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, it's 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

\[
20 \text{ TB} \times 2 \times 2.5 \times 28,537 \times 365
\]

\[
= 1,041,600,500 \text{ TB}
\]

\[
= 1 \text{ Zettabyte}
\]

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”

Is it really even desirable?
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
  o 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
  o We have seen how the automobile manufacturers have gone down this path
  o 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
  o 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
  o SPEC has done this on the SE level with Innoslate®
  o 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
  o 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

Structure and Integration Subsystem (SIS)

Data Handling Subsystem

Attitude Determination and Control Subsystem Actuator Box

Communication Subsystem

Payload

Electrical Power Subsystem, Power Distribution Board (PDM)

Attitude Determination and Control Module
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® to setup tests and record results
- Export Innoslate® 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
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
1.4 Import XML into LabView

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