

# Turning the Digital Thread into Reality

---

STEVEN H. DAM, PH.D., ESEP

JERRY J. SELLERS, PH.D.

OCTOBER 25, 2018



# Agenda

---

- What are some of the major hurdles in making the digital thread a reality?
- Don't we something like this already?
- Why do we want to create this “thread?”
- What's really missing?
- What are we doing to overcome these problems?

# What are some of the major hurdles in making the digital thread a reality?

---

- Tool interoperability
- Security
- Scalability



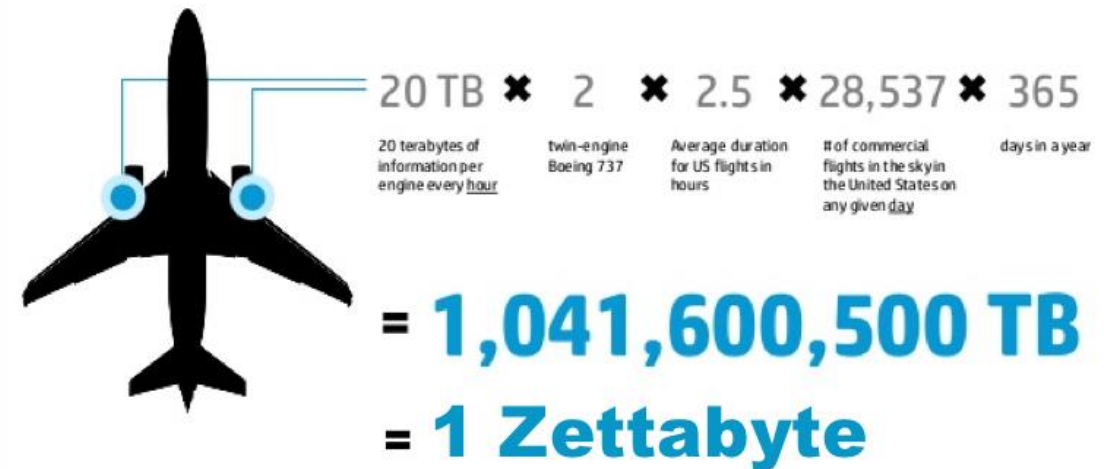
# Scalability Problem

- Today, complexity is going out of sight
- We no longer talk about Gigabytes of information, its now Zettabytes ( $1 \times 10^{21}$  bytes)
- How can we deal with this much data?
- Do we really need all this data?

## One example of one type of data in the world:

### A Real World Example:- Big Data- Micro-transactions

Sensor data collected from US commercial jet engines during 1 year



<https://www.slideshare.net/penumuru/harness-the-power-of-big-data-with-oracle-63438438>

# Other Concerns

---

- Tool interoperability
- Security
- Scalability
- Intellectual Property
- Cost
- “Rice Bowls”

# Don't We Do Something Like this Already?

---

- We have been creating physics-based models of systems for decades
- Many of those models have been coupled to CAD/CAM systems
- The gaps between tools are used as “inspection points” for analysis
  - Is that bad or good?

# Why Do We Want to Create this “Thread?”

---

- We think there will be significant saving accrued by having a seamless abstraction of an entire system
  - We have seen how the automobile manufacturers have gone down this path
  - The question is, “Are we making the same kind of product and incrementally improving it, as they do in the automotive world?”
- We will clearly save time and money if we can more easily move information between tools

# What's Really Missing?

---

- Need methods to capture and visualize tremendous amounts of information
- Massive storage and retrieval of information
- Need not only all the technical readouts, but also the programmatic information
- Capability to move data around easily, between applications
- A language that enables decomposition and abstraction
  - A systems engineering language, not a software engineering language



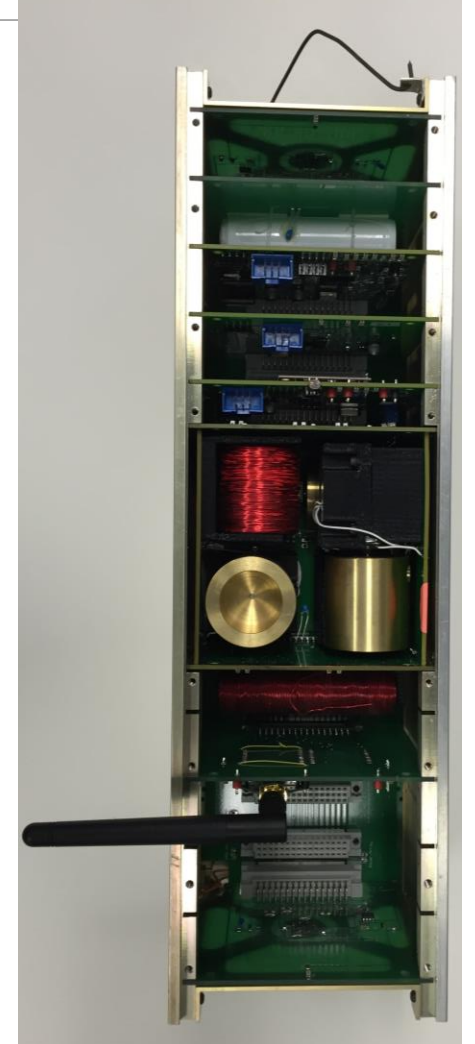
# What Are We Doing to Overcome these Problems?

---

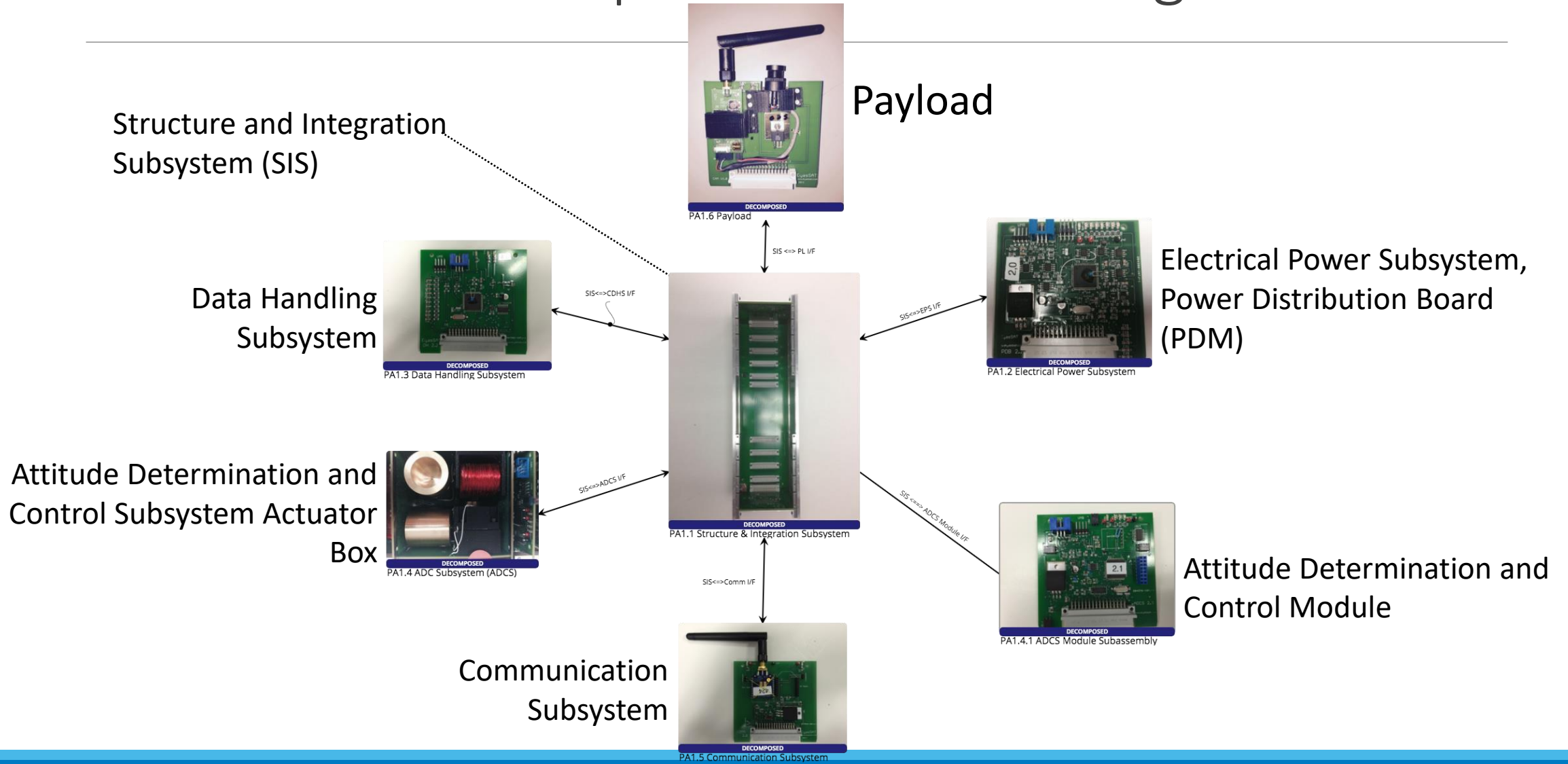
- Tool interoperability problems can be reduced by merging functionality into a common tool or tool set
  - SPEC has done this on the SE level with Innoslate®
  - Use of APIs can reduce the interoperability problem, if we have a common, generalized ontology to map tool data together (LML can provide this)
- Scalability, Security and IP problems can be resolved by continuing to partition the problem through decomposition
  - But this means that the databases at each level of decomposition must be able to interoperate (see bullet 1 above)
- Explore hardware-in-the-loop simulations, not just software

# Example: NanoMET

- NanoMet is hypothetical end-to-end systems engineering and project management case study designed for the education and training of space professionals
- NanoMet "spacecraft" are desktop training tools based on the EyasSAT3 (ES3) educational satellite bus
- All ES3 is "ITAR-free" and is not space qualified or qualifiable
- For the purpose of education and training, NanoMet is treated as a "real" space mission with representative systems engineering and project management artifacts and associated rigor



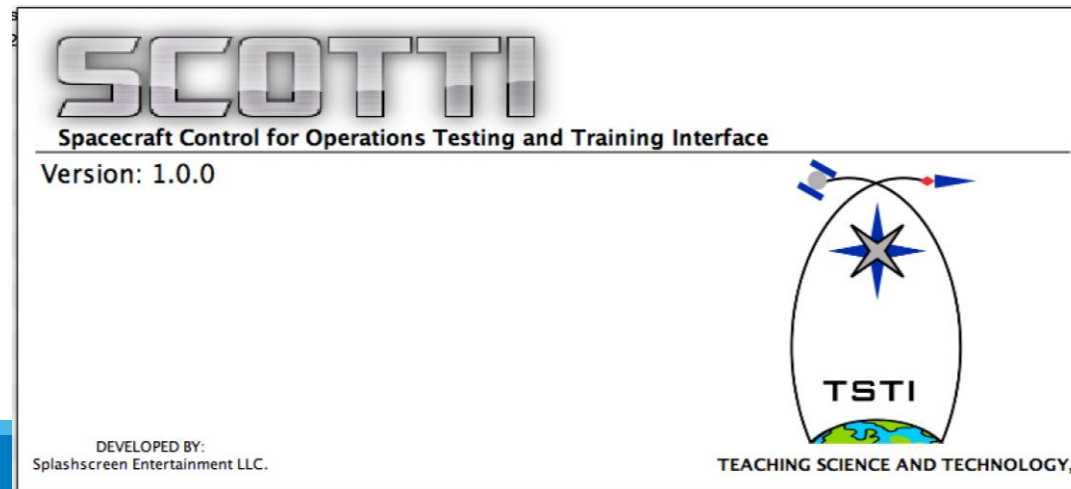
# NanoMET Spacecraft Asset Diagram



# Spacecraft Control and Operations Testing and Training Interface (SCOTTI)

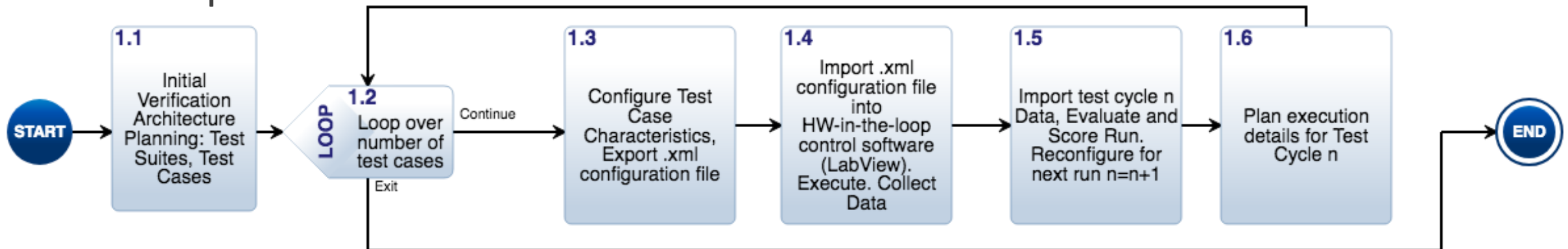
---

- SCOTTI is a versatile, LabView-based interface that provides a point and click graphics user interface (GUI) for all ES3/NanoMet lab and “operational” activities
- Provides insight into packet communication protocols (future A331 or SP200 lesson)
- Provides for real-time or Pass Planning execute-at-Time-X commands



# A Verification Round-trip Example

- Use Innoslate<sup>®</sup> to setup tests and record results
- Export Innoslate<sup>®</sup> XML to LabView “SCOTTI”
  - adding import capability to LabView took only a few hours
- Execute test cycle using LabView
- Import results to Innoslate
- Repeat



# 1.1 Initial Verification Planning

- Test Center provides means to create test cases and test suites
- The parameters we want to vary are also captured as Characteristics

TS.1000 CubeSAT Full System Testing	Expected Result
<p><b>TC.1001 Automated Bus Checkout</b> The purpose of this test case is to exercise the cubesat system using the auto-bus check out capability in SCOTTI. SCOTTI will send all commands, then receive and check all telemetry points. Regression tests will evaluate performance with differing (1) Number of Runs (burn in), (2) Run Delays, (2) Wheel Run Times. On Error = Continue will be the same for all runs.</p>	
<p><b>TC.1001.1 Test Run Procedures</b></p> <p>AUTOMATED BUS CHECKOUT</p> <ul style="list-style-type: none"><li><input type="checkbox"/> Power up the vehicle by CAREFULLY inserting Arm Plug into the pins NEXT TO the blue pins on the PDM, NOT INTO PINS OF THE BLUE CONNECTOR</li><li><input type="checkbox"/> Select Comm &gt;&gt; Connect.</li><li><input type="checkbox"/> Observe data streaming in STREAM window.</li><li><input type="checkbox"/> Select Role &gt;&gt; Bus Tester.</li><li><input type="checkbox"/> Click on Baseline Tests to populate the test script.</li><li><input type="checkbox"/> Check the Burn in Test Box, use default Number of Runs = 1, Run Delay = 1 min, Wheel Run Time = 10 sec, On Error = Continue.</li><li><input type="checkbox"/> Click Run.</li><li><input type="checkbox"/> Monitor results as it executes, scrolling down as needed.</li><li><input type="checkbox"/> When complete, verify all commands pass.</li><li><input type="checkbox"/> Verify all telemetry within parameters.</li></ul>	<p>(3.2.4.1.1) The test shall be considered successful if measured resistance from all temperature transducers (thermistors) varies as expected with temperature AND all telemetry values assigned to DHS are supplied and their calibrated values are within specified ranges as defined by the MASTER TELEMETRY DATABASE. (3.2.5.2) The inspection shall be considered successful if requirement 3.3.3 and all subordinate requirements have been successfully verified. (3.2.3.2.5) The test shall be considered successful if the ADCS responds to all command values allocated to it as defined by the MASTER COMMAND DATABASE. (3.2.3.1.6) The test shall be considered successful if all telemetry values assigned to ADCS respond as supplied</p>

# 1.3 Configure Test Case Characteristics

- Test Center provides means to create test cases and test suites
- The parameters we want to vary are also captured as Characteristics
- We change those Characteristics to reflect the test case we want to execute in Database View
- Export XML

TS.1000 CubeSAT Full System Testing

Expected Result

TC.1001 Automated Bus Checkout

The purpose of this test case is to exercise the cubesat system using the auto-bus check out capability in SCOTTI. SCOTTI will send all commands, then receive and check all telemetry points. Regression tests will evaluate performance with differing (1) Number of Runs (burn in), (2) Run Delays, (2) Wheel Run Times. On Error = Continue will be the same for all runs.

TC.1001.1 Test Run Procedures

(3.2.4.1.1) The test shall be considered successful if measured resistance from all temperature transducers (thermistors) varies as

AUTOMATED BUS CHECKOUT

MENU Dashboard Database Diagrams Documents Test Center Import Analyzer Schema Editor

Filter Saved Queries New Entity order:modified- class:"Characteristic" Reports Matrix

Entity	Value	Units
Wheel Run Time	20	seconds
Run delay	2	min
Number of Runs	1	

Save Current Query

My Queries

Characteristics Matrix

Click on Database tests to populate the test script.

- Check the Burn in Test Box, use default Number of Runs = 1, Run Delay = 1 min, Wheel Run Time = 10 sec, On Error = Continue.
- Click Run.
- Monitor results as it executes, scrolling down as needed.
- When complete, verify all commands pass.
- Verify all telemetry within parameters.

subordinate requirements have been successfully verified. (3.2.3.2.5) The test shall be considered successful if the ADCS responds to all command values allocated to it as defined by the MASTER COMMAND DATABASE. (3.2.3.1.6) The test shall be considered successful if all telemetry values assigned to ADCS respond as specified

# 1.4 Import XML into LabView

- Import XML into SCOTTI
- Execute Test via SCOTTI
- Collect Data

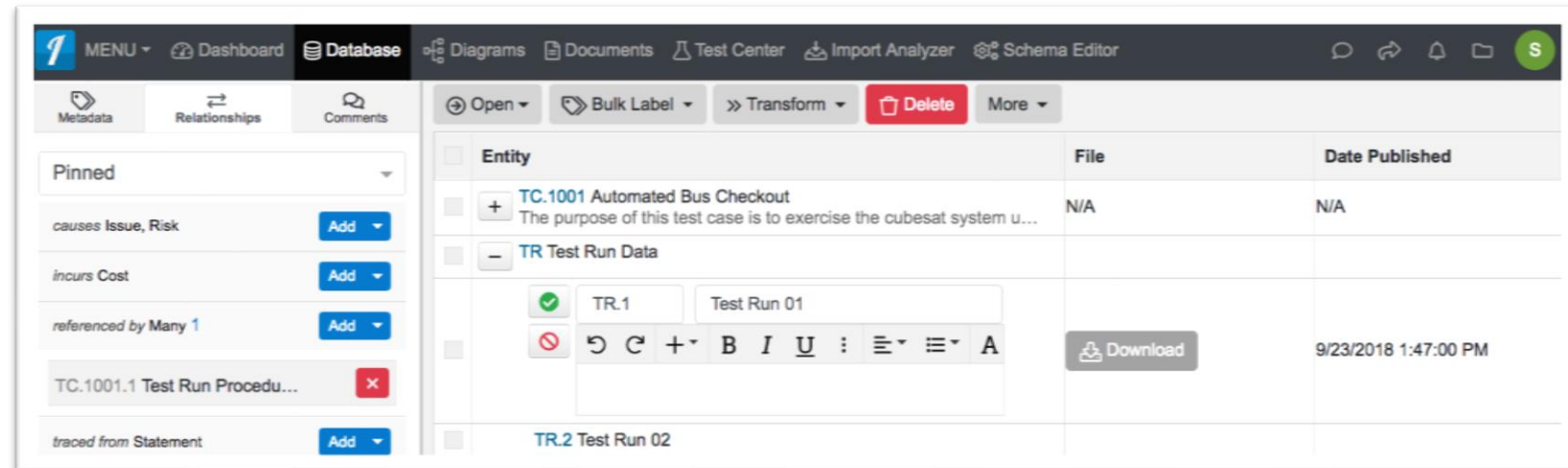
The screenshot displays the SCOTTI (Bus Tester) software interface. The main window is titled "SCOTTI (Bus Tester)" and contains several sections:

- Baseline Tests:** Includes buttons for "Load" and "Remove Test".
- Burn In Test:** Includes a checkbox, "Number of Runs" (set to 1), "Run Delay" (set to 1 Min.), "Wheel Run Time" (set to 10 Sec.), and "On Error" options (Continue, Stop, Retry).
- Test Results Table:** A table with columns: Status, Type, Action, and Timeout. It lists various test steps such as "Send Command", "Verify Telemetry", and "Wait" with their respective actions and completion times.
- TELEMETRY Table:** A table with columns: PID, Mnemonic, Raw, and Converted. It lists various telemetry points like "torque\_z", "ADCS\_MODE", "ADCS\_A", etc., with their raw and converted values.
- Status Bar:** At the bottom, it shows "Packet Count 2237", "EPS", "DHS", "ADCS Sensor", "ADCS Actuator", "Mode RF", and "Connected?".



# 1.5 Import Test Results and Score

- Capture data files as Artifacts in Innoslate®
- Add any notes or date/times for the execution
- Relate to specific tests or identify any Issues or Risks associated with the test



The screenshot displays the Innoslate Database interface. The top navigation bar includes 'MENU', 'Dashboard', 'Database', 'Diagrams', 'Documents', 'Test Center', 'Import Analyzer', and 'Schema Editor'. The left sidebar shows 'Pinned' items: 'causes Issue, Risk', 'incurs Cost', 'referenced by Many 1', 'TC.1001.1 Test Run Procedu...', and 'traced from Statement'. The main content area features a table with columns 'Entity', 'File', and 'Date Published'. The table contains the following data:

Entity	File	Date Published
TC.1001 Automated Bus Checkout The purpose of this test case is to exercise the cubesat system u...	N/A	N/A
TR Test Run Data		
TR.1 Test Run 01		
TR.2 Test Run 02		

The interface also includes a 'Download' button and a 'Date Published' of 9/23/2018 1:47:00 PM for the TR.2 Test Run 02 entry.

# 1.6 Configure for Next Test Cycle

- New Test Cycle builds on previous work creating a new baseline
- Repeat process until all configurations have been tested

The screenshot shows a software interface for test case management. At the top, there are navigation buttons: '+ New Test Case', 'Auto Number', 'New Test Cycle', 'Open', and 'Report'. Below this is a table with columns: 'Expected Result', 'Actual Result', 'Status', and 'Status Roll-Up'. The table contains two rows of test case data.

Expected Result	Actual Result	Status	Status Roll-Up
		Not Run	2
		Not Run	Not Run

The first row corresponds to the test case 'TC.1001 Automated Bus Checkout'. The description for this test case is: 'The purpose of this test case is to exercise the cubesat system using the auto-bus check out capability in SCOTTI. SCOTTI will send all commands, then receive and check all telemetry points. Regression tests will evaluate performance with differing (1) Number of Runs (burn in), (2) Run Delays, (2) Wheel Run Times. On Error = Continue will be the same for all runs.'

The second row corresponds to the sub-test case 'TC.1001.1 Test Run Procedures'. The description for this sub-test case is: 'AUTOMATED BUS CHECKOUT' followed by a list of steps: 'Power up the vehicle by CAREFULLY inserting Arm Plug into the pins NEXT TO the blue pins on the PDM, NOT INTO PINS OF THE BLUE CONNECTOR', 'Select Comm >> Connect.', 'Observe data streaming in STREAM window.', 'Select Role >> Bus Tester.', 'Click on Baseline Tests to populate the test script.', 'Check the Burn in Test Box, use default Number of Runs = 1, Run Delay = 2 min, Wheel Run Time = 20 sec, On Error = Continue.', 'Click Run.', 'Monitor results as it executes, scrolling down as needed.', 'When complete, verify all commands pass.', and 'Verify all telemetry within parameters.'

# Next Steps

---

- This process can be fully automated by using Java or REST APIs, but that would be up to the organization
- Many engineers would prefer to run their own tests and vary the parameters as needed when they don't see the need to continue a particular path
- This approach can be generalized to apply to many other typical tasks in the Digital Engineering process
- We plan to continue this development for DoD and other customers