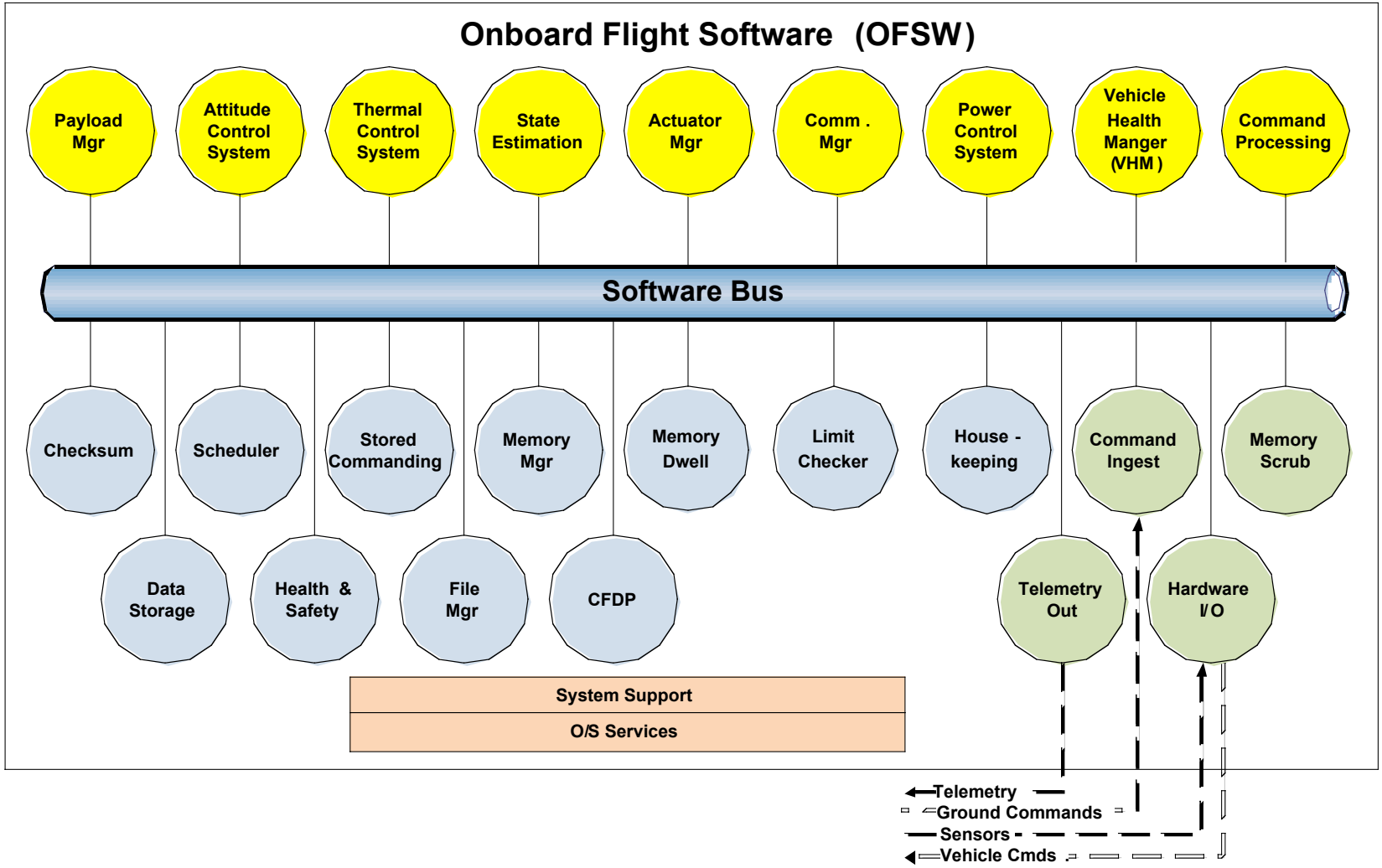
LADEE



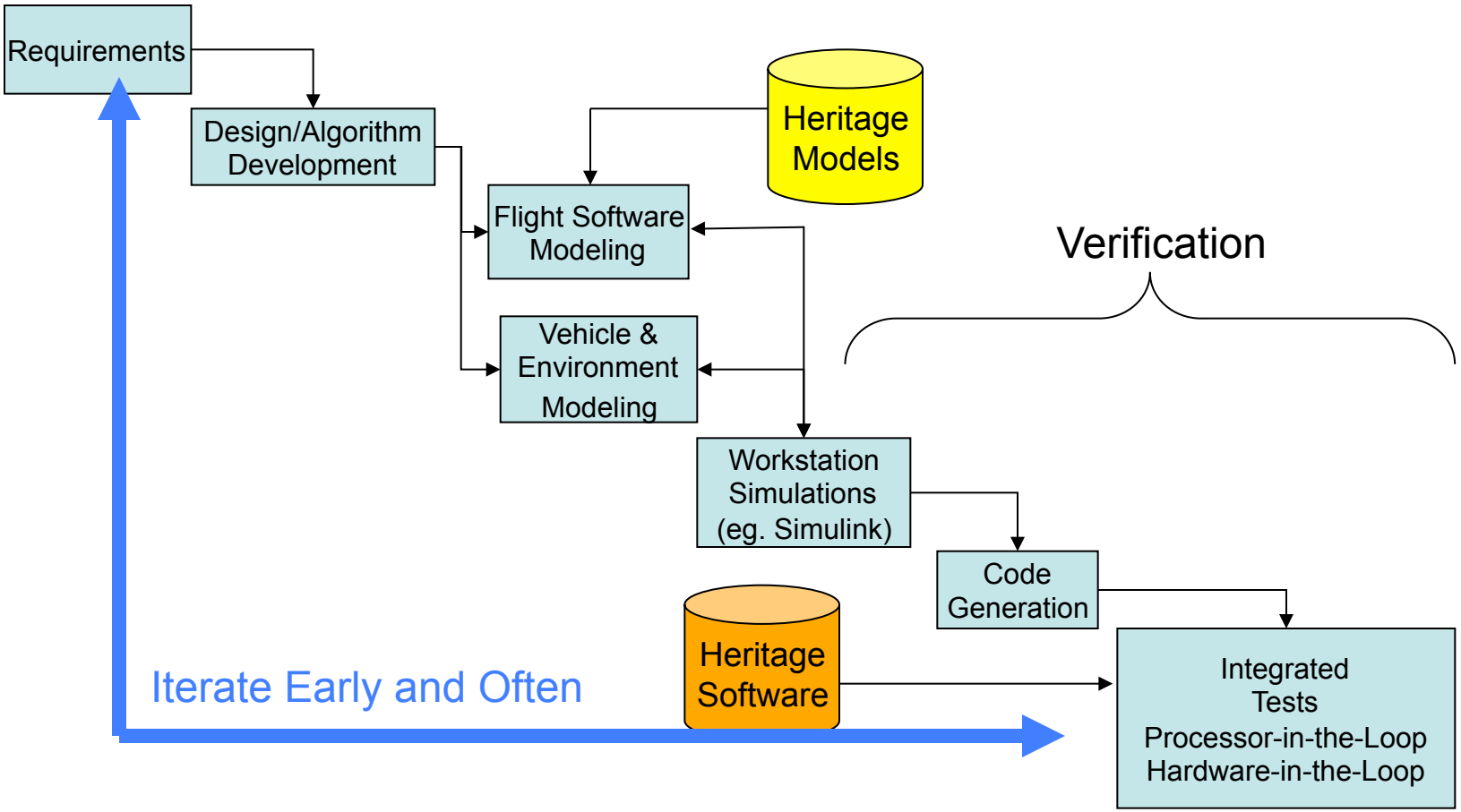
- Development Approach
  - Model Based Development Paradigm (prototyped process using a “Hover Test Vehicle”)
  - 4 Incremental Software Builds, 2 Major Releases
- Leverage Heritage Software
  - GSFC OSAL, cFE, cFS, ITOS
  - Broad Reach Drivers, VxWorks
  - Mathworks Matlab/Simulink & associated toolboxes

Flight Software Overview

- Approach
  - Layered Architecture with mixture of Model-Based auto-coded software, GOTS/COTS and minimal hand-developed glue code.



Model Based Development

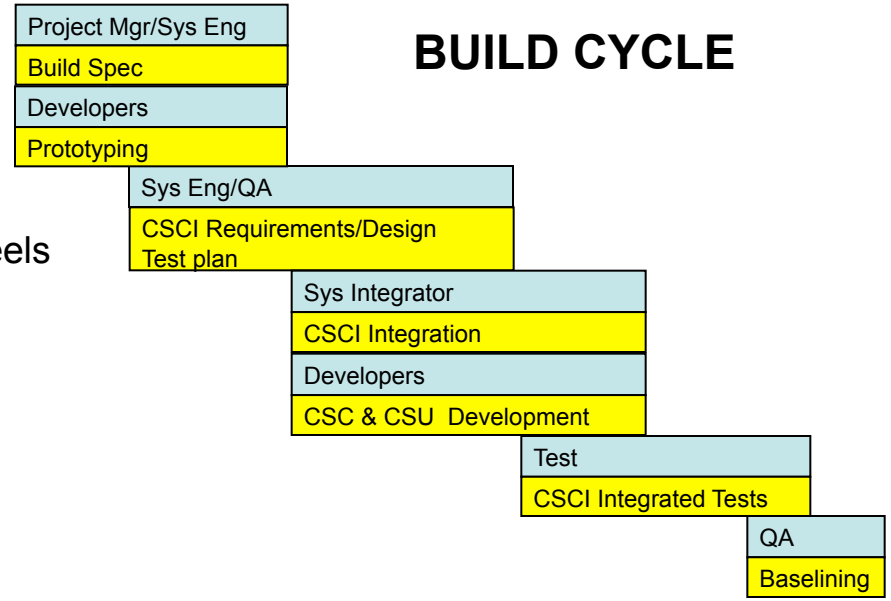


- Develop Models of FSW, Vehicle, and Environment in Simulink
- Automatically generate Software using RTW/EC.
- Integrate with hand-written and heritage software.
- Iterate while increasing fidelity of tests – Workstation Sim (WSIM), Processor-In-The-Loop (PIL), Hardware-in-the-Loop (HIL)

Build/Release Cycles

- Build 1
  - Attitude Control with Reaction Wheels
  - Basic & Stored Cmd
  - CCSDS Telemetry Packets
- Build 2
  - Orbital Maneuvers
  - CCSDS Frame Telemetry
  - COP-1 and CFDP Protocols
  - Data Storage
- Build 3
  - Trajectory Correction Maneuvers
  - Health Monitoring
  - Payload Data
  - Nominal End to End Ops
- Build 4
  - Off Nominal Recovery & Safing

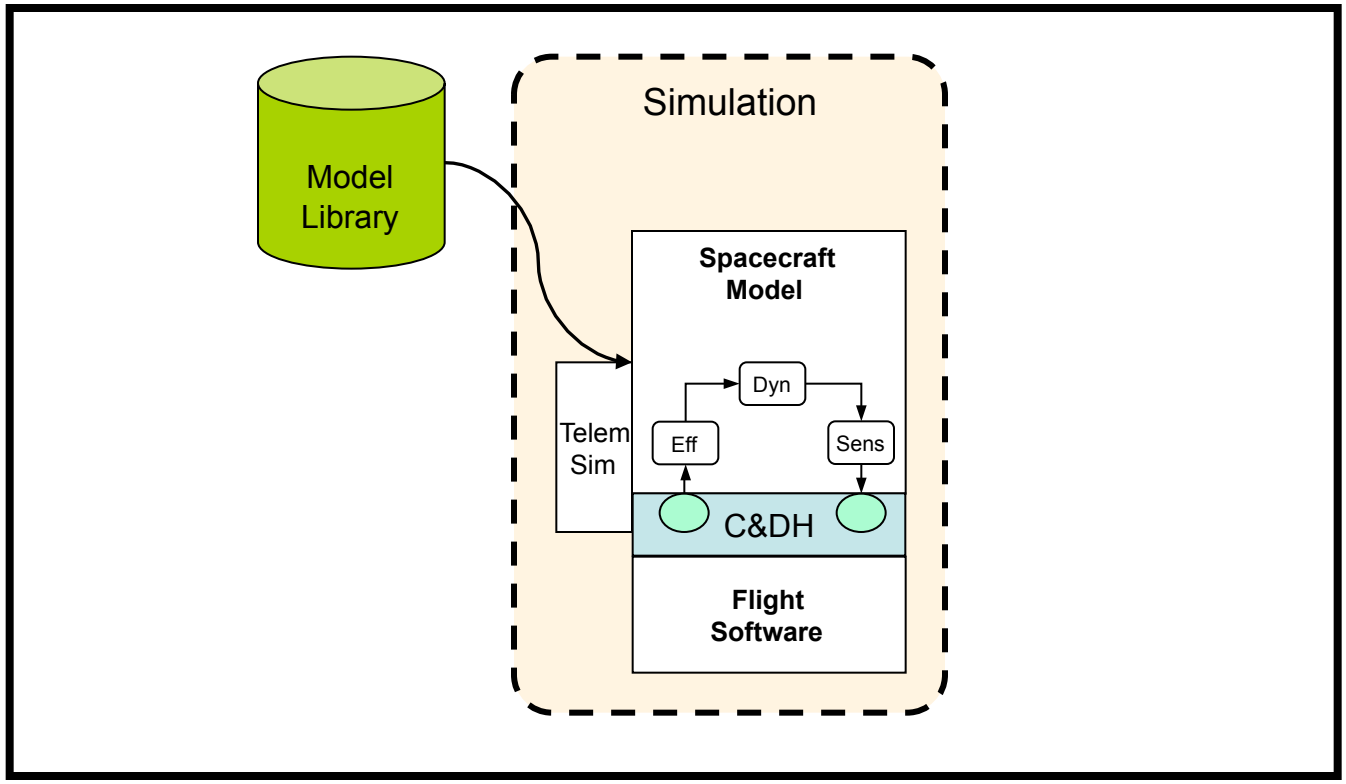
BUILD CYCLE



Release 1
-Ready for SC Integration at ARC

Release 2
-Ready for EV Tests at GSFC

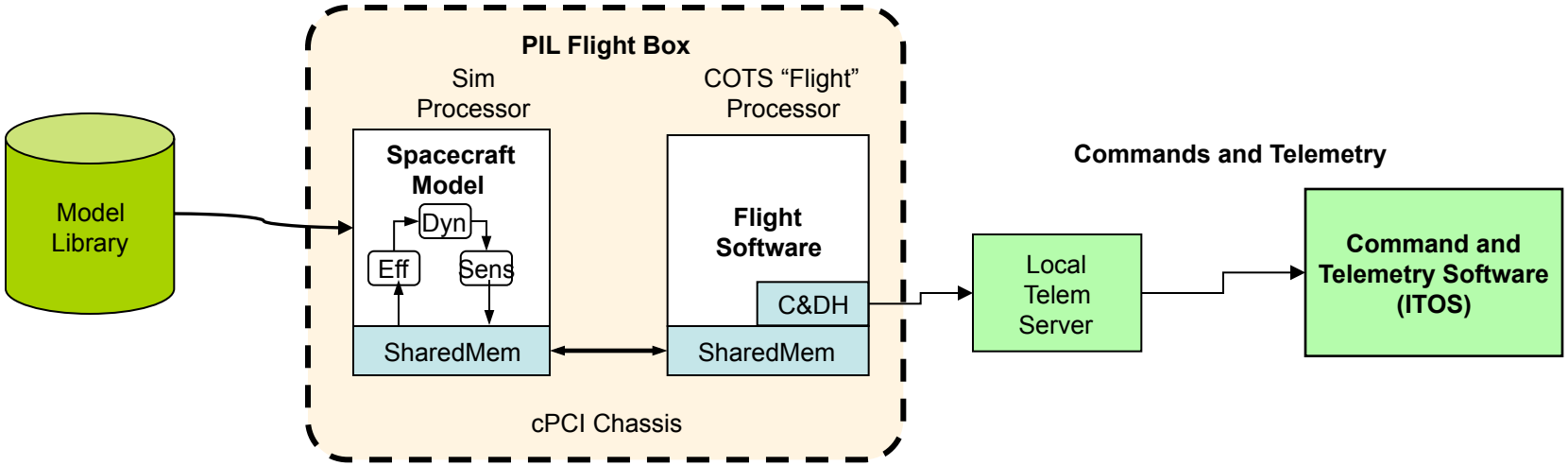
Workstation Simulation



Local Workstation

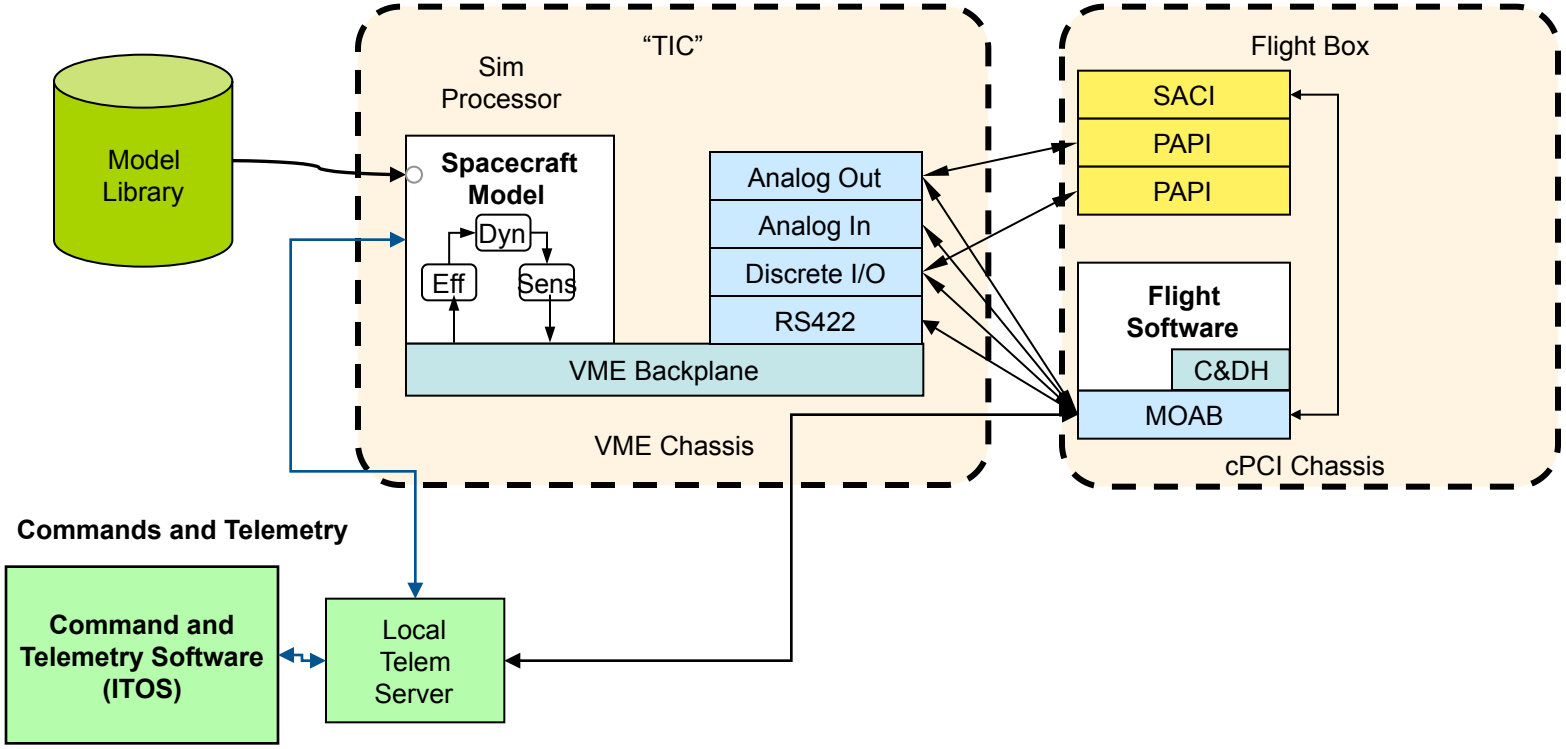
- Simulink Only (No Autocode)
- Early in development process
- Algorithm Development
- Requirements Analysis

Processor-in-the-Loop Simulation



- Models autocoded and running on RT processors
- Inexpensive "flight-like" processor
- Tests autocoding process & integration with C&DH software
- Integration with Telemetry Software allows early development/testing of downlink
- Used for code size and initial resource utilization metrics

Hardware-in-the-Loop Simulation



- Flight code runs on Flight Avionics EDU
- Provides testing of FSW with Avionics I/O
- Definitive answers on resource utilization
- Highest fidelity simulations for verification/validation

Model Verification Approach

- Design, Requirements, Functional Tests and Simulink Models are simultaneously prototyped in the WSIM environment
  - Unit test suite to verify low-level requirements.
  - Integrated test suite to verify system/subsystem compliance with associated requirements.
  - Naming Conventions utilized to link Models, Designs, Requirements and Unit Test Suite.
  - Reporting system
    - Uses Matlab/Simulink Report Generator: Custom-built reporting scripts for producing as-built vs. as-designed documentation, requirements verification reports, interface documentation.
    - 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
  - Custom Model Advisor checks to ensure compliance of Simulink models with Common Bus Modeling Guidelines.
  - Peer Review/Inspection program to assure quality of all artifacts

Requirements Verification Report

Requirements and Test Suite Results

ID Number	csu_sto_st_fov_obstr_0001
Title/Synopsis	STO ST FOV Obstruction Calculate Angular Distance to Obstructions CSU
System Class	STO ST FOV Obstruction Calculate Angular Distance to Obstructions CSU
Req. Type	Functional
Parent	csc_star_tracker_0001
Children	None
Priority	High
Requirement Description	The ST FOV Obstruction CSU shall calculate the angle between the star tracker boresight and the center of possible field of view obstructions including the Earth, Moon, and Sun.
Rationale	Used for calculating when a star tracker field of view obstruction occurs.
Verification	Test (csu_sto_st_fov_obstr_test_a.m)

ID Number	csu_sto_st_fov_obstr_0002
Title/Synopsis	STO ST FOV Obstruction Calculate Angular Size of Obstructions CSU
System Class	STO ST FOV Obstruction Calculate Angular Size of Obstructions CSU
Req. Type	Functional
Parent	csc_star_tracker_0001
Children	None
Priority	High
Requirement Description	The ST FOV Obstruction CSU shall calculate the angular size of possible field of view obstructions including the Earth, Moon, and Sun.
Rationale	Used for calculating when a star tracker field of view obstruction occurs.
Verification	Test (csu_sto_st_fov_obstr_test_a.m)

Unit Test for csu_sto_st_fov_obstr.

```
 sloc count for csu_sto_st_fov_obstr_test.m is 46


*** Beginning st_fov_obstr unit test ...

Max error in Test 1/2: 0 (PASS Threshold = 0)
Max error in Test 2/2: 0 (PASS Threshold = 0)

--- All Tests Passed ---
```

Requirements Verification Report

Chapter 88. Summary Data for Test Suite

The total number of libraries in this model is 88. Of these libraries, 40 had missing requirements, 18 were missing their unit test suite. and 19 were missing their integrated test suite. The total number of functional requirements in the build is 320 The test suite drivers contained 7745 SLOC. The following gives summary data for the unit test suite:

```
Unit Tests Passed: 190
Unit Tests Failed: 10
```

The following is a list of files with improper naming conventions:

```
csu_svi_sim_visual_interface.mdl
counter.mdl
dtu_st_lib_dtu.mdl
documentation_library.mdl
```

The following is a list of libraries with missing requirements:

```
'util_chebyshev_req.html'

'util_clock_req.html'

'util_noise_sim_req.html'

'util_tf_req.html'
```

Modeling Guidelines

NASA MODELING GUIDELINES USING MATLAB®, Simulink®, and Stateflow®

Version 1.0

Based on the MathWorks Automotive Advisory Board (MAAB) Modeling Guidelines Version 2.0 (July 27th, 2007)

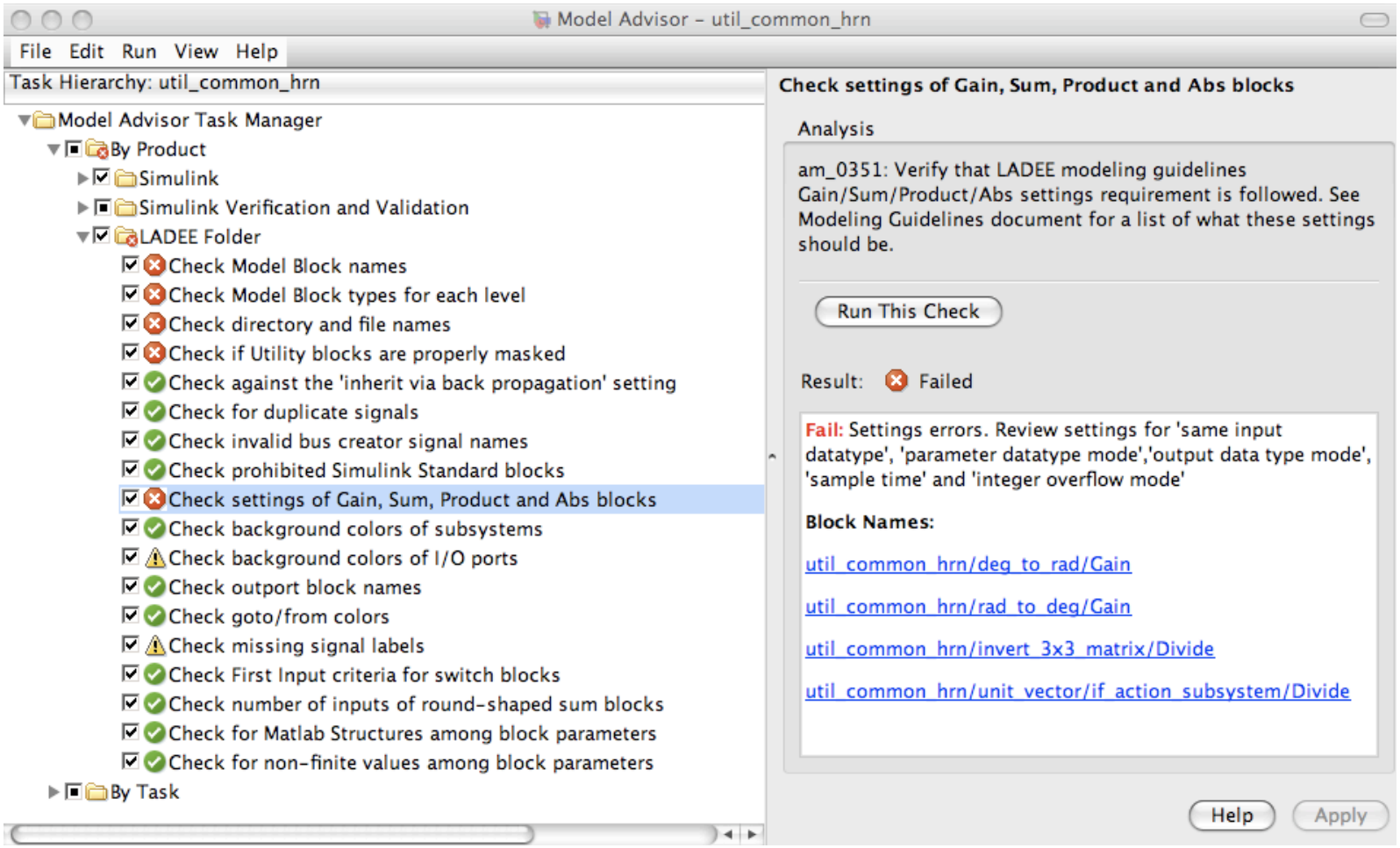
April 3rd, 2008

1

3

Model Advisor

- Customized checks enforcing our modeling guidelines



System V&V for Nominal Performance

- Utilize delivered GOTS/COTS unit test suites and compare with regression data.
- Comparison of modeled behavior on WSIM with PIL results
- Coding Standards.
- Static Analysis (Coverity, Klocwork) to ensure correctness of auto-generated code, hand-code & leveraged COTS/GOTS products.
- Investigating Mathworks Polyspace as adjunct to other static analysis tools.
- Validation of system nominal performance through scenario test scripts derived from ConOps document.
- Peer Review/Inspection program to assure quality of all artifacts

Problem Resolution & Reporting

- Utilizing TRAC – Wiki Based Software Management Tool
- Scheduled & Unscheduled Tasks Entered as Tickets
- Status and Change History tracked for all tickets.
- Estimated hours and Total hours to complete recorded
- Ticket completion requires approval of reporter

Ticket #49 (pending-approval defect: done)

RAD750 Integration into PIL

Opened 5 months ago
Last modified 3 months ago

Reported by:	hcannon	Owned by:	hcannon
Priority:	major	Milestone:	Build 1
Component:	6.5.4 GSE Software Development	Severity:	0 <= Hrs < 5
Keywords:		Cc:	
Estimated Number of Hours:	0.0		
Total Hours:	4h		

Description

Attachments

Attach file

Change History

Changed 5 months ago by schrista

- **status** changed from *new* to *accepted*

Reply

Changed 5 months ago by schrista

- **owner** changed from *schrista* to *hcannon*
- **status** changed from *accepted* to *pending-approval*
- **resolution** set to *done*

A PIL test was successfully conducted with Flight Software.

Reply

Changed 3 months ago by hcannon

- **hours** set to *0*
- **estimatedhours** set to *0*
- **status** changed from *pending-approval* to *assigned*
- **resolution** *done* deleted
- **owner** changed from *hcannon* to *schrista*

Please estimate hours to complete.

Reply

Changed 3 months ago by schrista

- **hours** changed from *0* to *4h*
- **owner** changed from *schrista* to *hcannon*
- **status** changed from *assigned* to *pending-approval*
- **resolution** set to *done*
- **totalhours** changed from *0* to *4h*

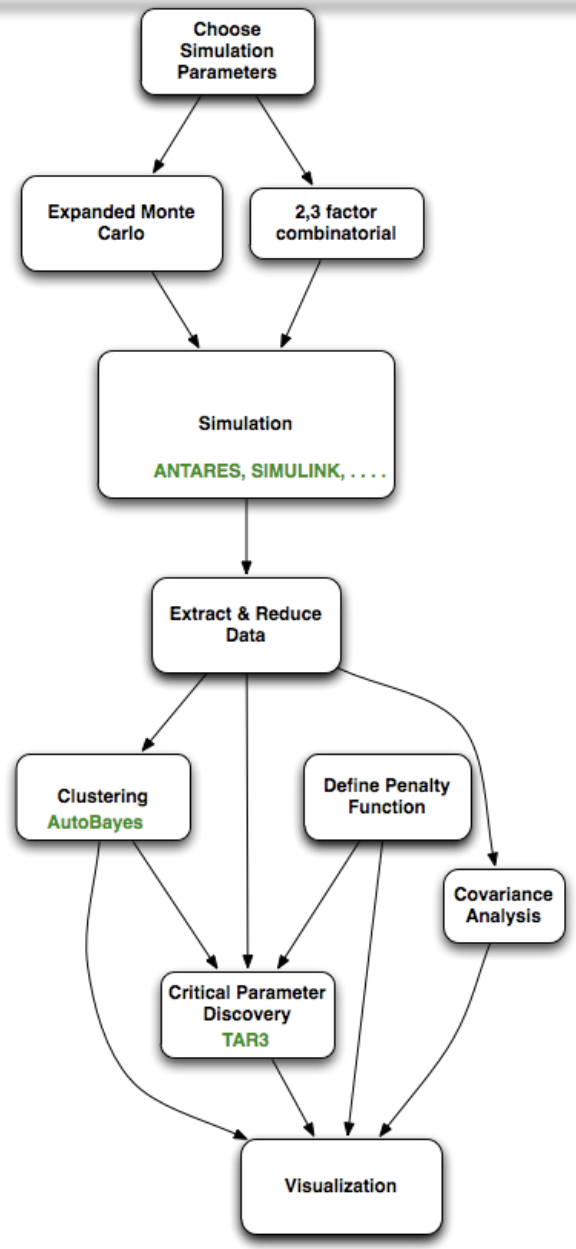
Reply

Add/Change #49 (RAD750 Integration into PIL)

Margins Analysis-System Off Nominal Performance

A multivariate analysis that

- generates inputs for high-fidelity simulations
- collect the outputs from many simulations
- then uses machine learning techniques on the output to find structure, sensitivities, failures (or other 'interesting' behaviors)
- identifies which input parameters and their ranges that caused the behaviors



Test Vectors: Combinatorial Explosion Problem

- **Consider a system having 13 test factors with 3 values each**
- **Exhaustive testing requires 3^{13} test cases (~1.5 million)**
- **Exhaustive testing is usually impractical, so how do we "cover" the test space in a *principled* way with a *tractable* number of tests that exercises important interactions in the system under test?**
- **Combinatorial design approach can do so with relatively few test cases**

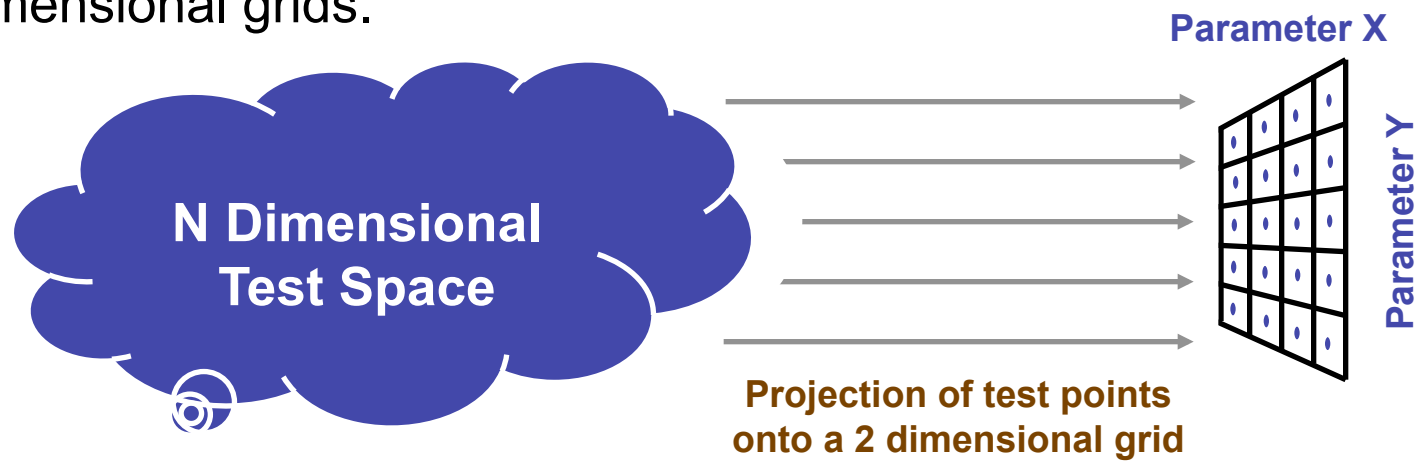
This test suite contains only 19 test cases, yet it exercises every pair-wise combination of 13 factors!

Test Factors

Test Cases	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13
1	0	0	0	0	1	2	2	1	0	2	0	1	2
2	1	0	2	2	2	0	1	0	1	0	1	2	0
3	2	0	1	1	0	1	0	2	2	1	2	0	1
4	1	2	0	1	1	0	0	1	1	1	0	0	0
5	0	2	2	0	0	1	1	2	0	0	1	1	1
6	2	2	0	2	2	2	2	2	2	2	2	2	2
7	0	1	1	2	1	0	2	0	0	1	2	1	0
8	1	1	0	0	0	2	1	0	2	2	0	0	1
9	2	1	2	1	1	2	1	1	1	0	1	2	2
10	1	2	1	0	2	1	2	1	1	2	2	1	1
11	2	1	2	0	2	0	0	2	0	2	0	2	0
12	0	2	1	1	2	2	0	0	0	0	0	0	2
13	0	1	2	2	0	0	2	1	2	0	1	0	2
14	1	1	0	2	1	1	0	2	0	0	0	1	2
15	2	2	0	1	1	1	2	0	2	2	1	1	0
16	0	0	1	0	0	2	1	2	1	1	1	2	0
17	2	2	2	2	2	1	1	0	2	1	2	2	2
18	0	1	2	2	1	0	0	2	0	0	2	2	1
19	1	0	0	0	1	2	0	0	0	1	1	0	0

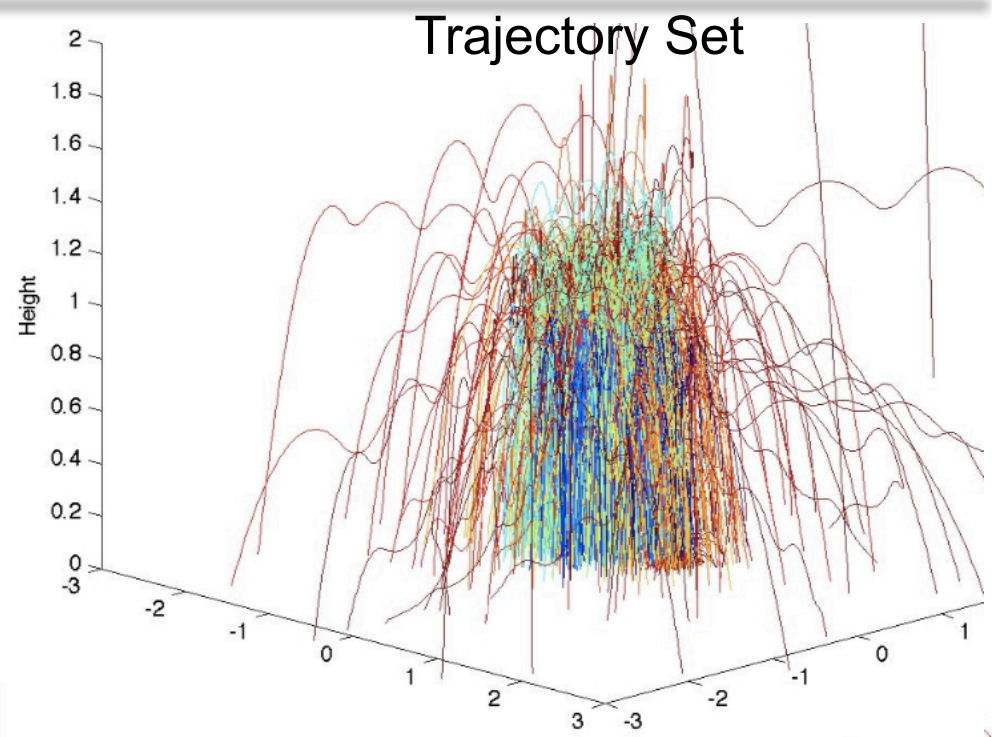
Combinatorial Test From a Graphical Perspective

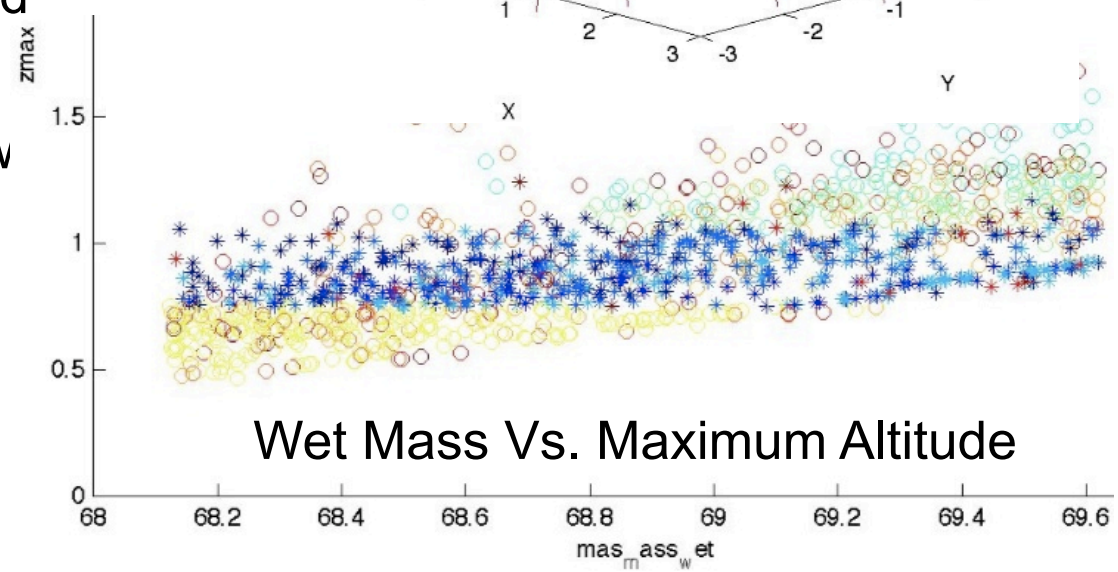
- For pair-wise coverage, every projection of the tests in an N dimensional test space onto a 2 factor plane fully covers the grid.
- Covering higher numbers of parameters projects to fully cover higher dimensional grids.



AUTOBAYES

- Undirected exploration of the dataset to find natural groupings/clusters
  - Algorithm is a multivariate mixture model (mixture of Gaussians here, now extended to other probability distributions).
  - Autocoded for inclusion as a matlab mex file from a high-level specification file.
- Colors indicate clusters, ranked by penalty function (blue = good, red = bad). Stars indicate success & circles show failures.
- Utility: Finding unexpected behaviors in the data.
- Weakness: Not obvious what the driving variables are for each cluster.





TAR-3

- Weighted-Class Minimal Contrast Set Association Rule Learner:
  - The dataset is classified according to some utility function(for instance, the penalty function, a failure class or a particular cluster)
  - Uses a stochastic search with pruning to determine the minimal parameter set that has the greatest influence on the behavior of the system.

Key Ranges for the Yellow Cluster (7)

68.1234 < Wet Mass < 68.6103

-1.1035 < Dry Ixy < -0.96766

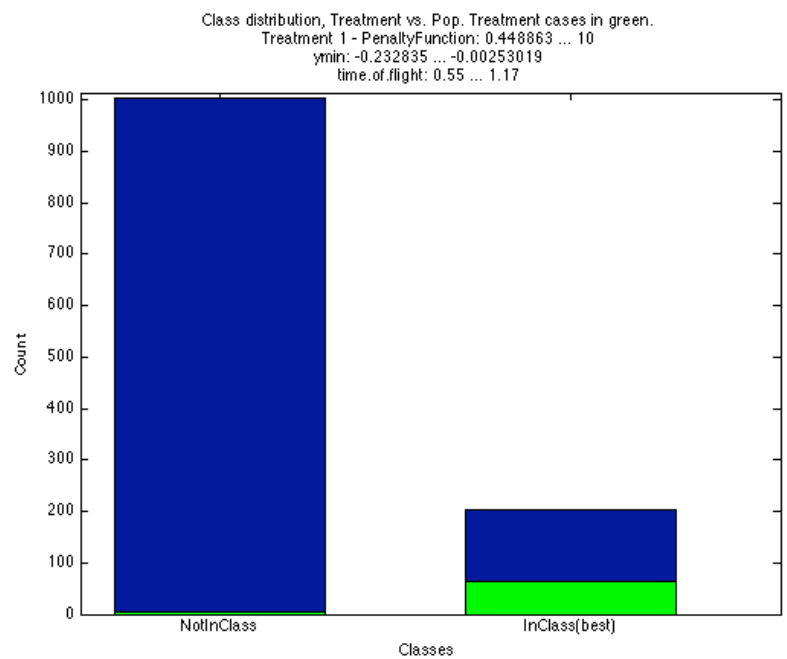
-0.670072 < Control parameter “a” < -0.335964

-0.154472 < Cg Z Dry location < -0.149611

Key Ranges to Minimize Penalty Function

68.6103 < Wet mass < 69.1325

0.657639 < Control parameter “b" < 0.999982



Summary

- Developed a flight software V&V strategy in compliance with NASA requirements:
  - NPR7150.2 Software Engineering Requirements
    - CMMI Level 2 or Equivalent
  - NASA-STD-8739.8 NASA Software Assurance Standard

- Methods are documented in required plans:
  - LADEE Software Management Plan
  - LADEE Software Test Plan
  - LADEE Software Assurance Plan
  - .
  - .
  - .

- Successful SCAMPI B audit for CMMI level 2.
  - Proceeding to SCAMPI A

- Refining testing and processes
  - Current in Build 2 cycle

Next Steps

Investigate Advanced Testing Tools for Simulink/Stateflow