Evaluation of a Candidate Flight Dynamics Model Simulation Standard Exchange Format

Bruce Jackson
NASA Langley Research Center
Hampton, Virginia

Brent W. York
Naval Air Systems Command
Patuxent River, Maryland

Bruce Hildreth
Science Applications International Corporation
Lexington Park, Maryland

William B. Cleveland
Northrop Grumman Info. Technology
Mountain View, California

2004 AIAA Modeling & Simulation Technology Conference
Providence, Rhode Island

AIAA 2004-5038
Problem: Simulation Rehosting

• Required whenever a model is shared
• Increasingly common, thanks to…
  – Increased reliance on numerical analysis
  – Contractor/government teaming
  – Moore's Law
  – Multiplicity of training devices
• Currently very labor intensive
  – Different languages / conventions / traditions

Typical: four to eight months to rehost & validate new sim
The Need for a Standard

• Standards promote productivity
  – Improved information exchange
    • More accurate simulations
    • More consistent simulations
    • Lower cost
  – Improved interoperability
  – Increased software reuse
• Rapid sim rehosting - minutes instead of months
• Potential for industry significant cost saving

2002 paper: $ 6+ M per year per aircraft model
Concept

The Standard (XML-based)

Site A (exporter)

Std. Model Editor

Export API

Report Editor

Import API

Site B (importer)

Other Commercial Tools

Import API

Site C (importer)
Concept

- Need for standard representation of vehicle dynamics/aerodynamics
- Get away from ad-hoc, site-specific “standards”
- Many are possible; we’re proposing one
- Standard is superset of typical site-specific standards

An exchange standard: no requirement for end use
Previous efforts

- MODCOMP - 1980's - attempt to standardize on software & hardware for all training simulations
- Similar attempts to standardize software modules
- DIS/HLA/SEDRIS – sim environment & network
- M&S T.C. started data standards effort in early 90’s
- Efforts focused on vehicle dynamics model
- Objective: to easily exchange a model from one site to another
Initial requirements for a standard

1. Function table data – required in most non-linear models. Standard will add:
   - Provenance (history / data source / modifications)
   - Statistics (uncertainty / Monte Carlo data)
   - Mathematics required to combine functions and inputs into force & moments acting at a specified location

2. Check case data – required to verify proper model transfer

3. Signal definitions (variable names) – required to clearly state what the transferred information is (units, axis system, sign convention, etc)
   - Includes methodology for naming new variables
   - Includes axis system definitions
The proposed AIAA Sim Data Standard includes a Standard Function Table with statistics & provenance. Each data point contains four elements:

- Independent variables: \( \delta_s, \text{Mach}, \alpha \)
- Dependent variable: \( C_L \)
- Provenance ID
- Statistics: \( -\sigma, \sigma \)

The statistics data (the confidence intervals) are optional. Here is a sample table:

<table>
<thead>
<tr>
<th>( \delta_s )</th>
<th>Mach</th>
<th>( \alpha )</th>
<th>( C_L )</th>
<th>Ref</th>
<th>( -\sigma )</th>
<th>( \sigma )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.8</td>
<td>60</td>
<td>0.60</td>
<td>C</td>
<td>-0.0032</td>
<td>0.0068</td>
</tr>
</tbody>
</table>

The statistics data are: \( \text{Ref} \), \( -\sigma \), \( \sigma \).
Proposed AIAA Sim Data Standard – axis systems

- Use the overlap of existing AIAA/ANSI Recommended Practice R-004-1992 and DIS Axis Systems
  - Body axis system
  - Earth fixed axis system
- Addition of a Flat Earth (local) axis system for convenience

Clearly defined axes are critical to successful exchange
Proposed AIAA Sim Data Standard – variable name definitions

• Standard dictionary of variable names

• Objective:
  – Clear definition of the significant components and parameters of a model and its validation data.
  – For example: "Angle of attack" means:
    • wing angle of attack?
    • fuselage angle of attack?
    • inertial angle of attack?
    • includes turbulence effects?
    • in degrees or radians?
    • ranging from ± 90 or ± 180 degrees?

• Extremely important in validation.

Clearly defined variable names are critical to successful exchange
An XML approach

- eXtensible Markup Language (XML) becoming popular way to encode data for on-line exchange
- Text-based human/machine readable files
- Lots of XML utility programs available
- Specialized set of markup tags developed: Dynamic Aerospace Vehicle Exchange Markup Language (DAVE-ML)
- First proposed in 2002 (AIAA M&ST Monterey)
DAVE-ML features

- Language- and facility-independent
- Encodes non-linear function tables
- Encodes build-up equations (via MathML)
- Encodes history & provenance of model
- Encodes statistical uncertainty of data
- Self-documenting (via XSLT)
- Can include validation data (checkcases)
Example: DAVE-ML transformed into XHTML

F-16 Subsonic Aerodynamics Model (a la Garza)

F-16 Aero Data File. Based on Morelli's adaptation of Stevens and Lewis' F-16 example [1] described in Garza work. Morelli's TM [2], obtaining from E. A. Morelli in the form of Matlab scripts [3] & [4]. This version has quotient's replaced with divide's.

Author: Bruce Jackson
Created: 10-JUNE-2003

Signal Definitions

<table>
<thead>
<tr>
<th>Name</th>
<th>Symbol</th>
<th>Units</th>
<th>Sign</th>
<th>Initial Value</th>
<th>Depends on</th>
</tr>
</thead>
<tbody>
<tr>
<td>absbeta</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>absCl0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>absCn0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Angle_of_Attack_deg</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Angle_of_Sideslip_deg</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b2v</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bspan</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example: DAVE-ML converted to Simulink®
Status of simulation standards efforts

• A data standard has been developed by M&S T.C.
  – The standard defines the information that will be exchanged

• A Recommended Practice (RP) for implementation has been developed by an informal DAVE-ML steering subcommittee
  – The RP defines how the information is exchanged

• The RP must be tested
  – To assure no critical components have been left out
  – To assure it is “user friendly”

DAVE-ML is a candidate RP and needs testing

- Two existing aerodynamic models encoded with DAVE-ML as examples
- Two simulation facilities (Ames and Pax) developed import tools
- Ames also developed an export tool
- Successfully demonstrated import and automatic validation of aero models
Example models used for demo I

**Fighter subsonic aero model**
- 51 variables, 18 tables, 744 points
- Switches & absolute value nonlinear elements
- 17 validation checkcases included
- 154 KB file with 2,712 lines

**Concept development lifting body aero model**
- Supersonic and subsonic regimes
- 361 variables, 168 tables, 6,240 points
- 24 validation checkcases included
- 1.2 MB file with 22,299 lines
Demonstration I results

- NASA Ames results
- NAVAIR Patuxent River results
Historically a FORTRAN-based facility

Employ Function Table Processor (FTP) precompiler to create FORTRAN table interpolation subroutines for each table

Wrote Perl scripts to import DAVE-ML into FTP source file, FORTRAN code snippets, checkcase routines

Wrote Perl scripts to export FTP input files into DAVE-ML files

Reduced import time from "several" to single week
NAVAIR Patuxent River

- Formerly FORTRAN, now C++ house
- DAVE-ML support planned for next release of CASTLE (v6.0) simulation executive
- Successfully imported DAVE-ML example model at run time into C++ aero model object
- No intermediate C++ code generated
Additional progress

- Informal DAVE-ML steering committee formed (Bruce Hildreth as chair – bruce.hildreth@saic.com)
- DAVE-ML website created: http://dcb.larc.nasa.gov/utils/fltsim/DAVE
- Discussion list created: simstds@larc.nasa.gov
- On-line DAVE-ML reference manual available
- Java tool to convert DAVE-ML into Simulink®
  – Tested with several internal NASA projects; one was 12.5 MB / 107 KLOC / 97 tables / 717 K pts
Next steps

• Invite additional participation / feedback
• Submit to AIAA; seek ANSI/ISO standard and recommended practice
• Develop model editor and report generator applications
• Distribute existing tools developed to test DAVE-ML for use by the modeling community
Conclusions

• The initial version of the standard is ready
  – Substantial savings of time & effort clearly possible
  – Improve efficiency of the simulation community

• DAVE-ML file can serve as model archive
  – Includes provenance, equations, data, statistics
  – Applicable to automatic Monte Carlo studies
  – Easy to grow and change as technology requires

• Exchange with NAVAIR and NASA Ames has demonstrated DAVE-ML is ready for submittal as the Recommended Practice for simulation data exchange
Questions?
Backup slides
Existing standards

- Simnet/DIS/HLA-networking/architecture
- SEDRIS- environmental data representation
- FAA Advisory Circulars (AC 120-40)
- Standard atmosphere
- Standard world (WGS –95?)
Existing Projects, Standards or Guideline Documents

• General
  – DATA Flight Simulator Design and Performance Data Requirements, 9th Ed. 1993
  – ANSI/AIAA Recommended Practice: Atmospheric and Space Flight Vehicle Coordinate Systems

• Simulation Networking/Architecture Standards
  – HLA / DIS - 4 Standards
    • Message Content
    • Communicative Architecture
    • Environment
    • Fidelity, Exercise Control and Feedback
Existing Projects, Standards or Guidelines (cont’d)

• Others
  – ARINC 610A Guidance for Use of Avionics Equipment and Software in Simulators
  – FAA AC 120-40B(c) Airplane Simulator and Visual System Evaluation
  – Project 2851 Visual Database Specs.
  – POSIX Computer Operating System Standard
  – MIL-STD-1815 Ada language
  – MIL-STD-2167A S/W Development
  – Joint Modeling and Simulation System (JMASS)
  – Software Technology for Adaptable and Reliable Systems (STARS)
  – CALS Standards (apply to computer data formats?)
Variable Names – key points

• Variable Naming convention includes:
  – Identification of Simulation States and Inputs
  – Units- either English or SI

• Linear System Formulation
  – $x = \text{states}$
  – $u = \text{inputs (or controls)}$

$$\frac{dx}{dt} = Ax + Bu$$
$$Y = Cx + Du$$

States and Inputs are key – everything in the dynamic simulation depends upon them

They should be easily identifiable for good software documentation and maintainability

Units for clarity and documentation purposes
Variable Naming Convention

• Each name has up to six components
  – (prefix) (variable source domain) (axis or reference system)
    (specific axis or reference) (core name) (units)
  – Follows "camelCase" naming convention

• Examples
  – s_bodyXVelocity_fps
  – sd_bodyXAcceleration_fps2
  – aeroXBodyForceCoefficient
  – thrustYBodyForce_lbf

Standard variable names clearly define the information being exchanged
Variable Names – Guidelines

• Meaningful name, not mnemonics, standard abbreviations okay
• Distinct parts of the variable names separated by underscores
• No more than 60 characters in length
• First letter of each part of word is capitalized
  – Abbreviations are all CAPS
  – Units are all lower case
Variable Names

- Names database and definition complete
- Naming convention taken from STARS Simulation work (Lead by NAWCTSD)
- Short names taken from NASA Ames

### Example Table of Names

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Short Name</th>
<th>Long Name</th>
<th>Same as</th>
<th>Description</th>
<th>Units Sign</th>
<th>Initial Value</th>
<th>Minimum Value</th>
<th>Maximum Value</th>
<th>Reference</th>
<th>Note</th>
<th>Date Last Changed</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHI</td>
<td>Euler Roll Angle deg y</td>
<td>Roll Euler Angle, L (local) Frame</td>
<td>DEG RWD</td>
<td>-180</td>
<td>180</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>THET</td>
<td>Euler Pitch Angle deg y</td>
<td>Pitch Euler Angle, L (local) Frame</td>
<td>DEG ANU</td>
<td>-90</td>
<td>90</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSI</td>
<td>Euler Yaw Angle deg m</td>
<td>Yaw Euler Angle, L (local) Frame</td>
<td>DEG ANR</td>
<td>-180</td>
<td>180</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Φ</td>
<td>Euler Roll Angle rad y</td>
<td>Roll Angle, L frame</td>
<td>RAD RWD</td>
<td>-π</td>
<td>π</td>
<td>10) 1.3.3.3</td>
<td>1,2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Θ</td>
<td>Euler Pitch Angle rad y</td>
<td>Pitch Angle, L frame</td>
<td>RAD ANU</td>
<td>-π/2</td>
<td>π/2</td>
<td>10) 1.3.3.2</td>
<td>1,2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ψ</td>
<td>Euler Yaw Angle rad m</td>
<td>Yaw Angle, L frame</td>
<td>RAD ANR</td>
<td>-π</td>
<td>π</td>
<td>10) 1.3.3.1</td>
<td>1,2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PHIRD</td>
<td>Euler Roll Angle Rate rad_p_s</td>
<td>Euler roll rate, L frame</td>
<td>RAD/SEC RWD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>THED</td>
<td>Euler Pitch Angle Rate rad_p_s</td>
<td>Euler pitch rate, L frame</td>
<td>RAD/SEC ANU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PSID</td>
<td>Euler Yaw Angle Rate rad_p_s</td>
<td>Euler yaw rate, L frame</td>
<td>RAD/SEC ANR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Variable Names – Issues – Units

• Why units? Compare

\[
\text{CLFlaps0} = \text{CLAlfa}*\text{angleOfAttack} + \text{CLDe} * \text{De} + \\
\text{CLQ}\text{QB}\text{chord}/(2.0*\text{trueAirspeed})
\]

VS

\[
\text{CLFlaps0} = \text{CLALFA}_{\text{prad}}*\text{angleOfAttack}_{\text{rad}} + \\
\text{CLDe}_{\text{pdeg}}*\text{De}_{\text{deg}} + \\
\text{CLQ}_{s\text{bodyPitchRate}_{\text{radps}}*\text{chord}_f/(2.0*\text{trueAirspeed}_{\text{fps}})}
\]

VS

\[
\text{CLFlaps0} = \text{CLALFA}_{\text{pdeg}}*\text{angleOfAttack}_{\text{deg}} + \\
\text{CLDe}_{\text{pdeg}}*\text{De}_{\text{deg}} + \\
\text{CLQ}_{s\text{bodyPitchRate}_{\text{degps}}*\text{chord}_f/(2.0*\text{trueAirspeed}_{\text{fps}})}
\]
Variable Names – Units

- Conclusion – Units included makes code
  - More self documenting
  - Less ambiguous
  - Works for English or Metric System
  - Helps catch homogeneity of units errors
  - Longer to type (However typing is by far the shortest part of s/w development)
Example Variable Names

• s_BodyXVelocity_fps
• sd_BodyXAcceleration_fps2
• GEAxisZVelocity_fps
• s_BodyRollRate_radps
• YBodyThrustForce_lbf