Evaluation of a Candidate Flight Dynamics Model Simulation Standard Exchange Format

Bruce Jackson
NASA Langley Research Center
Hampton, Virginia

Bruce Hildreth
Science Applications International Corporation
Lexington Park, Maryland

Brent W. York
Naval Air Systems Command
Patuxent River, 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

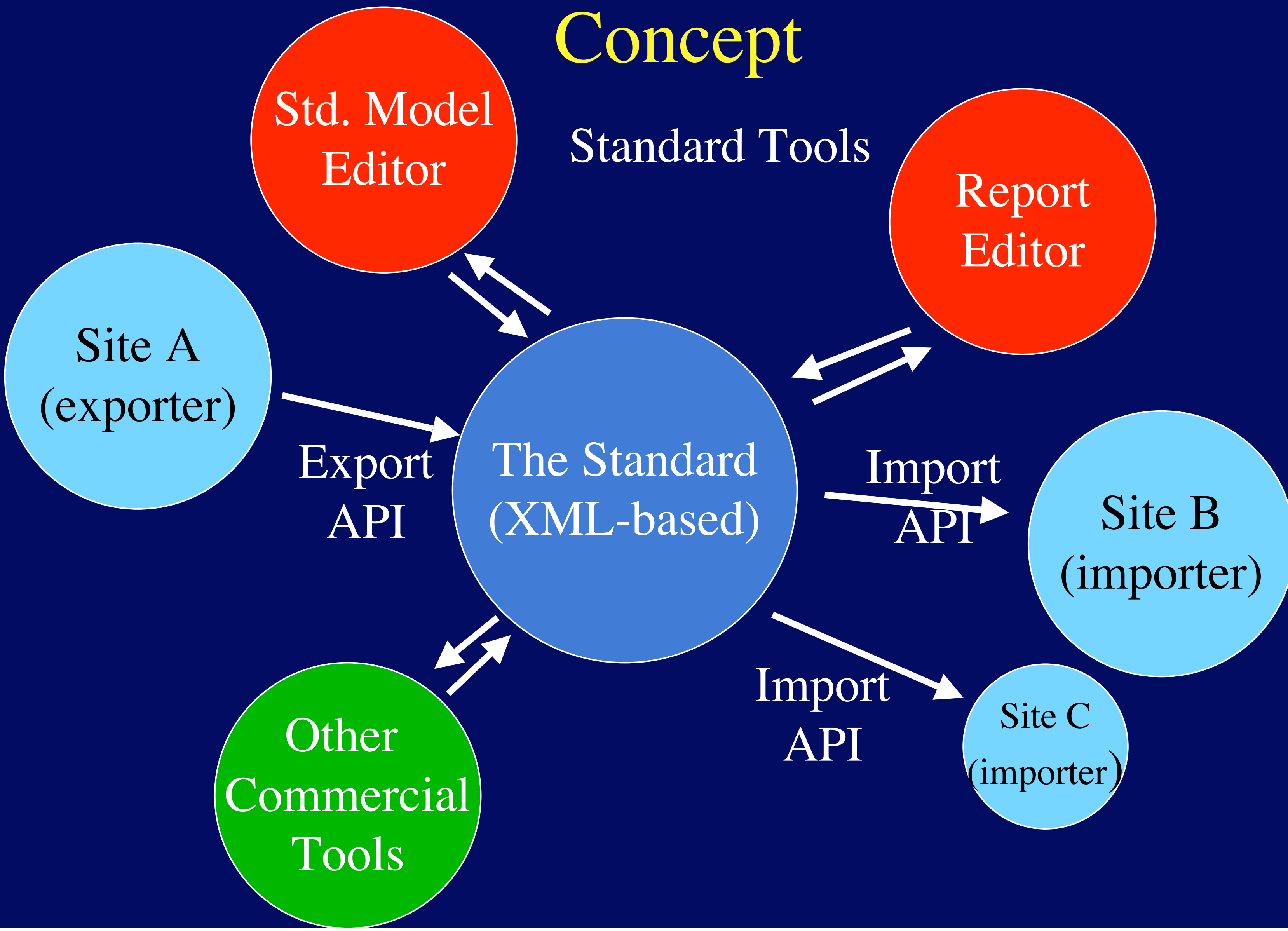
Typical: four to eight months to rehost & validate new sim

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


Std. Model Editor
Standard Tools
Report Editor
Site A (exporter)
Export API
The Standard (XML-based)
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

Proposed AIAA Sim Data Standard – Standard Function Table with statistics & provenance

Four elements per data point

Independent variables

Provenance ID

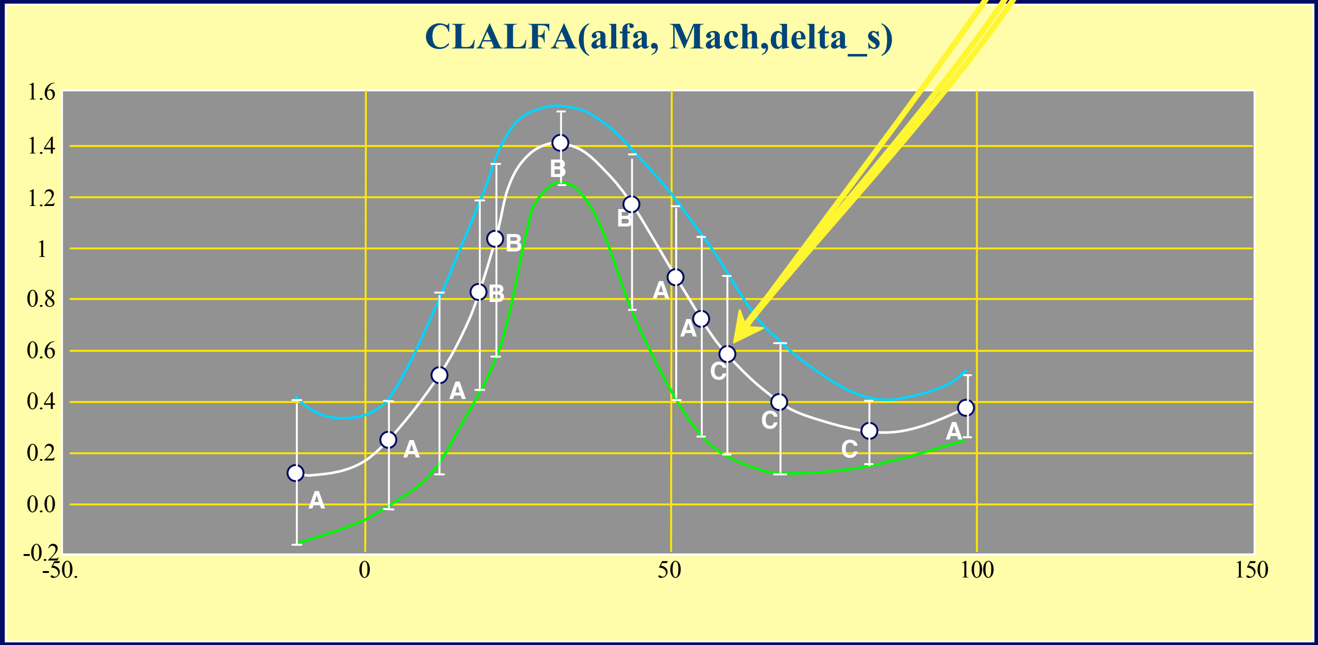
δ_s Mach α C_L Ref $-\sigma$ σ

0.0, 0.8, 60, 0.60, C , -.0032, .0068

Dependent variable

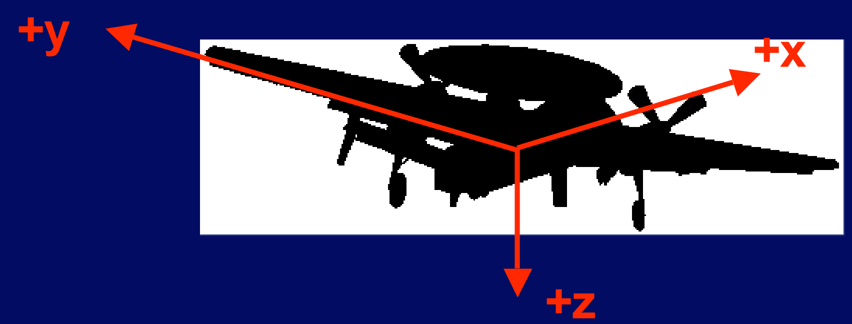
Statistics

The statistics data (the confidence intervals) are optional



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

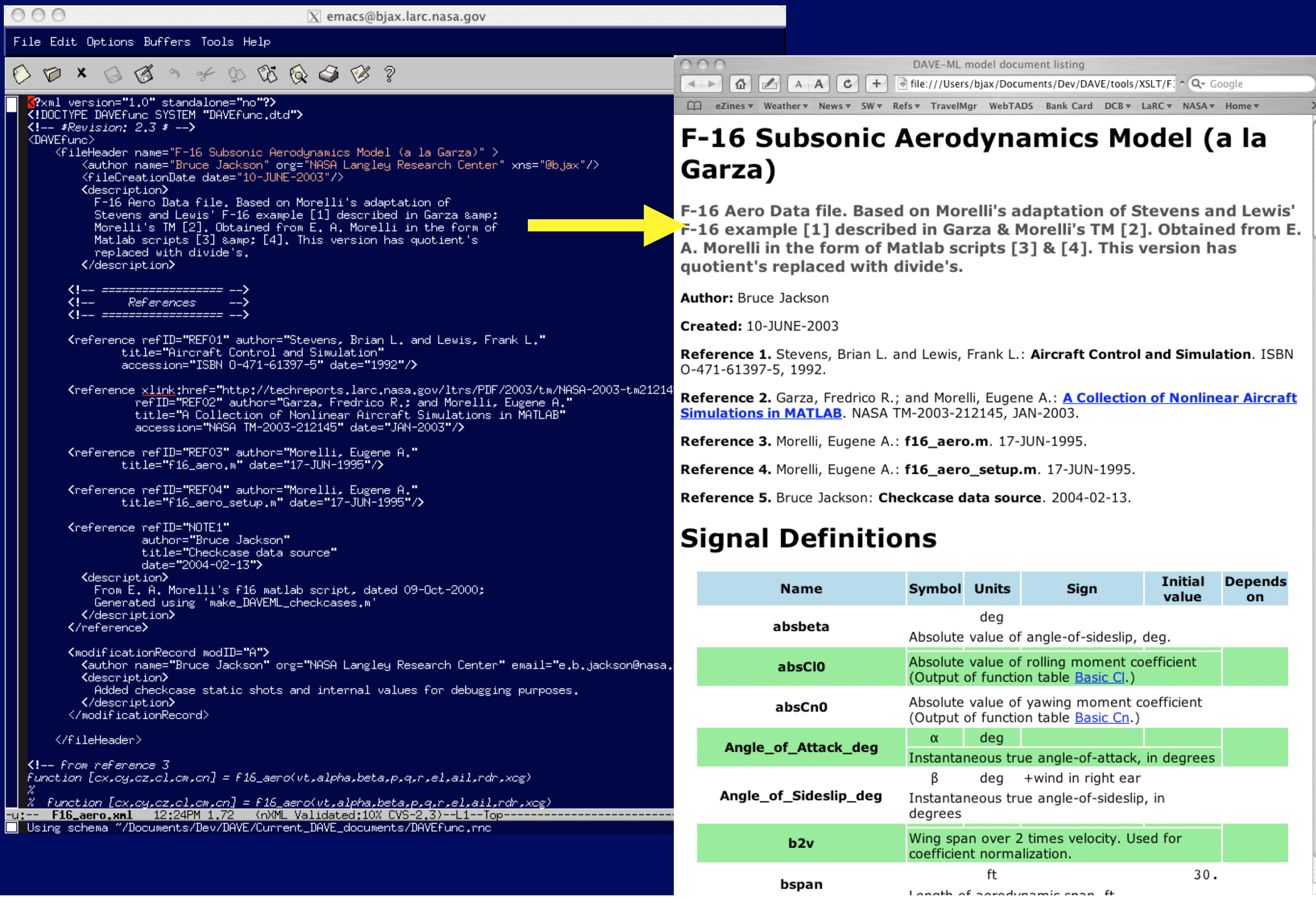
An XML approach

- 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)

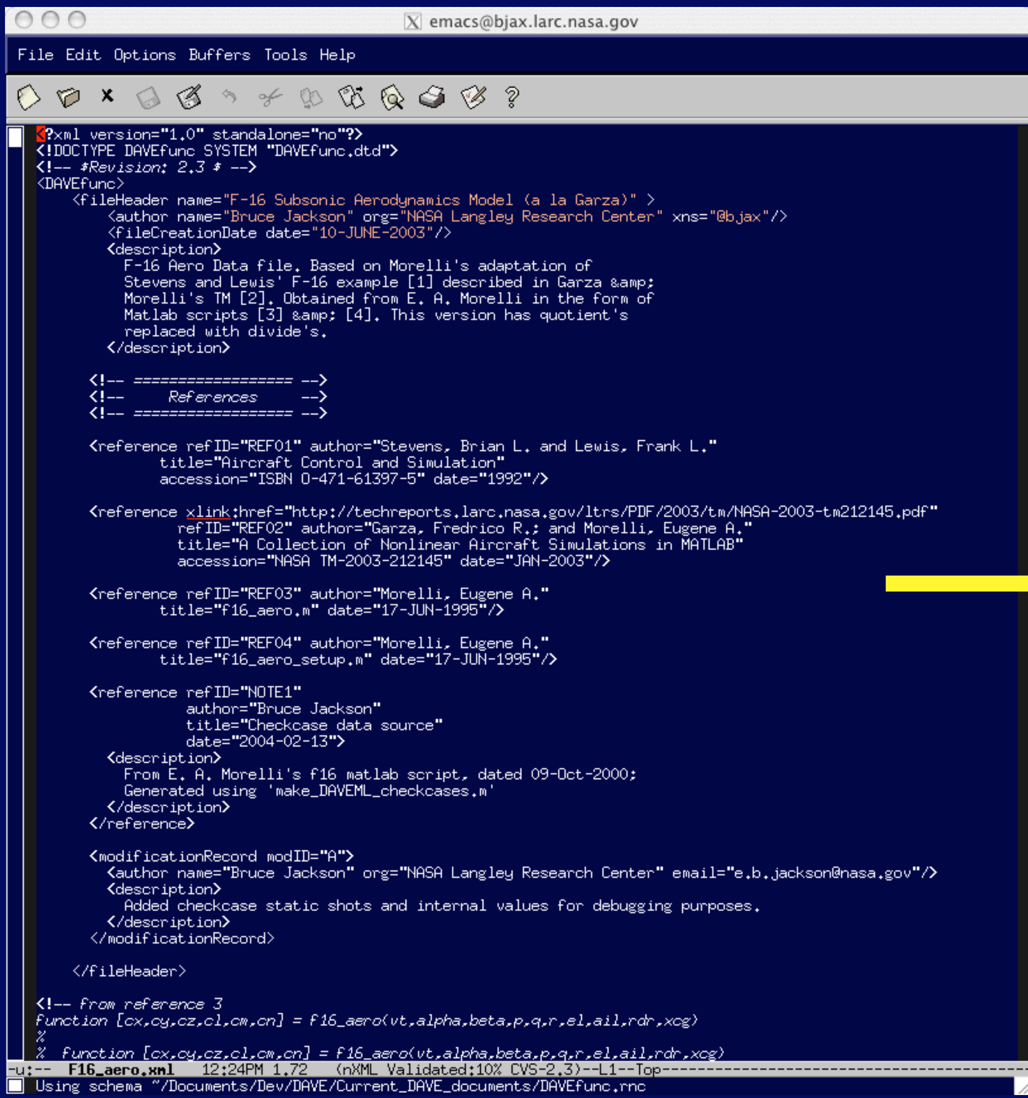
DAVE-ML features

- 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)

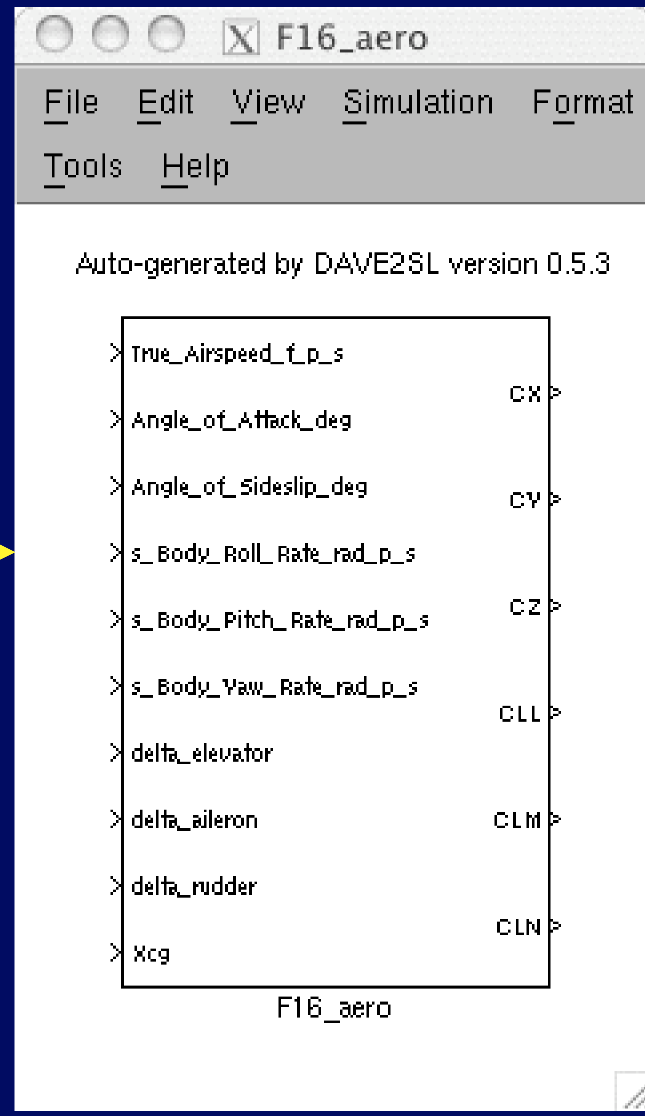
Example: DAVE-ML transformed into XHTML



Example: DAVE-ML converted to Simulink®



```
emacs@bjax.larc.nasa.gov
File Edit Options Buffers Tools Help
<?xml version="1.0" standalone="no"?>
<!DOCTYPE DAVEfunc SYSTEM "DAVEfunc.dtd">
<!-- $Revision: 2.3 $ -->
<DAVEfunc>
    <fileHeader name="F-16 Subsonic Aerodynamics Model (a la Garza)" >
        <author name="Bruce Jackson" org="NASA Langley Research Center" xns="@bjax"/>
        <fileCreationDate date="10-JUNE-2003"/>
        <description>
          F-16 Aero Data file. Based on Morelli's adaptation of
          Stevens and Lewis' F-16 example [1] described in Garza &amp;
          Morelli's TM [2]. Obtained from E. A. Morelli in the form of
          Matlab scripts [3] &amp; [4]. This version has quotient's
          replaced with divide's.
        </description>

      <!-- ================== -->
      <!--      References    -->
      <!-- ================== -->

      <reference refID="REF01" author="Stevens, Brian L. and Lewis, Frank L."
              title="Aircraft Control and Simulation"
              accession="ISBN 0-471-61397-5" date="1992"/>

      <reference xlink:href="http://techreports.larc.nasa.gov/ltrs/PDF/2003/tm/NASA-2003-tm212145.pdf"
                refID="REF02" author="Garza, Fredrico R.; and Morelli, Eugene A."
                title="A Collection of Nonlinear Aircraft Simulations in MATLAB"
                accession="NASA TM-2003-212145" date="JAN-2003"/>

      <reference refID="REF03" author="Morelli, Eugene A."
              title="f16_aero.m" date="17-JUN-1995"/>

      <reference refID="REF04" author="Morelli, Eugene A."
              title="f16_aero_setup.m" date="17-JUN-1995"/>

      <reference refID="NOTE1"
                 author="Bruce Jackson"
                 title="Checkcase data source"
                 date="2004-02-13">
        <description>
          From E. A. Morelli's f16 matlab script, dated 09-Oct-2000;
          Generated using 'make_DAVEML_checkcases.m'
        </description>
      </reference>

      <modificationRecord modID="A">
        <author name="Bruce Jackson" org="NASA Langley Research Center" email="e.b.jackson@nasa.gov"/>
        <description>
          Added checkcase static shots and internal values for debugging purposes.
        </description>
      </modificationRecord>

    </fileHeader>

<!-- from reference 3
function [cx,cy,cz,cl,cm,cn] = f16_aero(vt,alpha,beta,p,q,r,el,ail,rdr,xcg)
%
%  function [cx,cy,cz,cl,cm,cn] = f16_aero(vt,alpha,beta,p,q,r,el,ail,rdr,xcg)
-u:--  F16_aero.xml   12:24PM 1.72   (nXML Validated:10% CVS-2.3)--L1--Top----
Using schema ~/Documents/Dev/DAVE/Current_DAVE_documents/DAVEfunc.rnc
```



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

RP Demonstration I (2003-2004)

- 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

NASA Ames

- 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?)

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

SAIC An Employee-Owned Company

NAVAIR WWW.NAVAIR.NAVY.MIL

Variable Names – key points

- Variable Naming convention includes:
  - Identification of Simulation States and Inputs
  - Units- either English or SI
- Linear System Formulation
  - x = states
  - u = inputs (or controls)

$$dx/dt = Ax + Bu$$
$$Y = Cx + Du$$

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` — `s_` prefix indicates that this variable is a state
  - `sd_bodyXAcceleration_fps2` — `sd_` prefix indicates that this variable is a state derivative
  - `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

Symbol	Short Name 8 Character	Long Name 33 Character name	Same as STARS?	Description (including axis system if applicable)	Units	Sign Convention	Initial Value	Minimum Value	Maximum Value	Reference	Note	Date Last Changed
	PHI	Euler_Roll_Angle_deg	y	Roll Euler Angle, L (local) Frame	DEG	RWD		-180	180		2	
	THET	Euler_Pitch_Angle_deg	y	Pitch Euler Angle, L (local) Frame	DEG	ANU		-90	90		2	
	PSI	Euler_Yaw_Angle_deg	m	Yaw Euler Angle, L (local) Frame	DEG	ANR		-180	180		2	
Φ	PHIR	Euler_Roll_Angle_rad	y	Roll Angle, L frame	RAD	RWD		$-\pi$	π	10) 1.3.3.3	1,2	
Θ	THETR	Euler_Pitch_Angle_rad	y	Pitch Angle, L frame	RAD	ANU		$-\pi/2$	$\pi/2$	10) 1.3.3.2	1,2	
Ψ	PSIR	Euler_Yaw_Angle_rad	m	Yaw Angle, L frame	RAD	ANR		$-\pi$	π	10) 1.3.3.1	1,2	
	PHID	Euler_Roll_Angle_Rate_rad_p_s	y	Euler roll rate, L frame	RAD/SEC	RWD						
	THED	Euler_Pitch_Angle_Rate_rad_p_s	y	Euler pitch rate, L frame	RAD/SEC	ANU						
	PSID	Euler_Yaw_Angle_Rate_rad_p_s	m	Euler yaw rate, L frame	RAD/SEC	ANR						

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Variable Names – Issues – Units

- Why units? Compare

```
CLFlaps0 = CLAlfa*angleOfAttack + CLDe * De +
CLQ*QB*chord/(2.0*trueAirspeed)
```

vs

```
CLFlaps0 = CLALFA_prad*angleOfAttack_rad +
CLDe_pdeg*De_deg+
CLQ*s_bodyPitchRate_radps*chord_f/(2.0*trueAirspeed_fps)
```

vs

```
CLFlaps0 = CLALFA_pdeg*angleOfAttack_deg +
CLDe_pdeg*De_deg+
CLQ*s_bodyPitchRate_degps*chord_f/(2.0*trueAirspeed_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