

CODE / CONFIGURATION COVERAGE

In all affairs it's a healthy thing now and then to hang a question mark on the things you have long taken for granted. - Bertrand Russell, 1872-1970

Michael Aguilar – NASA Technical Fellow in Software – NASA NESC

michael.l.Aguilar@nasa.gov

	<h1>Verification Coverage</h1>	Presenter Michael Aguilar
		Date 11/12/2018

In software engineering, **test coverage** is a measure used to describe the degree to which the source code of a program is executed when a particular test suite runs.

Many different metrics can be used to calculate test coverage; some of the most basic are the percentage of program subroutines and the percentage of program statements called during execution of the test suite.

A program with high test coverage, measured as a percentage, has had more of its source code executed during testing, which suggests it has a lower chance of containing undetected software bugs compared to a program with low test coverage.

	<h1>DO178B Coverage Criteria</h1>	Presenter Michael Aguilar
		Date 11/12/2018

The following concepts illustrate the FAA and NASA definition of DO-178B/ED-12B for verifying critical software.

- a) Every statement in the program has been invoked at least once
- b) Every point of entry and exit in the program has been invoked at least once
- c) Every control statement (i.e., branch-point) in the program has taken all possible outcomes (i.e., branches) at least once
- d) Every non-constant Boolean expression in the program has evaluated to both a True and a False result
- e) Every non-constant condition in a Boolean expression in the program has evaluated to both a True and a False result
- f) Modified Condition/Decision Coverage (MC/DC) Every non-constant condition in a Boolean expression in the program has been shown to independently affect that expression's outcome.

Basic Range Rule Processing and Syntax

Presenter Michael Aguilar

Date
11/12/2018

These rules are not “expert system rules”, but could easily be written as C/C++ source code. The interpreter implements no forward chaining, nor backward chaining; all rules are evaluated in order every update cycle. An impact point is computed for evaluation against these rules.

The design implements static range rules (in boundary, outside boundary, gate passage) within the source code, along with an interpreter to process the loaded range rules that change per launch vehicle, launch, sensors, and range.

Note: The C/C++ source code verification of the interpreter approaches 100% coverage, as long as one of each conditional type and Lat/Lon type is defined in the rule upload.

Conditional Syntax

- IgnitionLogic describes stage ignition rules
- BurnoutLogic describes stage burnout rules
- QualifyLogic describes precondition to be met
- ApplyWhen describes trigger conditional gate
- FireWhen describes action conditional gate

Lat/Lon Boundaries, corridors, and gates

- Tables of segment endpoints in Lat/Lon

Core Autonomous Safety Software (CASS) Range Rules

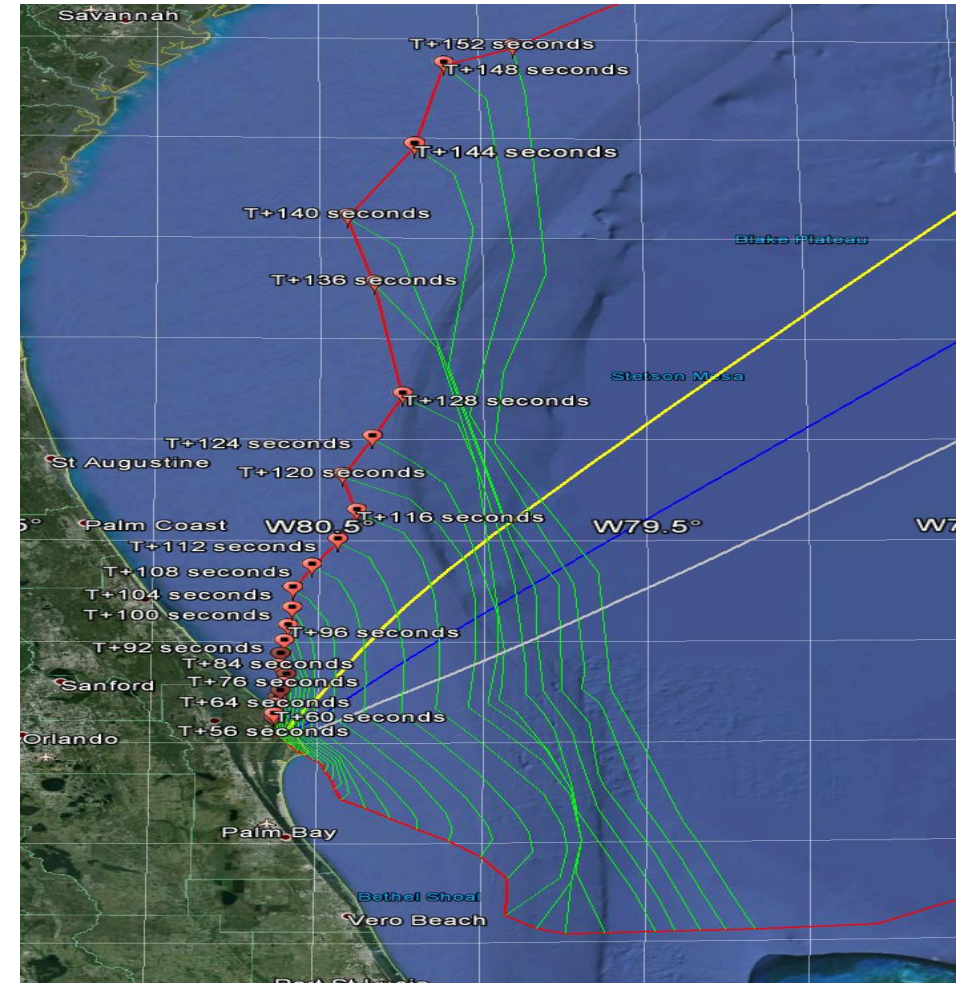
Presenter Michael Aguilar

Date
11/12/2018

If the range rules and configuration was implemented as C/C++ condition statements with the source code,

- a) IgnitionLogic, BurnoutLogic, QualifyLogic, ApplyWhen, and FireWhen conditions could then be measured for test coverage using tools
- b) If each boundary, corridor, or gate segment were implemented as “if statements” in the source code, the segments could be verified by trajectories passing outside, inside, and crossing every defined end-point using coverage tools.

The rules are implemented as a unique data file of non-executable Range Rules that are interpreted by the executable code. No coverage tool exists.




Conditional Range Rule Syntax

Presenter Michael Aguilar

Date
11/12/2018

IsValidNav Data	IsValidGPS Data	svCount > 4	PDOP < 10.0	Result
true	true	true	true	true
true	true	true	false	false
true	true	false	true	false
true	true	false	false	false
true	false	true	true	false
true	false	true	false	false
true	false	false	true	false
true	false	false	false	false
false	true	true	true	false
false	true	true	false	false
false	true	false	true	false
false	true	false	false	false
false	false	true	true	false
false	false	true	false	false
false	false	false	true	false
false	false	false	false	false

CASS Rule Coverage Syntax Example



```
</Settings>
- <NavSensors>
  - <Sensor id="GpsA">
    <velDotParam>0.93</velDotParam>
    <!--unitless-->
    - <QualifyLogic>
      <cond> GpsA.IsValidNavData is true </cond>
      <and/>
      <cond> GpsA.IsValidGPSData is true </cond>
      <and/>
      <cond> GpsA.svCount > 4</cond>
      <and/>
      <cond> GpsA.PDOP < 10.0</cond>
    </QualifyLogic>
  </Sensor>
  <PriorityList reselect="OnFail"> GpsA </PriorityList>
</NavSensors>
```


Conditional Rule Coverage

Presenter Michael Aguilar

Date
11/12/2018

IsValidNav Data	isValidGPS Data	svCount > 4	PDOP < 10.0	Result
true	true	true	true	TRUE
true	true	true	false	FALSE
true	true	false	true	FALSE
true	true	false	false	False
true	false	true	true	FALSE
true	false	true	false	False
true	false	false	true	False
true	false	false	false	False
false	true	true	true	FALSE
false	true	true	false	False
false	true	false	true	False
false	true	false	false	False
false	false	true	true	False
false	false	true	false	False
false	false	false	true	False
false	false	false	false	False

CASS Rule Coverage Example (Sufficient Testing)



```
</Settings>
- <NavSensors>
  - <Sensor id="GpsA">
    <velDotParam>0.93</velDotParam>
    <!--unitless-->
    - <QualifyLogic>
      <cond> GpsA.isValidNavData is true </cond>
      <and/>
      <cond> GpsA.isValidGPSData is true </cond>
      <and/>
      <cond>GpsA.svCount > 4</cond>
      <and/>
      <cond>GpsA.PDOP < 10.0</cond>
    </QualifyLogic>
  </Sensor>
  <PriorityList reselect="OnFail"> GpsA </PriorityList>
</NavSensors>
```


MC/DC Conditional Rule Coverage

Presenter Michael Aguilar

Date
11/12/2018

The verification would require test that exercise the AFTS rule with these conditions in order to quality for 100% DO178C MC/DC required coverage for critical code.

IsValidNav Data	isValidGPS Data	svCount > 4	PDOP < 10.0	Result
true	true	true	true	True
true	true	true	false	False
true	true	false	true	False
true	false	true	true	False
false	true	true	true	False

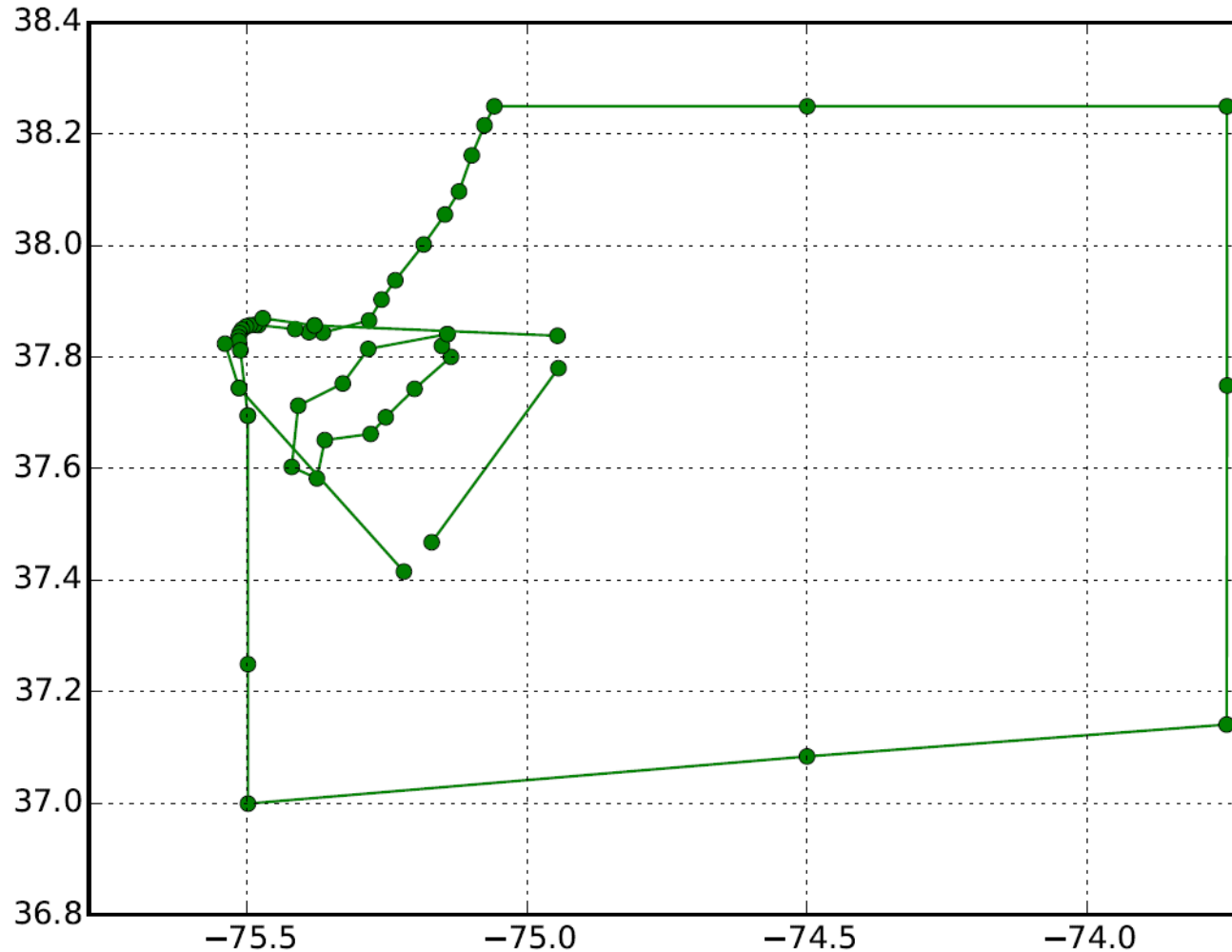


```
</Settings>
- <NavSensors>
  - <Sensor id="GpsA">
    <velDotParam>0.93</velDotParam>
    <!--unitless-->
    - <QualifyLogic>
      <cond> GpsA.isValidNavData is true </cond>
      <and/>
      <cond> GpsA.isValidGPSData is true </cond>
      <and/>
      <cond>GpsA.svCount > 4</cond>
      <and/>
      <cond>GpsA.PDOP < 10.0</cond>
    </QualifyLogic>
  </Sensor>
  <PriorityList reselect="OnFail"> GpsA </PriorityList>
</NavSensors>
```


Range Boundary and Gate Syntax

Presenter Michael Aguilar

Date
11/12/2018



- <Coordinates>
- <coord>
 <Lon> **-75.499371** </Lon>
 <Lat> **37.695069** </Lat>
 </coord>
- <coord>
 <Lon> **-75.499144** </Lon>
 <Lat> **37.249358** </Lat>
 </coord>
- <coord>
 <Lon> **-75.499144** </Lon>
 <Lat> **36.999259** </Lat>
 </coord>
- <coord>
 <Lon> **-74.500229** </Lon>
 <Lat> **37.083887** </Lat>
 </coord>
- <coord>
 <Lon> **-73.750507** </Lon>
 <Lat> **37.141092** </Lat>
 </coord>

First CASS Version Verification Tool Suite

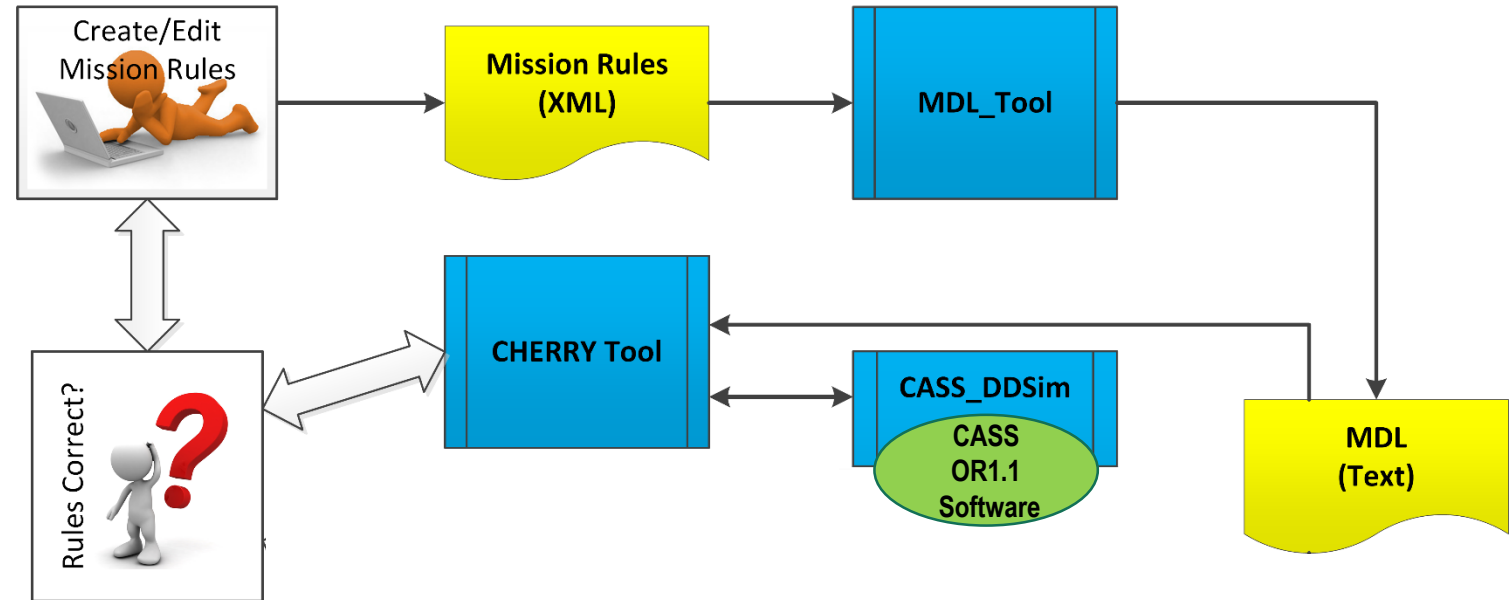
Presenter **Michael Aguilar**

Date
11/12/2018

CASS OR1.1 Process

1. Mission Rules are written, then checked for syntax errors (MDL_Tool)
2. Nominal and off-nominal trajectories are input to the simulator (CASS_DDSim), along with the Mission Rules.

In practice, this manual process produces 100 simulation runs used to validate and verify the CASS interpreter /rule behavior.



Enhanced CASS OR2 Government Tool Suite

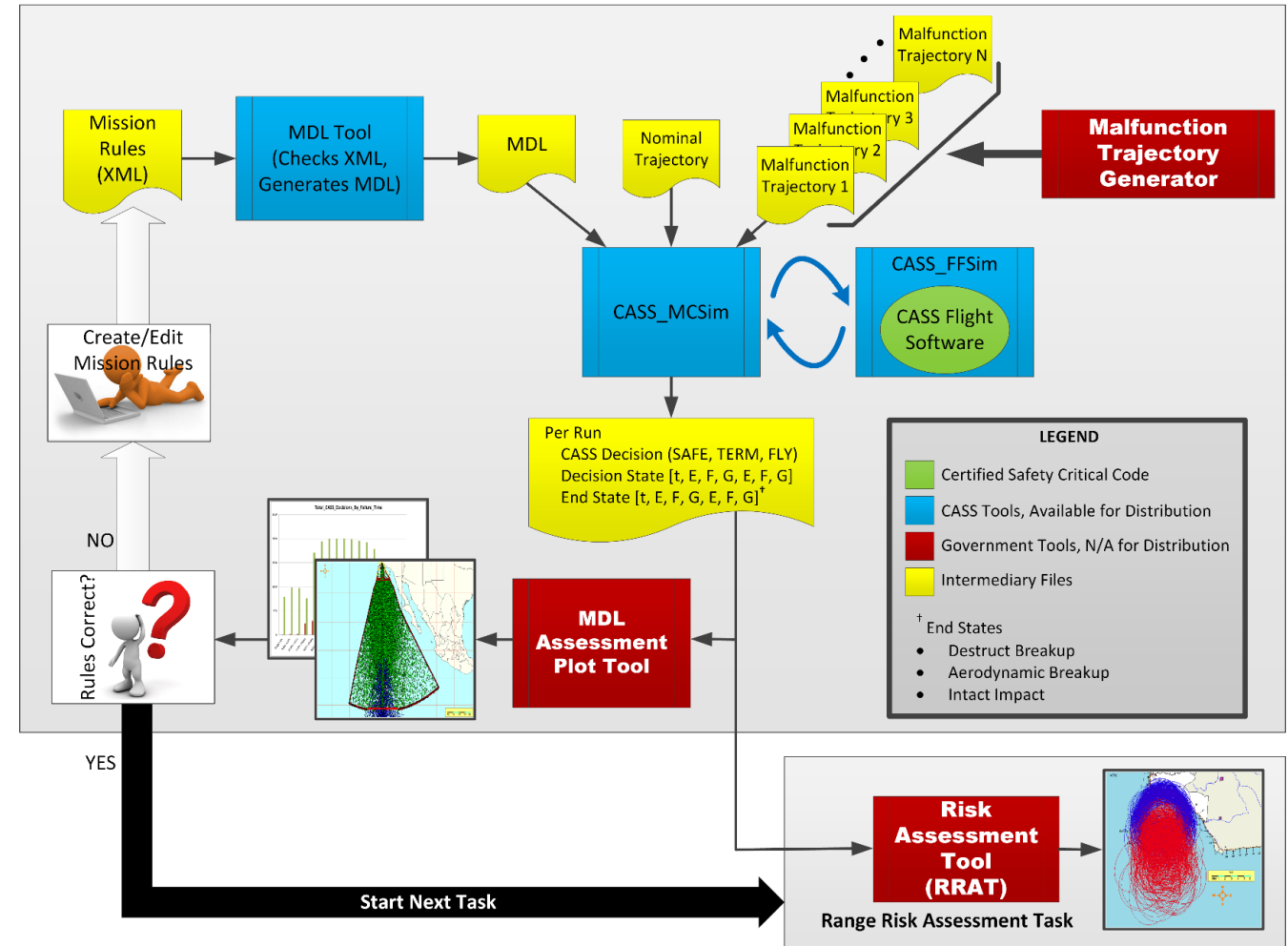
Presenter Michael Aguilar

Date
11/12/2018

Additional analysis and assessment tools have been developed for Government use

CASS OR2 Process

1. Mission Rules are written, then checked
2. Nominal and off-nominal trajectories are input simulated
3. Monte Carlo analysis is supported creating numerous nominal and off-nominal trajectories are created for simulation
4. The CASS_FFSim automates and executes rapid simulations: 500K runs on 300 Node Cluster → 8.8 Hours



So, what is *Coverage* of the Range Rules?

Presenter Michael Aguilar

Date
11/12/2018

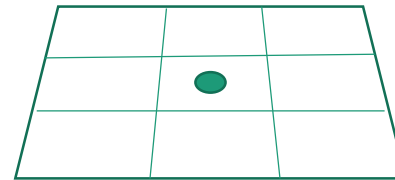
- **Running code coverage upon the C++ coded interpreter of range rules reports 100% coverage.**
- No definition exists that defines envelope or boundary coverage
- No tool exists that can instrument these range rules and capture conditional coverage
- No metrics are reported about boundary segments exercised or conditionals covered during Monte Carlo runs (500K – 1000K runs).
- The Monte Carlo tool design does not exercise all sensor input conditionals within range rules.

Sufficient for Range Rule file Coverage?

Presenter Michael Aguilar

Date
11/12/2018

- **Coverage Definition:** Utility to parse range rules into test scenarios requirement document.
- **MC/DC Conditional Coverage:** Generate specific input trajectories and conditions to “cover” conditionals in file, and modify the Simulation tools to execute and record these condition runs.
- **Boundary, Corridor, and Gate Coverage:** Develop CASS Unit Tests for boundary plane coverage for every range rule point 3x3 grid around each end point within file. Source code units are coded as Lat/Lon degrees, while input is GPS ECEF meters. Grid needs to be defined to cover the implementation of “equivalency”.



	<h1>Flight Configuration Load Failures in the News</h1> <h2>April 2018</h2>	Presenter Michael Aguilar
		Date 11/12/2018

The TESS mission Falcon 9 was scrubbed. It had an autocode script error in the autopilot propellant slosh model, affecting modal frequencies. This was first use, first flight, of this new script; it had not been validated. The slosh model itself had been previously V&V'd but now had the wrong inputs.

“Standing down today to conduct additional GNC analysis, and teams are now working towards a targeted launch of NASA TESS on Wednesday, April 18,” SpaceX Tweeted after the countdown was stopped.

	<h1>Flight Configuration Load Failures in the News</h1> <h2>Feb 2017</h2>	Presenter Michael Aguilar
		Date 11/12/2018

On CRS-10, an incorrect value was loaded in the Dragon spacecraft Global Positioning System affecting knowledge of its position relative to ISS and resulting in a 24 hour hold on its approach to ISS.

The problem was traced to an incorrect data value in the spacecraft's Global Positioning System, critical to operations as this data informs the vehicle of its relative position to the space station. The abort resulted in a 24-hour hold on its approach. The error was corrected in this time, during which the spacecraft entered a "racetrack" trajectory around the station to reset its approach.

Flight Configuration Load Failures in the News Dec 2017

Presenter Michael Aguilar

Date
11/12/2018

Russian space agency Roscosmos said it had lost contact with the newly-launched weather satellite - the Meteor-M - after it blasted off from Russia's new Vostochny cosmodrome in the Far East. "The rocket was really programmed as if it was taking off from Baikonur," said Rogozin. "They didn't get the coordinates right."



"Let's put it in the data load to simplify software system testing"

Presenter Michael Aguilar

Date
11/12/2018

- Data driven systems expect the loads to differ from the load used during software system verification.
- Data loads can dramatically alter the software system behavior and performance
- Data loads require some matching software analysis, design, or verification processes to allow for system verification for these late changes to system behavior and performance

Forward Work – Discussion

- What designs can support analysis and verification of these data loads?
- What analysis of these data loads formally identify the regression tests to be executed to verify software system behavior and performance?
- Are we doing enough?

	<h1>What is <i>Coverage</i> of these Data Driven Systems?</h1>	Presenter
		Michael Aguilar
		Date
		11/12/2018

Current trends in software design create systems where the software behavior is increasingly dependent on external data loads.

What is the verification and coverage metrics for the following?

- Closed-loop systems loading tuning and optimization values.
- Launch data loads to match the day of launch environment.
- Load of thousands of coefficients for day-of-launch differing from software system verification load.
- Control file downloads into complex electronics or control hardware to sequence ground support equipment.
- Autonomous systems loading fault management rules, expert system database, neural network coefficients, ...
- Necessary Tools that produce and process these loads.
- Reused code and Generated code

Coverage of these Data Driven Systems Suggestions

Presenter Michael Aguilar

Date
11/12/2018

- Define your projects definition of “coverage” requirements to include code and configuration.
- Partition uploads into files, tables, objects,... that encapsulate load affects, and link these load partitions to the necessary verification process. Document this upload to necessary verification pairing as risk and mitigation. Prepare for schedule/effort pressures.
- Include configuration upload defects as part of stress testing.
- Constrain unit input checks to flight-specific physical or engineering limits. Accept false positives to mitigate critical false negatives. Example: constrain GN&C pre-condition checks to accept ranges specific to the vehicle launch in the hopes of catching configuration upload direct or indirect effects that break GN&C.
- Develop FSW utilities and tools to support verification of uploads, perhaps using flight code subsystems. Include these tools with the FSW deployment and document the risk and mitigation in the use of these tools during operations.